Efficient Solution of Markov Decision Problems with Multiscale Representations

Jake Bouvrie[1] and Mauro Maggioni[2]

Abstract— **Many problems in sequential decision making and stochastic control naturally enjoy strong multiscale structure: sub-tasks are often assembled together to accomplish complex goals. However, systematically inferring and leveraging hierarchical structure has remained a longstanding challenge. We describe a fast multiscale procedure for repeatedly compressing or homogenizing Markov decision processes (MDPs), wherein a hierarchy of sub-problems at different scales is automatically determined. Coarsened MDPs are themselves independent, deterministic MDPs, and may be solved using any method. The multiscale representation delivered by the algorithm decouples sub-tasks from each other and improves conditioning. These advantages lead to potentially significant computational savings when solving a problem, as well as immediate transfer learning opportunities across related tasks.**

I. Introduction

Identifying and leveraging hierarchical structure has been a key, longstanding challenge for sequential decision making and planning research [1]–[3]. Hierarchical structure generally suggests a decomposition of a complex problem into smaller, simpler sub-tasks, which may be, ideally, considered independently [4]. One or more layers of abstraction may also provide a broad mechanism for reusing or transferring commonly occurring sub-tasks among related problems [5]–[8]. These themes are simply restatements of the divide-and-conquer principle: it is usually dramatically cheaper to solve a collection of small problems than a single big problem.

This paper considers the inference and use of hierarchical structure – *multiscale* structure in particular – in the context of discrete-time Markov decision problems. Fundamentally, inferring multiscale decompositions, learning, and planning across scales are intimately related concepts, and we have sought to couple these elements tightly within a unifying framework. Our main contribution is a multiscale procedure for partitioning and then repeatedly *compressing* or *homogenizing* Markov decision processes (MDPs). The result is a multiscale representation decomposing the original problem into a hierarchy of distinct sub-problems at multiple scales, each of which may be solved efficiently and independently of the others. Solutions to these sub-problems may also be transferred among related problems, giving a systematic means to approach transfer learning at multiple scales in planning and reinforcement learning domains.

[1]Department of Mathematics, Duke University, Durham, NC 27708, USA jvb@math.duke.edu.

[2]Departments of Mathematics and Computer Science, Duke University, Durham, NC 27708, USA mauro@math.duke.edu.

Research supported by DARPA FA8650-11-1-7150 SUB#7-3130298, MSEE FA8650-11-1-7150; Washington State U. SUB#113054 G002745; NSF IIS-08-03293, DMS-08-47388; ONR N00014-07-1-0625.

The homogenization we propose is consistent in that a compressed MDP is again another independent, deterministic MDP, and the statespace of the compressed MDP is a (small) subset of the original problem's statespace. Moreover, each coarse MDP in a multiscale hierarchy is consistent in the mean with the underlying fine scale problem. The compressed representation coarsely summarizes a problem's statespace, reward structure and Markov transition dynamics, and may be computed either analytically or by Monte-Carlo simulations.

Given a hierarchy of successively coarsened representations, an MDP may be solved efficiently. We describe a family of multiscale solution algorithms which realize computational savings in two ways: (1) *Localization*: computation involves small, decoupled sub-problems; and (2) *Conditioning*: sub-problems are comparatively well-conditioned, and obey a form of global consistency with each other through coarser scales. The key idea behind these algorithms is that sub-problems at a given scale decouple conditional on a solution at the next coarser scale, but must contribute constructively towards solving the overarching problem through the coarse solution; interleaved updates to solutions at pairs of fine and coarse scales are repeatedly applied until convergence. We present one particular algorithm, a localized variant of modified asynchronous policy iteration, that can achieve a cost of $\mathcal{O}(n \log n)$ per iteration, if there are n states in the original problem.

This paper describes preliminary results, and due to space limitations proofs and many details are omitted. A comprehensive manuscript, including a detailed discussion, proofs, experimental validation, and a wide-ranging literature review, is forthcoming from the authors.

II. Background

A. Markov Decision Processes and Stochastic Policies

Formally, a Markov decision process (MDP) (see e.g. [9], [10]) is a sequential decision problem defined by a tuple (S, A, P, R, Γ) consisting of a state space S, an action (or "control") set A, and for $s, s' \in S, a \in A$, a transition probability tensor $P(s, a, s')$, reward function $R(s, a, s')$ and collection of discount factors $\Gamma(s, a, s') \in (0, 1)$. We will assume that S, A are finite sets, and that R is bounded. The probability $P(s, a, s')$ refers to the probability that we transition to s' upon taking action a in s, while $R(s, a, s')$ is the reward collected in the event we transition from s to s' *after* taking action a in s.

1) Stochastic Policies: Let $\mathcal{P}(A)$ denote the set of all discrete probability distributions on A. A *stationary stochastic policy* (simply a *policy*, from now on) $\pi : S \to \mathcal{P}(A)$ is a function mapping states into distributions over the actions. A policy π may be thought of as a non-negative function on $S \times A$ satisfying $\sum_{a \in A} \pi(s,a) = 1$ for each $s \in S$, where $\pi(s,a)$ denotes the probability that we take action a in state s. We will often write $\pi(s)$ when referring to the *distribution* on actions associated to the (deterministic) state $s \in S$, so that $a \sim \pi(s)$ denotes the A-valued random variable a having law $\pi(s)$.

We may compute the policy-specific Markov transition matrix and reward function by averaging over the actions according to π:

$$P^{\pi}(s,s') = \mathop{\mathbb{E}}_{a \sim \pi(s)} [P(s,a,s')] = \sum_{a \in A} P(s,a,s')\pi(s,a) \quad (1)$$

and $R^{\pi}(s,s') = \mathbb{E}_{a \sim \pi(s)}[R(s,a,s')]$.

Deterministic policies can be recovered by placing unit masses on the desired actions[1]. Working with stochastic policies will allow convex combinations of policies. Finally, we will often make use of the uniform random or *diffusion* policy, denoted π^u, which always takes an action drawn randomly according to the uniform distribution on the feasible actions.

2) Value Functions: Given a policy, we may define a value function $V^{\pi} : S \to \mathbb{R}$ assigning to each state s the expected sum of discounted rewards collected over an infinite horizon by running the policy π starting in s:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}\left[R(s_0,a_1,s_1) \mid s_0 = s\right] + \\ &\mathbb{E}\left[\sum_{t=1}^{\infty} \prod_{\tau=0}^{t-1} \Gamma(s_\tau, a_{\tau+1}, s_{\tau+1}) R(s_t, a_{t+1}, s_{t+1}) \;\middle|\; s_0 = s\right] \end{aligned} \quad (2)$$

where the sequence of random variables $(s_i)_{i=1}^{\infty}$ is a Markov chain with transition probability matrix P^{π}. The expectation is taken over all sequences of state-action pairs $\{(s_t, a_t)\}_{t \geq 1}$, where a_t is an A-valued random variable representing the action which brings the Markov chain to state s_t from s_{t-1}: if s_{t-1} is observed, then $a_t \sim \pi(s_{t-1})$. The optimal value function V^* is defined as $V^*(s) = \sup_{\pi \in \Pi} V^{\pi}(s)$ for all $s \in S$, with Π the set of stationary, Markov policies, and the corresponding optimal policy π^* is any policy achieving the optimal value function. Under the assumptions we have imposed here, a deterministic optimal policy exists whenever an optimal policy (possibly stochastic) exists [10, Sec. 1.1.4]. We will make use of stochastic policies primarily to regularize a class of MDP solution algorithms.

The process of computing V^{π} given π is known as *value determination* (see e.g. [11] for a discussion regarding potential theory and Markov chains). Following the usual approach, a linear system describing V^{π} is obtained by conditioning on the first transition in (2) and applying the Markov property: for $s \in S$,

$$\begin{aligned} V^{\pi}(s) = \sum_{s',a} & P(s,a,s')\pi(s,a) \\ & \times \left[R(s,a,s') + \Gamma(s,a,s')V^{\pi}(s')\right]. \end{aligned} \quad (3)$$

[1]We will allow the set of actions available in state s to be limited to a nonempty state-dependent subset $\mathcal{A}(s) \subseteq A$ of *feasible* actions, but do not explicitly keep track of the sets $\mathcal{A}(s)$ to avoid cluttering the notation.

B. Notation

For $S' \subseteq S$, we define the *restriction* of $P : S \times A \times S \to \mathbb{R}_+$ to S' to be the transition tensor $P_{S'}$ defined by

$$P_{S'}(s,a,s') = \begin{cases} P(s,a,s'), & \text{if } s,s' \in S', s \neq s' \\ P(s,a,s) + \sum_{s'' \notin S'} P(s,a,s''), & s = s' \end{cases}. \quad (4)$$

The rewards associated to transitions between states in the subset S' are unmodified: $R_{S'}(s,a,s') = R(s,a,s')$, for all (s,a,s') such that $s,s' \in S', a \in A$. We will refer to this operation as *truncation*, to distinguish it from restriction as defined by (4). The sub-tensor $\Gamma_{S'}$ is similarly defined from Γ. Note that, by definition, $P_{S'}, R_{S'}, \Gamma_{S'}$ do not include tuples which start from a state s in the cluster but which end at a state s' outside of the cluster.

The restriction operation introduced above does *not* commute with taking expectations with respect to a policy. The matrix $P^{\pi}_{S'}$ will be defined by *first* restricting to S' by Equation (4), and *then* averaging $P_{S'}$ with respect to π as in Equation (1).

III. Multiscale Markov Decision Processes

The high-level procedure for efficiently solving a problem with a multiscale MDP hierarchy consists of the following steps, to be described individually in more detail below:

Step 1 **Partition** the statespace into subsets of states ("clusters") connected via "bottleneck" states.

Step 2 Given the decomposition into clusters by bottlenecks, **compress** or **homogenize** the MDP into another, smaller and *coarser* MDP, whose state space is the set of bottlenecks, and whose actions are given by following certain policies in clusters connecting bottlenecks ("sub-tasks").
Repeat the steps above with the compressed MDP as input, until the desired number of compression steps, obtaining a hierarchy of MDPs.

Step 3 **Solve the hierarchy of MDPs** from the top-down (coarse to fine) by pushing solutions of coarse MDPs to successively finer MDPs, down to the finest scale.

We say that the procedure above compresses or homogenizes, in a multiscale fashion, a given MDP. The construction is perfectly recursive, in the sense that the same steps and algorithms are used to proceed from one scale to the next coarser scale. It also enjoys various notions of consistency, as mentioned in the introduction. Actions at coarser scales are typically, as one may expect, complex, "higher-level" actions, and the above procedure may be thought of as producing different levels of "abstraction" of the original problem. While automating the process of hierarchically decomposing, in a novel fashion, large complex MDPs, the framework we propose may also yield significant computational savings. The details are discussed in Section VI-B. Finally, the framework facilitates knowledge transfer between related MDPs. Sub-tasks may be transferred anywhere within the hierarchies for a pair of problems, instead of mapping entire problems. We provide a brief overview of transfer ideas in Section VII.

The next three Sections are devoted to providing an overview of the steps (1) – (3) above. Details, algorithmic and theoretical, are omitted due to space constraints but may be found in a longer forthcoming paper.

IV. *Step 1:* Bottleneck Detection and Statespace Partitioning

The first step of the algorithm involves partitioning the MDP's statespace S by identifying a set $\mathcal{B} \subseteq S$ of bottlenecks, that induce a partitioning of $S \setminus \mathcal{B}$ into a family C of connected components. Typically $\mathcal{B}$ depends on a policy π, and when we want to emphasize this dependency, we will write $\mathcal{B}^\pi$. We always assume that $\mathcal{B}^\pi$ *includes all terminal states of* P^π. The partitioning of $\{S \setminus \mathcal{B}^\pi\}$ induced by the bottlenecks is the set of equivalence classes $S/\sim$, under the relation $s_i \sim s_j$ if $s_i, s_j \notin \mathcal{B}^\pi$ and there is a path from s_i to s_j not passing through any $\mathsf{b} \in \mathcal{B}^\pi$. Clearly these equivalence classes yield a partitioning of $S \setminus \mathcal{B}^\pi$. The term *cluster* will refer to an equivalence class plus any bottleneck states connected to states in the class: if $[s] := \{s' \mid s \sim s'\}$ is an equivalence class,

$$\mathsf{c}([s]) := [s] \cup \big\{\mathsf{b} \in \mathcal{B}^\pi \mid P^\pi(s', \mathsf{b}) > 0 \text{ or } P^\pi(\mathsf{b}, s') > 0 \text{ for some } s' \in [s]\big\}.$$

The set of clusters is denoted by C. If $\mathsf{c} = \mathsf{c}([s])$, $[s]$ will be referred to as the cluster's *interior* $\mathring{\mathsf{c}}$, and the bottlenecks attached to $[s]$ will be referred to as the cluster's *boundary*, denoted by $\partial\mathsf{c}$.

To each cluster c, and policy π (defined on at least c), we associate the Markov process with transition matrix P^π_{c}, defined according to Section II-B.

We also assume that a set of designated policies $\boldsymbol{\pi}_{\mathsf{c}}$ is provided for each cluster c. For example $\boldsymbol{\pi}_{\mathsf{c}}$ may be the singleton consisting of the diffusion policy in c. Or $\boldsymbol{\pi}_{\mathsf{c}}$ could be the set of optimal policies in c for the family of MDPs, parametrized by $s' \in \partial\mathsf{c}$ with reward equal to the original rewards plus an additional reward when s' is reached. Finally, we say that $\partial\mathsf{c}$ is π-*reachable*, for a policy π, if the set $\partial\mathsf{c}$ can be reached in a finite number of steps of P^π_{c}, starting from any initial state $s \in \mathsf{c}$.

A. Algorithms for bottleneck detection

Many algorithms may be used to partition the statespace, and detect bottlenecks (e.g. [12]–[14]). We consider one such possibility, a simple hierarchical spectral clustering algorithm, for illustrative purposes. Given a policy π, we can construct a weighted statespace graph G with vertices corresponding to states, and edge weights given by P^π. A policy that allows thorough exploration, such as the diffusion policy π^u, can be chosen to define the weighted statespace graph.

The hierarchical spectral clustering algorithm we will consider recursively splits the statespace graph into pieces by looking for low-conductance cuts. The spectrum of the symmetrized Laplacian for directed graphs [15] is used to determine the graph cuts at each step. The sequence of cuts establishes a partitioning of the statespace, and bottleneck states are states with edges that are severed by any of the cuts.

Algorithm 1 Recursive spectral partitioning.

```
1) Restrict $P^\pi$ to non-absorbing states.
2) Set $P^\pi_{\text{tel}} = (1-\eta)P^\pi + \eta n^{-1}\mathbf{1}\mathbf{1}^\top$, for some small, positive $\eta$.
3) Find the eigenvector (invariant distribution) $\mu$ satisfying $(P^\pi_{\text{tel}})^\top \mu = \mu$.
4) Let $\Phi = \operatorname{diag}(\mu)$, and compute the symmetrized Laplacian for directed graphs [15]
       $\mathcal{L} = I - \frac{1}{2}\big(\Phi^{1/2}P^\pi_{\text{tel}}\Phi^{-1/2} + \Phi^{-1/2}(P^\pi_{\text{tel}})^\top\Phi^{1/2}\big).$
5) Compute the $K$ eigenvectors of $\mathcal{L}$ corresponding to the $K$ smallest non-trivial eigenvalues $\lambda_1 < \cdots < \lambda_K$.
6) For each eigenvector $\Psi^{(i)}, i = 1, \ldots, K$, define a set of cuts by sweeping over thresholds ranging from the smallest entry of $\Psi^{(i)}$ to the largest. The points for which $\Psi^{(i)}$ are above/below the given threshold defines the states $Z, Z^c \subset S$ on either side of the cut.
7) Choose the cut $Z^*$ with minimum conductance
       $\varphi(Z) = \frac{\sum_{i\in Z}\sum_{j\in Z^c} P^\pi_{ij}}{\operatorname{vol}(Z) \wedge \operatorname{vol}(Z^c)},$
   where $\operatorname{vol}(Z) = \sum_{i\in Z}\sum_{j\in S} P^\pi_{ij}$.
8) Identify bottleneck states as the states on one side of the edges in $P^\pi$ severed by the cut, choosing the side which gives the smallest bottleneck set.
9) Store the partition of the statespace given by the cut.
10) Unless stopping criteria is met, run the algorithm again on each of the two subgraphs resulting from the cut.
```

Algorithm 1 describes the process. Localized variants of this algorithm may also be pursued, and one can also consider model-free versions that only have access to a "black-box" computing the results of running a process (e.g. truncated random walk, evolving sets process).

A recursive application of Algorithm 1 produces a set of bottlenecks $\mathcal{B}^\pi$. Each bottleneck and partition discovered by the clustering algorithm is associated with a spatial scale determined by the recursion depth. The finest scale consists of the finest partition and all bottlenecks. Due to the addition of a teleport matrix in Algorithm 1 (Step 2), the equivalence classes are strongly connected components of the graph induced by P^π_{tel} and are guaranteed to partition $\{S \setminus \mathcal{B}^\pi\}$.

Because graph weights are determined by P^π in this algorithm, which bottlenecks will be identified generally depends on the policy π. In this sense there are two types of "bottlenecks": problem bottlenecks and geometric bottlenecks. Geometric bottlenecks may be defined as interesting regions of the state space alone, as determined by a random walk exploration if π is a diffusion policy (e.g. π^u). Problem bottlenecks are regions of the state space which are interesting from a geometric standpoint *and* in light of the goal structure of the MDP. If the policy is already strongly directed according to the goals defined by the rewards, then the bottlenecks can be interpreted as choke points for a random walker in the presence of a strong potential.

V. *Step 2:* Multiscale Compression and The structure of Multiscale Markov Decision Problems

Given a set of bottlenecks $\mathcal{B}$, we can *compress* (or *homogenize*, or coarsen) an MDP into another MDP with statespace $\mathcal{B}$. The coarse MDP can be thought of as a low-resolution version of the original problem, where transitions *between* clusters are the events of interest, rather than what occurs within each cluster. As such, coarse MDPs may be vastly simpler: the size of the coarse statespace is on the order of the number of clusters, which may be small relative to the size of the original statespace. Indeed, clusters may be generally thought of as geometric properties of a *problem*, and are constrained by the inherent complexity of the problem, rather than the choice of statespace representation, discretization or sampling.

A solution to the coarse MDP may be viewed as a coarse solution to the original fine scale problem. An optimal coarse policy describes how to solve the original problem by specifying which sub-tasks to carry out and in which order. As we will describe in Section VI, a coarse value function provides an efficient means to obtain a fine scale value function and its associated policy. Coarse MDPs and their solutions also provide a framework for systematic transfer learning; these ideas are briefly discussed in Section VII.

A homogenized, coarse scale MDP will be denoted by the tuple $(\widetilde{S}, \widetilde{A}, \widetilde{P}, \widetilde{R}, \widetilde{\Gamma})$. We first give a brief description of the primary ingredients needed to define a coarse MDP, with a more detailed discussion to follow.

- **Statespace** $\widetilde{S}$: The coarse scale statespace $\widetilde{S}$ is the set of bottleneck states $\mathcal{B}$ for the fine scale, obtained by clustering the fine scale statespace graph, for example with the methods described in Section IV.
- **Action set** $\widetilde{A}$: A coarse action invoked from $\mathsf{b} \in \widetilde{S} = \mathcal{B}$ consists of executing a given fine scale policy $\pi_{\mathsf{c}} \in \boldsymbol{\pi}_{\mathsf{c}}$ within the fine scale cluster c, starting from $\mathsf{b} \in \partial\mathsf{c}$ (at a time that we may reset to 0), until the first positive time at which a bottleneck state in $\partial\mathsf{c}$ is hit.
- **Coarse scale transition probabilities** $\widetilde{P}(s, a, s')$: If $a \in \widetilde{A}$ is an action executing the policy $\pi_{\mathsf{c}} \in \boldsymbol{\pi}_{\mathsf{c}}$, then $\widetilde{P}(s, a, s')$ is defined as the probability that the Markov chain $P_{\mathsf{c}}^{\pi_{\mathsf{c}}}$ started from $s \in \widetilde{S}$, hits $s' \in \widetilde{S}$ before hitting any other bottleneck. In particular, $\widetilde{P}(s, a, s')$ may be nonzero only when $s, s' \in \partial\mathsf{c}$ for some $\mathsf{c} \in \mathsf{C}$.
- **Coarse scale rewards** $\widetilde{R}(s, a, s')$: The coarse reward $\widetilde{R}(s, a, s')$ is defined to be the expected total discounted reward collected along trajectories of the Markov chain associated to action a described above, which start at $s \in \widetilde{S}$ and end by hitting $s' \in \widetilde{S}$ before hitting any other bottleneck.
- **Coarse scale discount factors** $\widetilde{\Gamma}(s, a, s')$: The coarse discount factor $\widetilde{\Gamma}(s, a, s')$ is the expected product of the discounts applied to rewards along trajectories of the Markov chain $P_{\mathsf{c}}^{\pi_{\mathsf{c}}}$ associated to a action $a \in \widetilde{A}$, starting at $s \in \widetilde{S}$ and ending at $s' \in \widetilde{S}$.

We observe, before proceeding with details, that one of the important consequences of these definitions is that *the optimal fine scale value function on the bottlenecks is, in an appropriate sense, the solution to the coarse MDP, compressed with respect to the optimal fine scale policy*. The general philosophy is to reduce a large problem to a smaller one constructed by "locally averaging" parts of the original problem.

We now consider the objects constituting the coarser MDP in more detail, and show that, perhaps surprisingly, many of them may be quickly and locally computed in parallel by solving certain linear systems that we can describe analytically. The coarsening step may also be accomplished computationally by Monte Carlo simulations, as it involves computing the relevant statistics of certain functionals of Markov processes in each of the clusters. As such, the computation is embarrassingly parallel[2].

While this brings great flexibility to the framework above, it is interesting to note that many of those computations may in fact be carried out analytically, and that eventually they reduce to the solution of multiple, small and independent (and therefore embarrassingly parallelizable) linear systems, of size comparable to the size of a cluster. These linear systems uncover the natural structure of the multiscale organization we introduce, and lead to efficient, "explicit" algorithms for the solution of Markov decision problems.

A. Assumptions

We will always assume that the fine scale policy π used to compress has been *regularized*, by blending with a small amount of the diffusion policy π^u:

$$\pi(s, \cdot) \leftarrow \lambda \pi^u(s, \cdot) + (1 - \lambda)\pi(s, \cdot), \qquad s \in S$$

for some small, positive choice of the regularization parameter λ. In particular we will assume this is the case everywhere quantities such as P^π appear below. This form of regularization addresses certain pathological situations and helps enforce π-reachability of $\mathcal{B}$ (the boundary).

B. Actions

An action $\widetilde{A}$ at $s \in \widetilde{S}$ for the compressed MDP consists of executing a policy $\pi_{\mathsf{c}} \in \boldsymbol{\pi}_{\mathsf{c}}$ for some cluster c having s on its boundary, until hitting a bottleneck state in c. The number of actions is equal to the total number of policies across clusters,$|\widetilde{A}| = \sum_{\mathsf{c} \in \mathsf{C}} |\boldsymbol{\pi}_{\mathsf{c}}|$. We now fix a cluster c and a policy $\pi_{\mathsf{c}} \in \boldsymbol{\pi}_{\mathsf{c}}$. The corresponding local Markov transition matrix is $P_{\mathsf{c}}^{\pi_{\mathsf{c}}}$, and let $R_{\mathsf{c}}^{\pi_{\mathsf{c}}}$ denote the reward structure, and $\Gamma_{\mathsf{c}}^{\pi_{\mathsf{c}}}$ denote the system of discount factors, following Section II-B. Let $((X_{\mathsf{c}}^{\pi_{\mathsf{c}}})_n)_{n \geq 0}$ denote the Markov chain with transition matrix $P_{\mathsf{c}}^{\pi_{\mathsf{c}}}$. If the coarse action is invoked in state $s \in \widetilde{S}$, then we have $X_0 = s$. The set of actions available at $s \in \widetilde{S}$ for the compressed MDP is given by

$$\widetilde{A}(s) := \bigcup_{\mathsf{c} \in \mathsf{C}: s \in \partial\mathsf{c}} \left\{ \begin{array}{c} \text{“run the MRP } (P_{\mathsf{c}}^{\pi_{\mathsf{c}}}, R_{\mathsf{c}}^{\pi_{\mathsf{c}}}, \Gamma_{\mathsf{c}}^{\pi_{\mathsf{c}}}) \text{ in } \mathsf{c} \text{ until} \\ \text{the first } n > 0 : (X_{\mathsf{c}}^{\pi_{\mathsf{c}}})_n \in \mathcal{B}\text{”} \end{array} \right\}_{\pi_{\mathsf{c}} \in \boldsymbol{\pi}_{\mathsf{c}}}.$$

[2]Moreover, it does not require a priori knowledge of the fine details of the models in each cluster, but only requires the ability to call a "black box" which simulates the prescribed process in each cluster, and computes the corresponding functional (in this sense coarsening becomes model-free).

A Markov reward process (MRP) refers to an MDP with a fixed policy and corresponding P, R, Γ restricted to the fixed policy. The actions above involve running an MRP because while the action is being executed the policy remains fixed. In general, the compressed MDP will have action and state dependent rewards and discount factors, even if the fine scale problem does not.

C. Transition Probabilities

Consider the cluster c referred to by a coarse action $a \in \widetilde{A}$. *The transition probability $\widetilde{P}(s,a,s')$ for $s, s' \in \partial\mathsf{c} \subseteq \widetilde{S}$ is defined to be the probability that a trajectory in $\mathsf{c} \subset S$ hits state s' starting from s before hitting any other state in $\mathcal{B}$ (including itself) when running the fine scale MRP restricted to c and along the policy determined by the action a.* If s is a state not in the cluster associated to a, then a is not an available control when in state s.

These probabilities may be estimated either by sampling (Monte Carlo simulations), or computed analytically. The first approach is trivially implemented; here we develop the latter, which leads to a set of linear problems to be solved, and sheds light on both the mathematical and computational structure. As the bottlenecks partition the statespace into disjoint sets, the probabilities $\widetilde{P}(s,a,s')$ can be quickly computed in each cluster separately.

Proposition 1. *Let a be the action corresponding to executing a policy π_{c} in cluster c. Then*

$$\widetilde{P}(s,a,s') = H_{s,s'}, \quad \textit{for all } s, s' \in \partial\mathsf{c},$$

where H is the minimal non-negative solution, for each $s' \in \partial\mathsf{c}$, to the linear system

$$H_{s,s'} = P_{\mathsf{c}}^{\pi_{\mathsf{c}}}(s,s') + \sum_{s'' \in \mathring{\mathsf{c}}} P_{\mathsf{c}}^{\pi_{\mathsf{c}}}(s,s'') H_{s'',s'}, \quad s \in \mathsf{c}, s' \in \partial\mathsf{c}\,.$$

When deriving the the compressed rewards and discount factors below, we will need to reference the set of all pairs of bottlenecks s, s' for which the probability of reaching s' starting from s is positive, when executing the policy π_{c} associated to a. Having defined $\widetilde{P}$, this set may be easily characterized as

$$\mathrm{supp}_a(\widetilde{P}) := \{(s,s') \in \partial\mathsf{c} \mid \widetilde{P}(s,a,s') > 0\}$$

where c is the cluster associated to the coarse action a.

D. Rewards

The rewards $\widetilde{R} = \widetilde{R}(s,a,s')$, with $s, s' \in \partial\mathsf{c}$ and $a \in \widetilde{A}$, are defined to be the expected discounted rewards collected along trajectories that start from s and hit s' before hitting any other bottleneck state in $\partial\mathsf{c}$, when running the fine-scale MRP restricted to the cluster c associated to a.

In general, rewards under different policies and/or in other clusters are calculated by repeating the process described below for different choices of $\pi \in \boldsymbol{\pi}_{\mathsf{c}}, \mathsf{c} \in \mathsf{C}$. Even if the fine scale MDP rewards do not depend on the source state or actions, the compressed MDP's rewards will, in general, depend on the source, destination and action taken. As before, the relevant computations involve only the given cluster's subgraph.

Given a policy π_{c} on cluster c, consider the Markov chain $(X_t)_{t \geq 0}$ with transition matrix $P_{\mathsf{c}}^{\pi_{\mathsf{c}}}$. Let T and T' be two arbitrary stopping times satisfying $0 \leq T < T' < \infty$ (a.s.). The discounted reward accumulated over the interval $T \leq t \leq T'$ is given by the random variable

$$R_T^{T'} := R(X_T, a_{T+1}, X_{T+1}) + \sum_{t=T+1}^{T'-1} \left[\prod_{\tau=T}^{t-1} \Gamma\big(X_\tau, a_{\tau+1}, X_{\tau+1}\big) \right] R\big(X_t, a_{t+1}, X_{t+1}\big)$$

where $a_{t+1} \sim \pi_{\mathsf{c}}(X_t)$ for $t = T, \ldots, T'-1$, and we set $R_T^T \equiv 0$ for any T. Next, define the hitting times of $\partial\mathsf{c}$:

$$T_m = \inf\{t > T_{m-1} \mid X_t \in \partial\mathsf{c}\}, \qquad m = 1, 2, \ldots$$

with $T_0 = \inf\{t \geq 0 \mid X_t \in \partial\mathsf{c}\}$. Note that if the chain is started in a bottleneck state $X_0 = \mathsf{b} \in \partial\mathsf{c}$, then clearly $T_0 = 0$. We will be concerned with the rewards accumulated between these successive hitting times, and by the Markovianity of $(X_t)_t$, we may, without loss of generality, consider the reward between T_0 and T_1, namely $R_{T_0}^{T_1}$. The following proposition describes how to compute the expected discounted rewards by solving a collection of linear systems.

Proposition 2. *Suppose the coarse scale action a corresponds to executing a policy π_{c} in cluster c, and let $(X_t)_{t \geq 0}$ denote the Markov chain with transition matrix $P_{\mathsf{c}}^{\pi_{\mathsf{c}}}$. The state and action dependent rewards $\widetilde{R}$ at the coarse scale may be characterized as*

$$\widetilde{R}(s,a,s') = \mathbb{E}[R_0^{T_1} \mid X_0 = s, X_{T_1} = s'], (s,s') \in \mathrm{supp}_a(\widetilde{P})\,.$$

Moreover, for fixed a, $\widetilde{R}(s,a,s') =: H_{s,s'}$ may be computed by finding the (unique, bounded) solution H to the linear system

$$H_{s,s'} = \begin{cases} \displaystyle\sum_{s'' \in \mathring{\mathsf{c}} \cap \mathsf{c}'_{s'}, a \in A} P_{h_{s'}}(s,a,s'')\Gamma(s,a,s'')H_{s'',s'} \\ \displaystyle + \sum_{s'' \in \mathsf{c}'_{s'}, a \in A} P_{h_{s'}}(s,a,s'')R(s,a,s''), & \textit{if } s \in \mathring{\mathsf{c}} \cap \mathsf{c}'_{s'} \\ \displaystyle\sum_{s'' \in \mathring{\mathsf{c}} \cap \mathsf{c}'_{s'}, a \in A} P_{\tilde{h}_{s'}}(s,a,s'')\Gamma(s,a,s'')H_{s'',s'} \\ \displaystyle + \sum_{s'' \in \mathsf{c}'_{s'}, a \in A} P_{\tilde{h}_{s'}}(s,a,s'')R(s,a,s''), & \textit{if } (s,s') \in \mathrm{supp}_a(\widetilde{P}) \end{cases}$$

where $\mathsf{c}'_{s'} := \{s \in \mathsf{c} \mid h_{s'}(s) > 0\}$;

$$P_{h_{s'}}(s,a,s'') := \frac{P_{\mathsf{c}}(s,a,s'')\pi_{\mathsf{c}}(s,a)h_{s'}(s'')}{h_{s'}(s)}$$

for $s \in \mathring{\mathsf{c}} \cap \mathsf{c}'_{s'}$, $a \in A$, $s'' \in \mathsf{c}'_{s'}$; and

$$P_{\tilde{h}_{s'}}(s,a,s'') := \frac{P_{\mathsf{c}}(s,a,s'')\pi_{\mathsf{c}}(s,a)h_{s'}(s'')}{\widetilde{P}(s,a,s')}$$

for $(s,s') \in \mathrm{supp}_a(\widetilde{P})$, $a \in A$, $s'' \in \mathsf{c}'_{s'}$; with $h_{s'}(s) := \mathbb{P}_s(X_{T_0} = s')$, for $s \in \mathsf{c}, s' \in \partial\mathsf{c}$ denoting the minimal non-negative, harmonic function satisfying

$$h_{s'}(s) = \begin{cases} \delta_{s,s'} & s \in \partial\mathsf{c} \\ P_{\mathsf{c}}^{\pi_{\mathsf{c}}}(s,s') + \sum_{s'' \in \mathring{\mathsf{c}}} P_{\mathsf{c}}^{\pi_{\mathsf{c}}}(s,s'')h_{s'}(s'') & s \in \mathring{\mathsf{c}}\,. \end{cases}$$

Thus, the total cost of computing the compressed rewards $\widetilde{R}(s,a,s')$ with respect to a given fine policy is $\mathcal{O}\big(|\partial\mathsf{c}||\mathring{\mathsf{c}}|^3 + |\partial\mathsf{c}|^2|\mathring{\mathsf{c}}|\big)$ for each cluster $\mathsf{c} \in \mathsf{C}$.

E. Discount Factors

In the preceding sections, a coarse MDP was computed by averaging over paths between bottlenecks at a finer scale. Depending on the particular source/destination pair of states, the paths will in general have different length distributions. Thus, when solving a coarse MDP, rewards collected upon transitioning between states at the coarse scale should be discounted at different, state-dependent rates. The correct discount rate is a random variable, and transitions at the coarse scale implicitly depend on outcomes at the fine scale. We will partially correct for differing length distributions, and avoid the need to simulate at the fine scale, by imposing a coarse non-uniform discount factor based on the cumulative fine scale discount applied on average to paths between bottlenecks. The coarse discount factors $\widetilde{\Gamma}$ are incorporated when solving the coarse MDP so that the scale of the coarse value function is more compatible with the fine problem, and convergence towards the fine-scale policy may be accelerated.

The expected cumulative discounts may be computed using a procedure similar to the one given for computing expected rewards in Section V-D. As before, given a policy π_{c} on cluster c, consider the Markov chain $(X_n)_{n\geq 0}$ with transition matrix $P_{\mathsf{c}}^{\pi_{\mathsf{c}}}$, and let T, T' be two arbitrary stopping times satisfying $0 \leq T < T' < \infty$ (a.s.). The cumulative discount applied to trajectories $(X_T, X_{T+1}, \ldots, X_{T'})$ over the interval $T \leq t \leq T'$ is given by the random variable

$$\Delta_T^{T'} := \prod_{t=T}^{T'-1} \Gamma\big(X_t, a_{t+1}, X_{t+1}\big)\,,$$

where $a_{t+1} \sim \pi_{\mathsf{c}}(X_t)$ for $t = T, \ldots, T'-1$. The following proposition describes how to compute the expected discount factors by solving another set of linear problems.

Proposition 3. *Suppose the coarse scale action a corresponds to executing the policy π_{c} in cluster c. Let $(X_t)_{t\geq 0}$ denote the Markov chain with transition matrix $P_{\mathsf{c}}^{\pi_{\mathsf{c}}}$, and let $(T_m)_{m\geq 0}$ denote the boundary hitting times defined in Section V-D. The state and action dependent discount factors at the coarse scale may be characterized as*

$$\widetilde{\Gamma}(s,a,s') = \mathbb{E}[\Delta_0^{T_1} \mid X_0 = s, X_{T_1} = s'], (s,s') \in \mathrm{supp}_a(\widetilde{P})$$

and, letting $H_{s,s'} := \widetilde{\Gamma}(s,a,s')$, may be computed by finding the minimal non-negative solution H to the linear system

$$H_{s,s'} = \begin{cases} \displaystyle\sum_{s''\in\mathring{\mathsf{c}}\cap\mathsf{c}'_{s'},a\in A} P_{h_{s'}}(s,a,s'')\Gamma(s,a,s'')H_{s'',s'} \\ \displaystyle + \sum_{a\in A} P_{h_{s'}}(s,a,s')\Gamma(s,a,s'), & \textit{if } s \in \mathring{\mathsf{c}} \cap \mathsf{c}'_{s'} \\ \displaystyle\sum_{s''\in\mathring{\mathsf{c}}\cap\mathsf{c}'_{s'},a\in A} P_{\tilde{h}_{s'}}(s,a,s'')\Gamma(s,a,s'')H_{s'',s'} \\ \displaystyle + \sum_{a\in A} P_{\tilde{h}_{s'}}(s,a,s')\Gamma(s,a,s'), & \textit{if } (s,s') \in \\ & \mathrm{supp}_a(\widetilde{P}) \end{cases}$$

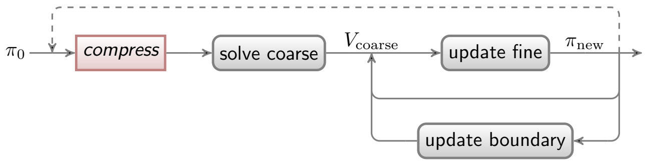

Fig. 1. *Different solution algorithms for solving a pair of coarse/fine MDPs are obtained by iterating over different paths in this flow graph. See text for details.*

where $h_{s'}(s)$, $\mathsf{c}'_{s'}$, $P_{h_{s'}}$, and $P_{\tilde{h}_{s'}}$ are as defined in Proposition 2.

The approach taken above is in the spirit of revealing the structure of the coarsening step and how it is possible to compute many coarser variables, or approximations thereof, by solutions of linear systems. Of course one may always use Monte-Carlo methods, which in addition to estimates of the expected values may be used to obtain more refined approximations to the law of the coarse random variables $\Delta_{T_0}^{T_1}$ and $R_{T_0}^{T_1}$.

VI. *Step 3:* Multiscale Solution of MDPs

Given a (fine) MDP and a coarsening as above, a solution to the fine scale MDP may be obtained by applying one of several possible algorithms suggested by the flow diagram in Figure 1. Solving for the finer scale's policy involves alternating between two main computational steps: (1) updating the fine solution given the coarse solution, and (2) updating the coarse solution given the fine solution. Given a coarse solution defined on bottleneck states, the fine scale problem decomposes into a collection of smaller independent sub-problems, each of which may be solved approximately or exactly. These are iterations along the inner loop surrounding "update fine" in Figure 1. After the fine scale problem has been updated, the solution on the bottlenecks may be updated either with or without a re-compression step. The former is represented by the long upper feedback loop in Figure 1, while the latter corresponds to the outer, lower loop passing through "update boundary". Updating without re-compressing may, for instance, take the form of the updates (e.g. Bellman, averaging) appearing in any of the asynchronous policy/value iteration algorithms. Updating by re-compression consists of re-compressing with respect to the current, updated fine policy and then solving the resulting coarse MDP.

A. An Alternating Interior-Boundary Algorithm

The particular algorithm we will consider here solves top-down, and employs localized policy iteration for fine-scale policy improvement, and local averaging for updating values at bottleneck states. We will describe the solution of a two layer hierarchy consisting of a fine scale problem and a single coarsened problem, although the main ideas may be readily extended to hierarchies of arbitrary depth; what is important is the handling of pairs of successive scales. Algorithm 2 gives the basic steps comprising the solution process.

The fine scale MDP is first compressed with respect to one or more policies. We suggest a *collection* of policies which

Algorithm 2 Top-down solution of MDPs: Alternating interior-boundary approach for pairs of layers.

```
Set the initial fine scale policy to random uniform if not otherwise given via transfer.
 1) Compress the MDP using one or more policies.
 2) Solve the coarse MDP using any algorithm, and save the resulting value function $V_{\text{coarse}}$.
 3) Fix the value function $V_{\text{fine}}$ of the fine MDP at bottleneck states $\mathcal{B}$ to $V_{\text{coarse}}$.
 4) Solve the local boundary value problems separately within each cluster to fill in the rest of $V_{\text{fine}}$, given the current
    fine scale policy.
 5) Recover a fine scale policy $\pi : \mathring{S} \times A \to \mathbb{R}_+$ on cluster interiors ($\mathring{S} := S \setminus \mathcal{B}$) from the resulting $V_{\text{fine}}$. For $s \in \mathring{S}$,
        $a^*(s) = \arg\max_{a \in A} \sum_{s' \in \mathsf{c}([s])} P(s,a,s')\big(R(s,a,s') + \Gamma(s,a,s')V_{\text{fine}}(s')\big)$   (5a)
        $\pi(s,\cdot) = \delta_{a^*(s)}$ .   (5b)
 6) Blend in a regularized fashion with the previous global policy. For $s \in \mathring{S}$,
        $\pi_{\text{new}}(s,\cdot) = \lambda\pi(s,\cdot) + (1-\lambda)\pi_{\text{old}}(s,\cdot)$ .   (6)
    where $\lambda \in (0,1]$ is a regularization parameter.
 7) (Optional - Local Policy Iteration) Set $\pi_{\text{old}} = \pi_{\text{new}}$. Repeat from step (4) until convergence criteria met.
 8) Update the fine policy on bottleneck states by applying Equations (5)-(6) for $s \in \mathcal{B}$.
 9) Update the boundary states' values either exactly, or by repeated local averaging,
        $V_{\text{fine}}(s) \leftarrow \mathbb{E}_{a \sim \pi_{\text{new}}(s)}\left[\sum_{s'} P(s,a,s')\big(R(s,a,s') + \Gamma(s,a,s')V_{\text{fine}}(s')\big)\right]$ , $s \in \mathcal{B}$
    where the number of averaging passes $N$ for each bottleneck state $s \in \mathcal{B}$ satisfies $N > \log_{\bar{\gamma}} \frac{1}{2}$ with $\bar{\gamma} :=$
    $\max_{s,a,s'}\big\{\Gamma(s,a,s')\mathbf{1}_{[P(s,a,s')>0]}\big\}$.
10) Set $\pi_{\text{old}} = \pi_{\text{new}}$. Repeat from step (4) until convergence criteria met.
```

provide all of the *coarse* actions an agent could possibly want to take involving each respective cluster. These coarse actions involve traversing a particular cluster towards each bottleneck along paths which vary in their directedness, and may be efficiently pre-computed by placing properly scaled artificial rewards at each bottleneck.

Next, the coarse MDP is solved to convergence. Solving the coarse MDP amounts to choosing the best fine policies (actions) from the available pool. Since the coarse MDP may itself be compressed and solved efficiently, this step is relatively inexpensive. The optimal value function for the coarse problem is then assigned to the set of bottleneck states for the fine problem.

With bottleneck values fixed, policy iteration is invoked within each cluster's interior independently (Steps (4)-(6)). The local value determination step can be thought of as a Poisson boundary value problem: For a given cluster c, we set $V(s) = V_{\text{coarse}}(s)$ for $s \in \partial\mathsf{c}$, and seek $V(s) := \mathbb{E}\big[R_0^{T_0} + \Delta_0^{T_0} V_{\text{coarse}}(X_{T_0}) \mid X_0 = s\big]$ for $s \in \mathring{\mathsf{c}}$, where $(X_t)_{t \geq 0} \sim P_{\mathsf{c}}^{\pi_{\mathsf{c}}}$. This amounts to solving the linear system

$$V(s) = \sum_{s' \in \mathsf{c}, a' \in A} P_{\mathsf{c}}(s,a,s')\pi_{\mathsf{c}}(s,a)\big[R(s,a,s') + \Gamma(s,a,s')V(s')\big]$$

for $s \in \mathring{\mathsf{c}}$, where P_{c} is the restriction of P to c defined by Equation (4). A greedy fine scale policy π on a cluster's interior states is computed from the interior values (Step (5)), however the new interior policy is a convex blend between the greedy policy and the previous policy (Step (6)). Policy blending allows one to regularize the solution and maintain a degree of stochasticity sufficient to repair coarse scale errors. When policy iteration has converged to the desired tolerance within each cluster independently, the individual cluster's value functions may be simply concatenated together along with the given values at the bottlenecks to obtain a globally defined value function.

Finally, information between clusters is exchanged by updating the policy on bottleneck states (Step (8)), and then using this (globally defined) policy in combination with the interior values to update bottleneck values by local averaging (Step (9)) via

$$V(\mathsf{b}) \leftarrow \sum_{s',a} P(\mathsf{b},a,s')\pi(\mathsf{b},a)\big(R(\mathsf{b},a,s') + \Gamma(\mathsf{b},a,s')V(s')\big)$$

for $\mathsf{b} \in \mathcal{B}$. Both of these steps are computationally inexpensive. Alternating updates to cluster interiors and boundaries are executed until convergence.

We emphasize that at each level of the hierarchy below the topmost level, the corresponding fine MDP may be decomposed into distinct pieces which are solved locally and independently of each other. Obtaining solutions at each resolution is an efficient process and at no point do we solve a large, global problem.

In practice, the multiscale algorithm we have discussed requires fewer iterations to converge than global, single-scale algorithms, for two primary reasons. First, the multiscale algorithm starts with a coarse approximation of the fine solution given by the solution to the compressed MDP, which provides a good warm start. Second, the multiscale approach can offer faster convergence since the solution of the sub-problems are decoupled from each other given the bottlenecks. Convergence of local (within cluster) policy iteration is thus constrained by what are likely to be faster mixing times *within* clusters, rather than slow global times across clusters, as clusters do not have strong geometric bottlenecks by construction.

B. Algorithm Analysis

The solution algorithm described above is an instance of modified asynchronous policy iteration, and can be shown to recover an optimal fine scale policy:

Theorem 1. *Fix any initial fine-scale policy π_0, and any collection of compression policies $\{\boldsymbol{\pi}_{\mathsf{c}}\}_{\mathsf{c}\in\mathsf{C}}$ such that each $\partial\mathsf{c}$ is π_{c}-reachable for all $\pi_{\mathsf{c}}\in\boldsymbol{\pi}_{\mathsf{c}}$. Let V^k denote the global fine scale value function after k passes of Steps (4)-(10) in Algorithm 2. For an appropriate number of updates per bottleneck per algorithm iteration,*

$$N > \log_{\bar{\gamma}}\tfrac{1}{2}$$

with $\bar{\gamma} := \max_{s,a,s'}\big\{\Gamma(s,a,s')\mathbf{1}_{[P(s,a,s')>0]}\big\}$, the sequence V^k generated by the alternating interior-boundary policy iteration Algorithm 2 satisfies

$$\lim_{k\to\infty}\max_{s\in S}|V^*(s)-V^k(s)| = 0.$$

Algorithm 2 may also afford substantial computational savings, as compared to the complexity of standard, global dynamic programming methods. Let n be the size of the original state space. If at a scale j there are r_j clusters of roughly equal size, and n_j states, an iteration at that scale has cost $\mathcal{O}\big(r_j(n_j/r_j)^3\big)$. If $r_j = n_j/C$ (the clusters are of roughly equal size across scales) and $n_j = n/C^j$ (the number of bottlenecks at each scale is about the number of clusters), then the computation time across $\log n$ scales is $\mathcal{O}\big(n\log n\big)$ per iteration. By contrast, DP methods typically require $O(n^3)$ time per iteration.

VII. Multiscale Transfer

The multiscale decomposition of MDPs we have described can be used to effect *knowledge transfer* between sufficiently related problems. We define transfer here as the process of transferring some aspect of a solution for one problem to another problem, such that the second problem may be solved faster or better (a better policy) than would otherwise be the case. Faster may refer to either less exploration (samples) or fewer computations, or both. Depending on the degree and type of relatedness among a pair of problems, transfer may entail small or large improvements, and may take on several different forms. It is our goal to be able to *systematically*: (1) Identify transfer opportunities; (2) Encode/represent the transferable information; and (3) Incorporate transferred knowledge into new problems. We describe only the broad contours of a transfer framework here; algorithmic details and experiments illustrating transfer in both discrete and continuous domains may be found in a longer forthcoming report.

A novel form of systematic knowledge transfer between sufficiently related MDPs is made possible by the multiscale framework discussed above. If a problem can be decomposed into a hierarchy of distinct parts then there is hope that both coarse policies governing transitions between the parts, as well as the parts themselves, may be transferred when appropriate. A key conceptual distinction is the transfer of *policies* and *potential operators* (quantities of the form $\big(I-(\Gamma_{\mathsf{c}}\circ P_{\mathsf{c}})^{\pi}\big)^{-1}$), rather than value functions.

At a high-level, transfer between two multiscale MDPs proceeds by matching sub-problems at various scales, testing whether transfer can actually be expected to help, transferring policies and/or potential operators where appropriate (along suitable statespace and action correspondences), and finally solving the unsolved problem using the transferred information as an initial condition. Transfer into sub-problems might also involve a database of pre-solved tasks: A new problem is solved by decomposing it into parts, identifying which parts are already in the database, and then stitching the pre-solved components together into a global policy by way of a coarse MDP. Any remaining unsolved parts may be solved for independently, and learning a meta policy on sub-tasks is comparatively inexpensive. The primary difficulty is determining suitable statespace correspondences, although this task is made easier by the multiscale partitioning, and the multiscale solution algorithms we have discussed can be robust to errors. As with the multiscale decomposition itself, the transfer process is inexpensive because it is inherently local.

References

[1] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, pp. 181–211, 1999.

[2] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *Journal of Artificial Intelligence Research*, vol. 13, pp. 227–303, 2000.

[3] R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines," in *Advances in Neural Information Processing Systems (NIPS)*, 1997.

[4] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dynamic Systems*, vol. 13, pp. 341–379, 2003.

[5] J. Barry, L. Kaelbling, and T. Lozano-Pérez, "DetH*: Approximate hierarchical solution of large markov decision processes," in *Proc. International Joint Conference on Artificial Intelligence*, 2011.

[6] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[7] V. Soni and S. Singh, "Using homomorphisms to transfer options across reinforcement learning domains," in *Proc. National Conference on Artificial Intelligence (AAAI)*, 2006.

[8] K. Ferguson and S. Mahadevan, "Proto-transfer learning in markov decision processes using spectral methods," in *ICML Workshop on Transfer Learning*, 2006.

[9] M. L. Puterman, *Markov Decision Processes*. Wiley, 1994.

[10] D. P. Bertsekas, *Dynamic Programming and Optimal Control (Vol. II)*, 3rd ed. Athena Scientific, 2007.

[11] J. R. Norris, *Markov Chains*. Cambridge University Press, 1997.

[12] O. Simsek and A. Barto, "Skill characterization based on betweenness," in *Advances in Neural Information Processing Systems (NIPS) 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2009.

[13] D. A. Spielman and S.-H. Teng, "A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning," *eprint*, 2008, arXiv:0809.3232.

[14] R. Andersen and Y. Peres, "Finding sparse cuts locally using evolving sets," in *Proceedings of the 41st annual ACM symposium on Theory of computing*, ser. STOC '09. ACM, 2009, pp. 235–244.

[15] F. Chung, "Laplacians and the cheeger inequality for directed graphs," *Annals of Combinatorics*, vol. 9, pp. 1–19, 2005.