Optimal Hierarchical Average-Reward Linearly-solvable Markov Decision Processes

Guillermo Infante
Universitat Pompeu Fabra
Barcelona, Spain
guillermo.infante@upf.edu

Anders Jonsson
Universitat Pompeu Fabra
Barcelona, Spain
anders.jonsson@upf.edu

Vicenç Gómez
Universitat Pompeu Fabra
Barcelona, Spain
vicen.gomez@upf.edu

Abstract

We introduce a novel approach to hierarchical reinforcement learning for Linearly-solvable Markov Decision Processes (LMDPs) in the infinite-horizon average-reward setting. Unlike previous work, our approach allows learning low-level and high-level tasks simultaneously, without imposing limiting restrictions on the low-level tasks. Our method relies on partitions of the state space that create smaller subtasks that are easier to solve, and the equivalence between such partitions to learn more efficiently. We then exploit the compositionality of low-level tasks to exactly represent the value function of the high-level task. Experiments show that our approach can outperform flat average-reward reinforcement learning by one or several orders of magnitude.

1 Introduction

Hierarchical reinforcement learning [Dayan and Hinton, 1992, Parr and Russell, 1997, Sutton et al., 1999, Dietterich, 2000, Barto and Mahadevan, 2003] aims to decompose a complex high-level task into several low-level tasks. After solving the low-level tasks, their solutions can be combined to form a solution to the high-level task. Ideally, the low-level tasks should be significantly easier to solve than the high-level task, in which case one can obtain an important speedup in learning [Nachum et al., 2018, Wen et al., 2020]. Hierarchical reinforcement learning has also been credited with other advantages, e.g. the ability to explore more efficiently [Nachum et al., 2019].

Most previous work on hierarchical reinforcement learning considers either the finite-horizon setting or the infinite-horizon setting with discounted rewards. In the few works on hierarchical reinforcement learning in the average-reward setting, either the low-level tasks are assumed to be solved beforehand [Fruit and Lazaric, 2017, Fruit et al., 2017, Wan et al., 2021b] or they have important restrictions that severely reduce their applicability, e.g. a single initial state [Ghavamzadeh and Mahadevan, 2007]. It is therefore an open question how to combine hierarchical reinforcement learning in the average-reward setting to learn the low-level and high-level tasks simultaneously.

In this paper we propose a novel framework for hierarchical reinforcement learning in the average-reward setting that simultaneously solves low-level and high-level tasks. Concretely, we consider the class of Linearly-solvable Markov Decision Processes (LMDPs) [Todorov, 2006]. This class of problems plays a key role in the framework of RL as probabilistic inference Kappen et al. [2012], Levine [2018]. Since the Bellman optimality equations are linear for LMDPs, it is possible to compose a solution to a novel task from the solutions to previously solved task without learning [Todorov, 2009b]. Such a property has been exploited in recent works about compositionality in RL Hunt et al. [2019], van Niekerk et al. [2019], Nangue Tasse et al. [2020] and in combination with Hierarchical RL in the finite-horizon setting Jonsson and Gómez [2016], Saxe et al. [2017], Infante et al. [2022]. Adapting this idea to the average-reward setting requires careful analysis.

16th European Workshop on Reinforcement Learning (EWRL 2023).

Unlike most frameworks for hierarchical reinforcement learning, our proposed approach does not decompose the policy, only the value function. Hence the agent never chooses a subtask to solve, and instead uses the subtasks to compose the value function of the high-level task. This avoids introducing non-stationarity at the higher level when updating the low-level policies. To the best of our knowledge, our work makes the following novel contributions:

- Learning low-level and high-level tasks simultaneously in the average-reward setting, without imposing restrictions on the low-level tasks.
- Representing low-level tasks as finite-horizon rather than average-reward decision processes.
- Extending the combination of compositionality and hiearchical reinforcement learning in the average-reward setting for LMDPs.

2 Related work

Most research on hierarchical reinforcement learning formulates problems as a Semi-Markov Decision Process (SMDP) with options [Sutton et al., 1999] or the MAXQ decomposition [Dietterich, 2000].

Fruit and Lazaric [2017] and Fruit et al. [2017] propose algorithms for solving SMDPs with options in the average-reward setting, proving that the regret of their algorithms is polynomial in the size of the SMDP components, which may be smaller than the components of the underlying Markov Decision Process (MDP). Wan et al. [2021b] present a version of differential Q-learning for SMDPs with options in the average-reward setting, proving that differential Q-learning converges to the optimal policy. However, the above work assumes that the option policies are given prior to learning. Ghavamzadeh and Mahadevan [2007] propose a framework for hierarchical average-reward reinforcement learning based on the MAXQ decomposition, in which low-level tasks are also modeled as average-reward decision processes. However, since the distribution over initial states can change as the high-level policy changes, the authors restrict low-level tasks to have a single initial state.

Wen et al. [2020] present an approach for partitioning the state space and exploiting the equivalence of low-level tasks, similar to our work. The authors present an algorithm whose regret is smaller than that of algorithms for solving the high-level task. However, their algorithm does not decompose the high-level task into low-level tasks, but instead exploits the fact that equivalent subtasks have shared dynamics. Infante et al. [2022] combine the compositionality of LMDPs with the equivalence of low-level tasks to develop a framework for hierarchical reinforcement learning in the finite-horizon setting. Compositionality is also a key idea in the option keyboard [Barreto et al., 2019], though in the case of MDPs, compositionality is not exact, so the resulting policy will only be approximately optimal. In contrast, our framework based on LMDPs can represent the globally optimal policy.

3 Background

Given a finite set $\mathcal{X}$, we use $\Delta(\mathcal{X})$ to denote the set of all probability distributions on $\mathcal{X}$.

3.1 First-exit Linearly-solvable Markov Decision Processes

A first-exit Linearly-solvable Markov Decision Process (LMDP, Todorov [2006]) is a tuple $\mathcal{L} = \langle \mathcal{S}, \mathcal{T}, \mathcal{P}, \mathcal{R}, \mathcal{J} \rangle$, where $\mathcal{S}$ is a set of non-terminal states, $\mathcal{T}$ is a set of terminal states, $\mathcal{P} : \mathcal{S} \to \Delta(\mathcal{S}^+)$ is an uncontrolled transition function, $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ is a reward function on non-terminal states, and $\mathcal{J} : \mathcal{T} \to \mathbb{R}$ is a reward function on terminal states. We use $S^+ = \mathcal{S} \cup \mathcal{T}$ to denote the full set of states. An agent interacts with the environment following a policy $\pi : \mathcal{S} \to \Delta(\mathcal{S}^+)$. At *timestep* t, it observes a state s_t, transitions to a new state $s_{t+1} \sim \pi(\cdot|s_t)$ and receives a reward

$$\mathcal{R}(s_t, s_{t+1}, \pi) = r_t - \frac{1}{\eta} \log \frac{\pi(s_{t+1}|s_t)}{\mathcal{P}(s_{t+1}|s_t)},$$

where r_t is a reward with mean $\mathcal{R}(s_t)$. The agent can modify the policy $\pi(\cdot|s_t)$, but gets penalized for deviating from the uncontrolled dynamics $\mathcal{P}(\cdot|s_t)$. The parameter $\eta > 0$ controls this penalty. Given $\eta > 0$, the value function of a policy π can be defined as follows:

$$v_\eta^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{T-1} \mathcal{R}(S_t, S_{t+1}, \pi) + \mathcal{J}(S_T) \,\middle|\, \pi, S_0 = s \right], \tag{1}$$

where T is a random variable representing the length of the episode, and S_t, $t \geq 0$, are random variables denoting the state at time t. The interaction ends when the agent reaches a terminal state S_T and receives reward $\mathcal{J}(S_T)$. The value function of a terminal state $\tau \in \mathcal{T}$ is simply $v_\eta^\pi(\tau) = \mathcal{J}(\tau)$.

The aim of the agent is to find a policy that maximizes expected future reward. For that it is useful to define the *optimal* value function $v_\eta^* : \mathcal{S} \to \mathbb{R}$ among all policies with parameter η. For simplicity, in what follows we omit the subscript and asterisk and refer to the optimal value function simply as the value function v. Such a value function is known to satisfy the following Bellman optimality equations [Todorov, 2006]:

$$v(s) = \frac{1}{\eta} \log \sum_{s' \in \mathcal{S}^+} \mathcal{P}(s'|s) \exp(\eta(\mathcal{R}(s) + v(s'))) \;\; \forall s \in \mathcal{S}. \tag{2}$$

Introducing the notation $z(s) = e^{\eta v(s)}$, $s \in \mathcal{S}^+$, results in the following system of linear equations:

$$z(s) = e^{\eta \mathcal{R}(s)} \sum_{s' \in \mathcal{S}^+} \mathcal{P}(s'|s) z(s') \;\; \forall s \in \mathcal{S}. \tag{3}$$

We abuse notation and refer to $z(s)$ as the value of s. Given z, an optimal policy is given by

$$\pi(s'|s) = \frac{\mathcal{P}(s'|s) z(s')}{\sum_{s''} \mathcal{P}(s''|s) z(s'')} \equiv \frac{\mathcal{P}(s'|s) z(s')}{G[z](s)}. \tag{4}$$

In what follows, we make the following assumption about the rewards of any first-exit LMDP.

Assumption 1. *For any first-exit LMDP $\mathcal{L}$, the reward $\mathcal{R}(s)$ is bounded in $(-\infty, 0)$ for all $s \in \mathcal{S}$ and $\mathcal{J}(\tau)$ is bounded in $(-\infty, 0]$ for all $\tau \in \mathcal{T}$, i.e. $z(s)$ and $z(\tau)$ are bounded in $[0, 1]$.*

The system of linear equations in (3) can then be written in matrix form when we know the uncontrolled dynamics and the reward functions. We let $P \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}^+|}$ be a matrix such that $P_{(s,s')} = \mathcal{P}(s'|s)$ and $R \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ a diagonal matrix such that $R_{(s,s)} = e^{\eta \mathcal{R}(s)}$. We also let $\mathbf{z}$ be the vector form of the value $z(s)$ for all states $s \in \mathcal{S}$ and $\mathbf{z}^+$ an extended vector that also includes the known value $z(\tau) = e^{\eta \mathcal{J}(\tau)}$ for all terminal states $\tau \in \mathcal{T}$. The problem is then expressed as

$$\mathbf{z} = RP\mathbf{z}^+. \tag{5}$$

We can use the power iteration method over (5) to obtain the solution for $\mathbf{z}$ [Todorov, 2006].

Alternatively, when $\mathcal{P}$ and $\mathcal{R}$ are not known, the agent can learn an estimate $\hat{z}$ of the optimal value function in an online manner, using samples (s_t, r_t, s_{t+1}) generated when following some estimated policy $\hat{\pi}_t$. The update rule for the so-called Z-learning algorithm is given by

$$\hat{z}(s_t) \leftarrow (1 - \alpha_t)\hat{z}(s_t) + \alpha_t e^{\eta \mathcal{R}(s_t, s_{t+1}, \hat{\pi}_t)} \hat{z}(s_{t+1}) = (1 - \alpha_t)\hat{z}(s_t) + \alpha_t e^{\eta r_t} \frac{\mathcal{P}(s_{t+1}|s_t)}{\hat{\pi}_t(s_{t+1}|s_t)} \hat{z}(s_{t+1}). \tag{6}$$

Here, α_t is a learning rate and $\mathcal{P}(s_{t+1}|s_t)/\hat{\pi}_t(s_{t+1}|s_t)$ acts as an importance weight.

In the first-exit case, the solution of a set of *component* problems can be combined to retrieve the optimal solution for new *composite* problems with no further learning [Todorov, 2009b]. Assume we have a set of first-exit LMDPs $\{\mathcal{L}_i\}_{i=1}^K$, which share $\mathcal{S}, \mathcal{T}$, $\mathcal{P}$ and $\mathcal{R}$, but differ in the values $z_i(\tau) = e^{\eta \mathcal{J}_i(\tau)}$ of terminal states. Let $z_1, \ldots, z_K$ be the optimal value functions of $\mathcal{L}_1, \ldots, \mathcal{L}_K$. Now consider a new composite problem $\mathcal{L}$ that also shares the aforementioned elements with the component problems. If the value at terminal states can be expressed as a weighted sum as follows:

$$z(\tau) = \sum_{i=1}^K w_i z_i(\tau) \;\; \forall \tau \in \mathcal{T},$$

then by linearity of the value function, the same expression holds for non-terminal states:

$$z(s) = \sum_{i=1}^K w_i z_i(s) \;\; \forall s \in \mathcal{S}.$$

3.2 Hierarchical Decomposition for LMDPs

Infante et al. [2022] introduce a hierarchical decomposition for LMDPs. Given a first-exit LMDP $\mathcal{L} = \langle \mathcal{S}, \mathcal{T}, \mathcal{P}, \mathcal{R}, \mathcal{J} \rangle$, the set of non-terminal states $\mathcal{S}$ is partitioned into L subsets $\{\mathcal{S}_i\}_{i=1}^{L}$. Each subset $\mathcal{S}_i$ induces a subtask $\mathcal{L}_i = \langle \mathcal{S}_i, \mathcal{T}_i, \mathcal{P}_i, \mathcal{R}_i, \mathcal{J}_i \rangle$, i.e. an LMDP whose components are given by:

- The set of non-terminal states is $\mathcal{S}_i$.
- The set of terminal states $\mathcal{T}_i = \{\tau \in \mathcal{S}^+ \setminus \mathcal{S}_i : \exists s \in \mathcal{S}_i \text{ s.t. } \mathcal{P}(\tau|s) > 0\}$ includes all states in $\mathcal{S}^+ \setminus \mathcal{S}_i$ (terminal or non-terminal) that are reachable in one step from a state in $\mathcal{S}_i$.
- $\mathcal{P}_i : \mathcal{S}_i \to \Delta(\mathcal{S}_i^+)$ and $\mathcal{R}_i : \mathcal{S}_i \to \mathbb{R}$ are the restrictions of $\mathcal{P}$ and $\mathcal{R}$ to $\mathcal{S}_i$, where $\mathcal{S}_i^+ = \mathcal{S}_i \cup \mathcal{T}_i$ denotes the full set of subtask states.
- The reward of a terminal state $\tau \in \mathcal{T}_i$ equals $\mathcal{J}_i(\tau) = \mathcal{J}(\tau)$ if $\tau \in \mathcal{T}$, and $\mathcal{J}_i(\tau) = \hat{v}(\tau)$ otherwise, where $\hat{v}(\tau)$ is the estimated value in $\mathcal{L}$ of the non-terminal state in $\tau \in \mathcal{S} \setminus \mathcal{S}_i$.

Intuitively, if the value $z_i(\tau)$ of each terminal state $\tau \in \mathcal{T}_i$ equals its optimal value $z(\tau)$ for the original LMDP $\mathcal{L}$, then solving the subtask $\mathcal{L}_i$ yields the optimal values of the states in $\mathcal{S}_i$. In practice, however, the agent only has access to an estimate $\hat{z}(\tau)$ of the optimal value. In this case, the subtask $\mathcal{L}_i$ is *parameterized* on the value estimate $\hat{v}$ (or $\hat{z}$) of terminal states in $\mathcal{T}_i$, and each time the value estimate changes, the agent can solve $\mathcal{L}_i$ to obtain a new value estimate $\hat{z}(s)$ for each state $s \in \mathcal{S}_i$.

The set of *exit states* $\mathcal{E} = \cup_{i=1}^{L} \mathcal{T}_i$ is the union of the terminal states of each subtask in $\{\mathcal{L}_1, \ldots, \mathcal{L}_L\}$. For convenience, let $\mathcal{E}_i = \mathcal{E} \cap \mathcal{S}_i$ denote the set of (non-terminal) exit states in the subtask $\mathcal{L}_i$. Also define the notation $K = \max_{i=1}^{L} |\mathcal{S}_i|$, $N = \max_{i=1}^{L} |\mathcal{T}_i|$ and $E = |\mathcal{E}|$.

Definition 1. *Two subtasks $\mathcal{L}_i$ and $\mathcal{L}_j$ are equivalent if there exists a bijection $f : \mathcal{S}_i \to \mathcal{S}_j$ such that the transition probabilities and rewards of non-terminal states are equivalent through f.*

Using the above definition, we can define a set of equivalence classes $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_C\}$, $C \leq L$, i.e. a partition of the set of subtasks $\{\mathcal{L}_1, \ldots, \mathcal{L}_L\}$ such that all subtasks in a given partition are equivalent. Each equivalence class can be represented by a single subtask $\mathcal{L}_j = \langle \mathcal{S}_j, \mathcal{T}_j, \mathcal{P}_j, \mathcal{R}_j, \mathcal{J}_j \rangle$. As before, the reward $\mathcal{J}_j$ of terminal states is parameterized on a given value estimate $\hat{v}$. We assume that each non-terminal state $s \in \mathcal{S}$ can be mapped to its subtask $\mathcal{L}_i$ and equivalence class $\mathcal{C}_j$.

For each subtask $\mathcal{L}_j$ and each terminal state $\tau^k \in \mathcal{T}_j$, Infante et al. [2022] introduce a *base LMDP* $\mathcal{L}_j^k = \langle \mathcal{S}_j, \mathcal{T}_j, \mathcal{P}_j, \mathcal{R}_j, \mathcal{J}_j^k \rangle$ that shares all components with $\mathcal{L}_j$ except the reward function on terminal states, which is defined as $\mathcal{J}_j^k(\tau) = 1$ if $\tau = \tau^k$, and $\mathcal{J}_j^k(\tau) = 0$ otherwise. Let $z_j^1, \ldots, z_j^n$ be the optimal value functions of the base LMDPs for $\mathcal{L}_j$, with $n = |\mathcal{T}_i|$. Given a value estimate $\hat{z}$ on each terminal state in $\mathcal{T}_j$, due to compositionality we can express the value estimate of each state $s \in \mathcal{S}_j$ as

$$\hat{z}(s) = \sum_{k=1}^{n} \hat{z}(\tau^k) z_j^k(s).$$

To solve the original LMDP, we can now define an optimal value function on exit states as $z_{\mathcal{E}} : \mathcal{E} \to \mathbb{R}$, and construct a matrix $G = \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$ whose element $G_{(s,s')}$ equals $z_j^k(s)$ if $s' = \tau^k$ is the k-th terminal state in the subtask $\mathcal{L}_j$ corresponding to the partition $\mathcal{S}_i$ to which s belongs, and 0 otherwise. By writing $\mathbf{z}_{\mathcal{E}}$ in vector form, the optimal value function satisfies the following matrix equation:

$$\mathbf{z}_{\mathcal{E}} = G\mathbf{z}_{\mathcal{E}}.$$

The total number of values required to represent the optimal value function equals $E + CKN$, since there are C equivalence classes with at most K states and N base LMDPs.

4 Average-reward Linearly-solvable Markov Decision Process

An Average-reward Linearly-solvable Markov Decision Process (ALMDP) is a tuple $\mathcal{L} = \langle \mathcal{S}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{P} : \mathcal{S} \to \Delta(\mathcal{S})$ is an uncontrolled transition function and $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ is a reward function. Since ALMDPs represent continuing tasks, there are no terminal states.

Throughout the paper, we make the following assumptions about ALMDPs.

Assumption 2. *The ALMDP $\mathcal{L}$ is unichain [Puterman, 1994]: the transition probability distribution induced by all stationary policies admit a single recurrent class.*

Assumption 3. *For all states $s \in \mathcal{S}$ the reward $\mathcal{R}(s)$ is bounded in $(-\infty, 0]$.*

In the average-reward setting, the value function is defined as the expected average reward when following a policy $\pi : \mathcal{S} \to \Delta(\mathcal{S}^+)$ starting from a state $s \in \mathcal{S}$. This is expressed as

$$v_\eta^\pi(s) = \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T} \mathcal{R}(S_t, S_{t+1}, \pi) \;\middle|\; \pi, S_0 = s\right]. \tag{7}$$

Again, we are interested in obtaining the *optimal* value function v. Similarly to the first-exit case, we obtain the following Bellman optimality equations:

$$v(s) = \frac{1}{\eta}\log \sum_{s'\in\mathcal{S}} \mathcal{P}(s'|s)\exp(\eta(\mathcal{R}(s) - \rho + v(s'))) \;\; \forall s \in \mathcal{S} \tag{8}$$

where ρ refers to the optimal one-step average reward (i.e. gain). After exponentiating, we obtain

$$z(s) = e^{\eta(\mathcal{R}(s)-\rho)} \sum_{s'\in\mathcal{S}} \mathcal{P}(s'|s) z(s') \;\; \forall s \in \mathcal{S}. \tag{9}$$

For the optimal value function z, the optimal policy is given by the same expression as in (4).

4.1 Solving an ALMDP

We introduce the notation $\Gamma = e^{\eta\rho}$ (exponentiated gain). Analogously to the first-exit case, the problem in (9) can then expressed in matrix form as

$$\Gamma \mathbf{z} = RP\mathbf{z}. \tag{10}$$

Here the matrices $P \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ and $R \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ are appropiately defined as in (5). The exponentiated gain Γ can be shown to correspond to the largest eigenvalue of RP [Todorov, 2009a]. An ALMDP can be solved using *relative value iteration* by initializing $\hat{\mathbf{z}}_0 = \mathbf{1}$ and iteratively applying

$$\hat{\mathbf{z}}_{k+\frac{1}{2}} \leftarrow RP\hat{\mathbf{z}}_k, \qquad \hat{\mathbf{z}}_{k+1} \leftarrow \hat{\mathbf{z}}_{k+\frac{1}{2}} / \hat{z}_{k+\frac{1}{2}}(s^*),$$

where $s^* \in \mathcal{S}$ is a reference state. After convergence, the exponentiated gain equals $\Gamma = \hat{z}_{k+\frac{1}{2}}(s^*)$.

An alternative method for solving an ALMDP $\mathcal{L}$ is to transform it to a first-exit LMDP. Given a reference state s^*, define a first-exit LMDP $\mathcal{L}' = \langle \mathcal{S} \setminus \{s^*\}, \{s^*\}, \mathcal{P}', \mathcal{R}', \mathcal{J}' \rangle$, where $\mathcal{P}'(s'|s) = \mathcal{P}(s'|s)$ for all state pairs $(s, s') \in (\mathcal{S} \setminus \{s^*\}) \times \mathcal{S}$, and $\mathcal{J}(s^*) = 0$. By inspection of (3) and (9), the Bellman optimality equation of $\mathcal{L}'$ is identical to that of $\mathcal{L}$ if $\mathcal{R}'(s) = \mathcal{R}(s) - \rho$. Even though the agent has no prior knowledge of the exponentiated gain $\Gamma = e^{\eta\rho}$, we can perform binary search to find Γ. For a given estimate $\hat{\Gamma}$ of Γ, after solving $\mathcal{L}'$, we can compare $\hat{\Gamma} z(s^*)$ with $e^{\eta\mathcal{R}(s^*)}\sum_s \mathcal{P}(s|s^*)z(s)$. If $\hat{\Gamma} z(s^*)$ is greater, then $\hat{\Gamma}$ is too large, else it is too small.

Alternatively, when $\mathcal{P}$ and $\mathcal{R}$ are not known, we can use samples (s_t, r_t, s_{t+1}) generated by some policy $\widehat{\pi}_t$ to get an estimate $\hat{z}$ of the optimal value z using *differential Z-learning*, similar to differential Q-learning [Wan et al., 2021a]. Simultaneously, we need to keep an estimate $\widehat{\Gamma}$ of the exponentiated gain Γ. We can derive update rules for $\hat{z}$ and $\widehat{\Gamma}$ from (9) as follows:

$$\widehat{z}_{t+1}(s_t) \leftarrow \widehat{z}_t(s_t) + \alpha_t\left(\frac{e^{\eta r_t}}{\widehat{\Gamma}_t}\frac{\mathcal{P}(s_{t+1}|s_t)}{\widehat{\pi}_t(s_{t+1}|s_t)} - \widehat{z}_t(s_t)\right), \tag{11}$$

$$\widehat{\Gamma}_{t+1} \leftarrow \widehat{\Gamma}_t + \beta_t\left(\frac{e^{\eta r_t}}{\widehat{z}_t(s_t)}\frac{\mathcal{P}(s_{t+1}|s_t)}{\widehat{\pi}_t(s_{t+1}|s_t)} - \widehat{\Gamma}_t\right). \tag{12}$$

The learning rates α_t and β_t can be chosen independently.

5 Hierarchical Average-Reward LMDPs

In this section we present our approach for hierarchical average-reward LMDPs. The idea is to take advantage of the similarity of the value functions in the first-exit and average-reward settings, and use compositionality to compose the value functions of the subtask LMDPs without additional learning.

5.1 Hierarchical Decomposition

Consider an ALMDP $\langle \mathcal{S}, \mathcal{P}, \mathcal{T} \rangle$. Similarly to Infante et al. [2022], we assume that the state space $\mathcal{S}$ is partitioned into subsets $\{\mathcal{S}_i\}_{i=1}^{L}$, with each partition $\mathcal{S}_i$ inducing a first-exit LMDP $\mathcal{L}_i = \langle \mathcal{S}_i, \mathcal{T}_i, \mathcal{P}_i, \mathcal{R}_i, \mathcal{J}_i \rangle$. The components of each such subtask $\mathcal{L}_i$ are defined as follows:

- The set of states is $\mathcal{S}_i$.
- The set of terminal states $\mathcal{T}_i = \{\tau \in \mathcal{S} \setminus \mathcal{S}_i : \exists s \in \mathcal{S}_i \text{ s.t. } \mathcal{P}(\tau|s) > 0\}$ contains states not in $\mathcal{S}_i$ that are reachable in one step from any state inside the partition.
- The transition function $\mathcal{P}_i$ and reward function $\mathcal{R}_i$ are projections of $\mathcal{P}$ and $\mathcal{R}$ onto $\mathcal{S}_i$.
- $\mathcal{J}_i$ is defined for each $\tau \in \mathcal{T}_i$ as $\mathcal{J}_i(\tau) = \hat{v}(\tau)$, where $\hat{v}$ is a current value estimate (hence $z_i(\tau) = e^{\eta \hat{v}(\tau)} = \hat{z}(\tau)$ is defined by a current exponentiated value estimate $\hat{z}$).

The Bellman optimality equations of each subtask $\mathcal{L}_i$ are given by

$$z_i(s) = e^{\eta \mathcal{R}_i(s)} \sum_{s'} \mathcal{P}_i(s'|s) z_i(s') \;\; \forall s \in \mathcal{S}_i. \tag{13}$$

By inspection of the Bellman optimality equations in (9) and (13), they are equal if we set $\mathcal{R}_i(s) = \mathcal{R}(s) - \rho$. Thus, if $z_i(\tau) = z(\tau)$ for each $\tau \in \mathcal{T}_i$ then the solution of the subtask $\mathcal{L}_i$ corresponds to the optimal solution for each $s \in \mathcal{S}_i$. However, in general neither ρ nor $z(\tau)$ are known prior to learning and, therefore, we have to use estimates $\hat{\rho}$ and $\hat{z}(\tau)$. Each subtask $\mathcal{L}_i$ can be seen as being *parameterized* on the estimate values $\hat{z}(\tau)$ for each $\tau \in \mathcal{T}_j$. Every time that $\hat{z}(\tau)$ for $\tau \in \mathcal{T}_i$ changes, we obtain a new value estimate for each $s \in \mathcal{S}_i$ by solving the subtask for the new terminal values.

5.2 Subtask Compositionality

It is impractical to solve each subtask $\mathcal{L}_i$ every time the estimate $\hat{z}(\tau)$ changes for $\tau \in \mathcal{T}_j$. To alleviate this computation we leverage compositionality for LMDPs. The key insight is to build a basis of value functions that can be combined to obtain the solution for the subtasks.

Consider a subtask $\mathcal{L}_i = \langle \mathcal{S}_i, \mathcal{T}_i, \mathcal{P}_i, \mathcal{R}_i, \mathcal{J}_i \rangle$ and let $n = |\mathcal{T}_i|$. We introduce n base LMDPs $\{\mathcal{L}_i^1, \ldots, \mathcal{L}_i^n\}$ that are first-exit LMDPs and terminate in $\mathcal{T}_i$. These base LMDPs only differ from $\mathcal{L}_i$ in the reward of each terminal state $\tau^k \in \mathcal{T}_i$. For all $s \in \mathcal{S}_i$, the reward for each $\mathcal{L}_i^k$ is by definition $\mathcal{R}_i(s) = \mathcal{R}(s) - \rho$ for all $s \in \mathcal{S}_i$, while at terminal states $\tau \in \mathcal{T}_i$ we let the reward function be $z_i^k(\tau; \rho) = 1$ if $\tau = \tau^k$ and $z_i^k(\tau; \rho) = 0$ otherwise. Thus, the base LMDPs are parameterized by the gain ρ. This is equivalent to setting the reward to $\mathcal{J}_i^k(\tau) = 0$ if $\tau = \tau_k$ and $\mathcal{J}_i^k(\tau) = -\infty$ otherwise. Intuitively, each base LMDP solves the subtask of reaching one specific terminal state $\tau_k \in \mathcal{T}_i$.

Let us now assume that we have the solution $z_i^1(\cdot; \rho), \ldots, z_i^n(\cdot; \rho)$ for the base-LMDPs (for the optimal gain ρ) as well as the optimal value $z(\tau^k)$ of the original ALMDP for each terminal state $\tau^k \in \mathcal{T}_i$. Then by compositionality we could represent the value function of each terminal state as a weighted combination of the subtasks:

$$z(\tau) = \sum_{k=1}^{n} w_k z_i^k(\tau; \rho) = \sum_{k=1}^{n} z(\tau^k) z_i^k(\tau; \rho) \;\; \forall \tau \in \mathcal{T}_i. \tag{14}$$

Clearly, the RHS in the previous expression evaluates to $z(\tau)$ since $z(\tau^k) z_i^k(\tau; \rho) = z(\tau) \cdot 1$ when $\tau = \tau^k$, and $z(\tau^k) z_i^k(\tau; \rho) = z(\tau^k) \cdot 0$ otherwise.

Thanks to compositionality, we can also represent the value function for each state $s \in \mathcal{S}_i$ as

$$z(s) = \sum_{k=1}^{n} z(\tau^k) z_i^k(s; \rho) \;\; \forall s \in \mathcal{S}_i. \tag{15}$$

We remark that the base LMDPs depend on the gain ρ by the definition of the reward function. This parameter is not known prior to learning. The subtasks in practice are solved for the latest estimate $\hat{\rho}$ and must be re-learned for every update of this parameter until convergence.

Efficiency of the value representation. Similar to previous work [Wen et al., 2020, Infante et al., 2022] we can exploit the equivalence of subtasks to learn more efficiently. Let $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_C\}$,

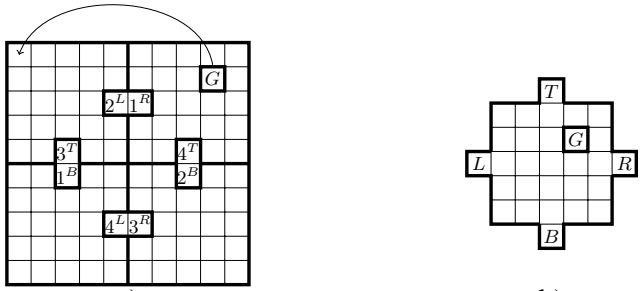

Figure 1: a) An example 4-room ALMDP; b) a single subtask with 5 terminal states G, L, R, T, B that is equivalent to all 4 room subtasks. Rooms are numbered 1 through 4, left-to-right, then top-to-bottom, and exit state 1^B refers to the exit B of room 1, etc.

$C \leq L$, be a set of equivalence classes, i.e. a partition of the set of subtasks $\{\mathcal{L}_1, \ldots, \mathcal{L}_L\}$ such that all subtasks in a given partition are equivalent. As before, we also define a set of exit states as $\mathcal{E} = \cup_{i=1}^{L} \mathcal{T}_i$. Due to the decomposition, we do not need to keep an explicit value estimate $\hat{z}(s)$ for every state $s \in \mathcal{S}$. Instead, it is sufficient to keep a value function for exit states $\hat{z}_\mathcal{E} : \mathcal{E} \to \mathbb{R}$ and a value function for each base LMDP of each equivalence class. This is enough to represent the value for any state $s \in \mathcal{S}$ by using the compositionality expression in (15).

Letting $K = \max_{i=1}^{L} |\mathcal{S}_i|$, $N = \max_{i=1}^{L} |\mathcal{T}_i|$ and $E = |\mathcal{E}|$, $O(KN)$ values are needed to represent the base LMDPs of a subtask, and we can thus represent the value function with $O(CKN + E)$ values. The decomposition leads to an efficient representation of the value function whenever $CKN + E \ll |\mathcal{S}|$. This is achieved when there are few equivalence classes, the size of each subtask is small (in terms of the number of states) and there are relatively few exit states.

Example 1: Figure 1a) shows an example 4-room ALMDP. When reaching the state marked G, separate from the room but reachable in one step from the highlighted location, the system transitions to a restart state (top left corner) and receives reward 0. In all other states the reward is -1. The rooms are connected via doorways, so the subtask corresponding to each room has two terminal states in other rooms, plus the terminal state G in the top right room. The 9 exit states in $\mathcal{E}$ are highlighted and correspond to states next to doorways, plus G. Figure 1b) shows a single subtask that is equivalent to all four room subtasks, since the dynamics is shared inside rooms and the set of terminal states is the same. There are five base LMDPs with value functions z^G, z^L, z^R, z^T and z^B, respectively. Given an initial value estimate $\hat{z}_\mathcal{E}$ for each exit state in $\mathcal{E}$, a value estimate of any state in the top left room is given by $\hat{z}(s) = \hat{z}_\mathcal{E}(1^B) z^B(s) + \hat{z}_\mathcal{E}(1^R) z^R(s)$, where we use $\hat{z}_\mathcal{E}(G) = \hat{z}_\mathcal{E}(L) = \hat{z}_\mathcal{E}(T) = 0$ to indicate that the terminal states G, L and T are not reachable in the top left room. The total number of values needed to store the optimal value function is $E + CKN = 9 + 125 = 134$, and the base LMDPs are faster to learn since they have smaller state space.

6 Algorithms

We now propose two algorithms for solving hierarchical ALMDPs. The first is a two-stage eigenvector approach that relies on first solving the subtasks. The second is an online algorithm in which an agent simultaneously learns the subtasks, the gain and the exit values from samples (s_t, r_t, s_{t+1}). Once again we recall that we do not explicitly represent the values for states $s \notin \mathcal{E}$.

Eigenvector approach. In previous work, the base LMDPs are only solved once, and the solutions are then reused to compute the value function $z_\mathcal{E}$ on exit states. However, in the case of ALMDPs, the reward functions of base LMDPs depend on the current gain estimate, which is initially unknown. If we could estimate the expected *duration* $d_i(s)$ of each subtask $\mathcal{L}_i$ from each state $s \in \mathcal{S}_i$, we could use the duration to adjust the value estimate $\hat{z}(s)$ of s according to the current gain estimate $\widehat{\Gamma}$ by a factor $\widehat{\Gamma}^{d_i(s)}$. The duration can be recursively defined as $d_i(\tau) = 0$ for each $\tau \in \mathcal{T}_i$ and

$$d_i(s) = 1 + \sum_{s'} \pi_i(s'|s) d_i(s').$$

However, even though the subtask policy π_i can be expressed in terms of the base LMDP policies π_i^k, we have been unable to formulate d_i as a composition of the durations d_i^k of base LMDPs, and hence the duration would have to be recomputed each time we resolve a subtask $\mathcal{L}_i$, which is inefficient.

Instead, we propose the procedure described in Algorithm 1. The intuition is that in each iteration, we first solve the subtasks for the latest estimate of the exponentiated gain $\widehat{\Gamma}$. For this, we use (13) with the appropiate definition of ρ to solve the base LMDPs. We then apply (15) restricted to $\mathcal{E}$ to obtain an estimate of the value for the exit states. This yields the system of linear equations

$$\mathbf{z}_{\mathcal{E}} = G_{\mathcal{E}} \mathbf{z}_{\mathcal{E}}.$$

Here, the matrix $G_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$ contains the optimal values of the base LMDPs and has elements defined as in (15). We use the previously introduced idea to transform the ALMDP $\mathcal{L}$ to a first-exit LMDP $\mathcal{L}'$, and find the optimal gain Γ using binary search. We keep a reference state $s^\star \in \mathcal{S}$ and use the test described above to decide how to update the search interval.

Algorithm 1 Eigenvector approach to solving a hierarchical ALMDP.

Input: $\mathcal{L}, \mathcal{S}_1, \ldots, \mathcal{S}_L, \mathcal{E}, \epsilon, \eta$
lo $\leftarrow 0$, hi $\leftarrow 1$
while hi $-$ lo $> \epsilon$ **do**
  $\widehat{\Gamma} \leftarrow$ (hi + lo) / 2
  Solve the base LMDPs $\mathcal{L}_i^1, \ldots, \mathcal{L}_i^n$ for each equivalence class $\mathcal{C}_i$ and $\widehat{\Gamma}$
  Form the matrix $G_{\mathcal{E}}$ from the optimal value functions of the base LMDPs
  Solve the system of equations $\widehat{\mathbf{z}}_{\mathcal{E}} = G_{\mathcal{E}} \widehat{\mathbf{z}}_{\mathcal{E}}$
  if $\widehat{\Gamma} \widehat{z}_{\mathcal{E}}(s^*) > e^{\eta \mathcal{R}(s^*)} \sum_{s \in \mathcal{S}} \mathcal{P}(s|s^\star) \widehat{z}_{\mathcal{E}}(s)$ **then**
    hi $\leftarrow \widehat{\Gamma}$
  else
    lo $\leftarrow \widehat{\Gamma}$
return value functions of all base LMDPs, $\widehat{\mathbf{z}}_{\mathcal{E}}$

Theorem 1. *Under mild assumptions, Algorithm 1 converges to the optimal value function of $\mathcal{L}$.*

The proof of Theorem 1 appears in Appendix A.

Online algorithm. In the online case (see Algorithm 2), the agent keeps an estimate of the exponentiated gain $\widehat{\Gamma} = e^{\eta \hat{\rho}}$ which is updated every timestep. It also keeps estimates of the value functions of the base LMDPs $\hat{z}_i^1(\cdot; \hat{\rho}), \ldots, \hat{z}_i^n(\cdot; \hat{\rho})$ for each equivalence class $\mathcal{C}_i$, and estimates of the value function on exit states $\hat{z}_{\mathcal{E}}$. All the base LMDPs of the same equivalence class can be updated with the same sample using intra-task learning with the appropriate *importance sampling weights* [Jonsson and Gómez, 2016]. For the estimates of the exit states, we only update them upon visitation of such states. In that case, we use the compositionality expression in (15) to derive the following update:

$$\hat{z}_{\mathcal{E}}(s) \leftarrow (1 - \alpha_\ell) \hat{z}_{\mathcal{E}}(s) + (1 - \alpha_\ell) \sum_{k=1}^{n} \hat{z}_{\mathcal{E}}(\tau^k) \hat{z}_i^k(s; \rho). \tag{16}$$

Here, α_ℓ is the learning rate. Each of the learned components (i.e., gain, base LMDPs and exit state value estimates) maintain independent learning rates.

Algorithm 2 Online algorithm.

Input: $\mathcal{L}, \mathcal{S}_1, \ldots, \mathcal{S}_L, \mathcal{E}, s_0, \epsilon, \eta$
$t \leftarrow 0, \widehat{\Gamma} \leftarrow 1$
while *not terminate* **do**
  Observe $(s_{t+1}, r_t, s_t) \sim \hat{\pi}_t$
  Update the base LMDPs $\hat{z}_i^1(\cdot; \hat{\rho}), \ldots, \hat{z}_i^n(\cdot; \hat{\rho})$ using intra-task learning and equation (11)
  Update $\widehat{\Gamma}$ using equation (12)
  if $s_t \in \mathcal{E}$ **then**
    Update $\hat{z}_{\mathcal{E}}(s_t)$ using (16)

7 Experiments

In this section we describe the empirical results of the our proposed online algorithm and compare it against *flat* differential Z-learning. We measure the Mean Absolute Error (MAE) between the estimated value function $\hat{z}$ and the true optimal value function z. For each algorithm, we report mean and standard deviation on five seeds. The learning rates have been optimized independently for each of the instances. We adapt two episodic benchmark tasks [Infante et al., 2022] and transform them into infinite-horizon tasks as follows:

N-room domain, cf. Example 1. As in the episodic case, there are some 'goal' states with high reward (i.e. 0). When the agent enters a goal state, the next action causes it to receive the given reward and transition to a *restart* location. We vary the size of the rooms as well as the size of the rooms.

Taxi domain. In this variant of the original domain [Dietterich, 2000], once the passenger has been dropped off, the system transitions to a state in which the driver is in the last drop-off location, but a new passenger appears randomly at another location.

Figures 2 and 3 show the results. Our algorithm is able to speed up learning and converges to the optimal solution faster than flat average-reward reinforcement learning (note the log scale). This is in line with previous results for the episodic case [Infante et al., 2022]. The difference in the error scale in the figures is due to the initialization of the base LMDPs. The average reward setting poses an extra challenge since the 'sparsity' of the reward can make the estimates of the gain oscillate. This ultimately has an impact on the estimates of the base LMDPs and the value estimates of the exit states, and it is likely the reason why in Figure 3 the error increases before decreasing down to zero.

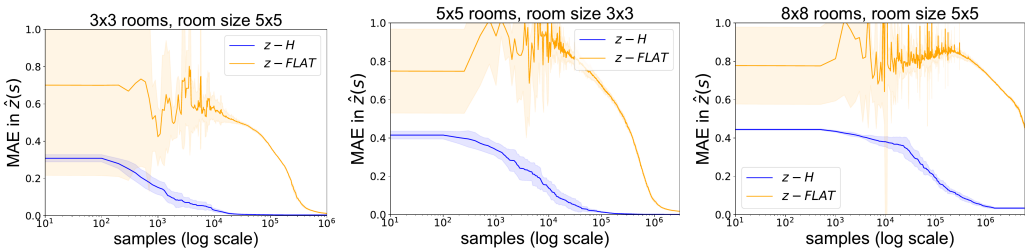

Figure 2: Results in N-room when varying the number of rooms and the size of the rooms.

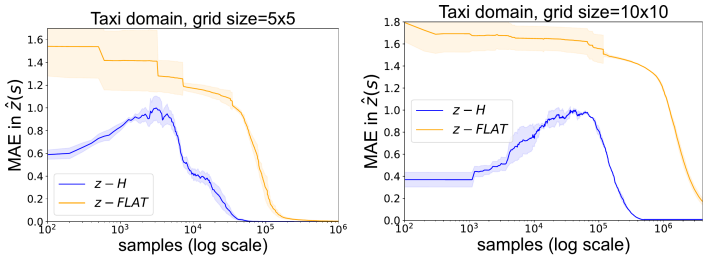

Figure 3: Results for 5×5 (left) and 10×10 (right) grids of the Taxi domain.

8 Conclusion

In this paper we present a novel framework for hierarchical average-reward reinforcement learning which makes it possible to learn the low-level and high-level tasks simultaneously. We propose an eigenvector approach and an online algorithm for solving problems in our framework, and show that the former converges to the optimal value function. In the future we would like to prove convergence also for the proposed online algorithm.

Acknowledgments and Disclosure of Funding

Anders Jonsson is partially funded by TAILOR (EU H2020 #952215), AGAUR SGR and Spanish grant PID2019-108141GB-I00. This publication is part of the action CNS2022-136178 financed by MCIN/AEI/10.13039/501100011033 and by the European Union Next Generation EU/PRTR

References

André Barreto, Diana Borsa, Shaobo Hou, Gheorghe Comanici, Eser Aygün, Philippe Hamel, Daniel Toyama, Jonathan J. Hunt, Shibl Mourad, David Silver, and Doina Precup. The Option Keyboard: Combining Skills in Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 32, pages 13031–13041, 2019.

Andrew Barto and Sridhar Mahadevan. Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13:41–77, 2003.

Peter Dayan and Geoffrey E Hinton. Feudal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 5, pages 271–278, 1992.

T. G. Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

Ronan Fruit and Alessandro Lazaric. Exploration-Exploitation in MDPs with Options. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 576–584, 2017.

Ronan Fruit, Matteo Pirotta, Alessandro Lazaric, and Emma Brunskill. Regret Minimization in MDPs with Options without Prior Knowledge. In *Advances in Neural Information Processing Systems*, volume 30, pages 3166–3176, 2017.

Mohammad Ghavamzadeh and Sridhar Mahadevan. Hierarchical Average Reward Reinforcement Learning. *Journal of Machine Learning Research*, 8:2629–2669, 2007.

Jonathan Hunt, Andre Barreto, Timothy Lillicrap, and Nicolas Heess. Composing entropic policies using divergence correction. In *International Conference on Machine Learning*, pages 2911–2920. PMLR, 2019.

Guillermo Infante, Anders Jonsson, and Vicenç Gómez. Globally Optimal Hierarchical Reinforcement Learning for Linearly-Solvable Markov Decision Processes. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pages 6970–6977, 2022.

Anders Jonsson and Vicenç Gómez. Hierarchical Linearly-Solvable Markov Decision Problems. *Proceedings of the 26th International Conference on Automated Planning and Scheduling*, pages 193–201, 2016.

Hilbert J. Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine Learning*, 87(2):159–182, May 2012. ISSN 1573-0565.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Ofir Nachum, Shixiang (Shane) Gu, Honglak Lee, and Sergey Levine. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang (Shane) Gu, Honglak Lee, and Sergey Levine. Why Does Hierarchy (Sometimes) Work So Well in Reinforcement Learning? *CoRR*, abs/1909.10618, 2019.

Geraud Nangue Tasse, Steven James, and Benjamin Rosman. A boolean task algebra for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:9497–9507, 2020.

Ronald Parr and Stuart Russell. Reinforcement Learning with Hierarchies of Machines. In *Advances in Neural Information Processing Systems*, volume 10, 1997.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

Andrew M. Saxe, Adam C. Earle, and Benjamin Rosman. Hierarchy through composition with multitask LMDPs. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3017–3026, 2017.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.*, 112(1-2):181–211, 1999.

Emanuel Todorov. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems*, volume 19, 2006.

Emanuel Todorov. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478–11483, 2009a.

Emanuel Todorov. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems*, volume 22, 2009b.

Benjamin van Niekerk, Steven D. James, Adam Christopher Earle, and Benjamin Rosman. Composing value functions in reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 6401–6409, 2019.

Yi Wan, Abhishek Naik, and Richard S Sutton. Learning and Planning in Average-Reward Markov Decision Processes. In *Proceedings of the 38th International Conference on Machine Learning*, pages 10653–10662, 2021a.

Yi Wan, Abhishek Naik, and Richard S. Sutton. Average-Reward Learning and Planning with Options. In *Advances in Neural Information Processing Systems*, volume 34, pages 22758–22769, 2021b.

Zheng Wen, Doina Precup, Morteza Ibrahimi, Andre Barreto, Benjamin Van Roy, and Satinder Singh. On Efficiency in Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 6708–6718, 2020.

A Proof of Theorem 1

The proof idea is to directly exploit the proof of convergence for first-exit hierarchical LMDPs [Infante et al., 2022]. However, the existing proof assumes that rewards are bounded in $(-\infty, 0]$. In our case, the reward function of each subtask $\mathcal{L}_i$ is given by

$$\mathcal{R}_i(s) = \mathcal{R}(s) - \hat{\rho},$$

where $\hat{\rho}$ is the current estimate of the optimal gain. Even if $\mathcal{R}(s)$ is bounded in $(-\infty, 0]$, $\mathcal{R}_i(s)$ is larger than 0 in states such that $\hat{\rho} < \mathcal{R}(s)$.

In sparse-reward tasks such as our benchmark domains, we can just select the reference state s^* to be the only state with high reward, and then the condition on $\mathcal{R}_i$ is satisfied for all other states, which implies that the algorithm converges to the optimal value function for $\hat{\rho}$. Since binary search will find the optimal exponentiated gain $\Gamma = e^{\eta\rho}$, the algorithm converges to the optimal value function of $\mathcal{L}$ to arbitrary precision.

In the case of more complicated reward functions, we can instead partition the states according to the current gain estimate $\hat{\rho}$. Concretely, we designate as a terminal state each state $s \in \mathcal{S}$ such that $\hat{\rho} < \mathcal{R}(s)$. We then run the hierarchical algorithm for $\hat{\rho}$ and test the condition on line 8 of Algorithm 1 for an arbitrary terminal state. This ensures that the first-exit algorithm converges, and provides the desired condition for binary search to work.