Clustering Markov Decision Processes For Continual Transfer

M. M. Hassan Mahmud[†] HMAHMUD@STAFFMAIL.ED.AC.UK
Majd Hawasly[†] M.HAWASLY@ED.AC.UK
Benjamin Rosman[†,*] B.S.ROSMAN@ED.AC.UK
Subramanian Ramamoorthy[†] S.RAMAMOORTHY@ED.AC.UK
[†]*School of Informatics*
University of Edinburgh
Edinburgh, EH8 9AB, UK.
[*]*Mobile Intelligent Autonomous Systems Group*
CSIR, Pretoria, South Africa.

Editor: TBD

Abstract

We present algorithms to effectively represent a set of Markov decision processes (MDPs), whose optimal policies have already been learned, by a smaller *source* subset for lifelong, policy-reuse-based transfer learning in reinforcement learning. This is necessary when the number of *previous* tasks is large and the cost of measuring similarity counteracts the benefit of transfer. The source subset forms an 'ϵ-net' over the original set of MDPs, in the sense that for each previous MDP $\mathcal{M}_p$, there is a source $\mathcal{M}^s$ whose optimal policy has $< \epsilon$ regret in $\mathcal{M}_p$. Our contributions are as follows. We present EXP-3-Transfer, a principled policy-reuse algorithm that *optimally* reuses a given source policy set when learning for a new MDP. We present a framework to cluster the previous MDPs to extract a source subset. The framework consists of (i) a distance d_V over MDPs to measure policy-based similarity between MDPs; (ii) a cost function $g(\cdot)$ that uses d_V to measure how good a particular clustering is for generating useful source tasks for EXP-3-Transfer and (iii) a provably convergent algorithm, MHAV, for finding the optimal clustering. We validate our algorithms through experiments in a surveillance domain.

Keywords: Reinforcement learning, transfer learning, Markov decision process, MDP abstractions, policy reuse, discrete optimisation using MCMC.

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. Being able to transfer information is important if we want learning agents to scale up to autonomously and efficiently handle many different learning problems that are related.

In this paper we consider the problem of TLRL in the lifelong learning setting (Mitchell and Thrun, 1993; Thrun and Mitchell, 1995; Thrun, 1996), where the agent learns a (possibly very large) set of tasks presented in sequence on a continual basis. One issue that arises in this case is that the benefit of transfer may be outweighed by the resources spent in performing transfer. In

particular, when learning, we need to test whether the tasks we are transferring from are actually relevant to the current problem. We need to do this because otherwise we may end up with *negative transfer* (i.e. transfer hurts performance) by using the wrong task from which to transfer. However, this testing itself often results in loss both in terms of wasted learning time and accumulated negative rewards. So, if we are transferring from a large number of tasks, we need some way to encode them compactly so that we properly achieve the best tradeoff between loss due to testing and gain due to transferring.

As a motivating example, consider a surveillance agent that is monitoring a large geographical region (this is a variant of the kinds of problems that are considered, for instance, in (An et al., 2012)). The agent faces a sequence of monitoring problems where each problem corresponds to the pattern in which infiltrators appear in different locations. If two tasks have similar infiltration patterns, then the same surveillance policy may be good for both of them. During each task, the goal of the agent is to learn the regions where infiltrators appear and choose the appropriate surveillance policy. We do not expect the patterns to be completely different every time, but at the same time we cannot completely rule out a new pattern emerging. In the former case, we should recognize this repetition and take advantage of this fact by reusing old surveillance policies. In the latter case, we should also determine that the new scenario is novel and learn an appropriate policy for that scenario. Furthermore, if the number of previous patterns becomes too large, we also need to compactly re-represent them so that the procedure for determining the correct way to act is more sample efficient.

More formally, in this paper we consider TLRL for the case of a 'lifelong' learning agent that learns a (possibly never-ending) sequence of MDPs which are defined on the same state and action space but differ in terms of the transition and reward distributions. We assume the distribution generating the sequence is unknown and unlearnable (for instance, in the motivating surveillance problem described above, the infiltration pattern may depend on the internal variables of the infiltrators that are not known to us). In this setting, the goal of the agent is to, if possible, *reuse* the optimal policies in the previous MDPs in order to learn the new MDP more *efficiently*. In this continual setting, we assume that the agent operates in the new MDP for a fixed number of episodes, and hence we measure efficiency by the total reward accumulated while learning the new task during these fixed number of episodes. Reusing a policy means that we try the optimal policies of the previous MDPs in the new MDP and if one results in efficient behavior we should keep using it. However, as we described above, a problem in this setting is that, when the number of previous tasks become too large, transfer becomes ineffective as we spend too much time testing the old policies. In this instance, one possible solution to this problem is to find a subset of the N *previous* policies, which we call *source* policies, that are, in a well-defined and useful sense, representative of all the N previous policies (see Section 1.1 for alternative encodings). In other words, the source policies should form the analogue of an ϵ-net in a metric space (Kolmogorov and Fomin, 1970) over the space of previous MDPs with respect to an appropriate distance over MDPs. In this paper we present a clustering based approach to finding this smaller subset of source policies. Our main idea is to cluster the N previous MDPs into c clusters, where the number c and the clusters themselves are to be determined via discrete optimization, and then choose the representative element of each cluster to obtain the source MDPs. The optimal policies of the source MDPs then become the source policies. In our approach to choosing the clustering and the corresponding source policies, we attempt to ensure a-priori that the chosen source policies are a good representative of the previous tasks for the purposes of transferring to the unknown target task.

In particular, we define a transfer learning algorithm, EXP-3-Transfer, with performance bound $g(c)$ that depends on the number c of source policies. Hence this explicitly measures how good the size of the clustering c is. We are now left with the task of choosing the clustering and the corresponding source policies. To that end, we define a distance function d_V between two MDPs that measures how well the optimal policy of one MDP performs in the other. Hence, given that our goal is to reuse optimal policies of one MDP in another, we choose our clustering so that within each cluster the pairwise d_V distances between the elements of the cluster are low. Similarly, we choose the source policy for each cluster to be the optimal policy of the MDP in that cluster which has low d_V distance with respect to all the other elements. Hence, the cost of a clustering with c clusters is, roughly speaking, $g(c) + \epsilon$ where ϵ is a measure of the inter-element d_V distances in the clusters.

Given the cost function, we show that it is NP-hard to find the optimal clustering and so we introduce a Markov chain Monte Carlo based discrete optimization algorithm to find it. The algorithm is an extension of the Metropolis-Hastings algorithm, which we call Metropolis-Hastings with Auxiliary Variables (MHAV in short), and can also be thought of as an extension to simulated annealing (Kirkpatrick et al., 1983) with stochastic temperature changes. Simulated annealing is a well known algorithm for discrete optimization, but requires carefully setting of an infinite sequence of parameters known as the temperature schedule. Setting this schedule in practice to ensure convergence is considered very difficult, and an art form. In our version of the algorithm, we search over both the temperature and the optimal point simultaneously, thereby handling the problem of setting the schedule automatically.

To summarize, 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, at fixed intervals, 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. The clustering is chosen so as to optimize the regret of the transfer algorithm, and is found by using a convergent discrete optimization algorithm.

We conclude this brief introduction to our method by noting that our transfer algorithm EXP-3-Transfer is in fact an extension of the well known EXP-3 algorithm (Auer et al., 2002b) for non-stochastic multi-armed bandits, and our performance bound $g(c)$ is in fact a regret bound of the type well known in bandit algorithm literature. Our strategy is to cast the policy reuse transfer learning problem as a bandit problem, with 'pure reinforcement learning algorithm' as one arm, and the c source policies as the remaining arms. The regret bound for EXP-3 ensures that we minimize negative transfer by never performing much worse than pure reinforcement learning. We will now discuss related work.

1.1 Related Work

As evidenced by the survey paper (Taylor and Stone, 2009), a significant amount of work has been done in transfer learning in reinforcement learning. As mentioned previously, lifelong learning in reinforcement learning was first explicitly considered in (Mitchell and Thrun, 1993; Thrun and Mitchell, 1995; Thrun, 1996) in the context of learning in robots. In these works, the main aim was to learn the dynamics of robot motion in one circumstance using a function approximator (such as neural networks) and then use these learnt dynamics as an initial bias in a new situation using an explanation based learning framework.

In terms of recent work on TLRL, two different strands are relevant – the first is work on policy reuse and the second on task encoding. For policy reuse, the only papers that seem to have considered this are (Fernandez et al., 2006, 2010). The algorithms presented there, at the beginning of every episode, choose between different source policies by using a softmax criteria on accumulated reward and then use the chosen policy as an initial exploration policy before switching to Q-learning exclusively. In contrast, we extend the EXP-3 algorithm for multi-armed bandits to choose between source policies and Q-learning, and as a result inherit its theoretical guarantees. Additionally, they do not consider the problem of source task selection, whereas in our work this is a major focus. A related paper is (Azar et al., 2013) who consider the best way to choose between a set of stationary policies. While this work is quite interesting, it lacks a key ingredient of policy reuse which is to not do much worse than a base RL algorithm.

We now look at previous work that uses a smaller set of source tasks to represent the complete set of previous tasks. The problem of source task selection through clustering seems to have been considered only by Carroll and Seppi (2005). They introduce several measures for task similarity and then consider clustering tasks according to those measures. The distance functions introduced were heuristic, and the clustering algorithm used was a simple greedy approach. The evaluation of their method was on several toy problems. In contrast, we derive a cost function for clustering in a principled way to optimize the regret of our EXP-3-Transfer policy reuse algorithm. Additionally, instead of constructing the cluster in a greedy fashion, we search through clustering space using a convergent discrete optimization algorithm.

A recent paper that also chooses selectively from previous tasks is (Lazaric and Restilli, 2011). The setting for this paper is a collection of tasks defined on the same state-action space with the tasks and the state-action-state triples for the different tasks generated i.i.d. (similar to the multi-task transfer in classification setting considered in (Baxter, 2000)) rather than sequentially as is typical in RL. Under this setting the authors are able to bound the error when samples from one task are used to learn the new task. This is quite a different setting from us as it is 'batch' RL rather than the typical online and sequential RL and measures similarity in terms of the actual transition and reward functions rather than policies or values. Additionally, the analysis and algorithms are derived under the assumption of a fixed set of prior tasks rather than the continual lifelong learning setting we consider.

Source task selection is not the only possible way to represent previous tasks, and the overall goal of finding abstractions for exploiting commonality has received considerable attention in the transfer learning community. Most of the work done in deriving abstractions for the purposes of transfer has been for MDP homomorphisms (Ravindran and Barto, 2003; Ferns et al., 2004; Ravindran, 2013; Konidaris and Barto, 2007; Sorg and Singh, 2009; Castro and Precup, 2010). In these works, similarity between MDPs is defined in terms of bisimulation between states of different MDPs. Bisimulation is a concept borrowed from process algebra. In the context of transfer learning in MDPs, at its most general formulation, a bisimulation is an isomorphism f, g between the state and action spaces that is preserved under the transition distribution – that is for every state-action-state triple s, a, s', $T_1(s'|s, a) = T_2(f(s')|f(s), g(a))$ where T_i are the transition distribution of the two MDPs. Unfortunately, in this pure form, bisimulation is absolute (two MDPs are either bisimilar or not) and does not take into account the reward function. And so, in the papers mentioned above, this basic notion was extended in various ways to address both these issues. However, one of the main issues with bisimulation is computational cost, and this remains so in the extensions as well. Another issue with these approaches is that, as observed by Castro and Precup (2010), bisim-

ulation is a worst case metric (two states may have identical optimal actions but still be completely different according to the metric) and as a result requires heuristic modifications.

Technically, the main difference between our approach and bisimulation based methods is that the similarity between different MDPs are ultimately determined by distance between value functions. In our case, however, we are interested in distance in terms of *policy*. As a result, even though two tasks might be quite different in terms of the value function they might be identical in terms of the optimal policy, and our approach will capture this (as noted earlier, failing to do this was one of the issues with bisimulation based approaches).

Another interesting line of work that uses a different approach to abstracting MDPs is the proto-value function based approach of (Ferrante et al., 2008). Proto-value functions were introduced in (Mahadevan, 2005) as an efficient way to represent the value function for large state spaces as a linear combination of functions, which are called proto-value functions. The main innovation in this approach is that, in representing the value function as a real function over state space, the state-space is treated as a manifold where the distance between points/states is determined by the reachability graph of the MDP. This idea of a spectral-decomposition of the value function naturally lends itself to transfer learning, as, given a new task, we can imagine using the proto-value functions learned in a previous task to initialize the new value function in the new task. It has been noted that proto-value function based transfer has issues in terms of scalability and tractability. The main difference between this and our work is that, as with the homomorphism based approach, our similarity notion is based on policy similarity, while theirs is based on similarity between value functions. Identifying policy similarity is more desirable because tasks similar in terms of value function will be similar in terms of policy, but not necessarily the other way round.

1.2 Paper Organization

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 algorthm and framework for measuring distance in Sections 3 and 4 respectively. 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.

2. Preliminaries

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

A (stationary) policy π for $\mathcal{M}$ is a map $\pi : \mathcal{S} \to \mathcal{A}$. For a policy π, the Q function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is given by $Q^\pi(s, a) = \mathbb{E}[R(s, a)] + \gamma \sum_{s'} P(s'|s, a)Q(s, \pi(s'))$. The value function for π is defined as $V^\pi(s) = Q^\pi(s, \pi(s))$. An optimal policy π^* satisfies $V^{\pi^*}(s) \geq V^\pi(s)$ for all policy π and state s – it can be shown that every finite MDP has an optimal policy. V^{π^*} is written V^*, and the corresponding Q function is given by $Q^*(s, a) = \mathbb{E}[R(s, a)] + \gamma \sum_{s'} P(s'|s, a)Q^*(s', \pi^*(s'))$. When the agent acts in the MDP, at each step it takes an action a at a state s, and moves to the next state s' and the reward r. The goal of the agent is to learn π^* from these observations and then choose the action $\pi^*(s)$ at each state. If there are multiple optimal policies, we will designate the

first policy in a lexicographical order as the canonical policy. We will assume that $R_{\max}$ is a known upper bound on the reward of all the agents. Without loss of generality, in the sequel we assume that there is a single initial state s°. We define the *regret* of a policy π to be $V^*(s^\circ) - V^\pi(s^\circ)$. We call a policy π ϵ-optimal if its regret is at most ϵ.

In the transfer learning setting, we are given previous MDPs $\mathcal{M}_i$, $1 \leq i \leq N$ and we transfer from these tasks to solve the $N + 1^{st}$ MDP $\mathcal{M}_{N+1}$. We denote the optimal policy of the i^{th} previous MDP by π_i^*, and the value of a policy π in MDP i as V_i^π. Similarly, we denote the reward and transition functions of the i^{th} MDP by P_i and R_i respectively. We will assume that the rewards of all MDPs fall within the range $[R_{\min}, R_{\max}]$ and we define $\Delta R \triangleq R_{\max} - R_{\min}$.

3. Policy Reuse and Cost of Clusterings

In this section we first clarify the problem of policy-reuse for transfer learning and then we introduce our algorithm EXP-3-Transfer for this problem. EXP-3-Transfer extends the EXP-3 algorithm (Auer et al., 2002b) for multi-armed bandits to the setting of TLRL and inherits similar regret bounds. It essentially extends EXP-3 to the setting where the arms are MDP-policies run for a whole episode, and where we know that some of the arms have i.i.d. payoffs.

3.1 The Policy Reuse Problem

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). Policy reuse as a method for transfer was first introduced in (Fernandez et al., 2006, 2010). The algorithm introduced was called Policy Reuse (PR)[1], and the goal of the algorithm was to balance using the c source policies and pure reinforcement learning policy so that the learning algorithm converges faster than just running pure RL by itself. The basic idea in PR is as follows. At the beginning of each episode, PR chooses a a policy from among the previous policies *and* ϵ-greedy Q-learning using a softmax criterion on the observed returns of the policies in previous episodes. It then initiates a policy-reuse episode, where it probabilistically chooses between ϵ-greedy Q-learning and the chosen policy, with probability of choosing ϵ-greedy Q-learning going to 1 during the episode. In essence, the c source policies serve as a initial exploration policy, so that if they happened to take the agent through paths of the optimal policy, it will result in faster learning of the optimal policy.

There are several aspects of the above algorithm that are noteworthy. First, even if the c source policies contained the optimal policy, the algorithm would deterministically switch to Q-learning after the initial phase. Another aspect is that, while there is a intuitive connection between the Boltzmann distribution and the benefit of using a policy, the actual connection is not made rigorous. Both these issues arise from the fact that the goal of policy reuse was not defined concretely in (Fernandez et al., 2006). So, taking a cue from the definition of online learning algorithms (Vovk, 1990; Littlestone and Warmuth, 1994; Cesa-Bianchi and Lugosi, 2006), we define the policy reuse problem concretely as designing an algorithm that chooses policies at every episode such that it does not perform much worse than any of the c policies or Q-learning. Formally,

Definition 1 (The policy reuse problem) *Let transfer learning for the target task $\mathcal{M}$ be run for T episodes, and let $\bar{x}_i(t) \triangleq \sum_{n=1}^{K_t} \gamma^n r_n$ be the discounted return accumulated by running ρ_i (with*

1. We use the term 'policy reuse' to refer to the generic problem of transfer using policy reuse as defined in this section, and the upper-case 'Policy-Reuse' to denote the specific algorithm of (Fernandez et al., 2006).

ρ_{c+1} being the non-stationary Q-learning policy) at episode t, with r_n the reward at step n and K_t the length of the episode. Let the total discounted reward for policy ρ_i be $R_i(T) = \sum_{t=1}^{T} \bar{x}_i(t)$. Let $R_A(T)$ be the total discounted reward accumulated by a policy-reuse algorithm. Then we require that either $R_i(T) - R_A(T) = o(T)$, or, failing that, $\mathbb{E}[R_i(T) - R_A(T)] = o(T)$ where the expectation is with respect to randomization in the algorithm and the dynamics in $\mathcal{M}$.

The above definition subsumes that main goal of the Policy-Reuse algorithm, which is using the previous MDPs as an exploration policy, and also imposes the additional constraint that the choice should be made in a manner that eliminates negative transfer asymptotically. That is, the learner should choose the c source policies in a way so as to ensure that the exploration does not harm the base Q-learning performance. In the following section we present an algorithm to solve this problem.

3.2 Policy Reuse As Nonstochastic Bandits

In the following, we will frame the problem of policy-reuse as a non-stochastic multi-armed bandits problem (Auer et al., 2002b). In the non-stochastic multi-armed bandits problem, there are $c + 1$ *arms* where each arm i has a *payoff process* $x_i(t)$ associated with it. The learner runs for T steps and at each step t needs to *pull*/select one of the arms $f(t)$ and its payoff is $x_{f(t)}(t)$. Additionally, the learner only gets to view the payoff of the arm $f(t)$ it has chosen. The goal of the learner is to minimize its *regret* with respect to the best arm, that is minimize the quantity

$$\max_i \sum_{t=1}^{T} x_i(t) - \sum_{t=1}^{T} x_{f(t)}(t)$$

In general it is not possible to minimize this in any meaningful sense. An optimal algorithm in the general case, for minimizing the *expected regret*, was developed in (Auer et al., 2002b), called EXP-3. We now show how we can frame the policy reuse problem as a non-stochastic multi-armed bandit problem and then adapt the EXP-3 algorithm for our purpose.

Our adaptation of EXP-3, which we call EXP-3-Transfer, is listed as Algorithm 1. This algorithm is in essence EXP-3 with $c + 1$ arms, where arms 1 to c each correspond to a ρ_i and arm $c + 1$ corresponds to the Q-learning policy (Watkins, 1989)[2]. When arm i is pulled at iteration t, we run the corresponding policy/algorithm for one episode and observe the discounted payoff $\bar{x}_i(t) \triangleq \sum_{n=1}^{K_t} \gamma^n r_n$, where r_n is the reward obtained at time step n and K_t is the time step at which the episode t ended.

Now note, each arm ρ_i has i.i.d. payoff as it is stationary, while the payoff of Q-learning arm is non-stationary as the Q-learning policy is non-stationary. The main difference between EXP-3 and our EXP-3-Transfer algorithm is that we take advantage of the i.i.d. payoffs by eliminating ρ_i from consideration as soon as we determine with high probability that the arm is not the best arm. In particular, the algorithm is given as input $\delta \in (0, 0.5]$ and then (lines 10 – 16) it eliminates arm ρ_i as soon as it is certain that with probability at least $1 - \delta$, all the arms are better than that arm. We can now adapt the known bound for the EXP-3 algorithm from Corollary 3.2 (Auer et al., 2002b).

2. We could have used any other RL algorithm, but we use Q-learning to make the comparison with Policy Reuse in our experiments more meaningful.

Algorithm 1 EXP-3-Transfer($\mathcal{M}, \{\rho_1, \rho_2, \cdots, \rho_c\}, \beta, T, l, \Delta R$)

Input: MDP $\mathcal{M}$, arms 1 to c: the source policies $\rho_1, \cdots, \rho_c$ and EXP-3 parameters β and T; l the interval at which to eliminate arms; ΔR, the range of discounted returns

Initialize:

- Set Q-learning policy as the $c + 1^{th}$ arm
- Set $w_i(1) = 1$, let $x_i(t)$ be the payoff of the arm i at step t; let $rem = \emptyset$ be the set of arms removed.
- Set $n_i \leftarrow 0$, where $1 \leq i \leq c$, and n_i is the number of times ρ_i has been used; set $z_i \leftarrow 0$, where $1 \leq i \leq c$, and z_i is the total normalized discounted reward observed for ρ_i when it was run.

for $t = 1$ to T **do**
  If $i \notin rem$ **then** set $p_i(t) = (1-\beta)\frac{w_i(t)}{\sum_{i=1, i\notin rem}^{c+1} w_i(t)} + \frac{\beta}{c+1-|rem|}$; else set $p_i(t) = 0$.
  Select arm i_t for step t to be i with probability p_i, increment $n_{i_t} \leftarrow n_{i_t} + 1$.
  Run arm i_t for one-episode, and observe discounted payoff $\bar{x}_{i_t}(t)$; normalize $x_{i_t}(t) \leftarrow \bar{x}_{i_t}(t)/[\Delta R(1-\gamma)^{-1}]$.
  if i_t is not the Q-learning arm **then** set $z_{i_t} \leftarrow z_{i_t} + x_{i_t}$.
  For each $j \notin rem$, set

$$\hat{x}_j(t) \leftarrow \begin{cases} x_j/p_j(t) & \text{if } j = i_t \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

  For each $j \notin rem$, set $w_j(t+1) \leftarrow w_j(t) \exp[\beta \hat{x}_j(t)/(c+1)]$.
  if $t \mod l = 0$ **then**
    for $k = 1$ to c, $k \notin rem$ **do**
      if $\exists \rho_j$ arm $j \leq c, j \notin rem$, such that, for $\epsilon = z_j/n_j - z_k/n_k$, we have $\epsilon/2 > \sqrt{-\ln(\delta/2c)(2n_j)^{-1}}$ and $\epsilon/2 > \sqrt{-\ln(\delta/2c)(2n_k)^{-1}}$ **then**
        $rem \leftarrow rem \cup \{k\}$.
      end if
    end for
  end if
end for

Theorem 2 *EXP-3-Transfer, with probability $1-\delta$, with respect to randomization due to the target MDP $\mathcal{M}$, satisfies,*

$$\mathbb{E}[\sum_{t=1}^{T} x_j] - \mathbb{E}[G_{E3T}] \leq 2.63\sqrt{(c+1)\ln(c+1)T} \tag{2}$$

for an arm j when run with $\beta = \min\{1, \sqrt{(c+1)\ln(c+1)/[T(e-1)]}\}$. Here $\mathbb{E}[G_{E3T}]$ is the expected payoff of EXP-3-Transfer over T iterations given the c policies ρ_i and the Q-learning algorithm as arms, and the expectations are with respect to randomization in the EXP-3 arm-selection policy and the randomization due to P and R.

The proof is given in Appendix A.

Hence, this theorem tells us that if we use EXP-3-Transfer, then we will not do much worse than the best arm, be it one of the previous ρ_i policies or pure reinforcement learning policy like Q-learning. In particular, if one of the previous policies is close to optimal, we will do almost as well as playing with that policy right from the start, and if none of the policies are good enough, we will not do much worse than Q-learning run on its own. Hence, this essentially accomplishes the goal of policy reuse while minimizing negative transfer. The rest of the paper is devoted to showing how to compute the c policies ρ_i from a set of N previous MDPs. Before proceeding, we give a corollary to the above theorem which is obtained by noting that $\mathbb{E}[\sum_{t=1}^{T} \bar{x}_j] = TV^{\rho_j}$ when ρ_j is not the Q-learning arm, and taking into account the fact that above bound is obtained on quantities normalized by $\Delta R(1-\gamma)^{-1}$ at step 6 in EXP-3-Transfer.

Corollary 3 *EXP-3-Transfer, with probability $1-\delta$, with respect to randomization due to the target MDP $\mathcal{M}$, satisfies,*

$$TV^{\rho_j} - \mathbb{E}[G_{E3T}] \leq 2.63\Delta R(1-\gamma)^{-1}\sqrt{(c+1)\ln(c+1)T} \tag{3}$$

for a source policy ρ_j under the conditions of Theorem 2.

4. The Clustering Approach To Task Encoding

In this section we present our clustering based approach to encoding N previous MDPs into c source MDPs, where c will depend on the set of previous MDPs. First we present the basic idea of the clustering approach and derive the high level structure of a cost function for clusterings which helps us choose the best clustering for the purposes of transfer. After that, we derive two different versions of the cost function under worst-case and best-case assumptions on the target task. The worst-case cost function is pleasing theoretically, but empirically leads to poor results (please see Section 7), while for the best-case cost function the opposite turns out to be true.

4.1 Encoding By Clustering

Our goal is to, given N previous MDPs, choose a subset c of those MDPs whose optimal policies will be used as input to EXP-3-Transfer as the c source policies when solving the $N+1^{th}$ MDP $\mathcal{M}_{N+1}$. Here, the number c as well as the MDPs themselves are to be determined by our learning algorithm. To that end, we begin by noting that the regret bound of EXP-3-Transfer in Corollary 3 increases monotonically with the number of source policies. This indicates that we should have as few source policies as possible. However, at the same time, if we have fewer source policies, we risk leaving out a previous optimal policy π_k^* of a previous MDP $\mathcal{M}_k$ that may have obtained high discounted return in $\mathcal{M}_{N+1}$. Hence, in choosing the source policies, we need to achieve a tradeoff between the number of source policies and how well they represent the set of all previous policies.

Our approach to choosing the source policies will be as follows (illustrated in 1). We will divide the N previous MDPs into c different groups or clusters $\{A_1, A_2, \cdots, A_c\} \triangleq \mathbf{A}$ and then choose optimal policies of the representative element of the A_is as the c source policies. We will set the representative element $\mathcal{M}^i$ of A_i to be the MDP whose optimal policy ρ_i does as well as possible in the a-priori unknown $\mathcal{M}_{N+1}$ as the optimal policy of any other elements A_i. The problem now is to find the best possible clustering $\mathbf{A}$ and for that we need an objective or cost function for the clustering $\mathbf{A}$ when it is used to learn the unknown $\mathcal{M}_{N+1}$ using EXP-3-Transfer. We may define it

as follows:

$$\begin{aligned}&\text{regret of EXP-3-Transfer w.r.t. one of the } c \text{ source policies } \rho_i \\ &\quad + \text{ minimum regret of using } \pi_k^* \text{ in the unknown } \mathcal{M}_{N+1} \\ &\quad\quad + \text{ maximum over regret of a } \rho_i \text{ w.r.t. an MDP } \mathcal{M}_k \in A_i \end{aligned} \quad (4)$$

The first component in (4) is the bound in Corollary 3. In Sections 4.3 and 4.4 we derive two different ways to get the second and third component (recall from Section 2 that the regret of a policy π is $V_k^*(s^\circ) - V_k^\pi(s^\circ)$ where s° is the initial state). Both approaches build on the policy based distance between MDPs that we introduce in Section 4.2. But then they diverge in terms of the assumptions made on $\mathcal{M}_{N+1}$. The first cost function assumes that the $\mathcal{M}_{N+1}$ is arbitrary, while the second one assumes that $\mathcal{M}_{N+1}$ is one of the N previous MDPs. Correspondingly, we refer to these as, respectively, the worst case and best case assumption on $\mathcal{M}_{N+1}$.

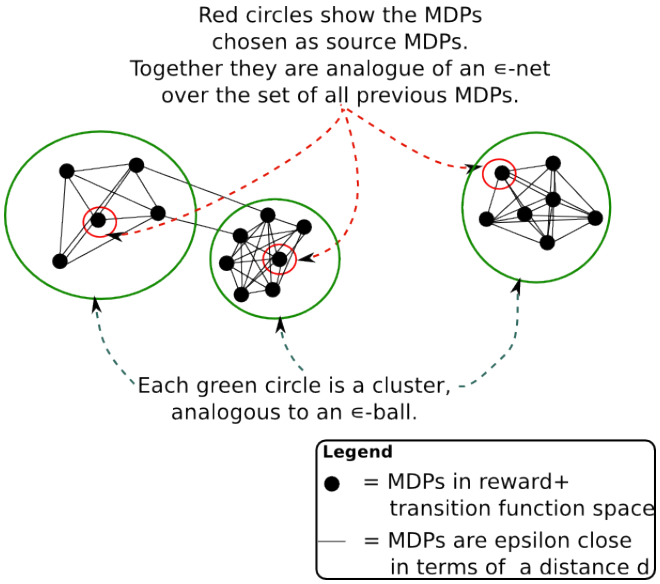


Figure 1: This figure sketches our basic approach to deriving the source policies. The black circles represent our set of previous N MDPs. The goal is to them into c clusters and then derive c source policies from the c source tasks. The figure illustrates the idea for $c = 3$. Each cluster is an analogue of an ϵ-ball (Kolmogorov and Fomin, 1970) according to an appropriate distance/similarity function d. The source policies are chosen so that together they form an analogue of an ϵ-net (Kolmogorov and Fomin, 1970) over the set of previous MDPs with respect to the same function d. The function d measures how well the policy of one MDP performs in the other – and hence the source policies being an ϵ-net implies that, given any MDP in the set of previous MDPs, there is at least one source policy which has performance that is 'ϵ-close' to the performance of the optimal policy of the previous MDP.

4.2 A Policy Based Distance Between MDPs

In this section we define a distance between MDPs based on their optimal policies. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two MDPs with the same state and action space but different transition and reward functions. Denote by V_i^π the value of policy π when executed in $\mathcal{M}_i$ at the initial state (this can be generalized to different initial states and/or distribution over initial states very easily – but we stick to the same initial state situation to keep the presentation simple). Letting the optimal policies for the two MDPs be π_1^*, π_2^*, we define the optimal policy similarity between two MDPs as follows.

$$d_V(\mathcal{M}_1, \mathcal{M}_2) \triangleq \max\{V_1^* - V_1^{\pi_2^*}, V_2^* - V_2^{\pi_1^*}\} \tag{5}$$

So this distance upper bounds how much we lose if we use the optimal policy of one MDP in the other – in particular we have the following lemma by construction.

Lemma 4 *If $d_V(\mathcal{M}_1, \mathcal{M}_2) \leq \epsilon$, then the optimal policy of $\mathcal{M}_1$ is at least ϵ-optimal in $\mathcal{M}_2$ and vice versa.*

This definition is motivated by the fact that the goal of policy reuse is to use the optimal policy of one MDP in another. Now given a clustering $\mathbf{A} = \{A_1, A_2, \cdots, A_c\}$ we may think of choosing the source policy for cluster A_i as the optimal policy ρ_i for the MDP defined as:

$$\mathcal{M}^i \triangleq \arg\min_{\mathcal{M} \in A_i} \max_{\mathcal{M}' \in A_i} d_V(\mathcal{M}, \mathcal{M}') \tag{6}$$

That is, $\mathcal{M}^i$ is the element of A_i that minimizes the maximum d_V distance to the other elements of the cluster, and hence, by Lemma 4, is in a worst case sense the best representative of the cluster A_i. Therefore $\min_i \max_{\mathcal{M} \in A_i} d_V(\mathcal{M}^i, \mathcal{M})$ gives us the second term of (4). For the third term we need to bound the d_V distance between any arbitrary $\mathcal{M}^i$ and $\mathcal{M}_{N+1}$ as a function of the d_V distance between a $\mathcal{M}^i$, $\mathcal{M} \in A_i$ and $\mathcal{M}$, $\mathcal{M}_{N+1}$. This is possible if either d_V is a metric or if $\mathcal{M}_{N+1}$ has optimal policy identical to another previous MDP. Since in the first case we need to make no assumptions about $\mathcal{M}_{N+1}$, this corresponds to a 'worst case' assumption, while the second case corresponds to a 'best case' assumption. However, the worst case is not directly achievable because d_V is not a metric (the proof is in appendix A):

Lemma 5 *There exists $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ all defined on the same state and action space such that the function d_V does not satisfy the triangle inequality: $d_V(\mathcal{M}_1, \mathcal{M}_3) \geq d_V(\mathcal{M}_1, \mathcal{M}_2) + d_V(\mathcal{M}_2, \mathcal{M}_3)$. Hence d_V is not a metric in general.*

So, in Section 4.3 we develop a Lipschitz metric that 'envelops' d_V, and develop a cost function under the worst-case assumption. In Section 4.4 we develop the cost function under the best case assumption.

4.3 Cost Function Under Worst Case Assumption

To derive the cost function for the worst case assumption, when $\mathcal{M}_{N+1}$ is arbitrary, we start by defining a class of functions that envelop d_V in a certain sense and is a Lipschitz metric. We call this class of functions value preserving Lipschitz metrics (abbreviated to VPL metric):

Definition 6 *Let M be a set of MDPs defined over the same state and action space. Then, we call function $d : \mathsf{M} \times \mathsf{M} \to \mathbb{R}$ a* value preserving Lipschitz (VPL) metric *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 on* 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 gives the qualifications 'value-preserving' and 'Lipschitz' to the metric. A VPL metric is useful because it allows us to operate on the space of MDPs almost as if it were 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. In our experiments, we use the following VPL metric (our results apply for any possible VPL metric).

$$d_M(\mathcal{M}, \mathcal{M}') \triangleq \max_{s,a} \max\{|R(s,a) - R'(s,a)|, \|P(\cdot|s,a) - P'(\cdot|s,a)\|_1\} \tag{7}$$

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

Lemma 7 *The function* d_M *defined in (7) is a VPL metric with* $K = 1$ *and* $k(\epsilon) = k_M(\epsilon) \triangleq \frac{\epsilon(1+\gamma R_{\max})}{(1-\gamma)^2}$.

To derive the cost function for a clustering $\mathbf{A}$, we define

$$\hat{\mathcal{M}}^i \triangleq \arg\min_{\mathcal{M} \in A_i} \max_{\mathcal{M}' \in A_i} d_M(\mathcal{M}, \mathcal{M}') \tag{8}$$

(this is an analogue of (6)) with corresponding optimal policy ρ_i. We ascribe parameters (c, ϵ) to each clustering $\mathbf{A}$, where c is the number of clusters and $\epsilon \triangleq \max_i \max_{\mathcal{M} \in A_i} d(\bar{\mathcal{M}}^i, \mathcal{M})$ (i.e. ϵ is the maximum diameter of a clustering A_i in $\mathbf{A}$).

Definition 8 *The worst-case cost of a clustering* $\mathbf{A}$ *with parameters* (c, ϵ) *is defined to be:*

$$cost_1(\mathbf{A}) \triangleq g(c) + k_M(\epsilon) \tag{9}$$

where $g(c) \triangleq \Delta R(1-\gamma)^{-1} 2.63 \sqrt{(c+1)\ln(c+1)/T}$.

This cost function is justified by the following result which derives immediately from Corollary 3.

Theorem 9 *For any clustering* $\mathbf{A}$ *with parameters* (c, ϵ)*, there is a constant* C*, independent of* $\mathbf{A}$*, but dependent on the previous* N *MDPs* $\mathcal{M}_i$*, such that* $V^*_{N+1} - \mathbb{E}[G_{E3T}]/T \leq g(c) + k_M(\epsilon) + C$*, when EXP-3-Transfer is run with source policies given by* $\mathbf{A}$ *with* β *set as in Theorem 2.*

The proof is in Appendix A. Hence, the above theorem shows us that our regret with respect to the best performing MDP is bounded by $g(c) + k_M(\epsilon) + C$. Hence, we can choose this as the cost of encoding the previous MDPs into c source policies using $\mathbf{A}$. Since C is independent of the clustering $\mathbf{A}$, we can ignore this, and use $cost_1(\mathbf{A})$ as our cost function.

4.4 Cost Function Under Best-Case Assumption

We now derive the cost function under the best case assumption which is that π^*_{N+1} is identical to a π^*_k, the optimal policy of a previous $\mathcal{M}_k$. In this case, we use d_V directly, instead of using d as a proxy as in the previous section. That is, given a clustering $\mathbf{A} = \{A_1, A_2, \cdots, A_c\}$, we set the source task corresponding to A_i to be $\mathcal{M}^i$ as defined in (6). We associate with $\mathbf{A}$ parameters $(c, \bar{\epsilon})$ where c is the number of clusters and$\bar{\epsilon}$ is the average diameter of the cluster, defined as follows. Let the diameter of a cluster A_i be:

$$\epsilon_i \triangleq \max_i \max_{\mathcal{M} \in A_i} d_V(\mathcal{M}^i, \mathcal{M}) \tag{10}$$

Then, we define:

$$\bar{\epsilon} \triangleq \frac{1}{n} \sum_{i=1}^{c} |A_i| \epsilon_i \tag{11}$$

So $\bar{\epsilon}$ is the 'average maximum d_V distance' from a previous MDP to the centroid of the cluster it belongs to.

Definition 10 *The best case cost of a clustering $\mathbf{A}$ with parameters $(c, \bar{\epsilon})$ is defined to be:*

$$cost_2(\mathbf{A}) \triangleq g(c) + \bar{\epsilon} \tag{12}$$

where $g(c)$ was defined in Definition 8.

This cost function is justified by the following result which derives immediately from Corollary 3.

Theorem 11 *For any clustering $\mathbf{A}$ with parameters $(c, \bar{\epsilon})$, under the assumption that $\mathcal{M}^{N+1}$ is a previous task drawn uniformly at random, we have $\mathbb{E}[V^*_{N+1}] - \mathbb{E}[G_{E3T}]/T \leq g(c) + \bar{\epsilon}$, when EXP-3-Transfer is run with source policies given by $\mathbf{A}$ with β set as in Theorem 2. Here the expectation is taken over the randomization of the task drawing process, randomization in EXP-3-Transfer and P_{N+1} and R_{N+1}.*

The proof is in Appendix A. Hence, the above theorem shows us that our regret with respect to the best performing MDP is bounded by $g(c) + \bar{\epsilon}$. As a result, we can choose this as the cost of encoding the previous MDPs into c source policies using $\mathbf{A}$.

5. Finding the Optimal Clustering

In this section we derive an algorithm to solve the discrete optimization problem of computing $\arg\min_{A \in \mathcal{C}} cost_j(A)$, $j \in \{1, 2\}$, where $\mathcal{C}$ is the set of all possible clusterings of $\mathcal{M}_i$s. To motivate the need for the algorithm, in Appendix B we show that it is NP-complete to optimize both $cost_1$, and an upper bound of $cost_2$ (regrettably we do not have the same result for $cost_2$ itself). After that we present our novel discrete optimization algorithm. Our basic strategy is to sample repeatedly from a distribution over $\mathcal{C}$, where the distribution concentrates around the optimum, and also around clusterings with low cost. This way, we are guaranteed with high probability to hit the optimum clustering, or at least a good clustering. Exact sampling from the distribution is difficult, and so our algorithm samples approximately from this distribution using a Markov chain Monte Carlo 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 (Kirkpatrick et al., 1983) 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 3 and the distributions used in the algorithm are described below.

5.1 Sampling Using Metropolis-Hastings Chains

In this subsection, we use upper-case Roman letters for random variables and lower-case letters to refer to their realized values. In the following, we use the theory of Markov chains as found in (for instance) Levin et al. (2009). A *Markov chain* over a (finite) state-space $\mathcal{X}$ is stochastic process X_n taking values in $\mathcal{X}$ such that $Pr(X_n = x|x_0, x_1, x_2, \cdots, x_{n-1}) = P_n(x|x_{n-1})$. The distribution $P_n(\cdot|\cdot)$ is called the *transition kernel* for the chain, and can be represented by a $|\mathcal{X}| \times |\mathcal{X}|$ matrix, also denoted by P_n, such that the entry (x, y) is $P_n(y|x)$ (here we have identified each element of $\mathcal{X}$ with an integer in $\{1, 2, \cdots, |\mathcal{X}|\}$ in some order). A Markov chain is said to be time-homogeneous if $P_n(x'|x) = P(x'|x)$, i.e. P_n is independent of time n. We will only consider time-homogeneous chains. A distribution Π over $\mathcal{X}$ is said to be *stationary* for the chain with kernel P if it satisfies:

$$\Pi(x') = \sum_x \Pi(x)P(x'|x) \tag{13}$$

Let $P_x(X_n = x')$ be the probability that $X_n = x'$ given that $X_0 = x$, that is

$$P_x(X_n = x') = \sum_{x_1, x_2, \cdots x_{n-1}} \prod_{i=0}^{n-1} P(x_{i+1}|x_i), \qquad \text{where } x_0 = x, x_n = x'$$

Then the chain X_n (equivalently, the kernel $P(\cdot|\cdot)$) is said to be *irreducible* if for each $x, x' \in \mathcal{X}$, $\exists n$ with $P_x(X_n = x') > 0$. It is called *a-periodic* if the set $\{n : P_x(X_n = x) > 0\}$ has greatest common divisor of 1 – that is there is no period to the set of time steps at which the chain returns to some state x, starting from x itself.

Theorem 12 *The following are true for any a-periodic and irreducible Markov chain with kernel P:*

1. *P has a stationary distribution Π and for any $y \in \mathcal{X}$,*

$$\lim_{n\to\infty} \|P_y(X_n = \cdot) - \Pi(\cdot)\|_{\text{TV}} = 0 \tag{14}$$

   where $\|P - P'\|_{\text{TV}} = \sup_{A\subset\mathcal{X}} |P(A) - P'(A)|$ is the total variation distance between any two distributions P, P' over $\mathcal{X}$.

2. *If Π is stationary for P and $\|P_y(X_n = \cdot) - \Pi(\cdot)\|_{\text{TV}} \leq k$ then $\|P_y(X_{n'} = \cdot) - \Pi(\cdot)\|_{\text{TV}} \leq k$ for all $n' > n$.*

Proof For the first part and second part, see (for instance), respectively, Theorem 4.9 and Lemma 4.12 in (Levin et al., 2009)). ■

This result is important because it can be used to approximately sample from a distribution Π that is

hard to sample from directly. The idea is to construct a Markov chain $\mathbf{X}$ with stationary distribution Π. Theorem 12 implies that if we simulate $\mathbf{X}$ long enough, then eventually we will start sampling from Π. To that end, the Metropolis-Hastings chain (MH chain in short) gives a standard way to define such a chain given Π as input (see Robert and Casella (2005) for an in-depth introduction).

A MH chain is defined via an irreducible kernel $\phi(x, x')$ over $\mathcal{X}$ and an acceptance probability $\mathsf{Acc}(x, x')$, which is a distribution over the second argument, indexed by the first. ϕ is problem dependent while Acc is defined as follows:

$$\mathsf{Acc}(x, x') \triangleq \min\left\{1, \frac{\phi(x', x)\Pi(x')}{\phi(x, x')\Pi(x)}\right\} \tag{15}$$

Given this, the MH chain has transition

$$P_{MH}(x, x') = \begin{cases} \phi(x, x')\mathsf{Acc}(x, x'), & \text{if } x \neq x' \\ 1 - \sum_{x' \neq x} \phi(x, x')\mathsf{Acc}(x, x'), & \text{otherwise}, \end{cases} \tag{16}$$

It can be easily checked that P_{MH} satisfies the *detailed balance* equation $\Pi(x)P_{MH}(x, x') = \Pi(x')P_{MH}(x', x)$ which in turn is equivalent to (13) (which can be seen by summing both sides over x'). So we have in our hands a chain which if simulated long enough will sample from the *target distribution* Π.

5.2 Discrete Optimization Using Metropolis Hastings With Auxiliary Variables (MHAV)

In this section, we show how we may use Metropolis-Hastings with an auxiliary variable as a discrete optimization algorithm. In particular, the algorithm may be thought of as simulated annealing without a temperature schedule. We discuss this connection in more detail below.

Assume that our goal is to minimize a cost function f defined over some finite set Y. In particular assume that there is a subset $\hat{Y} \subset Y$ for which $y \in \hat{Y}$ has acceptable cost $f(y)$. Let $\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_n\}$, $\lambda_i < \lambda_{i+1}$ such that $\exists \hat{\Lambda} \subset \Lambda$ which satisfies

$$\sum_{\lambda \in \hat{\Lambda}, y \in \hat{Y}} \lambda^{-f(y)} \geq \theta, \text{ where } \theta \text{ is close to } 1 \tag{17}$$

In our MH algorithm, we have $\mathcal{X} = Y \times \Lambda$ and our target distribution is

$$\bar{\Pi}(\lambda, y) \triangleq \lambda^{-f(y)} Z^{-1} \tag{18}$$

where $Z \triangleq \sum_{y,\lambda} \lambda^{-f(y)}$ is the normalization term. We briefly note here that if we plug $\bar{\Pi}$ into (15), then this normalization term cancels out, and in our algorithms there will never be any need to compute Z.

Continuing from above, given the above condition on Λ, if we draw repeatedly from Π, then after t draws, with probability at least

$$1 - (1 - \theta)^t \tag{19}$$

we will draw an element (λ, y) where $y \in \hat{Y}$. For θ sufficiently close to 1, t will be sufficiently small and we will discover an element from the acceptable set rapidly and hence this solves the discrete optimization problem.

Of course, in general it is not possible to sample from $\bar{\Pi}$ directly, so we will instead sample using $\bar{P}_{\mathrm{MH}}$ that has $\bar{\Pi}$ as the stationary distribution. To that end, let ϕ_Y be any irreducible kernel over Y (this will depend on the nature of Y and will be an input to the optimization algorithm – we discuss this below). Define the following transition kernel over Λ, parametrized by $\alpha' \in (0,1)$:

$$\phi_\Lambda(\lambda, \lambda') \triangleq \begin{cases} \alpha' & \text{if } i < N \text{ and } \lambda = \lambda_i, \lambda' = \lambda_{i+1} \\ 1-\alpha' & \text{if } i > 1 \text{ and } \lambda = \lambda_i, \lambda' = \lambda_{i-1} \\ 1 & \text{if } \lambda = \lambda_0, \lambda' = \lambda_1 \text{ or } \lambda = \lambda_N, \lambda' = \lambda_{N-1} \\ 0 & \text{otherwise} \end{cases}$$

Lemma 13 *ϕ_Λ is irreducible.*

The proof is given in Appendix A.

Given the above, the proposal distribution $\bar{\phi}[(\lambda, y), (\lambda', y')]$ for $\bar{P}_{\mathrm{MH}}$ is defined using the parameters $\alpha, \beta \in (0,1)$, $\alpha + \beta < 1$, as follows.

$$\bar{\phi}[(\lambda, y), (\lambda', y')] \triangleq \begin{cases} \alpha\phi_\Lambda(\lambda, \lambda') & \text{if } \lambda \neq \lambda', y = y' \\ \beta\phi_Y(y, y') & \text{if } \lambda = \lambda', y \neq y' \\ (1-\alpha-\beta) + \beta\phi_Y(y, y) & \text{otherwise} \end{cases} \tag{20}$$

The transition kernel $\bar{P}_{\mathrm{MH}}$ is now defined as in (16) using $\bar{\phi}$ as the proposal distribution and (18) as the target distribution:

$$\bar{P}_{\mathrm{MH}}(x, x') = \begin{cases} \bar{\phi}(x, x')\bar{\mathsf{Acc}}(x, x'), & \text{if } x \neq x' \\ 1 - \sum_{x' \neq x} \bar{\phi}(x, x')\bar{\mathsf{Acc}}(x, x'), & \text{otherwise} , \end{cases} \tag{21}$$

Given the above, the overall discrete optimization algorithm MHAV (Metropolis-Hastings with Auxiliary Variable) is listed in Algorithm 2.

Algorithm 2 MHAV$(\Lambda, Y, \bar{\Pi}, \bar{\phi}, T_M)$

1: **Input:** The set of auxiliary variables Λ, the search space Y, the target distribution $\bar{\Pi}$, and proposal distribution $\bar{\phi}$, T_M number of iterations to run algorithm.
2: **Output:** An near optimal element y.
3: **Initialize:** Initial, $\lambda(0) = \lambda_0$, $y(0) =$ arbitrary element of Y.
4: **for** $t = 1$ to T_M **do**
5: Sample $(\lambda', y') \sim \bar{\phi}[(\lambda(t), y(t)), \cdot]$
6: With probability $\bar{\mathsf{Acc}}[(\lambda(t), y(t)), (\lambda', y')]$, set $\lambda(t+1) = \lambda'$, $y(t+1) = y'$, and with probability $1 - \bar{\mathsf{Acc}}[(\lambda(t), y(t)), (\lambda', y')]$, set $\lambda(t+1) = \lambda(t)$, $y(t+1) = y(t)$.
7: **end for**
8: **return** $\arg\min_t f(y(t))$.

5.3 Analysis of the MHAV Algorithm

We begin analysis of our algorithm by showing that the kernel $\bar{P}_{\mathrm{MH}}$ for the $\bar{\phi}$ defined above is indeed irreducible and a-periodic.

Lemma 14 *If $\bar{\Pi}$ and ϕ_Y satisfy $\min_{x,x'} \frac{\bar{\Pi}(x')\phi_Y(x',x)}{\bar{\Pi}(x)\phi_Y(x,x')} > b > 0$, the kernel $\bar{P}_{\mathrm{MH}}$ defined using ϕ^* is irreducible and a-periodic.*

The following theorem establishes the probability with which we draw an element from the acceptable set $\hat{Y}$ when using $\bar{P}_{\mathrm{MH}}$ to sample.

Theorem 15 *$\bar{P}_{\mathrm{MH}}$ has $\bar{\Pi}$ as its stationary distribution, and hence for any initial state x_0 of the chain $\bar{P}_{\mathrm{MH}}$,*

$$\lim_{n\to\infty} \|\bar{P}_{\mathrm{MH}}(X_n = \cdot|X_0 = x_0) - \bar{\Pi}(\cdot)\|_{\mathrm{TV}} = 0 \tag{22}$$

In particular if at step t $\|\bar{P}_{\mathrm{MH}}(X_t = \cdot|X_0 = x_0) - \Pi(\cdot)\|_{\mathrm{TV}} \leq k$, then $\bar{P}_{\mathrm{MH}}(X_{t'} \in \hat{\lambda} \times \hat{Y}|X_0 = x_0) \geq \theta - k$ for all $t' > t$.

The proof is given in Appendix A.

We can also derive a convergence rate which establishes that for every k, there is a t_k for which $\|\bar{P}_{\mathrm{MH}}(X_{t_k} = \cdot|X_0 = x_0) - \Pi(\cdot)\|_{\mathrm{TV}} \leq k$. In the following, we introduce the notation $\bar{P}^n_{\mathrm{MH}}(x,x') \triangleq \bar{P}_{\mathrm{MH}}(X_n = x'|X_0 = x)$. Now define the *diameter* of $\mathcal{X}$ given the Markov chain $\bar{P}_{\mathrm{MH}}$ to be

$$D \triangleq \min\{l|\forall x, x', \bar{P}^l_{\mathrm{MH}}(x,x') > 0\} \tag{23}$$

Now define the ratio

$$\delta \triangleq \min_{x,x'} \frac{\bar{P}^D_{\mathrm{MH}}(x,x')}{\bar{\Pi}(x')} \tag{24}$$

Then we have the following:

Theorem 16 *The convergence rate of $\bar{P}_{\mathrm{MH}}$ to $\bar{\Pi}$ satisfies:*

$$\|\bar{P}_{\mathrm{MH}}(X_n = \cdot|X_0 = x_0) - \bar{\Pi}(\cdot)\|_{\mathrm{TV}} \leq (1-\delta)^{n/D}$$

for any initial state x_0.

This derives directly from the proof of Theorem 4.9 (Levin et al., 2009) and is given in Appendix A.

5.4 Setting the Optimization Parameters

We now discuss how to set the parameters α' in ϕ_Λ and α, β in $\bar{P}_{\mathrm{MH}}$ so as to optimize the convergence rate derived above. In setting these parameters, we are given the proposal distribution ϕ_Y, which was required to be an irreducible kernel on Y, and the target distribution $\bar{\Pi}$ over $\Lambda \times Y$. We start with the following result which simplifies deriving our result

Lemma 17 *D is independent of α', α, β.*

The proof is given in Appendix A.

Corollary 18 *Given f, ϕ_Y, the set of paths of positive probability under $\bar{P}_{\mathrm{MH}}$ is invariant with respect to α', α, β.*

Proof Follows directly from the proof of Lemma 17. ■

So, we need to set α', α, β to maximize δ. However, δ also depends on f and ϕ_Y, both of which are unknown and so it is difficult to specify optimal values for these a-priori. However, we can give heuristic arguments for setting these parameters in terms of increasing the 'flow' of the search process through the search space Y. First, α' is used to choose whether we should increase or decrease the λ value. We set α' to $1/2$ to ensure a neutral value and that we do not favor either direction and ensure maximum motion through the search space.

Now note that at each step the chain $\bar{P}_{\mathrm{MH}}$ moves either in Λ space or Y space. α and β determine, respectively, how often we move in the Λ and how often in Y. To make the search more effective (based on analysis of simulated annealing type algorithms), it seems we need to make sure that initially we need to explore the Y quite a bit and only settle down after we have explored sufficiently, by increasing the λ_i value. Hence, our recommendation is to set the α to be significantly smaller than β, ideally the ratio α/β should reflect how difficult we expect it to be to get close to the best y^* (with smaller ratio for greater difficulty). Even though our parameter settings are heuristic, we again stress that this only affects the convergence speed, but not the ultimate convergence. This is in contrast to simulated annealing where convergence itself is guaranteed only if we set the parameters carefully.

5.5 Searching for the Optimal Cluster

Searching for the optimal cluster can now be be solved using the Markov chain $\bar{P}_{\mathrm{MH}}$. In this case, $Y = \mathcal{C}$, and the objective function is $f(\mathbf{A}) = cost_i(\mathbf{A})$, where $i \in \{1, 2\}$. To complete the specification of MHAV for our problem, we now define the distribution $\phi_Y^M(\mathbf{A}, \mathbf{A'})$ as follows. Given a clustering $\mathbf{A} = A_1, \cdots, A_n$, 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 $1, 2, \cdots, |A_i|$:

$$PE(k; \theta_1) \triangleq \frac{1 - \exp(-\theta_1)}{\exp(-\theta_1)} \sum_{m=1}^{\infty} \exp[-\theta_1((m-1)|A_i| + k)]$$

Then we choose another $A_j \in (\mathbf{A} - \{A_i\}) \cup \{B\}$ uniformly at random, where B is an empty set representing a new cluster. We then move the chosen points of A_i 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 $\mathbf{A}$ to $\mathbf{A'}$. The following Lemma shows that this ϕ_Y is irreducible and hence satisfies the condition in Lemma 14 and hence ensures the convergence results in Section 5.3.

Lemma 19 *ϕ_Y defined above is irreducible.*

The proof is in Appendix A.

6. The Continual Transfer Algorithm

In this brief section we combine all the algorithms presented so far into the full continual transfer algorithm, which is listed as Algorithm 4. 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 $sourcePol(\mathbf{A}, d, cost)$ generates the c source policies $\rho_1, \rho_2, \cdots, \rho_c$ from clustering

Algorithm 3 Search-Clusterings($\mathsf{M}, d, \Lambda, T_M$)

1: **Input:** A set of MDPs $\mathsf{M} = \{\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_N\}$, the set of auxiliary variables Λ, a cost function $cost$, input condition $term$.
2: **Initialize:** ϕ_Y^M defined with respect to M; $\bar{\phi}$ defined using ϕ_Y^M using (20); define $\bar{\Pi}(\lambda, \mathcal{M}) = \lambda^{-cost(\mathcal{M})}$.
3: **return** MHAV($\Lambda, \mathsf{M}, \bar{\Pi}, \bar{\phi}, T_M$)

$\mathbf{A}$ such that ρ_j is the optimal policy for $\mathcal{M}^j$ where $\mathcal{M}^j$ is chosen from A_j according to (7) or (6) depending, respectively on whether $cost$ is either $cost_1$ or $cost_2$. If the current phase h satisfies $h \mod J = 0$, then it runs the Search-Clustering algorithm to find a new set of source tasks from the h tasks solved so far.

Algorithm 4 Continual-Transfer($d, \Lambda, cost, T_M, l, \Delta R, \beta, T$)

1: **Input:** A metric d, which is either a d_M or d_V; cost function $cost$, which is either $cost_1$ or $cost_2$; Search-Clustering parameters $T_M, l, \Delta R$, EXP-3-Transfer parameters β, T.
2: **Initialize:** Initial clustering $\mathbf{A} = \emptyset$, collection of previous MDPs M.
3: **for** $h = 1$ to ∞ **do**
4: Get unknown MDP $\mathcal{M}_h$ from the environment and run EXP-3-Transfer($\mathcal{M}_h, sourcePol(\mathbf{A}, d, cost), \beta, T_M, l$).
5: Set $\mathsf{M} \leftarrow \mathsf{M} \cup \{\mathcal{M}_h\}$
6: **if** $h \mod J = 0$ **then** $\mathbf{A}$ = Search-Clusterings($\mathsf{M}, \Lambda, cost, T_M$).
7: **end for**

7. Experiments

We performed two sets of experiments to illustrate various aspects and efficacy of our algorithm [3]. Our baseline algorithm for comparison was Policy Reuse of Fernandez et al. (2006) which, as we mentioned in Section 1.1, is the only prior work on policy reuse algorithms. Given this, we report results of various combinations of learning algorithms and clustering approaches as given in Table 1. For each graph we present in the subsequent sections, the results are averaged over 10 different target tasks with 10 trials per task. The various parameters used for the clustering and transfer algorithms are given in Table 2.

We present results on two different domains. In Section 7.1 we present results from a simple Windy Corridor domain to give an idea of the types of clusters found by our approach. The results show that the clusters found are natural. In Section 7.2, we present results on the more complex surveillance domain (described briefly in the Section 1) which is a variant of the kinds of problems that are considered, for instance, in (An et al., 2012). In this experiment we show the performance for the algorithm combinations in Table 1 for various numbers of previous tasks.

In the following sections, we only use $cost_2$. While $cost_1$ is better motivated as it is derived from much weaker conditions than $cost_2$, it is nonetheless too weak in the sense that the distance function d_M used in it highlights differences that may be irrelevant. For instance, if there are two

3. The code used in the experiments, as well as the data, can be found here `http://wcms.inf.ed.ac.uk/ipab/autonomy/code/MDP_Clustering_code.zip`

Algorithm	FULL	SANS	HANDPICKED	GREEDY
EXP-3-TRANSFER	✓	✓	✓	✓
POLICY REUSE	✓	✓	✓	✓
Q-LEARNING	N/A	N/A	N/A	N/A

Table 1: EXPERIMENT SETUP MATRIX. This table shows combinations of algorithms and clustering methods used in our experiments. 'Full' means Search-Clustering, 'sans' means without any kind of clustering, 'handpicked' means a set of source policies that we believe to be optimal. 'Greedy' means a clustering obtained as follows. We choose a threshold for the d_M or d_V distance and then greedily construct clusters. That is, we choose a MDP arbitrarily to seed a cluster and then add all the MDPs with distance $<$ threshold to that cluster. In our graphs, we present the results for the best/lowest cost clustering found by using various threshold values.

EXP-3-TRANSFER		SEARCH-CLUSTERINGS					RL
T	δ	α	β	α'	θ	T_M	γ
$1000, 5000$ and 10^4	0.1	0.1	0.8	0.5	1	10^5 (for each of 20 random restarts)	0.9

Table 2: VALUES OF ALGORITHM PARAMETERS. This table gives the values for the different parameters used in our various algorithms. RL refers to general parameters for reinforcement learning algorithms. The T parameter was chosen to illustrate effect of this parameter on algorithm performance. The δ parameter was chosen to allow for high degree of confidence in the performance of EXP-3-Transfer. The α, α' and β parameters where chosen according to Section 5.4, and θ was chosen heuristically. T_M was chosen because we found that restarting gave better results. γ was chosen as appropriate for our problems.

MDPs with identical optimal policies and transition distributions, but a state where the reward difference is 1000, the d_M distance between the two MDPs is $k_M(1000)$ (defined in Section 4.3) while the d_V distance between the two MDPs is 0. Indeed the d_M distance function is similar to bisimulation based distance functions studied in (Castro and Precup, 2010) which, while theoretically well motivated, were also found to be inadequate for applications.

7.1 Windy-Corridor Domain

The windy corridor domain is illustrated in Figure 2. The domain consists of a row of 10 corridors with a 'wind' blowing from the South to the North along the cells right in front of the entrance to the corridor. The agent has one action for each possible cardinal direction which moves it in that direction deterministically. In a windy cell, the motion of the agent becomes probabilistic with the probability of moving North being p, and moving in the desired direction being $1 - p$. p depends on the strength of wind, increasing from 0 to 0.9. The MDPs in the domain are distinguished by the location of the goal state and the strength of the wind. There are 10 possible wind speeds, and given 10 possible goal locations, this gives us a total of 100 possible MDPs.

For this domain, we learned the value function for each of the MDPs, and from that computed the distance between every pair of MDPs. This was then used to cluster the MDPs using our clustering

algorithm. Figure 3 presents the final cluster we found for this domain using $cost_2$. This figure shows that the best clustering we found put domains with the same goal state in the same cluster. This makes sense, because despite the wind speed, the policies requried for MDPs with identical goal states will be identical, while different for states with different goal states. So this shows that our algorithm recovers the clusters we expect to find, and provides a basic sanity check for our algorithm.

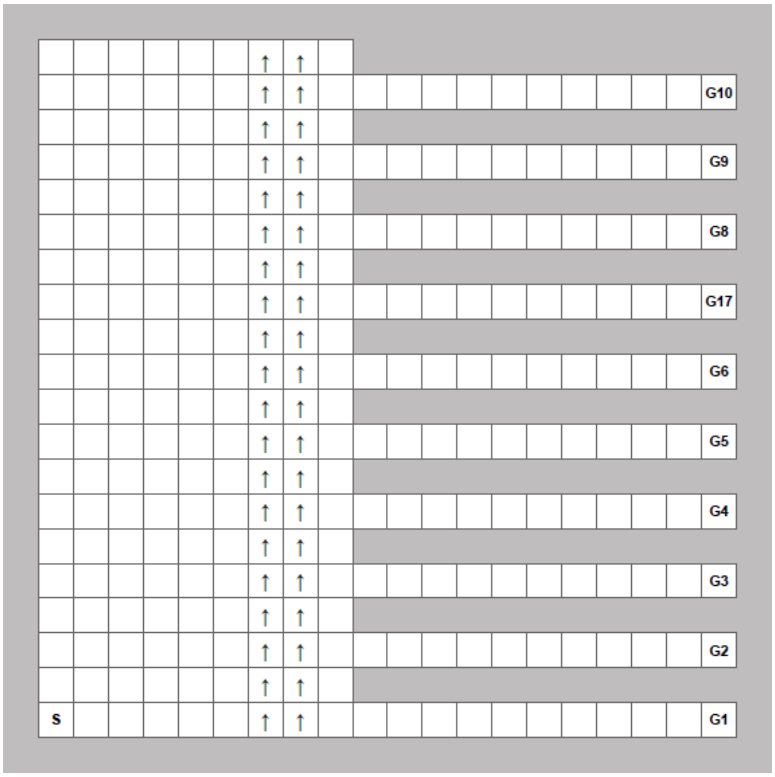


Figure 2: THE WINDY CORRIDOR DOMAIN. This shows the 10 corridors, the location of the goal states and the direction of the wind (the small arrows). The start state is marked by **s**.

7.2 Surveillence Domain Setup

The surveillance domain is illustrated in Figure 4. In this domain, the goal of the agent is to catch infiltrators who wish to break into a target region. There are L different vulnerable locations (abbreviated *v-locations*) in the domain, and the infiltrators only choose a subset of those v-locations to infiltrate through – we call these *target v-locations*. The type of the infiltrators is defined by the sequence in which they visit the target v-locations and the goal of the agent is to find out where the target locations are and surveil them in the right sequence. The actions available to the agent are the motion actions in the cardinal directions and a surveil action. Every action results in a reward of -1, an unsuccessful inspect action (i.e. inspecting the target v-location in the wrong order) results in a reward of -10, while a successful inspect action (i.e. inspecting the correct v-location at the right time) results in a reward of 200. The task relatedness takes the following form. If instead of

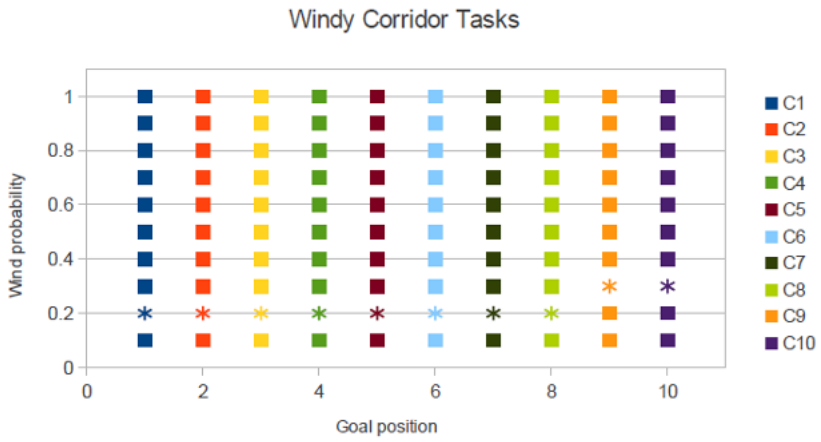


Figure 3: CLUSTERING FOR THE WINDY CORRIDOR DOMAIN. The clustering was obtained by running Search-Clustering and $cost_2$. Each point in the 2D grid is a MDP with the goal location and wind-speed given by the x and y axis respectively. The colors indicate the cluster to which the MDP was found to belong to, and shows that all the MDPs belonging with the same goal state belong to the same cluster. Additionally, the cluster centers $\mathcal{M}^i$ are marked with $*$s.

surveilling the right v-location, the agent visits a v-location in the same block (see Figure 4) as the true location, then the agent receives a reward of 190. These notions are illustrated in Figure 5.

We present the following results for a combination of previous MDPs and number of target locations. The results show that the more complex the transfer task is the better EXP-3-Transfer+clustering is compared to Policy Reuse, where complexity is measured by the number of previous MDPs and the difficulty of the target task.

- We compare the performance of EXP-3-Transfer, Policy-Reuse and Q-learning as the complexity of the transfer problem increases. Here, the complexity of the transfer problem is both the number of previous tasks, and the complexity of the MDP itself (i.e. the number of target v-locations). We call these results clustering gains.
- We compare the effect of having different types of clusterings (in Table 1) for EXP-3-Transfer with $T = 10,000$. We call these results clustering comparisons.
- We compare the effect of having different $T \in \{1000, 5000, 10000\}$ for EXP-3-Transfer with clustering for various number of previous tasks. We call these results time comparisons.

We now present these results below.

7.2.1 CLUSTERING GAINS

We begin presentation of our experimental results with two summary graphs which show the benefit of clustering vs. not clustering for our algorithm EXP-3-Transfer and Policy-Reuse. These results are presented in Figures 6 and 7 respectively. The full results that these graphs summarize are

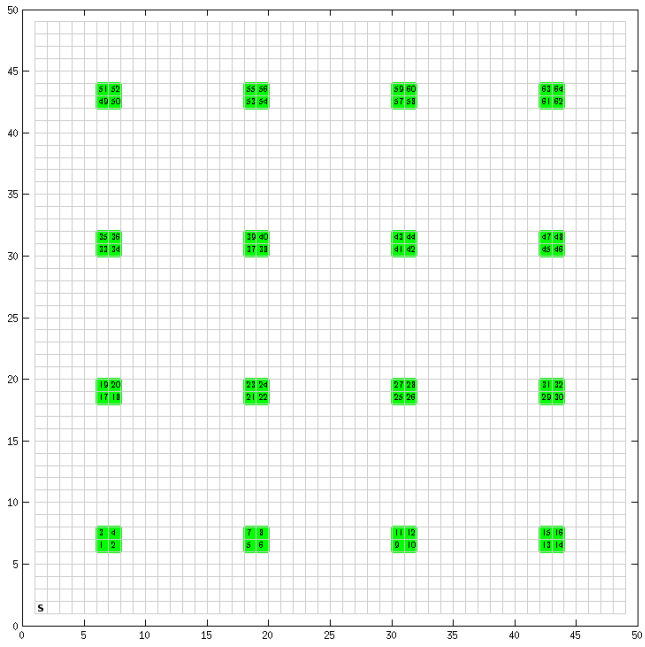


Figure 4: THE SURVEILLANCE DOMAIN. The domain is 48×48 gridworld with 64 possible surveillance locations (v-locations) marked in green. Each MDP in the domain requires the agent to surveil i different locations, $i \in \{1, 2, 3, 4\}$ in a particular sequence to receive positive reward of 200 for each location surveilled. A wrong location results a negative reward of -10 (the infiltrators have escaped). Each step gives a reward of -1. The target v-locations are clustered in groups of 4, such that surveilling one location in the cluster instead of the other results in a reward of 190 (a penalty of $200 - 190 = 10$) but does not end the episode (please see Figure 5 for further details).

given in Appendix D. Figures 6a and 6b both show that EXP-3-Transfer always benefits from using clustering and in fact the more complex the task is, the better the performance is. This is observed in the general upwards trend in the curves with increasing number of previous MDPs and the fact that the curve for the 3 v-locations lies above the curve for the 2 v-locations. This result is in complete agreement with our expectations, that in a bandit like algorithm lowering the number of arms will result in lower regret. In addition, it also shows that our clustering algorithm retains the correct arms so that with the removal of arms, the performance of EXP-3-Transfer is not affected adversely. The above figure does not give comparison with Q-learning. In the full results given in the Appendix D, we also show that EXP-3 consistently outperforms Q-learning by a large margin which shows that our algorithm escapes negative transfer.

Interestingly, for Policy-Reuse the trend is reversed. It appears that clustering does not help this algorithm, and the more complex the task is, the more harmful clustering is. Our conjecture regarding the reason for this is that Policy-Reuse does not reuse policies in the sense of using them as potential optimal policies, but as exploration devices. By clustering the MDPs, we remove arms and hence reduce the number of exploration policies and hence lower Policy-Reuse's scope for exploration. This in turn results in negative performance gain for Policy-Reuse.

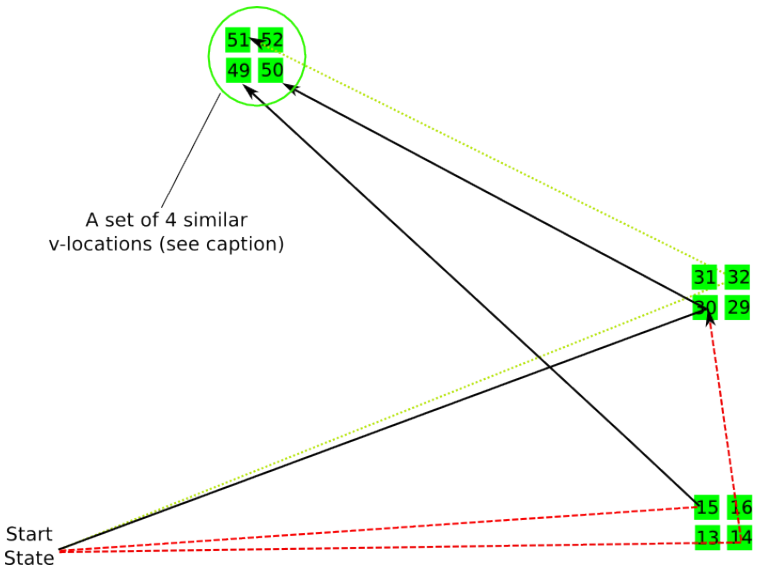


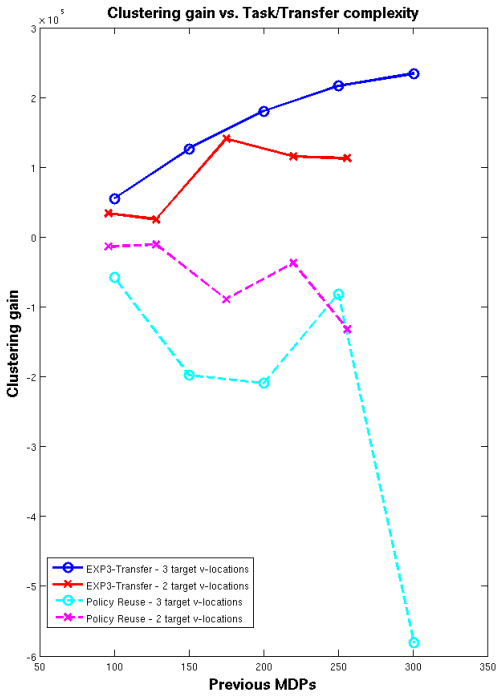
Figure 5: TRAJECTORY EXAMPLES. Example of different types of trajectories for the surveillance domain for a given MDP with two v-locations, 30 and 50 in sequence. The optimal trajectory for this MDP is shown in black/solid line. A trajectory that obtains a reward of 190 per target v-location is shown in green/dotted line (it visits two v-locations in the same block in the right sequence). The red/dashed trajectory (result in rewards of $-10, -10$ when the surveil action is taken at the two v-location. The red/dashed-black/solid trajectory result in rewards of $-10, 200$.

It is also interesting to see that in Figure 6b, which shows the gain in terms of the final reward obtained, the initial gain for EXP-3-Transfer and Policy-Reuse are both negative, and the gain for Policy-Reuse is higher. But as the transfer complexity increases (both in terms of previous MDPs and task complexity) the cumulative reward gain becomes positive for EXP-3-Transfer, while for Policy-Reuse it continues to decrease.

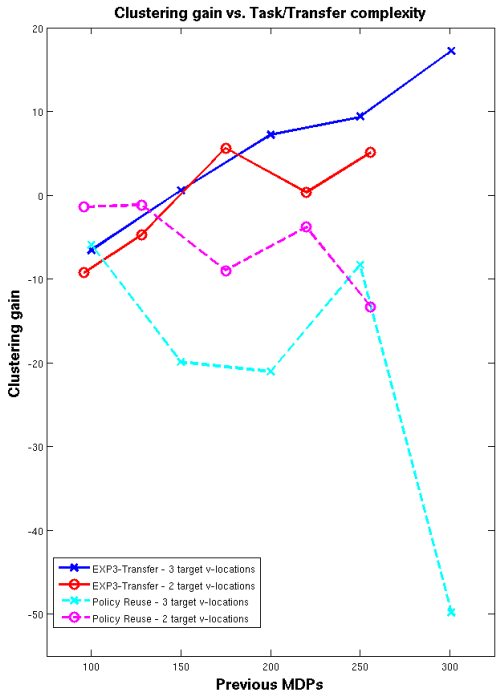
Given that the previous figures show that Policy-Reuse does not benefit from clustering, we compare the cumulative reward obtained by EXP-3-Transfer with clustering and Policy-Reuse without clustering for the complex 3-target-v-locations problem in Figure 7. This result shows that EXP-3-Transfer completely dominates Policy-Reuse, with the difference becoming particularly stark when the number of previous tasks increases to 300.

7.2.2 CLUSTERING COMPARISON

In this section we compare performance of EXP-3-Transfer using the different types of clustering methods in Table 1. As in the previous section, we look at the change in performance with increasing complexity of the transfer tasks. The summary of these results is given in Figure 8. As can be seen, EXP-3-Transfer using using Search-Clustering to obtain the source policy outperfoms the case when we do not cluster the previous tasks. This was the result reported in the previous section. However, in addition we also see that using greedy clustering is as good as using Search-Clustering. This is largely due to the structure of the domain, where groups of tasks are all similar to each other. To



(a) CUMULATIVE CLUSTERING GAINS.



(b) FINAL CLUSTERING GAINS.

Figure 6: CLUSTERING GAINS. The above figures shows the **clustering gain** for EXP-3-Transfer and Policy-Reuse. For each (x, y) data-point in each curve, the y-value is the difference in performance between using clustering and not using it when there are x previous MDPs. The performance measure in cumulative clustering gains (Figure 6a) is the total cumulative discounted reward over $10,000$ episodes. The performance measure in final clustering gains (Figure 6b) is the discounted reward in the final episode. We re-note that each (x, y) point is averaged over 10 different target tasks with 10 trials per task.

show cases where greedy clustering fails, we performed additional experiments, which are described in the next section.

7.2.3 GREEDY CLUSTERING MAY FAIL

In this section, we present results from an extension of the surveillance domain where the greedy clustering may fail and using Search-Clustering is necessary. In the original surveillance domain, if the agent surveilled v-locations in the same block, then it received rewards close to the optimal (see Figure 5), and this defined the similarity between the domain. In this subsection only, we consider a different similarity measure defined by a graph over the v-locations. If there is an edge between the v-locations then the reward obtained is 190, otherwise it is 200. This is illustrated in Figure 9. This can be interpreted as the center v-location being at the top of a 'hill-top' and the 4 other surrounding points being at sea-level. Hence, if we surveil the hill-top location we can also

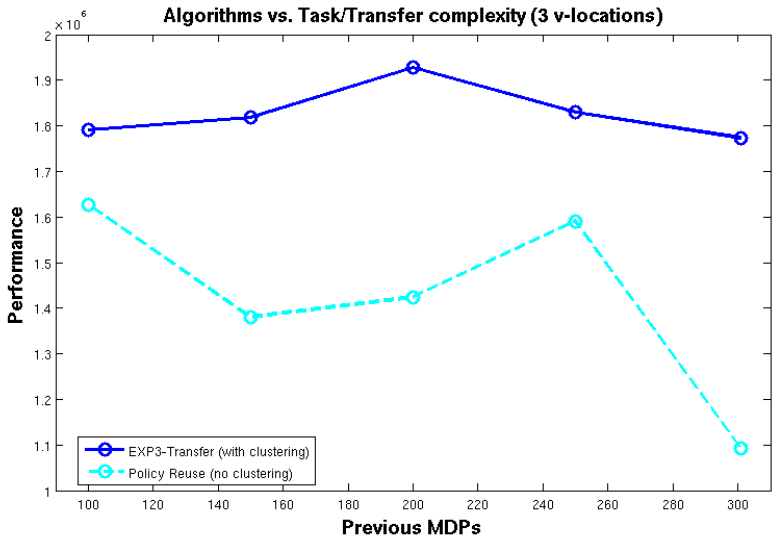


Figure 7: CUMULATIVE REWARD SUMMARY. This figure shows the final cumulative rewards after $10,000$ episodes for EXP-3-Transfer with clustering and Policy-Reuse without clustering for the surveillance domain with 3 target-v-locations. The x-axis is the number of previous MDPs.

surveil the lower locations automatically. Surveilling one sea level location means we may surveil the hill-top location, but not necessarily the other locations.

In this case, the results of Greedy clustering and Search-Clustering are illustrated in Figure 10. As the figure shows, greedy clustering performs very poorly in this case, while Search-Clustering finds the right clusters.

7.2.4 TIME COMPARISONS

In this section we look at the effect of the T parameter on performance. Recall that the T parameter affects both the clustering algorithm Search-Clustering and EXP-3-Transfer. We performed experiments with 7 different combinations of previous MDPs and MDP complexity. The results are more or less identical, and so we present only two graphs in Figure 11 for the most complex and the least complex transfer problem we have considered. We relegate the remaining graphs for the rest of the experiments to Appendix D.2 since these two figures are representative of the other graphs. As they show, the curve for EXP-3-Transfer with parameter T lies above the curves of EXP-3-Transfer with parameter $T' > T$. This confirms that setting this parameter is actually important.

8. Conclusion

In this paper we developed a framework to concisely represent a large number of previous MDPs by a smaller number of source MDPs for transfer learning. We presented a principled online transfer learning algorithm, a principled way to evaluate source sets for use in this algorithm and way to find the source set. The key idea was to cluster the previous MDPs and then use the representative

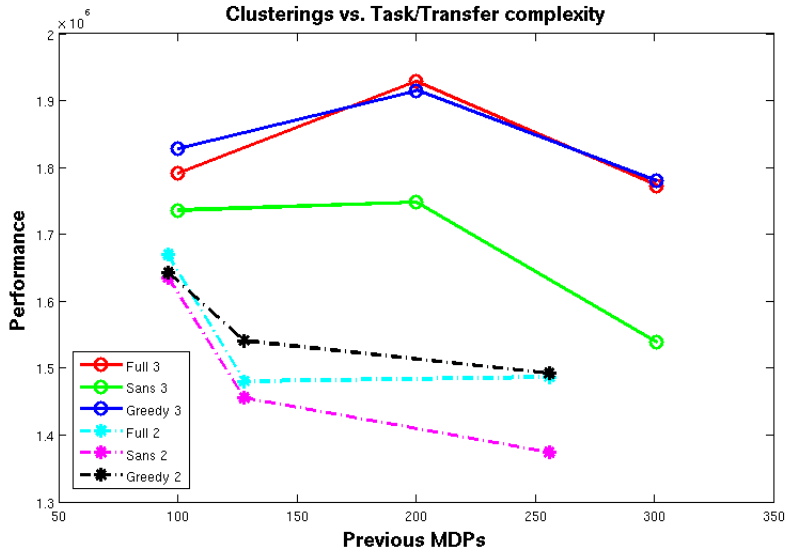


Figure 8: EFFECT OF CLUSTERING METHODS. This figure compares the performance of Greedy-Clustering and Search-Clustering for the domain with graph based distance. The performance is measured in terms of the total cumulative discounted rewards over $10,000$ episodes. In this figure 'Full' refers to EXP-3-Transfer run with Search-Clustering.

element of each cluster as the source tasks. We also presented extensive experiments to show the efficacy of our method. We now discuss several interesting directions for future work.

In this paper we only considered discrete domains. However, it is possible to translate the overall approach to the continuous setting. In particular, to apply our approach to continuous space problem, all we will need is a pure RL algorithm (as an arm in EXP-3-Transfer) and a way to evaluate policies (to compute the d_V distances). All our definitions, algorithms and results will then hold true in this setting. This is because our algorithms EXP-3-Transfer, MHAV and Search-Clustering and distance function d_V treats the underlying MDPs and policies as black boxes with certain properties. The discreteness of the MDP is never exploited or required in either the algorithms or their analysis.

We also only looked at only one possible VPL metric. As we pointed out in the introductory material of Section 7, this metric is not particularly interesting, and additionally, after extensive search we were unable to discover another one. So another possible interesting line of future enquiry is to fully develop the theory of VPL metrics as it results in a theory that holds under weaker assumptions. This work may involve the development of measures of similarity used in related work on bisimulation (Castro and Precup, 2010).

Finally, we end by pointing out that the idea of clustering a set of tasks to obtain a representative set is much more general. For instance, any other cost function derived under different assumptions can be applied with the clustering approach. As another example, the clustering approach may also be used in multi-agent systems to group together opponents according to whether the same policy of ours is equally effective against opponents in the same group. It will also be interesting to implement these methods and algorithms on scaled up, real version of the types of problems considered in this paper. We plan to pursue these and other extensions to the above in future work.

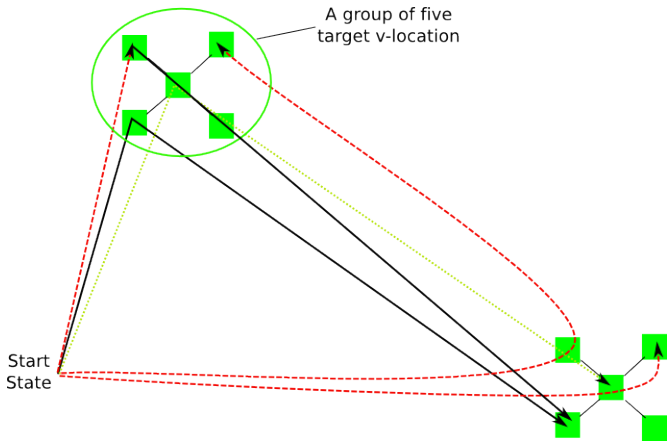


Figure 9: THE EXTENDED GRAPH DOMAIN. The extended graph domain consists of groups of target v-locations, with 5 v-locations per group. In our experiments we used 16 groups but for simplicity we only show two groups here. If there is an edge between two v-locations, then surveilling in one instead of the other location is acceptable with a penalty (of $200 - 190 = 10$)– otherwise it is not acceptable. As before, each MDP consists of surveiling the correct sequence of target v-locations. As in Figure 5, the black/solid line shows the optimal trajectory for a particular MDP. The green/dotted line shows a trajectory that incurs a penalty but is acceptable. A red/dashed line indicates a trajectory where the surveil action at the end incurs a reward of -10.

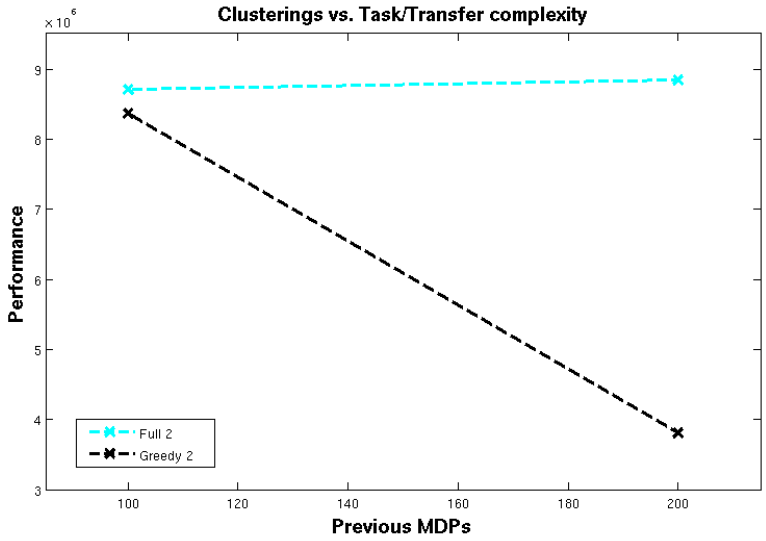
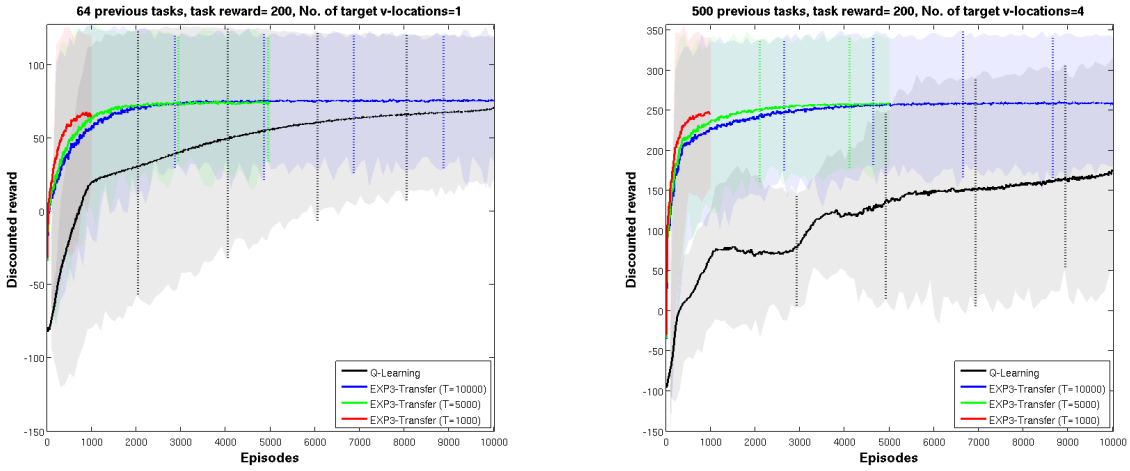


Figure 10: CLUSTERING IN EXTENDED SURVEILLANCE DOMAIN. This figure compares performance of EXP-3-Transfer in the extended surveillance domain when using greedy clustering vs. Search-Clusterings (referred to by Full in the legends). The performance is measured in terms of the total cumulative discounted rewards over $10,000$ episodes.



(a) LOW COMPLEXITY TRANSFER PROBLEM.

(b) HIGH COMPLEXITY TRANSFER PROBLEM.

Figure 11: EFFECT OF T PARAMETER. The above figures show the learning curve of EXP-3-Transfer when run for different numbers of time steps (parameter T). This affects both the clustering and the arms chosen by EXP-3-Transfer. The parmaters for the experiments are given in the title of the figure. As the figure shows, for shorter T, the EXP-3-Transfer run with the lowest $T = 1000$ is optimal. For the intermediate duration, $T = 5000$ is optimal, and for the remaining time $T = 10,000$ is optimal. Figures 11a and 11b respectively give the curves for the lowest and highest complexity task. The shaded areas give the standard deviation for the learning curves.

Appendix A. Proofs

Proof [Proof of Theorem 2] A direct application of Corollary 3.2 in (Auer et al., 2002b) is not possible because our algorithm diverges from EXP-3 because the number of arms possibly decreases across time steps. But the steps in the first part of the proof correspond closely to Theorem 3.1 (often identically), from which Corollary 3.2 is derived. The second part, where we deal with arms that were removed, is novel.

Let $C_t \triangleq \{1, 2, \cdot, c+1\} - rem_t$ be the arms that are in play at line 4 of EXP-3-Transfer, and let $c_t = |C_t|$. Let $W_t \triangleq \sum_{i \in C_t} w_i(t)$, and $\tilde{W}_{t-1} \triangleq \sum_{i \in C_t} w_i(t-1)$ (note the C_t, rather than C_{t-1} in the summation in $\tilde{W}$). Then for all sequences of policies $i_1, i_2, \cdots, i_T$, drawn by EXP-3-Transfer,

$$\frac{W_{t+1}}{W_t} \leq \frac{W_{t+1}}{\tilde{W}_t} \leq 1 + \frac{\beta/c_t}{1-\beta} x_{i_t}(t) + \frac{(e-2)(\beta/c_t)^2}{1-\beta} \sum_{i \in C_t} \hat{x}_i \tag{25}$$

The first inequality follows because $W_t \leq \tilde{W}_t$ as the latter contains fewer elements, while the final inequality follows from (8) in the proof of Theorem 3.1 (Auer et al., 2002a). Using the fact that $1 + x \leq \exp(x)$ gives us

$$\ln \frac{W_{t+1}}{W_t} \leq \frac{\beta/c_t}{1-\beta} x_{i_t}(t) + \frac{(e-2)(\beta/c_t)^2}{1-\beta} \sum_{i \in C_t} \hat{x}_i$$

Now since c_t is non-increasing, $\gamma/c_t \leq \gamma/c_T$, and so

$$\ln \frac{W_{t+1}}{W_t} \leq \frac{\beta/c_T}{1-\beta} x_{i_t}(t) + \frac{(e-2)(\beta/c_T)^2}{1-\beta} \sum_{i \in C_t} \hat{x}_i$$

Summing over t telescopes and gives us

$$\ln \frac{W_T}{W_1} \leq \frac{\beta/c_T}{1-\beta} G_{E3T} + \frac{(e-2)(\beta/c_T)^2}{1-\beta} \sum_{t=1}^{T} \sum_{i \in C_T} \hat{x}_i$$

Now, going in the opposite direction, for each $j \in C_T$ we have

$$\ln \frac{W_T}{W_1} \geq \ln \frac{w_j}{c+1} = \frac{\beta}{c_T} \sum_{t=1}^{T} \hat{x}_j(t) - \ln(c+1)$$

Putting these together, we have for each $j \in C_T$,

$$G_{E3T} \geq (1-\beta) \sum_{t=1}^{T} \hat{x}_j^t - \frac{c_T \ln(c+1)}{\beta} - \frac{(e-2)\beta}{c_T} \sum_{i \in C_T} \hat{x}_i \tag{26}$$

Taking expectation in terms of the randomization in the algorithm in both sides above, we have

$$\mathbb{E}[G_{E3T}] \geq (1-\beta) \sum_{t=1}^{T} x_j(t) - \frac{c_T \ln(c+1)}{\beta} - \frac{(e-2)\beta}{c_T} \sum_{t=1}^{T} \sum_{i \in C_T} x_i(t) \tag{27}$$

where we used that fact that $\mathbb{E}[\hat{x}_i(t)|i_1, i_2, \cdots, i_{t-1}] = x_i(t)$ for any i. Using the fact that $\sum_{t=1}^{T}\sum_{i\in C_T} x_i(t) \leq c_T G_{\max}$ and then rearranging, we get for each $j \in C_T$,

$$\sum_{t=1}^{T} x_j(t) - \mathbb{E}[G_{E3T}] \leq \beta \sum_{t=1}^{T} x_j(t) + c_T \ln(c+1) + (e-2)\beta G_{\max}$$
$$= \frac{c_T \ln(c+1)}{\beta} + (e-1)\beta G_{\max}$$

Plugging in the value of β and using the fact that $G_{\max} \leq T$ and then taking expectations with respect to randomness due to P and R, we get that

$$\mathbb{E}[\sum_{t=1}^{T} x_j(t)] - \mathbb{E}[G_{E3T}] \leq 2.63\sqrt{c\ln(c)T} \tag{28}$$

Taking expectation with respect to the transition and reward distributions P and R of the target task, and putting back the normalization term, now gives us the required result for an arm that was not removed. We now need to show that (28) is true for an arm x_k that is removed at some $t \leq T$.

To that end, we first need the Hoeffding bound (see, for instance, (Dubhashi and Panconesi, 2009) for an exposition) which states that if $y_{1:n} \triangleq y_1, y_2, \cdots, y_n$ are i.i.d. draws of a random variable Y, with $Y_i \in [a, b]$, and $\bar{y}_n$ is the empirical mean of the y_i, then

$$Pr[|\bar{y}_n - \mathbb{E}(Y)| > \epsilon] \leq \exp[-2n\epsilon^2/(b-a)^2] \tag{29}$$

In the sequel, we will assume that $b = 1, a = 0$, and so the denominator in the exponent on the left hand side of the equation is just 1. This bound then has the following simple and well known consequence. Assume we have two i.i.d. samples $y_{1:n}$ and $y'_{1:m}$, drawn from two random variables Y and Y'. Assume that $\bar{y}_n - \bar{y'}_m > \epsilon$, and n and m both satisfy $\exp[-2n\epsilon^2/4] \leq \delta'/2$ and $\exp[-2m\epsilon^2/4] \leq \delta'/2$. Then, by (29)

$$Pr[|\bar{y}_n - \mathbb{E}(Y)| > \epsilon/2] \leq \delta'/2, \qquad Pr[|\bar{y'}_n - \mathbb{E}(Y')| > \epsilon/2] \leq \delta'/2 \tag{30}$$

Then, by the triangle inequality and the union bound, with probability at least $1-\delta'$, $\mathbb{E}(Y) > \mathbb{E}(Y')$.

Now in line 12 of EXP-3-Transfer, we remove a source policy arm if $\epsilon = z_j/n_j - z_k/n_k$, we have $\epsilon/2 > \sqrt{-\ln(\delta/2c)(2n_j)^{-1}}$ and $\epsilon/2 > \sqrt{-\ln(\delta/2c)(2n_k)^{-1}}$. This implies, that with probability $> 1-\delta/c$, $V^{\rho_j} > V^{\rho_k}$. Since there are c arms, this implies that if there is an arm that was removed, by the union bound with probability at least $> 1-\delta$, $V^{\rho_j} > V^{\rho_k}$ for some arm j that is never removed for every arm k that is eventually removed. Now note that the expectation of the first term $\sum_{t=1}^{T} x_j(t)$ in (28) is TV^{ρ_j} when ρ_j is a stationary source policy. Hence, coupling the results of this paragraph with (28) we get that

$$\mathbb{E}[\sum_{t=1}^{T} x_k] - \mathbb{E}[G_{E3T}] \leq 2.63\sqrt{c\ln(c)T}$$

which is what we are required to prove. ■

Proof [Proof of Lemma 5] Let there be three MDPs $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ defined on a state space with a single state and three actions a_1, a_2, a_3. Assume that $R_1(a_1) = 100, R_1(a_2) = 90, R_1(a_3) = -100$ and for $i \in \{2, 3\}$, $R_i(a_i) = 100$, and $R_i(a_j) = 90$ when $i \neq j$. So the optimal action for $\mathcal{M}_i$ is a_i. But now, $d_V(\mathcal{M}_1, \mathcal{M}_2) = 100 - 90 = 10$, $d_V(\mathcal{M}_2, \mathcal{M}_3) = 100 - 90$, but $d_V(\mathcal{M}_1, \mathcal{M}_3) = 100 - (-100) = 200$, showing that d_V does not satisfy the triangle inequality and hence is not a metric. ■

For the next two proofs, we need to restate Lemma 1 in (Strehl et al., 2009).

Lemma 20 *[Strehl, Li and Littman] If $\mathcal{M}_1$ and $\mathcal{M}_2$ are two MDPs defined on the same state-action space, and $\epsilon_1 = |R_1(s,a) - R_2(s,a)|$, and $\epsilon_2 = \|P_1(\cdot|s,a) - P_2(\cdot|s,a)\|_1$, then for any policy π and state s,*

$$|V_1^\pi(s) - V_2^\pi(s)| \leq (\epsilon_1 + \gamma R_{\max}\epsilon_2)(1-\gamma)^{-2} \tag{31}$$

□

Now we can state our proofs.

Proof [Proof of Lemma 7] For d_M, 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 20 with $k(\epsilon) = \frac{\epsilon(1+\gamma R_{\max})}{(1-\gamma)^2}$, where $R_{\max}$ was defined in Section 2. To see this last fact, if $d_M(\mathcal{M}_1, \mathcal{M}_2) \leq \epsilon$ then $\max_{s,a} \max(|R_1(s,a) - R_2(s,a)|, \|P_1(\cdot|s,a) - P_2(\cdot|s,a)\|_1 \leq \epsilon$. Plugging this into the statement of Lemma 20, we get that $|V_1^\pi - V_2^\pi| \leq k(\epsilon)$ for any policy π. In particular, this is true for π_1^*, and π_2^*, and hence by the definition of d_V, it follows that $d_V(\mathcal{M}_1, \mathcal{M}_2) \leq k(\epsilon)$. ■

Proof [Proof of Theorem 9] Let $\mathcal{M}^* \triangleq_{1 \leq k \leq N} \arg\min d_M(\mathcal{M}_{N+1}, \mathcal{M}_k)$ and let $d_M(\mathcal{M}_{N+1}, \mathcal{M}_k) = \epsilon^*$. Let $\hat{\mathcal{M}}^*$ be the centroid of the cluster A_i (as defined in (8)) such that $\mathcal{M}^* \in A_i$. Then $d_M(\mathcal{M}^*, \hat{\mathcal{M}}^*) \leq \epsilon$ by definition of ϵ. By the Lipschitz property of d_M, $d_M(\mathcal{M}_{N+1}, \hat{\mathcal{M}}^*) \leq \epsilon^* + \epsilon$ (because by Lemma 7 $K = 1$ for d_M). This implies that $d_V(\mathcal{M}_{N+1}, \hat{\mathcal{M}}^*) \leq k(\epsilon^*) + k(\epsilon)$, and in particular $V_{N+1}^* - V_{N+1}^\rho \leq k(\epsilon^*) + k(\epsilon)$ where ρ is the optimal policy of $\hat{\mathcal{M}}^*$, which is used as an arm in EXP-3-Transfer. The definition of d_V implies. By Corollary 3, $V_{N+1}^\rho - \mathbb{E}[G_{E3T}] \leq g(c)$. Putting this altogether, this implies that $V_{N+1}^* - \mathbb{E}[G_{E3T}] \leq g(c) + k(\epsilon) + k(\epsilon^*)$. The theorem now follows with $\hat{K} = k(\epsilon^*)$. ■

Proof [Proof of Theorem 11] Fix any previous $\mathcal{M}_k$ and let $\mathcal{M}^{j_k}$ be the centroid of the cluster A_{j_k} of $\mathbf{A}$ that $\mathcal{M}_k$ belongs to. Let the optimal policy of $\mathcal{M}^{j_k}$, used as an arm in EXP-3-Transfer, be ρ_{j_k}. Then by Corollary 3, $V_{N+1}^{\rho_{j_k}} - \mathbb{E}[G_{E3T}]/T \leq g(c)$, and therefore $V_{N+1}^{\pi_k^*} - \mathbb{E}[G_{E3T}]/T \leq g(c) + \epsilon_{j_k}$, where ϵ_{j_k} was defined in (10). Summing over k, and dividing by N yields $\frac{1}{N}[\sum_{k=1}^N V_{N+1}^{\pi_k^*} - N\mathbb{E}[G_{E3T}]/T \leq g(c) + \frac{1}{N}\sum_{k=1}^N \epsilon_{j_k}$ which is equivalent to

$$\mathbb{E}[V_{N+1}^{\pi_k^*}] - \mathbb{E}[G_{E3T}]/T \leq g(c) + \bar{\epsilon}$$

which completes the proof. ■

Proof [Proof of Lemma 14] To show irreducibility we have to show that for any (λ, y) and (λ', y') there exists a n such that $\bar{P}_{\text{MH}}[X_n = (\lambda', y')|X_0 = (\lambda, y)] > 0$. To see this, first note that ϕ_Y was

assumed to be irreducible. So, there exists a n_1 such that with $\phi_Y(Y_n = y'|Y_0 = y) > 0$. Now consider a particular path $\mathbf{y} \triangleq y_0 y_1 y_2, \cdots y_{n-1}, y_n$ (where $y_0 = y, y_n = y'$) with probability > 0 under ϕ_Y. From the definition in (16), the probability under $\bar{P}_{\text{MH}}$ of each transition $y_i \to y_{i+1}$ is

$$\beta\bar{\phi}[(\lambda, y_i), (\lambda, y_{i+1})]\mathsf{Acc}[(\lambda, y_i), (\lambda, y_{i+1})] > \beta b \phi_Y[(\lambda, y_i), (\lambda, y_{i+1})]$$

where the inequality follows as $\mathsf{Acc}[\cdot, \cdot] > b$ by definition of b and Acc. Hence, the total probability of the path $yy_1y_2, \cdots y_{n-1}, y'$ under $\bar{P}_{\text{MH}}$ is lower bounded by $b^n\beta^n\phi_Y(\mathbf{y})$ (where $\phi(\mathbf{y}) = \prod_{i=0}^{n}\phi(y_i, y_{i+1})$). Summing over all possible paths of length n going from y to y' gives that the probability of each (λ, y') from (λ, y) is lower bounded by $b^n\beta^n\phi_Y(Y_n = y'|Y_0 = y)$.

Now assume that $\lambda = \lambda_k$ while $\lambda' = \lambda_{k'}$. If $k < k'$, we can bound the probability under $\bar{P}_{\text{MH}}$ of going from (λ_i, y') to (λ_{i+1}, y'), where $k \le i < k'$, by $z_i \triangleq \alpha\alpha' \frac{(1-\alpha)}{\alpha}(\lambda_{i+1}/\lambda_i)^{-f(y)}$ (this follows from definition of $\bar{P}_{\text{MH}}$ and Acc). Hence we reach (λ', y') from (λ, y') with probability $z \triangleq \prod_{i=k}^{k'-1} z_i$. By a symmetric argument, if $k' < k$, we reach λ' from λ with probability at least $z' \triangleq \prod_{i=k}^{k'-1} z'_i$, where $z'_i \triangleq \alpha(1-\alpha')\frac{\alpha}{(1-\alpha)}(\lambda_i/\lambda_{i+1})^{-f(y)}$. Both z, z' are positive by the finiteness of $f(y)$ and λ_is. Putting all the above together, we have that the probability of transitioning from (λ, y) to (λ', y') is lower bounded by

$$b^n\beta^n\phi_Y(Y_n = y'|Y_0 = y)\min\{z, z'\} > 0$$

which shows that $\bar{P}_{\text{MH}}$ is irreducible.

To show that $\bar{P}_{\text{MH}}$ is a-periodic, it is sufficient to note that $\alpha + \beta < 1$. Then, with probability $1 - \alpha - \beta$, $\bar{P}_{\text{MH}}$ returns to the same state in 1 step, which ensures that the g.c.d. of the set of time steps where $\bar{P}_{\text{MH}}$ returns to the same state is 1. ■

Proof [Proof of Theorem 15] $\bar{\phi}$ is irreducible by Lemma 14 and by construction of an MH chain, $\bar{P}_{\text{MH}}$ has $\bar{\Pi}$ stationary distribution. Hence, by the first part of Theorem 12 $\bar{P}_{\text{MH}}$ converges to $\bar{\Pi}$ in total variation. By the second part of the same theorem, if $\|\bar{P}_{\text{MH}}(X_t = \cdot|x_0) - \bar{\Pi}(\cdot)\|_{\text{TV}} \le k$, then for all $t' > t$, $\|\bar{P}_{\text{MH}}(X_{t'} = \cdot|x_0) - \bar{\Pi}(\cdot)\|_{\text{TV}} \le k$. ■

Proof [Proof of Theorem 16] As we mentioned above, this proof follows very closely the proof of Theorem 4.9 in (Levin et al., 2009). To begin with, first we note that by irreducibility of $\bar{P}_{\text{MH}}$, the diameter D (defined in (23)) is finite. Hence, by definition of δ in (24), for each x, x' we have that $\bar{P}_{\text{MH}}(x, x') \ge \delta\bar{\Pi}(x')$.

Let $\bar{P}_{\text{MH}}$ denote the transition matrix for the kernel $\bar{P}_{\text{MH}}$ and let $\bar{\mathbf{\Pi}}$ denote the transition matrix where each row is $\bar{\Pi}$. Then, setting $\theta \triangleq (1 - \delta)$, we can write

$$\bar{P}_{\text{MH}} = (1 - \theta)\bar{\mathbf{\Pi}} + \theta Q$$

where Q is another transition matrix. To see that Q is a valid transition matrix, note that row i of Q is given by $\theta^{-1}[\bar{P}_{\text{MH}}(x, \cdot) - (1 - \theta)\bar{\Pi}(\cdot)]$. Summing the elements of this row, we get $\sum_{x'} \bar{P}_{\text{MH}}(x, x') - (1 - \theta)\bar{\Pi}(x') = \theta$, whence each row of Q sums to 1. Furthermore, by the definition that $(1 - \theta) = \delta$, each entry is also positive, showing that Q is indeed a valid transition matrix.

Now note that for any transition matrix $\mathbf{M}$, $\mathbf{M}\Pi^T = \Pi^T$ (where ^{T} indicates the transpose), whence $\mathbf{M}\bar{\mathbf{\Pi}} = \bar{\mathbf{\Pi}}$. Additionally, since $\bar{\Pi}$ is stationary for $\bar{P}_{\mathrm{MH}}$, $\bar{\mathbf{\Pi}}\bar{P}_{\mathrm{MH}} = \bar{\mathbf{\Pi}}$. We will now use the above facts to show by induction on k that

$$\bar{P}_{\mathrm{MH}}^{Dk} = (1-\theta^k)\bar{\mathbf{\Pi}} + \theta^k Q^k \tag{32}$$

which will imply the convergence we seek.

Clearly (32) is true for $k = 0$. Assume, as the inductive hypothesis, that it is true for $k \leq n$. Then, we have

$$\begin{aligned}
\bar{P}_{\mathrm{MH}}^{D(n+1)} &= \bar{P}_{\mathrm{MH}}^{Dn}\bar{P}_{\mathrm{MH}}^{D} \\
&= (1-\theta^n)\bar{\mathbf{\Pi}} + \theta^n Q^n \bar{P}_{\mathrm{MH}}^{D} \\
&= (1-\theta^n)\bar{\mathbf{\Pi}} + \theta^n Q^n[(1-\theta)\bar{\mathbf{\Pi}} + \theta Q] \\
&= (1-\theta^n)\bar{\mathbf{\Pi}} - \theta^{n+1}\bar{\mathbf{\Pi}} + \theta^n\bar{\mathbf{\Pi}} + \theta^{n+1}Q^{n+1} \\
&= (1-\theta^{n+1})\bar{\mathbf{\Pi}} + \theta^{n+1}Q^{n+1}
\end{aligned}$$

The first equality is just the definition of k-step transitions. The second equality is obtained by applying the inductive hypothesis and because $\bar{\mathbf{\Pi}}\bar{P}_{\mathrm{MH}} = \bar{\mathbf{\Pi}}$. The third and fourth equality follows from applying the inductive hypothesis on $\bar{P}_{\mathrm{MH}}^{D}$ and the two facts about $\bar{\mathbf{\Pi}}$ established above. The final equality is obtained by cancelling out the terms.

Now $\theta^k \to 0$ as $k \to \infty$, and so each row of $\bar{P}_{\mathrm{MH}}$ converges to $\bar{\Pi}$. In other words for each x, $\lim_{t\to\infty}\sum_{x'} \bar{P}_{\mathrm{MH}}^t(x') - \bar{\Pi}(x') = 0$. This implies $\lim_{t\to\infty}\|\bar{P}_{\mathrm{MH}}^t(x,\cdot) - \bar{\Pi}(\cdot)\|_1 = 0$. Now since $\|\bar{P}_{\mathrm{MH}}^t(x,\cdot) - \bar{\Pi}(\cdot)\|_{\mathrm{TV}} = \frac{1}{2}\|\bar{P}_{\mathrm{MH}}^t(x,\cdot) - \bar{\Pi}(\cdot)\|_1$, this completes the proof. ■

Proof [Proof of Lemma 17] Fix any two (λ, y) and (λ, y') and let $\mathbf{x} \triangleq x_0 x_1 \cdots x_n$ be a path with $x_0 = (\lambda, y)$ and $x_n = (\lambda', y')$. Assume that this path has positive probability under $\bar{P}_{\mathrm{MH}}$ for certain value a, b, c, respectively of α', α, β. Then, by definition (21) of $\bar{P}_{\mathrm{MH}}$, the probability of this path has the form $Ca^k(1-a)^{k_2}b^{k_2}c^{k_3}(1-b-c)^{k_4}$ where the k_i are integers and C is a constant. Then, under a difference set of values a', b', c', the probability of this path has the form $Ca'^k(1-a')^{k_2}b'^{k_2}c'^{k_3}(1-b'-c')^{k_4}$. Since $\alpha', \alpha, \beta \in (0,1)$, this probability must also be non-zero. Hence the set of paths of positive probability are invariant with respect to the values of α', α and β. Since D is the length of the shortest path of positive probability, this proves the lemma. ■

Proof [Proof of Lemma 19] We just need to show that, for any two clusterings $\mathbf{A}$ and $\mathbf{A}'$, only a finite number of re-arrangement steps is sufficient to obtain $\mathbf{A}'$ from $\mathbf{A}$. Let the clusters of $\mathbf{A}'$, in some order, be $A'_1, A'_2, \cdots, A'_n$. Assume that the points of A'_i are spread across $A_{i_1}, \cdots, A_{i_k}$ with $n_1, n_2, \cdots, n_k$ points respectively. Then, with non-zero probability A'_i will be created with n_1 points from A_{i_1} (see Appendix C for the explicit computation). And from then on, with non-zero probability (again, see the computations given) the points of A'_i in A_{i_j} will be added to A'_i. Hence with non-zero probability A'_i will be created. This holds for each A'_i, and hence we have a non-zero probability of constructing $\mathbf{A}'$ from $\mathbf{A}$. ■

Appendix B. Hardness of the Clustering Problem

In this section we show that it is hard to optimize $cost_1$ and an upper bound $cost_{2m}$ of $cost_2$, where $cost_1$ and $cost_2$ are given in Definitions 8 and 10 respectively. To that end, define the average max-diameter of a clustering $\mathbf{A}$ to be:

$$\bar{\epsilon}_m = \frac{1}{N}\sum_i |A_i|\bar{\epsilon}^i_m, \text{ where } \epsilon^i_m = \frac{1}{|A_i|}\sum_{\mathcal{M}\in A_i}\max_{\mathcal{M}'\in A_i} d_V(\mathcal{M},\mathcal{M}') \tag{33}$$

Now define,

Definition 21 *Define* $cost_{2m}(\mathbf{A}) \triangleq g(c) + \bar{\epsilon}_m$.

We have the following relationships:

Lemma 22 *The parameter* $\bar{\epsilon}_m$ *of* $\mathbf{A}$ *is an upper bound on the parameter* $\bar{\epsilon}$ *of* $\mathbf{A}$ *defined in (11). Furthermore,* $cost_{2m}$ *upper bounds* $cost_2$*:*

Proof This follows directly from the definition of $\bar{\epsilon}$ – in particular, $\bar{\epsilon}_m$ upper bounds $\bar{\epsilon}$ because $\max_{\mathcal{M},\mathcal{M}'\in A_i} d_V(\mathcal{M},\mathcal{M}') > \min_{\mathcal{M}}\max_{\mathcal{M}'} d_V(\mathcal{M},\mathcal{M}') = d_V(\mathcal{M}^i,\mathcal{M})$. The second part of the lemma now follows by the definitions of the functions. ■

We show that finding the clusterings optimizing $cost_1$ and $cost_{2m}$ are in fact NP-complete by reducing the minimum clique-cover problem (Karp, 1972) to finding the optimal clustering of a given set of MDPs. We start by describing the clique cover problem. Let $G = (V, E)$ be a graph where V is the set of vertices and E is the set of edges. A subset $V' \subset V$ is a clique if for any $v, v' \in V$, there is an edge $(v, v') \in E$. The minimum clique cover problem is finding a partition $V_1, V_2, \cdots, V_n$ of V such that each V_i is a clique and n is minimum – that is there exists no other partition with $V'_1, V'_2, \cdots, V'_m$ of V such that each V'_i is a clique and $m < n$. We have the following theorem for $cost_1$.

Theorem 23 *Given a graph* $G = (V, E)$*, in time polynomial in the* $|V|$ *and* $|E|$*, we can reduce the minimum clique cover problem for* G *to finding the clustering* $\mathbf{A}^*$ *of MDPs* $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_{|V|}$*, with all* $\mathcal{M}_i$ *defined on the same state and action spaces, where* $\mathbf{A}^* \triangleq \arg\min_{\mathbf{A}\in\mathcal{C}} cost_1(\mathbf{A})$.

An analogous theorem, using essentially the same proof, holds for $cost_{2m}$.

Theorem 24 *Theorem 23 is also true if* $\mathbf{A}^* \triangleq \arg\min_{\mathbf{A}\in\mathcal{C}} cost_{2m}(\mathbf{A})$.

The proofs are given below. Since the clique cover problem is NP-complete, we immediately have the following corollary, which motivates the need for an algorithm to find the optimal clustering.

Corollary 25 *Finding the clustering optimizing either* $cost_1$ *or the upper bound* $cost_{2m}$ *of* $cost_2$ *is NP-complete.*

We now prove the theorems.

Proof [Proof of Theorem 23] First, let $|V| = M$, and given any ordering of the elements of V, identify each vertex $v \in V$ with its position in the ordering – so we can take $V = \{1, 2, \cdots, M\}$. Let $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_M$ be a set of MDPs defined on a state space $\mathcal{S} = \{s\}$, and action space

$\mathcal{A} = \{1, 2, \cdots, M\}$. The transition function for the MDPs in this case is trivial (all actions transition with probability 1 from s to s). The reward function for MDP i defined as follows. $R_i(s, a_i) = 0$; if $(i, j) \in E$ then $R_i(s, a_j) = 0$, otherwise $R_i(s, a_j) = -\epsilon^*$ where ϵ^* satisfies $k_M(\epsilon^*) = hg(M)$, where $h > 1$ and g is the function used in Definition 8 to define the cost function for clusters. This ϵ^* exists because k_M is invertible. In the following we will identify MDP $\mathcal{M}_i$ with vertex $i \in V$ and this way show that the optimal clustering for this corresponds to a maximal clique under the mapping $i \to v_i$ and $A \to V_i$.

By construction, $\pi_i^*(s) = a_i$, $V_i^* = 0$, $V_i^{\pi_j^*} = V_i^{a_j} = 0$ iff $(i, j) \in E$, and $V_i^{a_j} = -\epsilon^*$ otherwise. Hence, by the definition in (5),

$$d_M(\mathcal{M}_i, \mathcal{M}_j) = \begin{cases} 0 & \text{iff } (i, j) \in E \\ \epsilon^* & \text{otherwise} \end{cases} \tag{34}$$

First recall that by Definition 8, the cost of a clustering $\mathbf{A}$ of an MDP is $g(c) + \epsilon$ where c is the number of clusters and ϵ is the maximum over the diameter of the clusters in $\mathbf{A}$. Let an optimal clustering be $\mathbf{A}^*$ and let $A(\mathcal{M}_i)$ denote the cluster in $\mathbf{A}^*$ that $\mathcal{M}_i$ belongs to.

We now show that if $\mathcal{M}_{i_1}, \mathcal{M}_{i_1}, \cdots, \mathcal{M}_{i_l} \in A \in \mathbf{A}^*$, then $i_1, i_2, \cdots, i_l$ form a clique in G. In other words, we show that, if $A(\mathcal{M}_i) = A(\mathcal{M}_j)$ then $(i, j) \in E$, or equivalently $(i, j) \notin E$ then $A(\mathcal{M}_i) \neq A(\mathcal{M}_j)$. By way of contradiction, assume that $(i, j) \notin E$ but $A(\mathcal{M}_i) = A(\mathcal{M}_j)$. Since i, j do not have an edge between them, by (34) the diameter of $\mathbf{A}^*$ is at least $d_M(\mathcal{M}_i, \mathcal{M}_j) = \epsilon^*$. This in turn implies that $cost_1(\mathbf{A}^*) = g(|\mathbf{A}^*|) + k(\epsilon^*) = g(|\mathbf{A}^*|) + hg(M)$. Now consider the clustering $\mathbf{A}'$ obtained by putting each MDP $\mathcal{M}_i$ in its own cluster. This clustering has cost $g(M) + 0 < g(|\mathbf{A}^*|) + hg(M)$ – contradicting the optimality of $\mathbf{A}^*$. Hence, the clusters of $\mathbf{A}^*$ have cost $g(|\mathbf{A}^*|)$ and corresponds to a collection of cliques that partition V – denote this collection of cliques by J^*.

Now note that each collection of cliques $V_1, V_2, \cdots V_j$ that partition V correspond to a clustering $\mathbf{A}$ such that $\mathcal{M}_i, \mathcal{M}_j \in A$ iff $(i, j) \in V_l$ for some l; in this case $j = |\mathbf{A}|$. Now assume that there is a clique I such that $|I| < |J^*|$ and let the corresponding clustering be $\mathbf{A}_I$. Then we show that $cost_1(\mathbf{A}_I) < cost_1(\mathbf{A}^*)$, resulting in a contradiction. To see this note that each $\mathcal{M}_i, \mathcal{M}_j \in A \in \mathbf{A}_I$ then $d_M(\mathcal{M}_i, \mathcal{M}_j) = 0$ by (34). Hence the diameter of $\mathbf{A}_I = 0$. So the cost of $\mathbf{A}_I$ is $g(|\mathbf{A}_I|) + 0 < g(|\mathbf{A}^*|)$ since by definition of $\mathbf{A}_I$, $|\mathbf{A}_I| < |\mathbf{A}^*|$.

Because of the contradiction, J^* is indeed a minimum clique cover, showing that the problem of minimum clique cover can be reduced to the problem of finding the optimal clustering. To complete the proof, we need to show that this reduction takes polynomial time. The only cost in computing a $\mathcal{M}_i$ is setting the reward function, which takes time $C|V|$ for some constant C. ■

Proof [Proof of Theorem 24] The overall steps in the proof are quite similar to the proof of Theorem 23 but some important details vary. First, let $|V| = M$, and given any ordering of the elements of V, identify each vertex $v \in V$ with its position in the ordering – so we can take $V = \{1, 2, \cdots, M\}$. Let $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_M$ be a set of MDPs defined on a state space $\mathcal{S} = \{s\}$, and action space $\mathcal{A} = \{1, 2, \cdots, M\}$. The transition function for the MDPs in this case is trivial (all actions transition with probability 1 from s to s). The reward function for MDP i defined as follows. $R_i(s, a_i) = 0$; if $(i, j) \in E$ then $R_i(s, a_j) = 0$, otherwise $R_i(s, a_j) = -hMg(M)$ where $h > 1$ and g is the function used in Definition 10 to define the cost function for clusters. In the following we will identify MDP

$\mathcal{M}_i$ with vertex $i \in V$ and this way show that the optimal clustering for this corresponds to a maximal clique under the mapping $i \to v_i$ and $A \to V_i$.

By construction, $\pi_i^*(s) = a_i$, $V_i^* = 0$, $V_i^{\pi_j^*} = V_i^{a_j} = 0$ iff $(i,j) \in E$, and $V_i^{a_j} = -hMg(M)$ otherwise. Hence, by the definition in (5),

$$d_V(\mathcal{M}_i, \mathcal{M}_j) = \begin{cases} 0 & \text{iff } (i,j) \in E \\ hMg(M) & \text{otherwise} \end{cases} \tag{35}$$

Now recall that $cost_{2m} \triangleq g(c) + \bar{\epsilon}_m$. Let an optimal clustering be $\mathbf{A}^*$ and let $A(\mathcal{M}_i)$ denote the cluster in $\mathbf{A}^*$ that $\mathcal{M}_i$ belongs to. We now show that if $\mathcal{M}_{i_1}, \mathcal{M}_{i_1}, \cdots, \mathcal{M}_{i_l} \in A \in \mathbf{A}^*$, then $i_1, i_2, \cdots, i_l$ form a clique in G. In other words, we show that, if $A(\mathcal{M}_i) = A(\mathcal{M}_j)$ then $(i,j) \in E$, or equivalently $(i,j) \notin E$ then $A(\mathcal{M}_i) \neq A(\mathcal{M}_j)$. By way of contradiction, assume that $(i,j) \notin E$ but $A(\mathcal{M}_i) = A(\mathcal{M}_j)$. Since i, j do not have an edge between them, by (35) the diameter of $\mathbf{A}^*$ is at least $d_V(\mathcal{M}_i, \mathcal{M}_j) = hMg(M)/M = hg(M)$. Which in turn implies that $cost_{2m}(\mathbf{A}^*) = g(|\mathbf{A}^*|) + hg(M)$. Now consider the clustering $\mathbf{A}'$ obtained by putting each MDP $\mathcal{M}_i$ in its own cluster. This clustering has cost $g(M) + 0 < g(|\mathbf{A}^*|) + hg(M)$ – contradicting the optimality of $\mathbf{A}^*$. Hence, the clusters of $\mathbf{A}^*$ has cost $g(|\mathbf{A}^*|)$ and corresponds to a collection of cliques that partition V – denote this collection of cliques by J^*.

Now note that each collection of cliques $V_1, V_2, \cdots V_j$ that partition V correspond to a clustering $\mathbf{A}$ such that $\mathcal{M}_i, \mathcal{M}_j \in A$ iff $(i,j) \in V_l$ for some l; in this case $j = |\mathbf{A}|$. Now assume that there is a clique I such that $|I| < |J^*|$ and let the corresponding clustering be $\mathbf{A}_I$. Then we show that $cost_{2m}(\mathbf{A}_I) < cost_{2m}(\mathbf{A}^*)$, resulting in a contradiction. To see this note that each $\mathcal{M}_i, \mathcal{M}_j \in A \in \mathbf{A}_I$ then $d_V(\mathcal{M}_i, \mathcal{M}_j) = 0$ by (35). Hence the diameter of $\mathbf{A}_I$ = 0. So the cost of $\mathbf{A}_I$ is $g(|\mathbf{A}_I|) + 0 < g(|\mathbf{A}^*|)$ since by definition of $\mathbf{A}_I$, $|\mathbf{A}_I| < |\mathbf{A}^*|$.

Because of the contradiction, J^* is indeed a minimum clique cover, showing that the problem of minimum clique cover can be reduced to the problem of finding the optimal clustering. To complete the proof, we need to show that this reduction takes polynomial time. The only cost in computing a $\mathcal{M}_i$ is setting the reward function, which takes time $C|V|$ for some constant C. ■

Appendix C. Computations

Here we present the computation of the ratio $\bar{\phi}(\lambda', \mathbf{A}')/\bar{\phi}(\lambda, \mathbf{A})$ defined using (20) and constructed using $\bar{\phi}_Y^M$ defined in Section 5.5. For this section, we set $|\mathbf{A}| = N$. We have four cases to consider.

Case 1: With probability $\alpha\alpha'$, λ' increased and $\mathbf{A}' = \mathbf{A}$. In this case, we have $\phi[(\lambda, \mathbf{A}), (\lambda', \mathbf{A})] = \alpha\alpha'$, $\phi[(\lambda', \mathbf{A}), (\lambda, \mathbf{A})] = \alpha(1-\alpha')$ and $\frac{\phi(\lambda', \mathbf{A}')}{\phi(\lambda, \mathbf{A})} = (1-\alpha')/\alpha'$.

Case 2: With probability $\alpha(1-\alpha')$, λ' decreased and $\mathbf{A}' = \mathbf{A}$. In this case we have $\phi[(\lambda, \mathbf{A}), (\lambda', \mathbf{A})] = \alpha(1-\alpha')$, $\phi[(\lambda', \mathbf{A}), (\lambda, \mathbf{A})] = \alpha\alpha'$, and $\frac{\phi(\lambda', \mathbf{A}')}{\phi(\lambda, \mathbf{A})} = \alpha'/(1-\alpha')$.

Case 3: With probability $1-\alpha-\beta$, $\lambda' = \lambda$ and $\mathbf{A} = \mathbf{A}'$. $\phi[(\lambda, \mathbf{A}), (\lambda', \mathbf{A})] = \phi[(\lambda', \mathbf{A}), (\lambda, \mathbf{A})] = 1-\alpha-\beta$ and $\frac{\phi(\lambda', \mathbf{A}')}{\phi(\lambda, \mathbf{A})} = 1$.

Case 4: With probability $\beta\beta'$, $\lambda' = \lambda$ and is rearranged. Now the probability of moving k_i points from A_i to A_j is,

$$P(A_i, A_j; k_i) = N^{-2} PE(k_i; |A_i|, \theta_1) \binom{|A_i|}{k_i}^{-1}$$

The reverse probability now depends on what actually has been moved. We have 4 subcases:

Case 4.1: If k_i points are moved between clusters A_i and A_j from clustering , with $0 < k_i < |A_i|$:

$$P(A_j, A_i; k_i) = N^{-2} PE(k_i; |A_j| + k_i, \theta_1) \binom{|A_j| + k_i}{k_i}^{-1}$$

Now, we have that $\phi[(\lambda, \mathbf{A}), (\lambda, \mathbf{A}')] = \beta\beta' P(A_i, A_j; k_i)$ and $\phi[(\lambda, \mathbf{A}'), (\lambda, \mathbf{A})] = \beta\beta' P(A_j, A_i; k_i)$, so that, we have:

$$\frac{\phi(\lambda', \mathbf{A}')}{\phi(\lambda, \mathbf{A})} = \frac{PE(k_i; |A_j| + k_i, \theta_1) \binom{|A_i|}{k_i}}{PE(k_i; |A_i|, \theta_1) \binom{|A_j|+k_i}{k_i}}$$

Case 4.2: If k_i points are moved from cluster A_i to a new cluster $A_{|A|+1}$, with $0 < k_i < |A_i|$:

$$P(A_{N+1}, A_i; k_i) = (N+1)^{-2} PE(k_i; k_i, \theta_1)$$

note that $|A_{N+1}| = k_i$. Now, we have that $\phi[(\lambda, \mathbf{A}), (\lambda, \mathbf{A}')] = \beta\beta' P(A_i, A_{N+1}; k_i)$ and $\phi[(\lambda, \mathbf{A}'), (\lambda, \mathbf{A})] = \beta\beta' P(A_{N+1}, A_i; k_i)$. So the desired ratio is:

$$\frac{\phi(\lambda', \mathbf{A}')}{\phi(\lambda, \mathbf{A})} = \frac{N^2 PE(k_i; k_i, \theta_1) \binom{|A_i|}{k_i}}{(N+1)^2 PE(k_i; |A_i|, \theta_1)}$$

Case 4.3: If $|A_i|$ points are moved from cluster A_i to existing cluster A_j, now we have one less cluster so that,

$$P(A_j, A_i : |A_i|) = (N-1)^{-2} PE(|A_i|; |A_j| + |A_i|, \theta_1) \binom{|A_i| + |A_j|}{|A_i|}^{-1}$$

The ϕ values are: $\phi[(\lambda, \mathbf{A}), (\lambda, \mathbf{A}')] = \beta\beta' P(A_i \xrightarrow{|A_i|} A_j)$ and $\phi[(\lambda, \mathbf{A}'), (\lambda, \mathbf{A})] = \beta\beta' P(A_j \xrightarrow{|A_i|} A_i)$. Together, this gives us the ratio:

$$\frac{\phi(\lambda', \mathbf{A}')}{\phi(\lambda, \mathbf{A})} = \frac{N^2 PE(|A_i|; |A_j| + |A_i|, \theta_1)}{(N-1)^2 PE(|A_i|; |A_i|, \theta_1) \binom{|A_i|+|A_j|}{|A_i|}}$$

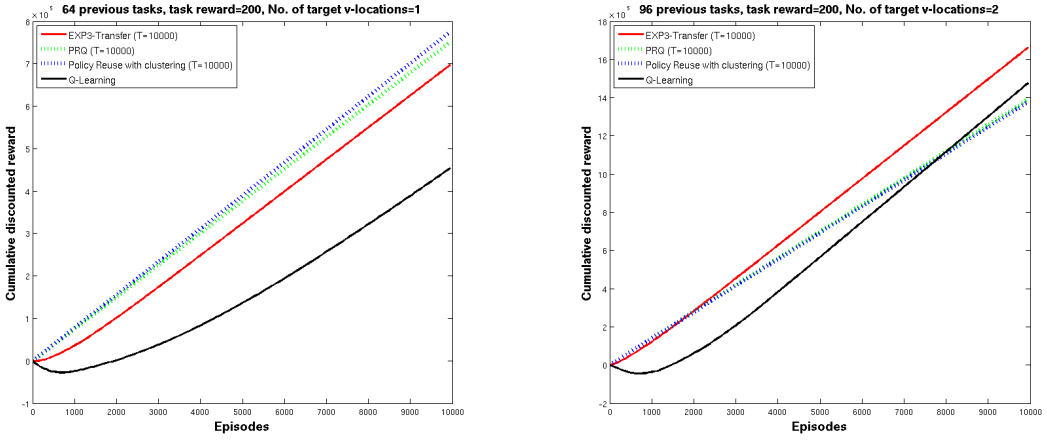
Case 4.4: If $|A_i|$ points are moved from cluster A_i to a new cluster $A_{|A|+1}$:

$$P(A_{N+1}, A_i : |A_i|) = N^{-2} PE(|A_i|; |A_i|, \theta_1)$$

The clustering $\mathbf{A}$ does not change in this case and the ϕ values are: $\phi[(\lambda, \mathbf{A}), (\lambda, \mathbf{A}')] = \beta\beta' P(A_i, A_{N+1}; |A_i|)$, $\phi[(\lambda, \mathbf{A}'), (\lambda, \mathbf{A})] = \beta\beta' P(A_{N+1}, A_i; |A_i|)$, which gives us $\frac{\phi(\lambda', \mathbf{A}')}{\phi(\lambda, \mathbf{A})} = 1$.

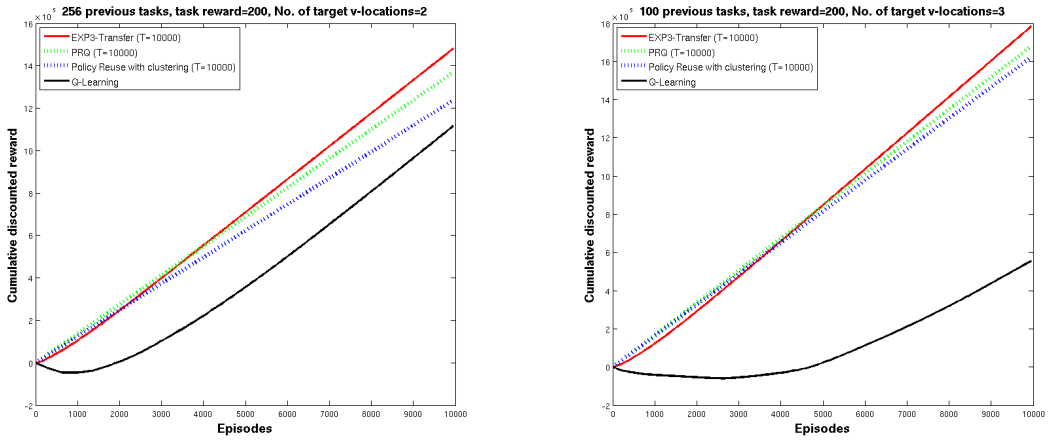
Appendix D. Surveillance Domain Experiments: Algorithm Comparisons

In this section we give detailed cumulative reward curves for the 4 algorithms: E3T with clustering, Policy-Reuse with clustering and Policy-Reuse with clustering. The results are given in Figures 12 to 14. The results more or less show what the summary graphs showed. In particular, when the number of previous tasks and the complexity of task is low, Policy-Reuse is better than our algorithm. However, as the complexity keeps increasing, our algorithm begins to dominate both versions of Policy-Reuse, showing that clustering is beneficial.



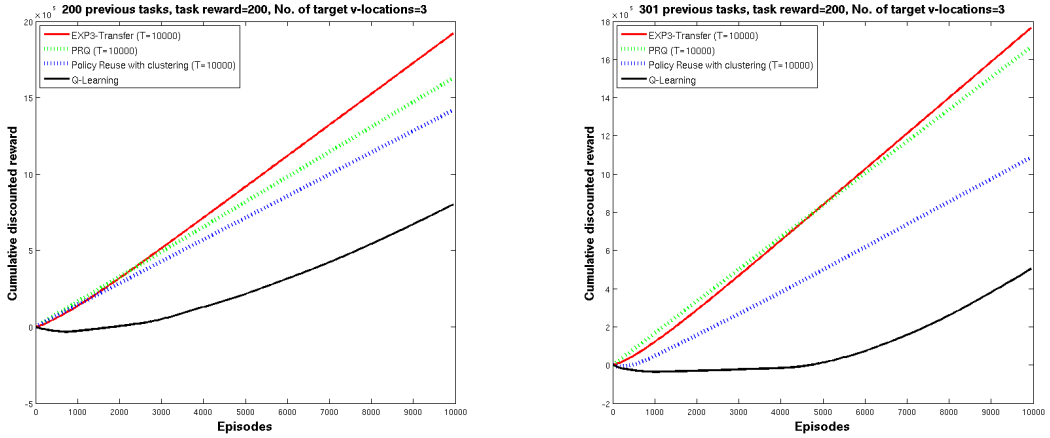
(a) (b)

Figure 12: ALGORITHM COMPARISONS. These figures compares the performance of EXP-3-Transfer with clustering, Policy-Reuse with and without clustering, and Q-learning for various settings of the task (see the figure title). These are the detailed plots of the summary results presented in Section 7.2.1.



(a) (b)

Figure 13: ALGORITHM COMPARISONS CONTINUED. These figures compares the performance of EXP-3-Transfer with clustering, Policy-Reuse with and without clustering, and Q-learning for various settings of the task (see the figure title). These are the detailed plots of the summary results presented in Section 7.2.1.

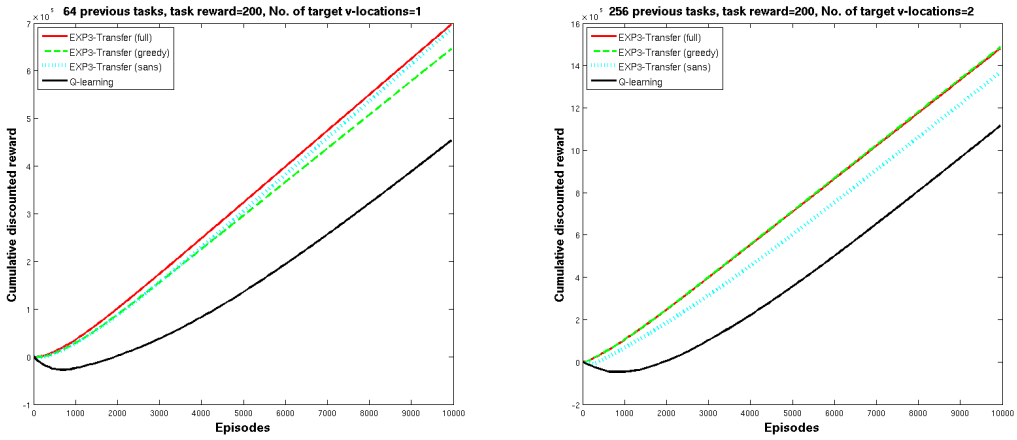


(a) (b)

Figure 14: ALGORITHM COMPARISONS CONTINUED. These figures compares the performance of EXP-3-Transfer with clustering, Policy-Reuse with and without clustering, and Q-learning for various settings of the task (see the figure title). These are the detailed plots of the summary results presented in Section 7.2.1.

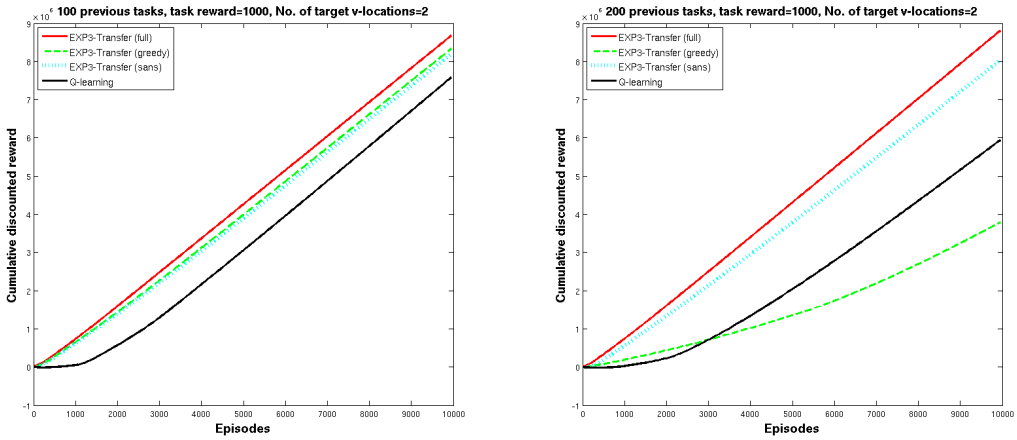
D.1 Surveillance Domain: Clustering Comparisons

In this section in figures 15 to 17 we present the learning curves summarized in figure 8. The general trend follows what was observed in Section 7.2.2.



(a) (b)

Figure 15: CLUSTERING COMPARISIONS EXTENDED RESULTS. These figures show the results that are summarized in Figure 8. The title of the graphs describe the experiment setup.



(a) (b)

Figure 16: CLUSTERING COMPARISIONS EXTENDED RESULTS CONTINUED. These figures show the results that are summarized in Figure 8. The title of the graphs describe the experiment setup.

D.2 Time Comparisons

Figures 18 to 20 gives the time comparison results for transfer problems not described in Figure 11.

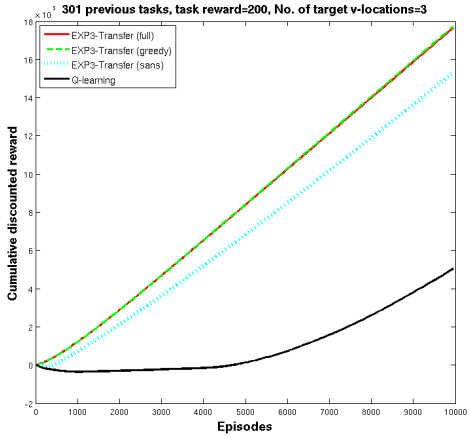
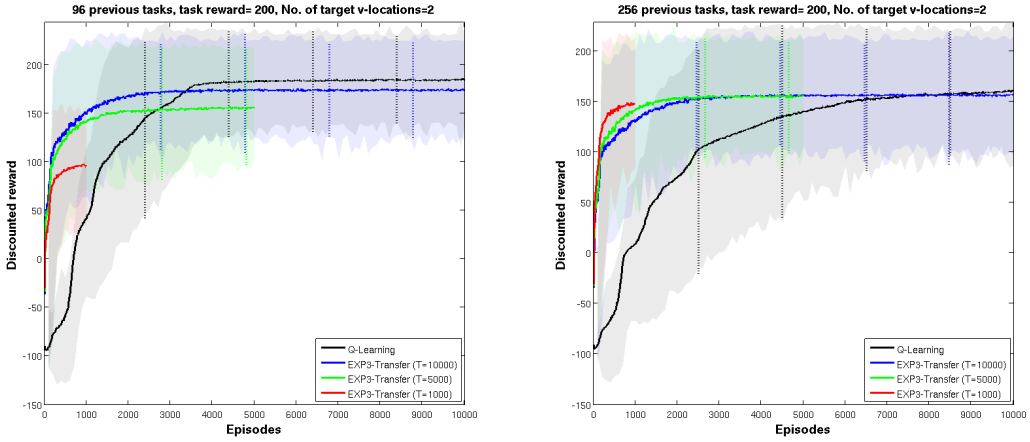
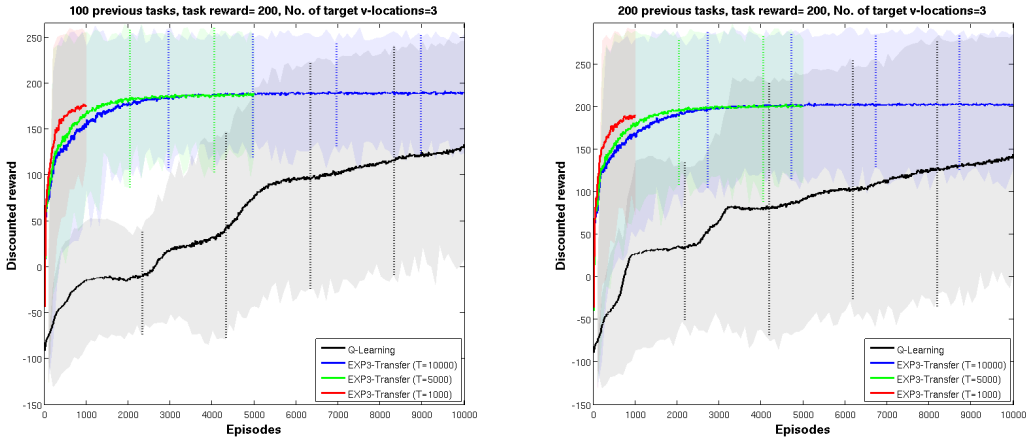


Figure 17: CLUSTERING COMPARISIONS EXTENDED RESULTS CONTINUED. These figures show the results that are summarized in Figure 8. The title of the graphs describe the experiment setup.



(a) (b)

Figure 18: TIME COMPARISIONS EXTENDED RESULTS. These figures show time comparsion results for transfer tasks in addition to Figure 11. The title of the graphs show the experiment setup. The shaded areas give the standard deviation for the learning curves.



(a) (b)

Figure 19: TIME COMPARISIONS EXTENDED CONTINUED. These figures show time comparsion results for transfer tasks in addition to Figure 11. The title of the graphs show the experiment setup. The shaded areas give the standard deviation for the learning curves.

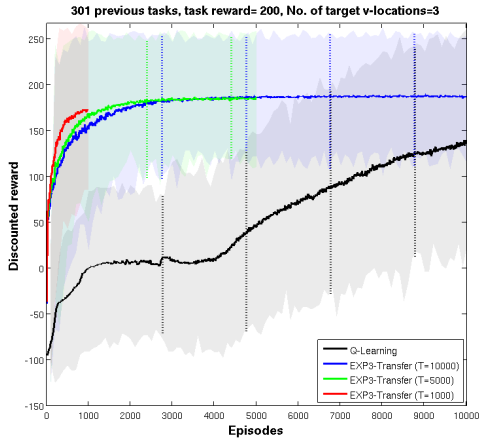


Figure 20: TIME COMPARISIONS EXTENDED CONTINUED. These figures show time comparsion results for transfer tasks in addition to Figure 11. The title of the graphs show the experiment setup. The shaded areas give the standard deviation for the learning curves.

References

Bo An, David Kempe, Christopher Kiekintveld, Eric Shieh, Satinder Singh, Milind Tambe, and Yevgeniy Vorobeychik. Security games with limited surveillance. In *Proceedings of the 26th Conference on Artificial Intelligence*, 2012.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002a.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32:48–77, 2002b.

Mohammad Gheshlaghi Azar, Alessandro Lazaric, and Emma Brunskill. Regret bounds for reinforcement learning with policy advice. In *Proceedings of* 23^{rd} *European Conference on Machine learning (ECML)*, 2013.

Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, March 2000.

James L. Carroll and Kevin Seppi. Task similarity measures for transfer in reinforcement learning task libraries. In *Proceedings of 2005 IEEE International Joint Conference on Neural Networks*, 2005.

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

Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction Learning and Games*. Cambridge University Press, 2006.

Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure Inequalities for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

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.

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

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2004.

Eliseo Ferrante, Alessandro Lazaric, and Marcello Restelli. Transfer of task representation in reinforcement learning using policy-based protovalue functions. In *Proceedings of the* 7^th *International Confernce on Autonomous Agent And Multiagent Systems*, 2008.

Richard Karp. Reducibility among combinatorial problems. In *Proceedings of a Symposium on the Complexity of Computer Computations*, 1972.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220: 671–680, 1983.

Andrey Kolmogorov and Sergei V. Fomin. *Introductory Real Analysis*. Dover Publications, 1970.

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.

Allesandro Lazaric and Marcello Restilli. Transferring from multiple mdps. In *Proceedings of the Neural Information Processing Systems Conference*, 2011.

David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

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

Sridhar Mahadevan. Proto-value functions: Developmental reinforcement learning. In *Proceedings of International Conference on Machine Learning*, 2005.

Tom M. Mitchell and Sebastian Thrun. Explanation-based neural network learning for robot control. pages 287–294, 1993.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

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

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

Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, Berling, 2005.

Jonathan Sorg and Satinder Singh. Transfer via soft homomorphisms. In *Proceedings of the* 8^th *International Conference on Autonomous Agents and Multiagent Systems*, 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.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

Matthew Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

Sebastian Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Kluwer Academic Publishers, Boston, MA, 1996.

Sebastian Thrun and Tom Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15:25–46, 1995.

Vladimir Vovk. Aggregating strategies. In *Proceedings of the* 3^{rd} *Internation Conference on Computational Learning Theory*, 1990.

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