Clustering Markov Decision Processes for Continual Transfer

Subramanian Ramamoorthy and M. M. Hassan Mahmud
Majd Hawasly and Benjamin Rosman
School Of Informatics
University of Edinburgh

Abstract

We consider the problem of a lifelong learning agent that continually encounters a sequence of tasks. After sufficient exposure to the domain, it is conceivable that the agent has developed a large library of task-specific optimal policies that may then be used as a basis for transfer learning - to speed up operation in a novel instance of a task from this domain. We present an algorithm for continual transfer learning in this setting of reinforcement learning with MDPs. A key question now is - is it possible to concisely encode the library to enable efficient transfer and search? Towards this end, we present a theory, framework and algorithms to cluster MDPs to obtain a source MDP set which optimize the reuse performance of our transfer algorithm. Our contributions are as follows. We present a principled policy-reuse algorithm that optimally reuses a given set of source policies when solving a particular MDP. We then present a framework for clustering the previously solved MDPs to obtain a source MDP set which optimize the reuse performance of our transfer algorithm. The framework consists of (i) a class of distance functions over MDPs which helps define clustering of MDPs; (ii) a cost function which measures how good a particular clustering is for generating useful source tasks for the transfer algorithm and (iii) provably convergent optimization algorithm for finding the optimal cluster. Finally, we present a set of experiments that illustrate the efficacy of our approach.

1 Introduction

Reinforcement learning (RL) in Markov decision processes (MDPs) is a well known framework in machine learning for modelling artificial agents [Puterman, 1994][Sutton and Barto, 1998]. In transfer learning for RL in MDPs (TLRL), the goal is to solve a particular RL problem (the target task) quicker by using information gained from previously solved related tasks (the source tasks) – see [Taylor and Stone, 2009] for a recent, comprehensive survey. In this paper we consider TLRL for the case of a ‘lifelong’ learning agent that learns many different MDPs and builds up a large collection of optimal policies for the MDPs learned. To reuse the optimal policies efficiently when solving a new target MDP, the agent needs to find a concise encoding of the previous MDPs so as to reduce loss incurred while trying to reuse irrelevant policies (indeed, not finding a concise encoding may make transfer computationally infeasible). Of course, the encoding must also ensure that relevant policies are well represented – so we need to trade-off these two contradictory objectives.

In this paper we derive a framework to do precisely the above. The encoding we choose is a set of source MDPs which are a subset of the set of all previous MDPs and are also a good representative of the latter. In particular we assume that the agent learns a long sequence of MDPs with the same state and action space, but different reward and transition function. In this setting, we consider a source subset to be a good representative if for each previous MDP, there is some source MDP, such that the optimal policy of the latter is $\epsilon$-optimal when used in the former. Indeed, to construct the source MDPs, we define a novel class of distance functions between pairs of MDPs that measure how well the optimal policy of one performs in the other and vice versa.

The source set is now generated by clustering all the previous MDPs using the distance function above as the distance between MDPs. Then for each cluster, we choose a ‘landmark’ MDP, which is the MDP within the cluster that best represents the whole cluster by minimizing the maximum distance to all other elements (see (3)). These set of landmark MDPs, one for each cluster form our source MDP set. There are many possible clusterings of previous MDPs, and to choose the right one, we set the cost of a clustering to be an upper bound on the worst case regret of our policy reuse algorithm when using the source tasks obtained from the clustering to perform transfer.

The goal of a so-called policy reuse algorithm [Fernandez et al., 2006] is to, given a set of source policies as input, make sure to not perform much worse than any one of the source policies and a pure-reinforcement learning algorithm like Q-learning. We derive a principled policy reuse algorithm by adapting the EXP-3 algorithm [Auer et al., 2002] for non-stochastic bandits to use the source policies and the Q-learning policy as arms. The regret bound of EXP-3 translates to a regret bound for our algorithm and this is used to assign costs to our clusterings. Given the cost function for clus-
terings, we now have a combinatorial optimization problem to find the best cluster. To solve this, we propose a novel optimization algorithm that can be thought of as a simulated annealing with stochastic cooling schedule (hence we avoid the very difficult problem of setting the cooling schedule, which makes the algorithm of independent interest).

Our overall continual transfer algorithm is as follows. We continually learn MDPs given to us in sequence. When learning a particular MDP, we use the optimal policies of previous MDPs in a policy reuse transfer learning algorithm. To make transfer more effective, every so often we cluster the previous MDPs and then derive a small subset as the set of source tasks and use those as input to the policy reuse algorithm.

In the following we proceed as follows. We present related work and then Preliminaries in Sections 1.1 and 2 respectively. Then we define our transfer learning algorithm and framework for measuring distance in Sections 3 and 4. Sections 5 and 6 presents our clustering algorithm and the full continual transfer algorithm. We then present our experiments in Section 7 and then end with a conclusion in Section 8. Due to lack of space and in the interest of presenting a complete method, most of the proofs are in sketch form. These are all available in the longer version.

1.1 Related Work

To the best of our knowledge, no prior work considers the problem of clustering MDPs for the purpose of extracting a core set for the purposes of transfer. But the overall goal of finding abstractions for exploiting commonality has received considerable attention in TLRL. These include MDP homomorphisms [Ravindran and Barto, 2003] [Ferns et al., 2004], [Ravindran, 2013], [Konidaris and Barto, 2007], [Sorg and Singh, 2009], [Castro and Precup, 2010] and the proto-value function based approach [Ferrante et al., 2008]. An important research issue in these methods (particularly MDP homomorphisms) is that of effective approximation and scaling. These methods are quite different from our approach in the details, but the most fundamental difference seems to be that we are interested in closeness in terms of the policies replicating optimal behavior, while these methods are interested in closeness in terms of the transition/reward/value functions replicating the same in the other MDP.

A recent related work is [Coviello et al., 2013] which cluster together HMMs that encode similar distributions over state space. Here again, this is not directly applicable to our case, where we need to determine if tasks are similar in terms of policies.

2 Preliminaries

We use $\hat{=} \equiv$ for definitions, $Pr$ for expectation. A finite MDP $M$ is defined by the tuple $(S, A, R, P, R, \gamma)$ where $S$ is a finite set of states, $A$ is a finite set of actions and $R = [1, u] \subset \mathbb{R}$ is the set of rewards. $P(s'|s, a)$ is a the state transition distribution for $s, s' \in S$ and $a \in A$ while and $R(s, a)$, the reward function, is a random variable taking values in $R$. Finally, $\gamma \in [0, 1)$ is the discount rate.

A (stationary) policy $\pi$ for $M$ is a map $\pi : S \rightarrow A$. For a policy $\pi$, the Q function $Q^\pi : S \times A \rightarrow \mathbb{R}$ is given by $Q^\pi(s, a) = \mathbb{E}[R(s, a)] + \gamma \sum_{s'} P(s'|s, a)Q^\pi(s', \pi(s'))$. The value function for $\pi$ is defined as $V^\pi(s) = Q^\pi(s, \pi(s))$. An optimal policy $\pi^*$ is defined as $\pi^* = \arg \max_{\pi} V^\pi$ – the Q function is given by $Q^*(s, a) = \mathbb{E}[R(s, a)] + \gamma \sum_{s'} P(s'|s, a) \max_{\pi} Q^*(s', \pi)$. For the optimal policy the value functions is denoted by $V^*$. The goal of the agent is to estimate $Q^*$ and then choose the action $\arg \max_{a} Q^*(s, a)$ at state $s$.

3 Policy Reuse As Nonstochastic Bandits

In the policy reuse transfer problem, we have a target task $\mathcal{M}$ and a set of $c$ source policies $\rho_1, \rho_2, \cdots, \rho_c$ (which are, in our case, optimal policies of $c$ source tasks). Our goal is to re-use the optimal policies $\rho_i, 1 \leq i \leq c$, of the $c$ source tasks as initial guesses for the optimal policy of $\mathcal{M}$ along the same lines as Policy Reuse algorithm of [Fernandez et al., 2006] [Fernandez et al., 2010]. The algorithm we present simply adapts the EXP-3 algorithm [Auer et al., 2002] for non-stochastic bandits, and can be considered as putting Policy-Reuse in a framework where we can obtain regret bounds. We run EXP-3 with $c + 1$ arms, where arms $1$ to $c$ each correspond to a $\rho_i$ and arm $c + 1$ corresponds to the Q-learning policy [Watkins, 1989]. As Q-learning is a pure RL algorithm, by using Q-learning as an arm, we are able to balance reuse the source task policies and pure reinforcement learning. Our algorithm, EXP-3-Transfer is given in Algorithm 1. When arm $i$ is pulled at iteration $t$, we run the corresponding policy/algorithm for one episode and observe the discounted payoff $x_i(t) \triangleq \sum_{n=1}^{T} \gamma^n r_n$, where $r_n$ is the reward obtained at time step $n$ and $T$ is the time step at which the episode ended. Other than this, the algorithm is identical to EXP-3. Given this, the known bound for the EXP-3 algorithm from Corollary 3.2 [Auer et al., 2002] applies, which we restate as Theorem 1:

Theorem 1. The regret bound of EXP-3-Transfer is as follows:

$$G_{\text{max}} - \mathbb{E}[X] \leq \sqrt{e - 1} \sqrt{G(c + 1) \ln(c + 1)}$$

$$\leq 2.63 \sqrt{(c + 1) \ln(c + 1) G} \quad (1)$$

where $G_{\text{max}}$ is the payoff of the best of the $c + 1$ arms and $G$ is an upper-bound on $G_{\text{max}}$.

In the next couple of sections, we discuss how to get the $c$ source policies from a set of $N$ previous tasks.

4 Distance Between MDPs

In this section we derive a framework for measuring distance between MDPs. In Section 5, we give an algorithm that uses this distance function to cluster $N$ previous MDPs to get $c$ source MDPs, where possibly $c \ll N$. These $c$ source MDPs are used as input to the EXP-3-Transfer algorithm from section 3. The smaller set reduces the number of tasks to transfer from, which trades off the loss incurred while trying to reuse unrelated policies, and not reusing a relevant policy.

Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two MDPs with same state and action space but different transition and reward functions. Denote by $V^*_{\mathcal{M}_i}$ the value of policy $\rho$ when executed in $\mathcal{M}_i$. Let the
Algorithm 1 EXP-3-Transfer($\mathcal{M}, \{\rho_1, \rho_2, \cdots, \rho_c\}, \beta, T$)

1. **Input:** MDP $\mathcal{M}$, arms 1 to $c$, the source policies $\rho_1, \cdots, \rho_c$, and EXP-3 parameters $\beta$ and $T$.
2. **Initialize:** Set $w_i(1) = 1$, let $x_i(t)$ be the per-episode payoff of the arms, Q-learning policy as the $c+1$th arm.
3. **for** $t = 1$ to $T$ **do**
   4. Set $p_i(t) = (1 - \beta)\frac{w_i(t)}{\sum_{j=1}^{c+1} w_j(t)} + \frac{\beta}{c+1}$.
   5. Select arm $i_t$ for step $t$ to be $i$ with probability $p_i$.
   6. Run chosen arm for one-episode, and observe discounted payoff $x_{i_t}(t)$.
   7. Set $x_j(t) \leftarrow x_j/p_j(t)$ if $j = i_t$; otherwise $x_j(t) \leftarrow 0$. Update $w_j(t + 1) \leftarrow w_j(t) \exp[\beta x_j(t)/(c + 1)]$.
8. **end for**

optimal policies for the two MDPs be $\pi_1^*, \pi_2^*$. Let the initial distribution over states of $\mathcal{M}_i$ be $In_i$. We define the optimal policy similarity between two MDPs as follows.

$$d_V(\mathcal{M}_1, \mathcal{M}_2) \triangleq \max\{\mathbb{E}_{x \sim In_i} |V_1^*(s) - V_2^*(s)|, \mathbb{E}_{x \sim In_2} |V_2^*(s) - V_1^*(s)|\}$$

We have the following Lemma by construction:

**Lemma 1.** If $d_V(\mathcal{M}_1, \mathcal{M}_2) \leq \epsilon$, then the optimal policy of $\mathcal{M}_1$ is at least $\epsilon$ optimal in $\mathcal{M}_2$ and vice versa.

We can now define the class of distance function that our method uses. These are called value preserving Lipschitz metrics and has a close relationship to $d_V$.

**Definition 1.** Let $\mathcal{M}$ be the set of MDPs defined over the same state and action space. Then, we call function $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ a value preserving Lipschitz metric (VPL metric in short) if it satisfies the following conditions:

1. $d(\mathcal{M}, \mathcal{M}') \geq 0$.
2. $d(\mathcal{M}, \mathcal{M}) = 0$.
3. $d(\mathcal{M}, \mathcal{M}') = d(\mathcal{M}', \mathcal{M})$.
4. $d(\mathcal{M}, \mathcal{M}') = \epsilon$ implies that $d_V(\mathcal{M}, \mathcal{M}') \leq k(\epsilon)$ where $k$ is a monotonically increasing function dependent only in $d$.
5. $d(\mathcal{M}, \mathcal{M}') \leq K[d(\mathcal{M}, \mathcal{M}'')] + d(\mathcal{M}', \mathcal{M}'')]$ where $K$ is a constant dependent only on $d$.

The first three conditions are standard conditions for metrics. The fourth condition dictates the dependency of $d$ on $d_V$. In the fifth condition the triangle inequality is replaced by a Lipschitz type inequality. These last two conditions give the qualifications ‘value-preserving’ and ‘Lipschitz’ to the metric. A VPL metric is useful because it allows us to almost operate on the space of MDPs as if it is a metric space while still ensuring that if two points are close in terms of the VPL metric, then they have optimal policies that work well in the other MDP.

The way a VPL metric $d$ may be used to obtain the $c$ source policies is illustrated in figure 1. Given the $N$ MDPs, we separate them into clusters $A_i$, $i = 1 \cdots c$ such that $d(\mathcal{M}_i, \mathcal{M}_j) \leq \epsilon$ for any two $\mathcal{M}_i, \mathcal{M}_j \in A_i$ (so $A_i$ is reminiscent of an $\epsilon$-ball in a metric space). The number $c$ is the number of clusters created by a clustering algorithm from the $N$ MDPs. For each cluster $A_i$, the landmark MDP is defined to be:

$$\mathcal{M}^i = \arg\min_{\mathcal{M} \in A_i} \max_{\mathcal{M}' \in A_i} d(\mathcal{M}, \mathcal{M}')$$

That is, the landmark for $A_i$ is the MDP whose optimal policy, in the worst case, performs better than all the other MDPs. By the Lipschitz condition 5, any new MDP that is close to one of the $N$ MDPs in terms of $d$ will also be close to some landmark. By the value preserving condition 4, the optimal policy of the landmark will also be useful in the new MDP. Hence, clustering using $d$ gives us a good way to select the $c$ source tasks from the $N$ previous tasks.

**A VPL Metric.** Our experiments use the following VPL metric (our results apply for any possible VPL metric).

$$d(\mathcal{M}, \mathcal{M}') \triangleq \max_{s,a} \left\{ |R(s, a) - R'(s, a)|, \|T(\cdot|s, a) - T'(\cdot|s, a)| \right\}$$

where $\| \cdot \|_1$ is the $L_1$ norm of the two probability vectors. So we have the following:

**Lemma 2.** The function $d$ defined in 4 is a VPL metric.

**Sketch.** Conditions 1-3 and 5 of a VPL metric follows from the fact that we are taking max of two metrics (with $K = 1$), while 4 follows from Lemma 1 of [Strehl et al., 2009] with $k(x) = \frac{x(1 + \gamma R_{max})}{(1 - \gamma)^2}$, were $R_{max} \triangleq \max_{s,a} R_i(s, a)$ ($\gamma$ is the MDP discount rate).

5 Clustering MDPs

In this section we show how to cluster a set of $N$ previous MDPs into $c$ clusters to derive $c$ source tasks for input to EXP-3-Transfer. To that end, we first derive present a cost-function for clusterings, which gives us a discrete optimization problem. To solve this, we present simulated annealing with stochastic temperature changes, a novel, provably convergent, version of simulated annealing. In the sequel we assume a fixed VPL metric $d$ and a set of previous MDPs $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_N$.
5.1 Cost Function For Clusters

Our goal is to divide the $\mathcal{M}_i$s into $c$ clusters $A_1, A_2, \ldots, A_c \triangleq A$ such that $\mathcal{M}, \mathcal{M}' \in A_i \Rightarrow d(\mathcal{M}, \mathcal{M}') \leq \epsilon$. The source tasks then correspond to the $c$ landmarks $\mathcal{M}_i$, chosen according to (3). The optimal policy $\rho^*$ of $\mathcal{M}'$ are then given as input to EXP-3-Transfer. Hence, an appropriate measure of the cost of a clustering $A$ is an upper bound on the regret of EXP-3-Transfer when given $\rho_i$ as input.

To that end, we ascribe parameters $(c, \epsilon)$ to each clustering $A$, where $c$ is the number of clusters and $\epsilon \triangleq \max_{i, \max_{\mathcal{M}_i \in A}, d(\mathcal{M}_i, \mathcal{M}_j)}$ (i.e., $\epsilon$ is the maximum diameter of a clustering in $A$). Let

$$g(c) \triangleq 2.63 \sqrt{(c + 1) \ln(c + 1)G} \quad (5)$$

which, by Theorem 1, upper bounds regret of EXP-3-Transfer when using $c$ source policies $\rho_i$. Let $\mathcal{M}_{N+1}$ denote the unknown target task and let $\epsilon' = \min_j d(\mathcal{M}_i, \mathcal{M}_{N+1})$ where $1 \leq j \leq N$. In the worst case setting $\mathcal{M} \triangleq \arg \min_i d(\mathcal{M}_i, \mathcal{M}_{N+1})$ can be any of the $N$ previous MDPS and so if we use $\pi^*$ in $\mathcal{M}_{N+1}$, then by Lemma 1 the regret is given by $k(\epsilon')$. Now if $A \in A_i$, by definition of the parameter $\epsilon$ and condition 5 in Definition 1, we have that

$$d(\mathcal{M}_{N+1}, \mathcal{M}_i) \leq K[d(\mathcal{M}_{N+1}, \mathcal{M}) + d(\mathcal{M}_i, \mathcal{M})]$$

$$\leq K[\epsilon' + \epsilon]$$

Condition 4 of Definition 1 implies $d_V(\mathcal{M}_{N+1}, \mathcal{M}_i) \leq k[K[\epsilon' + \epsilon] + \epsilon]$. Since the regret bound $g(c)$ holds against any of the arms/source policies, we have that the overall regret of EXP-3-Transfer against the optimal policy of the target task, when the clustering with parameters $(c, \epsilon)$ is used to generate the source policies, is

$$f_{\epsilon}(c, \epsilon) \triangleq k[K[\epsilon' + \epsilon] + \epsilon] + g(c) \quad (6)$$

Since $\epsilon'$ term of $f_{\epsilon}(\cdot, \cdot)$ is independent of the clustering (depending only on the previous MDPS that we do not control) we fix some finite arbitrary $\epsilon'$ and then use $f(c, \epsilon) \triangleq f_{\epsilon}(c, \epsilon)$ as the cost for a clustering with parameter $(c, \epsilon)$. Putting all the material in this subsection together, we get:

**Lemma 3.** The cost function $f(c, \epsilon)$ for a clustering $A$ upper bounds the worst case regret when EXP-3-Transfer is run with the sources selected from $A$ according to (3).

In the sequel, we will denote by $f(A)$ the number $f(c, \epsilon)$ where $A$ has parameters $c$ and $\epsilon$.

5.2 Finding the Optimal Cluster

In this section we derive an algorithm to solve the discrete optimization problem of computing $\arg \min_{A \in \mathcal{C}} f(A)$ where $\mathcal{C}$ is the set of all possible clusterings of $\mathcal{M}_i$s. Our basic strategy is to sample repeatedly from a distribution that concentrates around the optimum. This way, we are guaranteed with high probability to hit the optimum clustering. Exact sampling from the distribution is difficult, and so our algorithm samples approximately from this distribution using a MCMC approach – see [Robert and Casella, 2005] for a comprehensive introduction to MCMC and Metropolis Hastings Markov chains (MH chain in short) that we use. Our algorithm is essentially simulated annealing with stochastic temperature changes, which means that we bypass the very hard problem of setting the cooling schedule in the standard version of the algorithm. We give a simple proof of convergence and speed of convergence of the algorithm (see, for instance, [Locatelli, 2000] for contrast). The algorithm, Search-Clusterings is given in Algorithm 2 and the distributions used in the algorithm are described below.

**The Target Distribution.** The basic idea of the algorithm is to sample from the target distribution $\Pi(\lambda, A) = \lambda^{-f(A)}$ where $A$ is a clustering and $\lambda \in I$, where $I$ is a finite subset of $\mathbb{R}^+$. The requirement on $I$ is as follows. Let $\hat{A}^* \subseteq C$ be the set of clusterings with cost that we deem acceptable, where $C$ is the set of all clusterings of $\mathcal{M}_i$s. Then $I$ should have a subset $\hat{\lambda}$ such that $\Pi(\lambda, A)$ concentrates around $\hat{\lambda} \times \hat{A}^*$. This is satisfied if, for instance, $\exists \hat{\lambda} \subseteq I$ such that $\sum_{A \subseteq \hat{\lambda}, A \subseteq C, \lambda \leq \hat{\lambda}, A \subseteq \lambda^{-f(A)} \gg \sum_{A \subseteq \hat{\lambda}, A \subseteq C, \lambda \leq \hat{\lambda}, A \subseteq \lambda^{-f(A)}$. Hence, if we can sample repeatedly from the distribution $\Pi$, we will with high probability quickly find the acceptable clusterings. We will now construct the MH chain with $\Pi$ as the stationary distribution. This means that if we simulate the chain long enough, we will start sampling from $\Pi$. In fact this is precisely what Search-Clusterings does, i.e., simulates the MH chain.

**The MH Chain.** In the following, we use theory of Markov chains as found in (for instance) [Levin et al., 2009]. The MH chain has state space $I \times \mathcal{C}$ and is defined in terms of a proposal distribution $\phi(z, w)$ that gives the probability of choosing $w$ as a candidate for the state of the chain given that $z$ is the current state. Given $\phi$ (to be defined below) the (well-known) transition kernel $\mu$ for the MH chain is:

$$\text{Acc}[(\lambda, A), (\lambda', A')] \triangleq \min \left\{ \phi((\lambda', A'), (\lambda, A))\Pi(\lambda', A'), 1 \right\}$$

$$\mu([(\lambda, A), (\lambda', A')]) \triangleq \phi((\lambda, A), (\lambda', A'))\Pi(\lambda, A)\text{Acc}((\lambda, A), (\lambda', A'))$$

(7)

Where $\mu(z, w)$ is the probability of transitioning to state $w$ given current state $z$. We now have

**Lemma 4.** Search-Clusterings simulates the Markov chain with transition kernel $\mu$ defined in (7)

**Proof.** In lines 5 and 6 of Search-Clusterings, we sample a candidate next state using $\phi$ and then accept that candidate using Acc, which is precisely the definition of $\mu$. \qed

**The Proposal Distribution.** We will define $\phi$ in a way so that the above chain is irreducible and a-periodic. Together with the construction of MH, it can be shown that this chain in fact converges to $\Pi$. Let $I = \{\lambda_1, \lambda_2, \ldots, \lambda_L\}$ such that $\lambda_i > \lambda_{i-1}$. The proposal distribution $\phi((\lambda, A), (\lambda', A'))$ is defined using three parameters $\alpha, \alpha', \beta$, as follows.

- With probability $\alpha$, $\lambda \neq \lambda'$ and $A = A'$. Let $\lambda = \lambda_i$; then with probability $\alpha'$, $\lambda' = \lambda_{i+1}$, and probability $1 - \alpha'$ a $\lambda' = \lambda_{i-1}$.

- With probability $\beta$, $\lambda = \lambda'$, $A \neq A'$. $A'$ is generated from $A$ by the Rearrangement procedure, which converts $A$ to $A'$ and is described below.
With probability $1 - \alpha - \beta$, $\lambda' = \lambda$ and $A' = A$.

In the above we require that $1 - \alpha - \beta > 0$.

**Algorithm 2** Search-Clusterings($M, d, \text{term}$)

1: **Input:** A set of MDPs $M = \{M_1, M_2, \ldots, M_N\}$, a VPL metric $d$, termination predicate $\text{term}$.
2: **Output:** A Clustering $A$.
3: **Initialize:** Initial, arbitrary clustering $A(0)$ of $M$, initial value of $\lambda(0) = \lambda_0$, $l = 0$.
4: **while** $\text{term}(\lambda(0), \ldots, \lambda(l), A(0), \ldots, A(l))$ is **false**
5: Sample $(\lambda, A)$ according to $\phi(\lambda(l), A(l), \cdot)$.
6: Defining $\theta \triangleq \text{Acc}[\lambda(l), A(l), (\lambda, A)]$, set
   
   $(\lambda(l+1), A(l+1)) \leftarrow \begin{cases} (\lambda, A) & \text{with prob. } \theta \\ (\lambda(l), A(l)) & \text{with prob. } 1 - \theta \end{cases}$
7: $l \leftarrow l + 1$.
8: **end while**
9: **return** $\arg\min_I f(A(l))$.

**Rearrangement Procedure.** Given a clustering $A = A_1, \ldots, A_n$, we choose an $A_i$ uniformly at random. We then choose $k_i > 0$ points of $A_i$ according to the exponential distribution over $\mathbb{N}^+$ truncated to have support $1, 2, \ldots, |A_i|$: $P(E(k; \theta_i) \triangleq \frac{1 - \exp(-\theta_i)}{\exp(-\theta_i)} \sum_{m=1}^{\infty} \exp(-\theta_i((m-1)|A_i| + k)]$.

Then we choose another $A_j \in (A - \{A_i\}) \cup \{B\}$ uniformly at random, where $B$ is an empty set representing a new cluster. We then move the points of $A_i$ chosen to $A_j$. Note that if $A_j = B$, then $A_i$ loses points, which are then used to create a new cluster. The above procedure converts $A$ to $A'$.

**5.3 Convergence of Search-Clusterings**

The main result in this section is Theorem 2 which shows that the $t$ step distribution over states of Search-Clusterings converges to $\Pi$. Recall that the total variation distance between two distributions $P, P'$ over a space $\Omega$ is defined as $TV(P, P') \triangleq \max_{A \subset \Omega} |P(A) - P'(A)|$. We use notions from Markov chain theory as found in (for instance) [Levin et al., 2009].

**Lemma 5.** Let $\mu^t(w, z)$ denote the probability that we reach state $w$ from $z$ in $t$ steps. Then, the chain defined using the transition kernel in (7) is $\alpha$-periodic and irreducible with stationary distribution $\Pi(\lambda, A)$. In particular $\lim_{t \to \infty} TV(\Pi, \mu^t) = 0$.

**Proof. Sketch.** From the definition of $\phi$, it can be seen that there is a sequence of 1 step moves of finite probability that takes use from any $(\lambda, A)$ to any other $(\lambda', A')$, including $(\lambda, A)$ itself. This establishes irreducibility and $\alpha$-periodicity of $\mu$. Standard results on MH chain (see Chapters 3 and 4 of [Levin et al., 2009]) establishes the convergence.

We now consider the speed of convergence and show that this depends on the diameter of the space $I \times C$. Recall that in Lemma 5 we defined $\mu^t(w, z)$ to be the probability of transitioning to $z$ from state $w$ in $t$ steps.

Define the diameter of $I \times C$ given the Markov chain (7) to be $diam(I \times C) \triangleq \min_{l \in \mathbb{N}} |\lambda(l), (\lambda', A')|$. Now define the ratio $\delta \triangleq \min_{\lambda(0), (\lambda', A')} \mu^diam(I \times C)(\lambda(0), A(0)) \cdot (\lambda', A')$. Our main convergence theorem shows that our clustering algorithm converges geometrically to the target distribution $\Pi$.

**Theorem 2.** Let $SC^t[(\lambda, A)]$ be the probability that in Search-Clusterings, $(\lambda(l), A(l)) = (\lambda, A)$. Then, we have $TV[SC^t, \Pi] \leq (1 - \delta)^t \cdot diam(I \times C)$.

**Proof.** By Lemma 4, $SC^t[(\lambda, A)] = \mu^t[(\lambda(0), A(0)), (\lambda, A)]$ where $(\lambda(0), A(0))$ are the initial $\lambda$ and clustering chosen in Search-Clusterings. From the proof of Theorem 4.9 in [Levin et al., 2009] we have $TV[\mu^t[(\lambda(0), A(0)), \cdot], \Pi] \leq (1 - \delta)^t \cdot diam(I \times C)$ which gives us the theorem.

**6 The Continual Transfer Algorithm**

We now combine all the algorithms presented so far into the full continual transfer algorithm, which is given in Algorithm 3. The algorithm runs in phases and in each phase it solves a MDP using the EXP-3-Transfer algorithm and the current set of source policies as input. In line 4, the function $sourcePol(A)$ generates the $c$ source policies $\rho_1, \rho_2, \ldots, \rho_c$ from clustering $A$ such that $\rho_j$ is the optimal policy for $M^j$ where $M^j$ is chosen from $A_j$ according to (3). If the current phase $h$ satisfies $h \mod J = 0$, then it runs the Search-Clustering algorithm to find a new set of a source tasks from the $h$ tasks solved so far.

**Algorithm 3** Continual-Transfer$(d, J, \beta, T_1, \text{term})$

1: **Input:** A VPL metric $d$, clustering period $J$, EXP-3-Transfer parameters $\beta$, $T_1$, Search-Clustering termination condition $\text{term}$.
2: **Initialize:** Initial clustering $A = 0$, collection of previous MDPs $M$.
3: **for** $h = 1$ to $\infty$ **do**
4: Get unknown MDP $M_h$ from the environment and run EXP-3-Transfer($M_h, sourcePol(A), \beta, T_1$).
5: Set $M \leftarrow M \cup \{M_h\}$
6: **if** $h \mod J = 0$ **then** $A = \text{Search-Clusterings}(M, d, \text{term})$.
7: **end for**

**7 Experiments**

We performed two sets of experiments to illustrate various aspects and efficacy of our algorithm. Section 7.1 presents the first set of experiments, where we clustered a simple set of domains by themselves to illustrate how the Search-Clustering algorithm performs; these experiments show that our algorithm gives reasonable clusters (we used parameters $\alpha = 0.1, \alpha' = 0.5, \beta = 0.8, \text{term} = 30000$ steps, $\lambda_0 = 1.2, \Delta \Lambda = 0.1$). In Section 7.2 we present the second set of experiments that show that (1) clustering does not hurt when the
number of previous tasks are small and (2) helps significantly when the number of previous tasks is large.

### 7.1 Clustering Experiments

In these set of experiments we look at clustering 100 MDPs defined on a simple chain domain but with different transition and reward functions. The results are presented in Figure 2. Task $i$ differs from $i+1$ by 0.2 according to $d$ in (4) and in Figure the x-axis corresponds to these tasks. The y-axis shows the number of tasks present and each row shows the clusters (one color per cluster). The growth of the clusters over increasing number of tasks shows the tradeoff between $c$ and cluster diameter in $f(\epsilon, c)$. The figure shows that for this particular setting the optimal set of clusters is around 4. The stochastic nature of the search algorithm prevents the clustering from being perfect but they are quite close to the optimal.

### 7.2 Transfer And Clustering Experiments

In this section we ran Continual Transfer algorithm in a domain with the number of previous tasks increasing from 1 to 100 with $J$ set to 10 (i.e. we clustered every 10 tasks). We only report the results for when the we generate a new clustering (i.e. for iteration $10j + 1$, with $j = 1$ to 10). We compared this against transferring with no clustering when the previous tasks numbered in $10j$. We show the performance for 10 different target tasks.

The domain itself was a $20 \times 20$ gridworld with 10 horizontal corridors on the east end beginning from the $10^{th}$ cell. There were 10 ‘classes’ of MDPs, one corresponding to a goal state at the end of each corridor. There was also a wind blowing south at the entrance the corridors. The 100 tasks (differing in terms of the wind probability) were distributed uniformly across these 10 classes and each of the target tasks belonged to one of these classes. The results are presented in Figure 3. Due to space constraints, we only show the results when the previous tasks number 10, 20, 30 and 100. The transfer results for the smaller number of tasks show that clustering and no-clustering does not really make much difference. In this case, the number of clusters is similar to the number of previous tasks and so it does not help too much. However, as the number of tasks increase, the performance between the two changes significantly, and at the end of 100 tasks, we get fairly consistent performance improvement across all the tasks. So this shows that clustering is indeed quite beneficial.

### 8 Conclusion

In this paper we developed a framework to concisely represent a large number of previous tasks by a smaller number of source tasks for the purpose of transfer. This encoding trades off performance loss (possibly even infeasibility of transfer) due to having too many irrelevant tasks and loss incurred due to not having the relevant task in the source set. The framework introduced the notion of a VPL metric, a novel class of distance over MDPs and a convergent optimization method to cluster the MDPs (we believe this method is of independent interest). We also presented experiments that demonstrate that clustering is indeed needed for transferring when the number of previous tasks is large. There are many possible extensions for this work. We have only explored one VPL metric, and so it would be interesting to look at other metrics, particularly ones that look at the structure of optimal policies. It would also be interesting to see if there are more effective ways of putting topology on the space of MDPs, possibly involving hierarchies. Additionally, our version of simulated annealing algorithm is of independent interest and it will be interesting to develop it fully. And last but not least, we would like to use our method in more complex problems.
Acknowledgements

This work has taken place in the Robust Autonomy and Decisions group within the School of Informatics. Research of the RAD Group is supported by the UK Engineering and Physical Sciences Research Council (grant number EP/H012338/1) and the European Commission (TOMSY Grant Agreement 270436, under FP7-ICT-2009.2.1 Call 6).

References

[Auer et al., 2002] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. SIAM Journal on Computing, 32:48–77, 2002.

[Castro and Precup, 2010] Pablo Samuel Castro and Doina Precup. Using bisimulation for policy transfer in mdps. In Proceedings of the the 24\textsuperscript{th} AAAI Conference on Artificial Intelligence, 2010.

[Coviello et al., 2013] Emanuele Coviello, Antoni B. Chan, and Gert Lanckriet. variational hierarchical em algorithm for clustering hidden markov models. In Proceedings of the Conference on Neural Information Processing Systems, 2013.

[Fernandez et al., 2006] Fernando Fernandez, Javier Garcia, and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems, 2006.

[Fernandez et al., 2010] Fernando Fernandez, Javier Garcia, and Manuela Veloso. Probabilistic policy reuse for inter-task transfer learning. Robotics and Autonomous Systems, 58:866–871, 2010.

[Ferns et al., 2004] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. 2004.

[Ferrante et al., 2008] E. Ferrante, A Lazaric, and M. Restelli. Transfer of task representation in reinforcement learning using policy-based protovalue functions. In Proceedings of the 7\textsuperscript{th} International Conference on Autonomous Agent And Multiagent Systems, 2008.

[Konidaris and Barto, 2007] George Konidaris and Andrew G. Barto. Building portable options: skill transfer in reinforcement learning. In Proceedings of the 20th International Joint Conference on Artificial Intelligence, 2007.

[Levin et al., 2009] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. Markov Chains and Mixing Times. American Mathematical Society, 2009.

[Locatelli, 2000] M. Locatelli. Simulated annealing, algorithms for continuous global optimization: convergence conditions. Journal of Optimization Theory and Applications, 104(1):121–133, 2000.

[Puterman, 1994] M. L. Puterman. Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley and Sons, 1994.

[Ravindran and Barto, 2003] Balaraman Ravindran and Andrew G. Barto. SMDP homomorphisms: An algebraic approach to abstraction in semi-markov decision processes. In Proceedings of the International Joint Conference on Artificial Intelligence, 2003.

[Ravindran, 2013] Balaraman Ravindran. Relativized hierarchical decomposition of markov decision processes. Decision making: neural and behavioural approaches, 42:465–488, 2013.

[Robert and Casella, 2005] C. P. Robert and George Casella. Monte Carlo Statistical Methods. Springer, Berling, 2005.

[Sorg and Singh, 2009] Jonathan Sorg and Satinder Singh. Transfer via soft homomorphisms. In Proceedings of the 8\textsuperscript{th} International Conference on Autonomous Agents and Multiagent Systems, 2009.

[Strehl et al., 2009] Alexander L. Strehl, Lihong Li, and Michael L. Littman. Reinforcement learning in finite MDPs: PAC analysis. Journal of Machine Learning Research, 10:2413–2444, 2009.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998.

[Taylor and Stone, 2009] Matthew Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. 10:1633–1685, 2009.

[Watkins, 1989] Christopher Watkins. Learning From Delayed Rewards. PhD thesis, Cambridge University, 1989.