

Hierarchical Average-Reward Linearly-solvable Markov Decision Processes

Guillermo Infante^{a,*}, Anders Jonsson^a and Vicenç Gómez^a

^aAI&ML research group, Universitat Pompeu Fabra, Barcelona, Spain

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 (RL) [4, 18, 21, 5, 3] 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 [15, 28]. Hierarchical RL has also been credited with other advantages, e.g. the ability to explore more efficiently [16].

Most previous work on hierarchical RL considers either the finite-horizon setting or the infinite-horizon setting with discounted rewards. The average-reward setting is better suited for cyclical tasks characterized by continuous experience. In the few works on hierarchical RL in the average-reward setting, either the low-level tasks are assumed to be solved beforehand [6, 7, 27] or they have important restrictions that severely reduce their applicability, e.g. a single initial state [8]. It is therefore an open question how to develop algorithms for hierarchical RL in the average-reward setting in order to learn the low-level and high-level tasks simultaneously.

In this paper we propose a novel framework for hierarchical RL 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) [22]. LMDPs are a class of restricted MDPs for which the Bellman equation can be exactly transformed into a linear equation. This class of problems plays a key role in the framework of RL as probabilistic inference [13, 14]. One of the properties of LMDPs is compositionality: one can compute the solution to a novel task from the solutions to previously solved tasks without learning [24]. Such a property has been exploited in recent works about compositionality in RL [9, 25, 17] and in

combination with Hierarchical RL in the finite-horizon setting [12, 20, 10]. Adapting this idea to the average-reward setting requires careful analysis.

Unlike most frameworks for hierarchical RL, 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.

Our work makes the following novel contributions:

- Learning low-level and high-level tasks simultaneously in the average-reward setting, without imposing additional restrictions on the low-level tasks.
- Two novel algorithms for solving hierarchical RL in the average reward setting: the first one is based on the eigenvector approach used for solving LMDPs. The second is an online variant in which an agent learns simultaneously the low-level and high-level tasks.
- Two main theoretical contributions LMDPs: a converge proofs for both differential soft TD-learning for (non-hierarchical) LMDPs and also for the eigenvector approach in the hierarchical case.

To the best of our knowledge, this work is the first that extends the combination of compositionality and hierarchical RL to the average-reward setting.

2 Related work

Most research on hierarchical RL formulates problems as a Semi-Markov Decision Process (SMDP) with options [21] or the MAXQ decomposition [5].

Fruit and Lazaric [6] and [7] 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. [27] 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 [8] propose a framework for hierarchical average-reward RL 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. [28] present an approach for partitioning the state space and exploiting the equivalence of low-level tasks, similar to our work.

* Corresponding Author. Email: guillermo.infante@upf.edu

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. [10] combine the compositionality of LMDPs with the equivalence of low-level tasks to develop a framework for hierarchical RL in the finite-horizon setting. Compositionality is also a key idea in the option keyboard [2], 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 [22]) 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, $\mathcal{P} : \mathcal{S} \rightarrow \Delta(\mathcal{S}^+)$ is the passive dynamics, which describes state transitions in the absence of controls, $\mathcal{R} : \mathcal{S} \rightarrow \mathbb{R}$ is a reward function on non-terminal states, and $\mathcal{J} : \mathcal{T} \rightarrow \mathbb{R}$ is a reward function on terminal states. We use $\mathcal{S}^+ = \mathcal{S} \cup \mathcal{T}$ to denote the full set of states. An agent interacts with the environment following a policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{S}^+)$. 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 passive 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:

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

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 $v_\eta^* : \mathcal{S} \rightarrow \mathbb{R}$ among all policies. 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 [22]:

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

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

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

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

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

The system of linear equations in (3) can then be written in matrix form when we know the passive 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

$$\mathbf{z} = RP\mathbf{z}^+. \quad (5)$$

We can use the power iteration method over (5) to obtain the solution for \mathbf{z} [22]. Power iteration is guaranteed to converge as long as the diagonal matrix R is not too large, and a common assumption is that the rewards of non-terminal states are non-positive (i.e. $\mathcal{R}(s) \leq 0$ for each $s \in \mathcal{S}$). However, we refrain from making any such assumptions, and later we instead prove convergence in an alternative way.

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 the estimated policy $\hat{\pi}$ derived from \hat{z} using (4). The update rule for the so-called Z-learning algorithm is given by

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

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 [24]. 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, \dots, z_K be the optimal value functions of $\mathcal{L}_1, \dots, \mathcal{L}_K$. 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:

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

then by linearity of the value function, the same expression holds for non-terminal states [24]:

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

3.2 Hierarchical Decomposition for LMDPs

Infante et al. [10] 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 for which

- 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 .
- $\mathcal{P}_i : \mathcal{S}_i \rightarrow \Delta(\mathcal{S}_i^+)$ and $\mathcal{R}_i : \mathcal{S}_i \rightarrow \mathbb{R}$ 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 the terminal states \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* $\mathcal{E} = \bigcup_{i=1}^L \mathcal{T}_i$ is the union of the terminal states of each subtask in $\{\mathcal{L}_1, \dots, \mathcal{L}_L\}$. We 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 $f : \mathcal{S}_i \rightarrow \mathcal{S}_j$ 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 $\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_C\}$, $C \leq L$, i.e. a partition of the set of subtasks $\{\mathcal{L}_1, \dots, \mathcal{L}_L\}$ 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. [10] 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, \dots, z_j^n be the optimal value functions of the base LMDPs for \mathcal{L}_j , with $n = |\mathcal{T}_j|$. 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 $z_{\mathcal{E}} : \mathcal{E} \rightarrow \mathbb{R}$, 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 Processes

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,

$\mathcal{P} : \mathcal{S} \rightarrow \Delta(\mathcal{S})$ is the passive dynamics, which describes state transitions in the absence of controls, and $\mathcal{R} : \mathcal{S} \rightarrow \mathbb{R}$ is a reward function. Since ALMDPs represent infinite-horizon tasks, there are no terminal states.

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

Assumption 1. The ALMDP \mathcal{L} is communicating [19]: for each pair of states $s, s' \in \mathcal{S}$, there exists a policy π that has non-zero probability of reaching s' from s .

Assumption 2. The ALMDP \mathcal{L} is unichain [19]: the transition probability distribution induced by all stationary policies admit a single recurrent class.

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

$$v_{\pi}(s) = \lim_{T \rightarrow \infty} \mathbb{E} \left[\frac{1}{T} \sum_{t=0}^T \mathcal{R}(S_t, S_{t+1}, \pi) \mid \pi, S_0 = s \right], \quad (7)$$

where $\mathcal{R}(s_t, s_{t+1}, \pi)$ is defined as for first-exit LMDPs. Again, we are interested in obtaining the *optimal* value function v . Under Assumption 2, the Bellman optimality equations can be written as

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

where ρ is the optimal one-step average reward (i.e. gain), which is state-independent for unichain ALMDPs [22]. Exponentiating yields

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

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). Similar to the first-exit case, Equation (9) can be expressed in matrix form as

$$\Gamma \mathbf{z} = R P \mathbf{z}, \quad (10)$$

where the matrices $P \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ and $R \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ are appropriately defined as in (5). The exponentiated gain Γ can be shown to correspond to the largest eigenvalue of RP [23]. An ALMDP can be solved using *relative value iteration* by selecting a reference state $s^* \in \mathcal{S}$, initializing $\hat{\mathbf{z}}_0 = \mathbf{1}$ and iteratively applying

$$\hat{\mathbf{z}}_{k+\frac{1}{2}} \leftarrow R P \hat{\mathbf{z}}_k, \quad \hat{\mathbf{z}}_{k+1} \leftarrow \hat{\mathbf{z}}_{k+\frac{1}{2}} / \hat{z}_{k+\frac{1}{2}}(s^*).$$

The reference state s^* satisfies $z(s^*) = 1$, which makes the optimal value z unique (else any constant shift preserves optimality). After convergence, the exponentiated gain equals $\Gamma = \hat{z}_{k+\frac{1}{2}}(s^*)$. Under Assumption 1, relative value iteration converges to the unique optimal value z [23].

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 $\mathcal{J}'(s^*) = 0$ (implying $z(s^*) = 1$). 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 obtain an estimate \hat{v} of the optimal value v and an estimate $\hat{\rho}$ of the optimal gain ρ using *differential soft TD-learning*, similar to differential Q-learning [26]. We collect samples (s_t, r_t, s_{t+1}) generated by the estimated policy $\hat{\pi}$ derived from \hat{v} as in (4), and derive update rules for \hat{v} and $\hat{\rho}$ from (8) as follows:

$$\hat{v}_{t+1}(s_t) \leftarrow \hat{v}_t(s_t) + \alpha_t \delta_t, \quad (11)$$

$$\hat{\rho}_{t+1} \leftarrow \hat{\rho}_t + \lambda \alpha_t \delta_t. \quad (12)$$

Here, the TD error δ_t is given by

$$\begin{aligned} \delta_t &= r_t - \hat{\rho}_t - \frac{1}{\eta} \log \frac{\hat{\pi}_t(s_{t+1}|s_t)}{\mathcal{P}(s_{t+1}|s_t)} + \hat{v}_t(s_{t+1}) - \hat{v}_t(s_t) \\ &= r_t - \hat{\rho}_t + \frac{1}{\eta} \log \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s_t) e^{\eta \hat{v}_t(s')} - \hat{v}_t(s_t). \end{aligned}$$

Note that both updates use the same TD error. At any time, we can retrieve estimates of \hat{z} and $\hat{\Gamma}$ by exponentiating \hat{v} and $\hat{\rho}$, respectively.

Theorem 3. *Under mild assumptions, differential soft TD-learning in (11) and (12) converges to the optimal values of v and ρ in \mathcal{L} .*

Proof sketch. The proof is adapted from the proof of convergence of differential Q-learning [1, 26], which requires the ALMDP to be communicating (Assumption 1). Define a Bellman operator T as

$$T(v)(s) = \mathcal{R}(s) + \frac{1}{\eta} \log \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s) e^{\eta v(s')}.$$

To adapt the previous proof, it is sufficient to show that T is a non-expansion in the max norm, i.e. $\|T(x) - T(y)\|_\infty \leq \|x - y\|_\infty$ for each $x, y \in \mathbb{R}^{|\mathcal{S}|}$, and that T satisfies $T(x + c\mathbb{1}) = T(x) + c\mathbb{1}$ for each $x \in \mathbb{R}^{|\mathcal{S}|}$ and constant $c \in \mathbb{R}$, where $\mathbb{1}$ is the vector of $|\mathcal{S}|$ ones. To see the full proof we refer the reader to the Appendix in [11]. \square

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 $\langle \mathcal{S}, \mathcal{P}, \mathcal{R} \rangle$. Similarly to Infante et al. [10], 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 $\mathcal{T}_i = \{\tau \in \mathcal{S} \setminus \mathcal{S}_i : \exists s \in \mathcal{S}_i, \mathcal{P}(\tau|s) > 0\}$ 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 $\mathcal{R} - \hat{\rho}$ onto \mathcal{S}_i , where $\hat{\rho}$ is a gain estimate.
- \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

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

By inspection of the Bellman optimality equations in (9) and (13), they are equal if $\mathcal{R}_i(s) = \mathcal{R}(s) - \rho$. 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 value estimates $\hat{z}(\tau)$ for each $\tau \in \mathcal{T}_j$ and the gain estimate $\hat{\rho}$. Every time that $\hat{z}(\tau)$, $\tau \in \mathcal{T}_i$, and $\hat{\rho}$ change, we obtain a new value estimate for each $s \in \mathcal{S}_i$ by solving the subtask for the new parameters.

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 $\{\mathcal{L}_i^1, \dots, \mathcal{L}_i^n\}$ 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 $\mathcal{R}_i(s) = \mathcal{R}(s) - \hat{\rho}$ for all $s \in \mathcal{S}_i$, while at terminal states $\tau \in \mathcal{T}_i$ we let the reward function be $z_i^k(\tau; \hat{\rho}) = 1$ if $\tau = \tau^k$ and $z_i^k(\tau; \hat{\rho}) = 0$ otherwise. Thus, the base LMDPs are parameterized by the gain estimate $\hat{\rho}$. 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 $z_i^1(\cdot; \rho), \dots, z_i^n(\cdot; \rho)$ 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:

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

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 subtask state $s \in \mathcal{S}_i$ as

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

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.

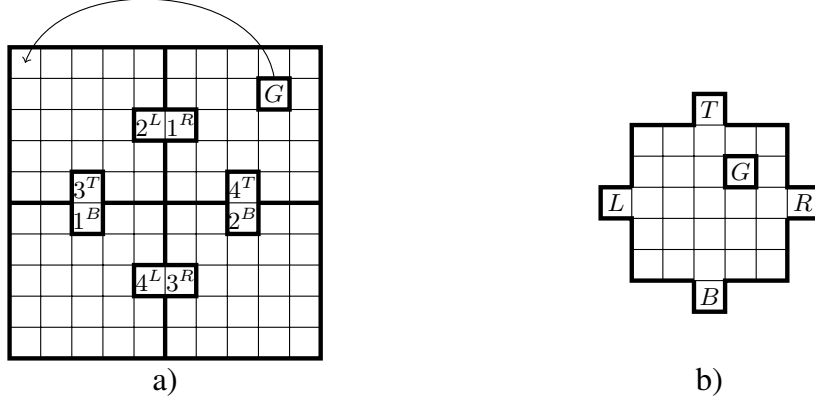


Figure 1: a) An example 4-room ALMDP; b) a single subtask with 5 terminal states G, L, R, T, B that is equivalent to all 4 room subtasks. Rooms are numbered 1 through 4, left-to-right, then top-to-bottom, and exit state 1^B refers to the exit B of room 1, etc.

5.3 Efficiency of the value representation

Similar to previous work [28, 10] we can exploit the equivalence of subtasks to learn more efficiently. Let $\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_C\}$, $C \leq L$, be a set of equivalence classes, i.e. a partition of the set of subtasks $\{\mathcal{L}_1, \dots, \mathcal{L}_L\}$ 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 $\hat{z}_{\mathcal{E}} : \mathcal{E} \rightarrow \mathbb{R}$ and a value function for each base LMDP of each equivalence class. This is enough to represent the value for any state $s \in \mathcal{S}$ 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 $s \notin \mathcal{E}$.

6.1 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 $\hat{\rho}$, which is initially unknown.

Our proposed algorithm appears in Algorithm 1. The intuition is that in each iteration, we first solve the subtasks for the latest estimate of the exponentiated gain $\hat{\Gamma}$. For this, we use (13) with the current value of $\hat{\rho}$ 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

$$\mathbf{z}_{\mathcal{E}} = G_{\mathcal{E}} \mathbf{z}_{\mathcal{E}}. \quad (16)$$

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}' parameterized on the estimated gain $\hat{\rho}$, and find the optimal exponentiated gain $\hat{\Gamma}$ using binary search. We keep a reference state $s^* \in \mathcal{S}$ (which is by definition an exit state) and use the test described above to decide how to update the search interval.

Theorem 4. *Algorithm 1 converges to the optimal value function z of \mathcal{L} as $\epsilon \rightarrow 0$.*

First note that the optimal value function z of \mathcal{L} exists and is unique due to Assumption 1. Due to the equivalence between \mathcal{L} and the corresponding first-exit LMDP \mathcal{L}' , this implies that \mathcal{L}' has a unique solution $z(\cdot; \rho)$ when the estimated gain $\hat{\rho}$ equals ρ , and that this solution equals $z(\cdot; \rho) = z$, the optimal solution to \mathcal{L} .

Lemma 5. *Given a first-exit LMDP \mathcal{L}' parameterized on $\hat{\rho}$, the optimal value $z(s; \hat{\rho})$ of each non-terminal state $s \in \mathcal{S}$ is strictly monotonically decreasing in $\hat{\rho}$.*

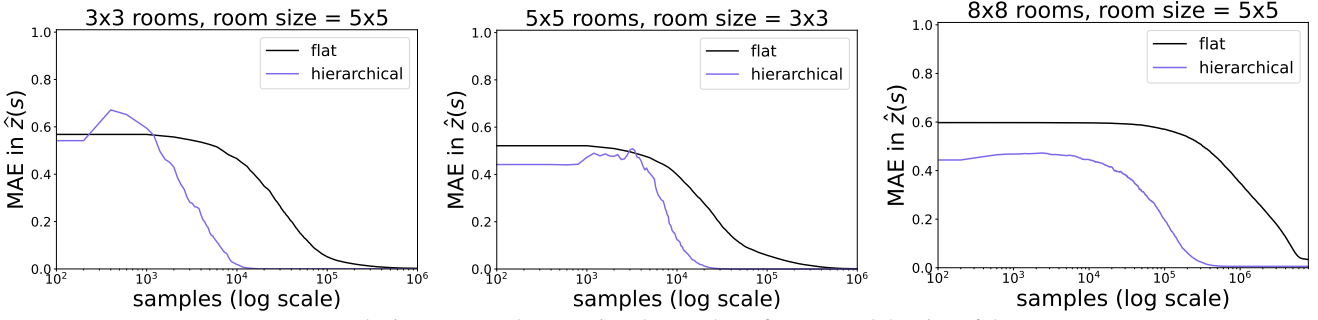


Figure 2: Results in N-room when varying the number of rooms and the size of the rooms.

Proof. Strict monotonicity requires that there exists $\varepsilon > 0$ such that $z(s; \hat{\rho} - \varepsilon) > z(s; \hat{\rho}) > z(s; \hat{\rho} + \varepsilon)$ when $\varepsilon \rightarrow 0$. We prove the first inequality by induction; the second is analogous. The base case is given by the terminal states $\tau \in \mathcal{T}$, for which $z(\tau; \hat{\rho} - \varepsilon) = z(\tau; \hat{\rho})$. The inductive case is given by

$$\begin{aligned} z(s; \hat{\rho} - \varepsilon) &= e^{\eta(\mathcal{R}(s) - (\hat{\rho} - \varepsilon))} \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s) z(s'; \hat{\rho} - \varepsilon) \\ &\geq e^{\eta\varepsilon} e^{\eta(\mathcal{R}(s) - \hat{\rho})} \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s) z(s'; \hat{\rho}) \\ &= e^{\eta\varepsilon} z(s; \hat{\rho}) > z(s; \hat{\rho}). \end{aligned}$$

This concludes the proof. \square

As a consequence of Lemma 5, \mathcal{L}' has a unique solution $z(\cdot, \hat{\rho})$ for each $\hat{\rho} \geq \rho$, since the values $z(\cdot, \hat{\rho})$ decrease as $\hat{\rho}$ increases. In contrast, there may be values of $\hat{\rho} > \rho$ for which power iteration does not converge.

Lemma 6. *Given a subtask \mathcal{L}_i , if the optimal value of each terminal state $\tau \in \mathcal{T}_i$ equals its optimal value in \mathcal{L} , i.e. $z_i(\tau) = z(\tau)$, and the optimal gain ρ in \mathcal{L} is known, then the optimal value of each non-terminal state $s \in \mathcal{S}_i$ is unique and equals $z_i(s) = z(s)$.*

Proof. Since \mathcal{R}_i and \mathcal{P}_i are restrictions of $\mathcal{R} - \rho$ and \mathcal{P} , respectively, to \mathcal{S}_i , we have

$$z_i(s) = e^{\eta\mathcal{R}_i(s)} \sum_{s'} \mathcal{P}_i(s'|s) z_i(s') = e^{\eta(\mathcal{R}(s) - \rho)} \sum_{s'} \mathcal{P}(s'|s) z_i(s'),$$

which is the same Bellman equation as for $z(s)$. Assuming that $z_i(\tau) = z(\tau)$ for each $\tau \in \mathcal{T}$, directly yields that $z_i(s) = z(s)$ for each non-terminal state $s \in \mathcal{S}_i$. \square

Corollary 7. *If the optimal gain ρ in \mathcal{L} is known, each base LMDP \mathcal{L}_j^k has a unique solution $z_j^k(\cdot; \rho)$.*

Algorithm 1 Eigenvector approach to solving a hierarchical ALMDP.

```

1: Input:  $\mathcal{L}, \mathcal{S}_1, \dots, \mathcal{S}_L, \mathcal{E}, \epsilon, \eta$ 
2:  $\text{lo} \leftarrow 0, \text{hi} \leftarrow 1$ 
3: while  $\text{hi} - \text{lo} > \epsilon$  do
4:    $\hat{\Gamma} \leftarrow (\text{hi} + \text{lo}) / 2$ 
5:   Solve base LMDPs  $\mathcal{L}_j^1, \dots, \mathcal{L}_j^n$  for each equivalence class  $\mathcal{C}_j$ 
6:   Form the matrix  $G_{\mathcal{E}}$  from the optimal value functions
7:   Solve the system of equations  $\hat{\mathbf{z}}_{\mathcal{E}} = G_{\mathcal{E}} \hat{\mathbf{z}}_{\mathcal{E}}$ 
8:   if  $\hat{\Gamma} \hat{\mathbf{z}}_{\mathcal{E}}(s^*) > e^{\eta\mathcal{R}(s^*)} \sum_{s \in \mathcal{S}} \mathcal{P}(s|s^*) \hat{\mathbf{z}}_{\mathcal{E}}(s)$  then
9:      $\text{hi} \leftarrow \hat{\Gamma}$ 
10:  else
11:     $\text{lo} \leftarrow \hat{\Gamma}$ 
return value functions of all base LMDPs,  $\hat{\mathbf{z}}_{\mathcal{E}}$ 

```

Proof. From (15), the optimal values of subtask states satisfy

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

Due to Lemma 6, the optimal value $z(s)$ is unique, which is only possible if $z_i^k(s; \rho)$ is unique for each $\tau^k \in \mathcal{T}_i$. \square

Combined with Lemma 5, Corollary 7 implies that each base LMDP \mathcal{L}_j^k has a unique solution $z_j^k(\cdot; \hat{\rho})$ whenever $\hat{\rho} \geq \rho$.

Lemma 8. *For $\hat{\rho} \geq \rho$, the equation $\hat{\mathbf{z}}_{\mathcal{E}} = G_{\mathcal{E}} \hat{\mathbf{z}}_{\mathcal{E}}$ has a unique solution that equals $z_{\mathcal{E}}(\tau) = z(\tau; \hat{\rho})$ for each exit $\tau \in \mathcal{E}$, where $z(\cdot; \hat{\rho})$ is the unique value of the first-exit LMDP \mathcal{L}' for $\hat{\rho}$.*

Proof. At convergence, due to (15) it has to hold for each non-terminal exit $\tau \in \mathcal{E}$ that

$$z_{\mathcal{E}}(\tau) = \sum_{k=1}^n z_{\mathcal{E}}(\tau^k) z_i^k(s; \hat{\rho}) \quad \forall s \in \mathcal{S}_i,$$

where each τ^k is also an exit and $z_i^k(s; \hat{\rho})$ is well-defined and unique since $\hat{\rho} \geq \rho$. This equation is satisfied when $z_{\mathcal{E}}(\tau) = z(\tau; \hat{\rho})$ for each exit. Since $z(\cdot; \hat{\rho})$ is unique, this is the only solution. \square

We now have all the ingredients to prove Theorem 4. When $\hat{\rho} \geq \rho$ (or equivalently, $\hat{\Gamma} \geq \Gamma$), each base LMDP has a unique solution, and $z_{\mathcal{E}}$ is also unique. Moreover, when $\hat{\rho} > \rho$, the condition on line 8 is true, which causes binary search to discard all values greater than $\hat{\rho}$. If the base LMDPs or $z_{\mathcal{E}}$ do not have a unique solution, we know that $\hat{\rho}$ is too small, and we can hence discard all values less than $\hat{\rho}$. Since the solution $z(\cdot; \hat{\rho})$ is monotonically decreasing in $\hat{\rho}$, binary search is guaranteed to find the optimal gain ρ within a factor of ϵ .

6.2 Online algorithm

In the online case (see Algorithm 2), the agent keeps an estimate of the exponentiated gain $\hat{\Gamma} = e^{\eta\hat{\rho}}$ which is updated every timestep. It also keeps estimates of the value functions of the base LMDPs $\hat{z}_i^1(\cdot; \hat{\rho}), \dots, \hat{z}_i^n(\cdot; \hat{\rho})$ 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* [12]. 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:

$$\hat{z}_{\mathcal{E}}(s) \leftarrow (1 - \alpha_{\mathcal{E}}) \hat{z}_{\mathcal{E}}(s) + \alpha_{\mathcal{E}} \sum_{k=1}^n \hat{z}_{\mathcal{E}}(\tau^k) \hat{z}_i^k(s; \rho). \quad (17)$$

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.

```

1: Input:  $\mathcal{L}, \mathcal{S}_1, \dots, \mathcal{S}_L, \mathcal{E}, s_0, \epsilon, \eta$ 
2:  $t \leftarrow 0, \hat{\Gamma} \leftarrow 1$ 
3: while not terminate do
4:   Observe  $(s_t, r_t, s_{t+1}) \sim \hat{\pi}_t$ 
5:   Update  $\hat{z}_j^1(\cdot; \hat{\rho}), \dots, \hat{z}_j^n(\cdot; \hat{\rho})$  using Equation (11)
6:   Compute  $\hat{z}(s_t)$  and  $\sum_{s'} \mathcal{P}(s'|s_t) \hat{z}(s')$  using Equation (15)
7:   Update  $\hat{\Gamma}$  using Equation (12)
8:   if  $s_t \in \mathcal{E}$  then
9:     Update  $\hat{z}_\mathcal{E}(s_t)$  using Equation (17)
```

7 Experiments

In this section we compare experimentally Algorithm 2 with differential soft TD-learning in the flat representation of the ALMDP (code available at [11]). 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 [10] 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 number of rooms as well as the size of the rooms.

Taxi domain. In this variant of the original domain [5], 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 [10]. 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.

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 eigen-vector approach and an online algorithm for solving problems in our framework, and show that the former converges to the optimal value function. As a by-product of our analysis, we also provide a convergence theorem in the non-hierarchical case for average-reward LMDPs, which to the extent of our knowledge, was not previously done. In the future we would like to prove convergence also for the proposed online algorithm.

Acknowledgements

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

- [1] J. Abounadi, D. Bertsekas, and V. S. Borkar. Learning algorithms for markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40(3):681–698, 2001.
- [2] A. Barreto, D. Borsa, S. Hou, G. Comanici, E. Aygün, P. Hamel, D. Toyama, J. J. Hunt, S. Mourad, D. Silver, and D. Precup. The option keyboard: Combining skills in reinforcement learning. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13031–13041, 2019. URL <https://proceedings.neurips.cc/paper/2019/hash/251c5ff6b62cc21c446c963c76cf214-Abstract.html>.
- [3] A. Barto and S. Mahadevan. Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13:41–77, 2003.
- [4] P. Dayan and G. E. Hinton. Feudal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992. URL <https://proceedings.neurips.cc/paper/1992/hash/d14220ee66aee73c49038385428ec4c-Abstract.html>.
- [5] T. G. Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, Nov. 2000. ISSN 1076-9757. doi: 10.1613/jair.639. URL <https://www.jair.org/index.php/jair/article/view/10266>.
- [6] R. Fruit and A. Lazaric. Exploration-exploitation in mdps with options. In A. Singh and X. J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 576–584. PMLR, 2017. URL <http://proceedings.mlr.press/v54/fruit17a.html>.
- [7] R. Fruit, M. Pirota, A. Lazaric, and E. Brunskill. Regret Minimization in MDPs with Options without Prior Knowledge. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/90599c8fdd2f6e7a03ad173e2f535751-Abstract.html.

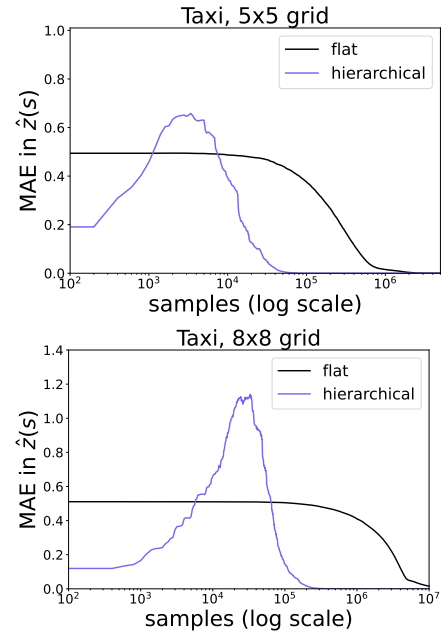


Figure 3: Results for 5×5 (top) and 8×8 (bottom) grids of the Taxi domain.

- [8] M. Ghavamzadeh and S. Mahadevan. Hierarchical average reward reinforcement learning. *Journal of Machine Learning Research*, 8(87): 2629–2669, 2007. URL <http://jmlr.org/papers/v8/ghavamzadeh07a.html>.
- [9] J. Hunt, A. Barreto, T. Lillicrap, and N. Heess. Composing Entropic Policies using Divergence Correction. pages 2911–2920. PMLR, May 2019. URL <https://proceedings.mlr.press/v97/hunt19a.html>.
- [10] G. Infante, A. Jonsson, and V. Gómez. Globally optimal hierarchical reinforcement learning for linearly-solvable markov decision processes. volume 36, pages 6970–6977, Jun. 2022. doi: 10.1609/aaai.v36i6.20655. URL <https://ojs.aaai.org/index.php/AAAI/article/view/20655>.
- [11] G. Infante, A. Jonsson, and V. Gómez. Hierarchical average-reward linearly-solvable markov decision processes. *arXiv preprint arXiv:2407.06690*, 2024. URL <https://arxiv.org/abs/2407.06690>.
- [12] A. Jonsson and V. Gómez. Hierarchical Linearly-Solvable Markov Decision Problems. *Proceedings of the International Conference on Automated Planning and Scheduling*, 26:193–201, Mar. 2016. ISSN 2334-0843. doi: 10.1609/icaps.v26i1.13750. URL <https://ojs.aaai.org/index.php/ICAPS/article/view/13750>.
- [13] H. J. Kappen, V. Gómez, and M. Opper. Optimal control as a graphical model inference problem. *Machine Learning*, 87(2):159–182, May 2012. ISSN 1573-0565. doi: 10.1007/s10994-012-5278-7. URL <https://doi.org/10.1007/s10994-012-5278-7>.
- [14] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.
- [15] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://papers.nips.cc/paper_files/paper/2018/hash/e6384711491713d29bc63fc5eeb5ba4f-Abstract.html.
- [16] O. Nachum, H. Tang, X. Lu, S. Gu, H. Lee, and S. Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019.
- [17] G. Nangue Tasse, S. James, and B. Rosman. A Boolean Task Algebra for Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 9497–9507. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/hash/6ba3af5d7b2790e73f0de32e5c8c1798-Abstract.html>.
- [18] R. Parr and S. Russell. Reinforcement Learning with Hierarchies of Machines. In *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1997. URL <https://proceedings.neurips.cc/paper/1997/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html>.
- [19] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. ISBN 978-0-47161977-2. doi: 10.1002/9780470316887. URL <https://doi.org/10.1002/9780470316887>.
- [20] A. M. Saxe, A. C. Earle, and B. Rosman. Hierarchy Through Composition with Multitask LMDPs. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3017–3026. PMLR, July 2017. URL <https://proceedings.mlr.press/v70/saxe17a.html>.
- [21] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1.
- [22] E. Todorov. Linearly-solvable markov decision problems. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL <https://proceedings.neurips.cc/paper/2006/file/d806ca13ca3449af72a1ea5aedbed26a-Paper.pdf>.
- [23] E. Todorov. Efficient computation of optimal actions. *Proceedings of the national academy of sciences*, 106(28):11478–11483, 2009.
- [24] E. Todorov. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://papers.nips.cc/paper_files/paper/2009/hash/3eb71f6293a2a31f3569e10af6552658-Abstract.html.
- [25] B. van Niekerk, S. D. James, A. C. Earle, and B. Rosman. Composing value functions in reinforcement learning. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6401–6409. PMLR, 2019. URL <http://proceedings.mlr.press/v97/van-niekerk19a.html>.
- [26] Y. Wan, A. Naik, and R. S. Sutton. Learning and planning in average-reward markov decision processes. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10653–10662. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/wan21a.html>.
- [27] Y. Wan, A. Naik, and R. S. Sutton. Average-reward learning and planning with options. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 22758–22769, 2021. URL <https://proceedings.neurips.cc/paper/2021/hash/c058f544c737782deacefa532d9add4c-Abstract.html>.
- [28] Z. Wen, D. Precup, M. Ibrahim, A. Barreto, B. Van Roy, and S. Singh. On Efficiency in Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 6708–6718. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/hash/4a5cfa9281924139db466a8a19291aff-Abstract.html>.