Sample Complexity of Hierarchical Decompositions in Markov Decision Processes

Arnaud Robert [1] Ciara Pike-Burke [2] Aldo A. Faisal [1]

Abstract

Hierarchical Reinforcement Learning (HRL) algorithms perform planning at multiple levels of abstraction. Algorithms that leverage states or temporal abstractions have empirically demonstrated a gain in sample efficiency. Yet, the basis of those efficiency gains is not fully understood and we still lack theoretically-grounded design rules to implement HRL algorithms. Here, we derive a lower bound on the sample complexity for the proposed class of goal-conditioned HRL algorithms (such as Dot-2-Dot (Beyret et al., 2019)) that inspires a novel Q-learning algorithm and highlights the relationship between the properties of the decomposition and the sample complexity. Specifically, the proposed lower bound on the sample complexity of such HRL algorithms allows to quantify the benefits of hierarchical decomposition. These theoretical findings guide the formulation of a simple Q-learning-type algorithm that leverages goal hierarchical decomposition. We then empirically validate our lower bound by investigating the sample complexity of the proposed hierarchical algorithm on a spectrum of tasks. Our tasks were designed to allow us to dial up or down their complexity over multiple orders of magnitude. Our theoretical and algorithmic results provide a clear step towards understanding the foundational question of quantifying the efficiency gains induced by hierarchies in reinforcement learning.

[1]Brain & Behaviour Lab: Department of Computing, Imperial College London, UK [2]Department of Mathematics, Imperial College London, UK. Correspondence to: Arnaud Robert <ar4220@ic.ac.uk>.

Workshop on New Frontiers in Learning, Control, and Dynamical Systems at the International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA, 2023. Copyright 2023 by the author(s).

1. Motivation

Hierarchical Reinforcement Learning (HRL) (Sutton et al., 1999b; Dayan & Hinton, 1992; Dietterich, 2000; Beyret et al., 2019) leverages the hierarchical decomposition of a problem to build algorithms that are more sample efficient. While there is significant empirical evidence that hierarchical implementations can drastically improve the sample efficiency of Reinforcement Learning (RL) algorithms (Nachum et al., 2018; 2019; Vezhnevets et al., 2017; Dayan & Hinton, 1992), there are also cases where temporal abstraction worsens the empirical sample complexity (Jong et al., 2008). Therefore, a natural question to ask is when does HRL lead to improved sample complexity and how much of an improvement can it provide?

Previous theoretical work on sample-complexity bounds in Machine Learning has been integral to the development of the field. Understanding sample complexity in RL is particularly beneficial because, in contrast to supervised learning, RL is substantially more expensive to train, more sample inefficient and uses more expensive interaction-type data.

In addition, theoretical results (such as (Dann & Brunskill, 2015; Li et al., 2022; Auer & Ortner, 2005; Jin et al., 2018; Sutton et al., 1999a)) often uncover interesting principles that are useful for improving algorithm design. For example, the analysis of the Q-learning algorithm in (Jin et al., 2018) improved our understanding of exploration strategies in model-free RL and the derivation of the policy gradient theorem (Sutton et al., 1999a) gave birth to a wide range of new RL methods. In contrast, there are few theoretical results in hierarchical RL and many key studies are empirical in nature, e.g. hierarchies of states (Dayan & Hinton, 1992; Dietterich et al., 1998), time (Precup & Sutton, 1997), or action (Vezhnevets et al., 2016; Pickett & Barto, 2002; Abramova et al., 2012).

To address this gap in the literature, we consider a tabular version of the goal-based approach to HRL (Nachum et al., 2018; Beyret et al., 2019) and we analyze the induced MDP decomposition to derive a lower bound on the sample complexity of this specific HRL framework. The lower bound we establish provides insights into the circumstances where a hierarchical decomposition proves advantageous. Those

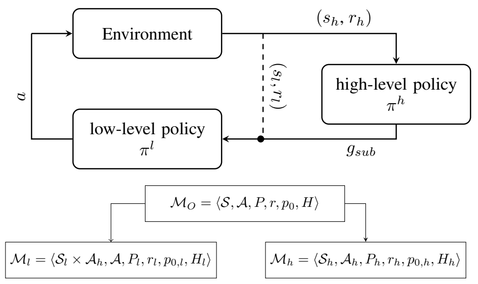
Environment
(s_h, r_h)
(s_l, r_l)
high-level policy π^h
a
low-level policy π^l
g_{sub}
$\mathcal{M}_O = \langle \mathcal{S}, \mathcal{A}, P, r, p_0, H \rangle$
$\mathcal{M}_l = \langle \mathcal{S}_l \times \mathcal{A}_h, \mathcal{A}, P_l, r_l, p_{0,l}, H_l \rangle$
$\mathcal{M}_h = \langle \mathcal{S}_h, \mathcal{A}_h, P_h, r_h, p_{0,h}, H_h \rangle$

Figure 1. The top diagram illustrates the interaction between the different components of a goal-conditioned hierarchical agent. The pair (s_h, r_h) denotes the high-level state and reward, while g_{sub} and r_l denote the sub-goal chosen by the high-level policy as well as the reward associated with it. S_l is the low-level state space and lastly, a is the primitive action used by the low-level policy to interact with the environment. The bottom diagram depicts the MDP decomposition considered.

insight drives the development of a novel hierarchical Q-learning algorithm, specifically designed to leverage the hierarchical structure and improve the sample efficiency. In the goal-based HRL framework that we consider, a high-level policy and a low-level policy are jointly learned to solve an overarching goal together. In such a goal-hierarchical RL system, the high-level policy chooses a sub-goal for the low-level policy, which in turn executes primitive actions in order to solve the sub-goal (Fig. 1, top diagram). This natural way to break down tasks is universal (i.e. can be applied to a wide range of tasks) and it induces a decomposition of the original MDP into two sub-MDPs (detailed in Sec. 2.2).

This paper improves our understanding of HRL through the following contributions:

- We derive a lower bound on the sample complexity associated with the hierarchical decomposition (see Sec. 3). This lower bound allows practitioners to quantify the gain in efficiency they might obtain from decomposing their task.

- We propose a simple, yet novel, algorithm that performs Q-learning-type updates for goal-hierarchical RL, inspired by the type of decomposition considered (see Sec. 4).

- We empirically validate our theoretical findings using a set of synthetic tasks with hierarchical properties that can be scaled in complexity (see Sec. 5). Our experiments confirm that the derived bound is able to successfully identify instances where a hierarchical decomposition could be beneficial (see Sec. 5).

2. Background

Online reinforcement learning (Sutton & Barto, 2018) algorithms aim to learn an optimal policy (i.e. a policy that maximizes the sum of observed rewards) only through interactions with their environment. When a task is too complex, the number of interactions required to learn a near-optimal policy becomes prohibitive. The complexity of the task typically depends on the difficulty of temporal credit assignment (which is directly related to the episode length) and the size of the state space (McGovern et al., 1997). To address this difficulty, HRL leverages temporal abstractions (Sutton et al., 1999b) and state abstractions (Dayan & Hinton, 1992) to reduce the number of interactions required to learn an optimal policy. There exists a wide range of HRL frameworks, see (Hutsebaut-Buysse et al., 2022) for a survey. In this paper, we focus on the goal-conditioned HRL framework (Nachum et al., 2018; Beyret et al., 2019). Of the other HRL frameworks, only the options framework (Sutton et al., 1999b) and it's associated semi-Markov Decision Process (Fruit et al., 2017; Wen et al., 2020; Brunskill & Li, 2014; Fruit & Lazaric, 2017) are supported by a well-developed theory. However, in practice, the goal-conditioned hierarchical framework presented in figure 1 is much more common. Unlike the options framework, the goal-conditioned HRL framework requires no prior knowledge about the task (Hutsebaut-Buysse et al., 2022) and introduces the opportunity to generalize over the goal space, leading to significant performance gains in benchmark tasks (Vezhnevets et al., 2017; Nachum et al., 2018; Haarnoja et al., 2018). Additionally, the option framework does not directly allow for state abstraction and often considers that the transition function is known. These differences mean that we cannot directly apply the analysis in the options literature (Fruit & Lazaric, 2017; Fruit et al., 2017; Wen et al., 2020) to our goal-conditioned HRL setting, where we consider state abstraction, action abstraction, time abstraction and jointly learn all levels of the hierarchy through interaction with the environment.

For the remainder of this section, we formally define the episodic finite-horizon MDPs setting and the hierarchical decomposition we consider.

2.1. Episodic finite-horizon Markov Decision Process

An episodic finite-horizon Markov Decision Process (MDP) is described by the following tuple: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, r, P, p_0, H \rangle$. Where $\mathcal{S}$ is a finite state space of size $|\mathcal{S}|$ and $\mathcal{A}$ is a finite action space of size $|\mathcal{A}|$. The goal of the task is encoded in a terminal state $g \in \mathcal{S}$. We assume the reward function $r(s, g) \in [-a, b]$ (for $a, b \geq 0$) is known $\forall s \in \mathcal{S}, g \in \mathcal{S}$. The initial state distribution p_0 is a distribution over states that is used to sample the starting state of an episode. The learner interacts with the MDP in episodes that last at most H time steps. In each time step $t = 0, \dots, H-1$ the learner observes a state s_t and chooses an action a_t. Given a state action pair (s_t, a_t) the next state $s_{t+1} \sim P(\cdot|s_t, a_t)$ is drawn from the transition kernel. Eventually, the episode ends either because the agent interacted with the environment for H time-steps, or because it reached the terminal state.

The objective of the agent is to select actions that maximize the expected return over the duration of an episode. Actions are chosen according to a policy, $a_t \sim \pi(s_t)$, where π is a function that maps each state and time step pair to a distribution over actions $\pi : \mathcal{S} \times [H-1] \to \Delta_{\mathcal{A}}$, and $\Delta_{\mathcal{A}}$ is the set of all probability distributions over $\mathcal{A}$. The agent's aim is to select a policy π to maximize the sum of expected rewards, $\mathbb{E}[\sum_{t=1}^{H} r_t | a_t \sim \pi(s_t)]$, where the expectation is over the initial state distribution, the policy and the stochastic transitions. Note, that it is often the case for finite-horizon MDPs that the policy depends on the current time step. To simplify notation we do not make this relation explicit.

For a given policy π, we write the value function, $V^\pi_\tau(s)$, and the Q-function, $Q^\pi_\tau(s,a)$, at time step $\tau \in [H-1]$ as follows:

$$V^\pi_\tau(s) = \mathbb{E}\left[\sum_{t=\tau}^{H-1} r_t | s_\tau = s, a_{\tau:H-1} \sim \pi\right], \tag{1}$$

$$Q^\pi_\tau(s,a) = \mathbb{E}\left[\sum_{t=\tau}^{H-1} r_t | s_\tau = s, a_\tau = a, a_{\tau+1:H-1} \sim \pi\right], \tag{2}$$

where $s \in \mathcal{S}$ and $a \in \mathcal{A}$ represent the current state and action. The notation $a_{\tau:H-1} \sim \pi$ is used to specify that actions between time step τ and time step $H-1$ were selected using π. The optimal policy, π^*, is the policy with the highest value function for every time step and every state, $V^{\pi^*}_\tau(s) = V^*_\tau(s) = \max_\pi V^\pi_\tau(s) \, \forall \tau \in [H-1], \forall s \in \mathcal{S}$. Note that there always exists a deterministic Markov policy that maximizes the total expected reward in a finite-horizon MDP (Puterman, 2014).

In this article, we evaluate the quality of a policy by its expected value at the beginning of an episode. To lighten the notation, we define $V^\pi = \mathbb{E}_{s_0 \sim p_0}[V^\pi_0(s)]$ to be the expected value from the beginning of an episode where the expectation is taken over initial states.

2.2. Episodic finite-horizon hierarchical MDP

For a given episodic finite-horizon MDP $\mathcal{M}_o$ we assume it can be hierarchically decomposed into a pair of MDPs $(\mathcal{M}_l, \mathcal{M}_h)$ as depicted on the bottom diagram of figure 1. To avoid any ambiguity, when necessary, we use the following notation: the subscript o is used to denote the original MDP, while subscripts l and h are used to denote low-level and high-level MDPs, respectively.

The low-level and high-level MDPs consist of the following tuples $\mathcal{M}_l = \langle \mathcal{S}_l \times \mathcal{A}_h, \mathcal{A}, r_l, P_l, p_{0,l}, H_l \rangle$ and $\mathcal{M}_h = \langle \mathcal{S}_h, \mathcal{A}_h, r_h, P_h, p_{0,h}, H_h \rangle$, respectively. In order to be a valid hierarchical decomposition we enforce the MDPs to satisfy the following set of conditions:

Action space: The low-level action space is equal to the set of primitive actions that the agent can use to interact with the environment. It is then equivalent to the original MDP action space $\mathcal{A}$. The high-level action space $\mathcal{A}_h$ is the set of sub-goals the high-level agent can instruct to the low-level agent. Note that the set of available actions $\mathcal{A}_h(s_h)$ depends on the current high-level state s_h. For the sake of simplicity, we do not make this relationship explicit in our notation.

State spaces: The low-level state s_l and the high-level state s_h combined contain all the necessary information to reconstruct the corresponding state, s, in the original MDP. States $s \in \mathcal{S} \subset \mathbb{R}^d$ are usually described as multi-dimensional vectors, where each dimension encodes a specific feature of the state. For example, a state description can be factored in a tuple $(s_l, s_h) \in \mathcal{S}_l \times \mathcal{S}_h$ with a part of the state description that belongs to the low-level MDP and another part to the high-level MDP. Hence any state $s \in \mathcal{S}_o$ can be represented by a tuple $(s_l, s_h) \in \mathcal{S}_l \times \mathcal{S}_h$. Additionally, since the low-level policy is goal conditioned, its state space also contains the goal description leading to the following state space for the low-level MDP: $\mathcal{S}_l \times \mathcal{A}_h$, a complete low-level state consists of the concatenation of the low-level state description s_l and the sub-goal description a_h.

Initial state distribution: The high-level initial state distribution $p_{0,h}$ is a restriction of the original state distribution p_0 on $\mathcal{S}_h$. The low-level initial state distribution $p_{0,l}(\cdot|s_{h,0})$ is conditioned on the initial high-level state $s_{h,0}$ and spans the low-level space, ensuring that $p_0(s) = p_{0,h}(s_h) p_{0,l}(s_l|s_h)$, where s_l and s_h are the corresponding decomposition of s.

Transition functions: The low-level transition function P_l is the restriction of P on $\mathcal{S}_l \times \mathcal{A}_h$. One important challenge in HRL is that the high-level transition function, P_h, depends on the low-level policy since the quality of the low-level policy influences the likelihood of reaching a sub-goal state. The high-level transition probability $P_h(s'_h|s_h, a_h, \pi_l)$ is the probability that the agent transitions to s'_h given the current high-level state s_h, the sub-goal a_h

and low level policy π_l. Since P_h depends on the low-level policy it is non-stationary because the low-level policy will change as it learns to respond to sub-goals. This makes the learning task more challenging for the high-level policy.

Reward functions: Since the terminal states for the original MDP belong to $\mathcal{S}$ and the sub-goals for the low-level MDP lie in $\mathcal{S}_l$. The low-level reward function can be obtained from the original reward function, $r_l(s_l, g_{sub}) = 2r(s, g)$, where s and g are the reconstruction of the low-level state and the sub-goal in the original MDP, using the current high-level state. The high-level reward function is the sum of rewards obtained by the low level during the sub-episode, where the high-level action plays the role of a sub-goal: $r_h(s, a_h) = \sum_{t=1}^{H_l} r_l(s_t, a_h)$.

Horizons: The original MDP allows for an episode to last at most H steps. Consequently, the horizons of the high-level, H_h, and low-level, H_l, MDPs must satisfy the following equality $H = H_h H_l$.

Note that we can always produce a decomposition that satisfies these assumptions; a naive way to decompose any MDP would be to consider a high-level agent whose only action encodes the end goal of the task and a low-level with complete state information (i.e. it does not use state abstraction). While the above decomposition is valid, it is not necessarily useful. Here, our goal is to identify when a decomposition is useful (i.e. when it leads to an improvement in sample efficiency).

We denote by π_l a policy that interacts with the low-level MDP $\mathcal{M}_l$, and π_h a policy that interacts with the high-level MDP $\mathcal{M}_h$. In goal-conditioned HRL, the low-level policy maps a pair (low-level state, sub-goal) to an action: $\pi_l : \mathcal{S}_l \times \mathcal{A}_h \to \mathcal{A}_l$ and the high-level policy maps a high-level state to a high-level action: $\pi_h : \mathcal{S}_h \to \mathcal{A}_h$. Each policy can be assessed using the corresponding high and low-level value functions $V_l^{\pi_l}$ and $V_h^{\pi_h}$. Similar to the non-hierarchical case, we can define optimal high-level and low-level policies as $\pi_l^* = \operatorname{argmax}_{\pi_l} V_l^{\pi_l}$ for the low-level policy and $\pi_l^* = \operatorname{argmax}_{\pi_h} V_h^{\pi_h}$ for the high-level policy. Moreover, as we show below, every pair of policies (π_l, π_h) can be combined to produce a policy π that interacts with the original MDP $\mathcal{M}_o$.

Definition 2.1. A hierarchical policy consists of a pair (π_l, π_h) that can be mapped to a policy π in the original MDP $\mathcal{M}_o$ as follows:

$$\pi(a|s) = \pi(a|s_l, s_h) = \sum_{a_h \in \mathcal{A}_h} \pi_h(a_h|s_h)\pi_l(a|a_h, s_l). \tag{3}$$

The optimal hierarchical policy denotes the policy obtained when combining the optimal pair (π_l^*, π_h^*). It is important to note that not all policies π in the original MDP have a corresponding decomposition (π_l, π_h), and in particular, there is no guarantee that the decomposition of the optimal policy in the original MDP exists. Intuitively, this is principally due to the absence of an interruption mechanism. After several steps in a stochastic MDP, the current sub-goal might not remain the most valuable sub-goal. A non-hierarchical policy will immediately be able to adapt its trajectory, while the hierarchical policy will first have to complete the current sub-goal before being able to make adjustments.

Our goal is to understand when a hierarchical decomposition of the MDP allows us to learn a near-optimal policy with fewer interactions with the environment. Therefore, we are interested in assessing the performance of the combination of π_l and π_h while they interact with the original MDP $\mathcal{M}_o$. To convey the fact that we are evaluating a hierarchical policy in the original MDP, we use the following notation: given a pair of policies (π_l, π_h) and their associated policy in the original MDP, π, the value function of the hierarchical policy is denoted by $V_o^{\pi_l, \pi_h} = \mathbb{E}_{s_0 \sim p_0}[V_{o,0}^{\pi}(s_0)]$, where the subscript o is a reminder that we are evaluating a policy while it interacts with the original MDP $\mathcal{M}_o$.

When learning in a decomposed MDP, the algorithm has to learn two policies, the high-level policy, π_h, and the low-level policy, π_l. This is done in an episodic setting where an episode unfolds as follows. The learner first observes the initial state and uses the high-level policy to find the most appropriate sub-goal. For the next H_l time steps the low-level policy attempts to solve the sub-goal. The low-level agent updates its policy at the end of each low-level step. Once the H_l time steps are over or if the sub-goal has been reached, the high-level agent observes a new high-level state and can finally perform an update to its policy. If the overall task is not completed, the high-level agent instructs a new sub-goal. These interactions are repeated until the task is completed or the horizon H is reached. We can now think of HRL as two agents that interact with the environment. Where the objective is to find a hierarchical policy that maximizes the value in the original MDP. To approximate this, each agent can find the policy that maximizes their own value function, $\max_{\pi_l} V_l^{\pi_l}$ and $\max_{\pi_h} V_h^{\pi_h}$.

2.3. Probably-Approximately Correct RL

Our aim is to find, in as few episodes as possible, a pair of policies (π_l, π_h) which have a near-optimal value. To formalize this, we introduce the Probably-Approximately Correct (PAC) RL notion. We denote by Δ_k the sub-optimality gap, that is the difference between the optimal (non-hierarchical) policy π^* and the current hierarchical policy (π_l^k, π_h^k): $\Delta_k = V_o^* - V_o^{\pi_l^k, \pi_h^k}$. When defining the sub-optimality gap both policies are evaluated on the original MDP $\mathcal{M}_o$. We use the PAC guarantee defined in (Dann, 2019), for completeness, we recall its definition below.

Definition 2.2. An algorithm satisfies a PAC bound N if for a given input $\epsilon, \delta > 0$, it satisfies the following condition for any episodic fixed-horizon MDP: with probability at least $1-\delta$, the algorithm plays policies that are at least ϵ-optimal after at most N episodes. That is, with probability at least $1-\delta$,

$$\max\{k \in \mathbb{N} : \Delta_k > \epsilon\} \leq N,$$

where N is a polynomial that can depend on the properties of the problem instance.

In Section 3, we bound the sample complexity of HRL algorithms. In this context, the sample complexity refers to the number of episodes, N, during which the algorithm may not follow a policy that is at least ϵ-optimal with probability at least $1-\delta$.

2.4. Running Example

In order to concretise our setting, we consider the following companion example. The original MDP describes the task of solving a maze in a grid-world environment. The state consists of a tuple (R, C) that indicates in which room, R, and which cell within that room, C, the agent is currently in. The reward function incurs a small cost, $-a$, at each time step unless the agent reaches the absorbing goal state. Once the goal state is reached, the agent stops receiving penalties and receives a reward of 0 for all the remaining time steps. Mathematically, $r(s) = -a\mathbb{1}\{s \neq g\}$ where $g \in \mathcal{S}$ denotes the goal state, and $\mathbb{1}$ denotes the indicator function.

We can decompose this MDP as follows. The high-level MDP describes a similar maze, but instead of moving from cell to cell, the agent is moving from room to room so the state is just the current room it is in. The aim of the high-level agent is to find the sequence of rooms that lead to the goal. Hence at each (high-level) time step, it indicates the most valuable exit the low-level agent should take from the room. As detailed in section 2.2 the high-level reward for a sub-goal is the sum of the rewards accumulated by the low-level agent during that sub-episode. The low-level agent is myopic to other rooms - it only sees the current room and the exit it has to reach (i.e. its sub-goal), and it receives a penalty of $-2a$ for each action it takes unless it reaches the sub-goal, in which case it does not receive any penalty. Hence, if g_{sub} is the sub-goal, it receives reward $r(s) = -2a\mathbb{1}\{s \neq g_{sub}\}$.

We will return to this example throughout the paper, but it should be noted that the framework we consider is general enough to be applied to a wide range of tasks. One such example is robotics, where the low-level agent would be tasked to control the joints of the robot to produce movements selected by the high-level policy whose goal is to perform tasks that require a sequence of distinct movements (i.e. navigational tasks, manipulation tasks or a combination of both).

3. Lower bound on the sample complexity of HRL

It has been proven in (Dann & Brunskill, 2015) that, for any RL algorithm, the number of sample episodes required to obtain an (ϵ, δ)-accurate policy (in the original MDP) is lower bounded by:

$$\mathbb{E}[N] = \Omega\bigg(\frac{|\mathcal{S}||\mathcal{A}|H^2}{\epsilon^2}\ln\big(\frac{1}{\delta + c}\big)\bigg), \tag{4}$$

where c is a positive constant.

We now extend this result to hierarchical MDPs. Before doing so, it is important to note that even the best hierarchical policy (as constructed in Eq. (3)) might be sub-optimal. This is a direct consequence of the goal-conditioned architecture and the absence of an interruption mechanism. If while executing a sub-episode it appears that another sub-goal becomes more valuable the architecture proposed does not allow interruptions. The agent will first have to complete the current sub-episode before being able to adapt its trajectory to the new circumstances. Let $V_o^{\pi_l^*, \pi_h^*}$ denote the value of the optimal hierarchical policy value function in the original MDP. Then, the sub-optimality gap is larger than the gap between the current policy pair and the optimal hierarchical policy $\Delta_k = V_o^* - V_o^{\pi_l^k, \pi_h^k} \geq V_o^{\pi_l^*, \pi_h^*} - V_o^{\pi_l^k, \pi_h^k}$. Therefore, if for some N, $V_o^{\pi_l^*, \pi_h^*} - V_o^{\pi_l^k, \pi_h^k} \geq \epsilon$ for at least N episodes, it must also be the case that $\Delta_k \geq \epsilon$ for at least N episodes. Hence, N is a lower bound on the number of episodes where the algorithm must follow a sub-optimal policy.

In the following theorem, we provide a lower bound on the number of episodes required to learn a pair of policies (π_l, π_h) which are ϵ-accurate with respect to the optimal hierarchical policy (π_l^*, π_h^*). By the above argument, this bound will also be a lower bound on the number of episodes necessary to learn an ϵ-accurate policy with respect to the optimal policy π^*.

Theorem 3.1. *There exist positive constants c_l, c_h and δ_0 such that for every $\delta \in (0, \delta_0)$ and for every algorithm A that satisfies a PAC guarantee for (ϵ, δ) and outputs a deterministic policy, there is a fixed horizon MDP such that A must interact for*

$$\mathbb{E}[N] = \Omega\Bigg(\max\bigg(\frac{|\mathcal{S}_l||\mathcal{A}_h||\mathcal{A}|H_l^2}{\epsilon^2}\ln\big(\frac{1}{\delta + c_l}\big), \frac{|\mathcal{S}_h||\mathcal{A}_h|H_h^2}{\epsilon^2}\ln\big(\frac{1}{\delta + c_h}\big)\bigg)\Bigg) \tag{5}$$

episodes until the policy is (ϵ, δ)-accurate.

A complete version of the proof is given in Appendix A.1. In the following, we simply highlight the main steps of the proof.

Sketch of the proof: An ϵ-accurate pair of policies must satisfy the following inequality, $|V_o^{\pi_l^*,\pi_h^*} - V_o^{\pi_l,\pi_h}| \leq \epsilon$. To find a lower bound on the number of episodes N before we obtain an ϵ-accurate pair of policies (π_l, π_h) we use the following steps:

(i) We decompose the objective applying the triangle inequality, $|V_o^{\pi_l^*,\pi_h^*} - V_o^{\pi_l^*,\pi_h}| + |V_o^{\pi_l^*,\pi_h} - V_o^{\pi_l,\pi_h}| \leq \epsilon$.

(ii) We show that the number of samples required to guarantee $|V_o^{\pi_l^*,\pi_h^*} - V_o^{\pi_l^*,\pi_h}| \leq \epsilon/2$ is bounded by $\Omega\Big(\frac{|\mathcal{S}_h||\mathcal{A}_h|H_h^2}{\epsilon^2}\ln\big(\frac{1}{\delta+c_h}\big)\Big)$

(iii) We show that the number of samples required to guarantee $|V_o^{\pi_l^*,\pi_h} - V_o^{\pi_l,\pi_h}| \leq \epsilon/2$ is bounded by $\Omega\Big(\frac{|\mathcal{S}_l||\mathcal{A}_H||\mathcal{A}|H_l^2}{\epsilon^2}\ln\big(\frac{1}{\delta+c_l}\big)\Big)$

These three steps together give us the result in Theorem 3.1, see A.1 for more details.

Interpretation of the sample complexity bound: We argue that this lower bound[1] is quite insightful and allows us to identify characteristics of the decomposition that might lead to improved sample efficiency. We discuss some of the key insights below:

State abstraction: Only one of the two state space cardinalities will dominate the bound in eq. 5. This suggests that an efficient decomposition must separate the original state space as evenly as possible between the two levels of the hierarchy. Another phenomenon at stake is the low-level re-usability. Due to the state abstraction, the low-level agent can re-use its learned policy in different states (i.e. different states $s_1, s_2 \in \mathcal{S}$ whose low-level component s_l are the same). To highlight of the re-usability on the sample efficiency we rewrite the lower bound 5 in terms of the re-usability index $\kappa = \frac{|\mathcal{S}|}{|\mathcal{S}_l|}$.

$$\mathbb{E}[N] = \Omega\Bigg(\max\Big(\frac{\frac{|\mathcal{S}\times\mathcal{A}_H|}{\kappa}|\mathcal{A}|H_l^2}{\epsilon^2}\ln\big(\frac{1}{\delta+c_l}\big), \frac{|\mathcal{S}_H||\mathcal{A}_H|H_h^2}{\epsilon^2}\ln\big(\frac{1}{\delta+c_h}\big)\Big)\Bigg). \quad (6)$$

[1]Note that this is a lower bound - it does not tell us if there exist algorithms which achieve this lower bound.

From equation 6 it clearly appears that a large re-usability index improves the sample efficiency.

Temporal abstraction: Similarly, only one of the two time horizons will dominate the bound, again suggesting that a fair repartition of the load is beneficial. The temporal abstraction (reducing H to H_h and H_l) simplifies the credit assignment problem for both (the high-level and the low-level) policies by giving denser feedback. The low-level agent is rewarded for successfully completing sub-tasks that are significantly shorter (in horizon) than the original task and the high-level trajectory consists of significantly fewer (high-level) steps than a trajectory in the original MDP.

High-level action space: This is the only term that appears on both sides of the $\max(\cdot,\cdot)$ in eq. 5. This suggests that both the high-level and the low-level benefit from a small high-level action space.

As explained above, there are aspects where both agents are aligned (i.e. small high-level action space) and other aspects where an equilibrium needs to be found as both agents would benefit from short horizon and small state space.

It is important to note that our bound also shows that a hierarchical decomposition does not always improve the sample efficiency. Indeed, there will be some settings where using a "bad" hierarchical decomposition does not lead to any improvement in the sample complexity. Our bound can therefore provide a sanity check to determine whether a hierarchical decomposition *could* lead to an improved sample complexity. Although we note that finding an algorithm that achieves this improved sample complexity can still be challenging. In section 5, we consider several MDP decompositions and empirically validate that when our bound suggests the hierarchical decomposition is beneficial, our algorithm (see Sec. 4) leverages this to achieve lower sample complexity.

4. Stationary Hierarchical Q-learning

Once we know that we are in an MDP where the hierarchical decomposition could lead to improved sample complexity, the next challenge is to design an algorithm which can leverage this decomposition. In this section, we propose the *Stationary Hierarchical Q-learning* algorithm (SHQL).

One of the most challenging aspects of jointly learning a pair of policies is the non-stationarity of the high-level transition dynamics, P_h. It was briefly mentioned (in Sec. 2.2) that the high-level transition function, P_h, is non-stationary since it depends on the low-level policy, π_l with the next high-

Algorithm 1 Stationary Hierarchical Q-learning (SHQL)

```
1: **Initialize:** $Q^L_{:,:,:} = 0$, $Q^H_{:,:} = 0$
2: done$_H = False$
3: $t = k = 0$
4: **while** not done$_H$ and $k < K$ **do**
5:     Observe s$^H_k$, s$^L_t$
6:     g$_{sub}$ = $\pi^H(s^H_k)$
7:     **while** not done$_L$ and $t < T$ **do**
8:         a$^L_t = \pi^L(s^L_t)$
9:         Observe s$^L_{t+1}$, $r^L_t$
10:        *LowLevelUpdate*((s$^L_t$, a$^L_t$, r$^L_t$, s$^L_{t+1}$, g$_{sub}$))
11:        s$_t$ = s$_{t+1}$
12:        $t = t + 1$
13:    **end while**
14:    Observe s$^H_{k+1}$ r$^H_k$
15:    **if** done$_L$ **then**
16:        Q$^H_{nxt}$ = $\max_a$ Q$^H_{s^H_{k+1},a}$
17:        Q$^L_{s_k,a_k}$ = Q$^H_{s_k,a_k} + \alpha * (r^H_k + \gamma Q^H_{nxt})$
18:    **end if**
19:    k = k + 1
20: **end while**
```

Algorithm 2 LowLevelUpdate

```
1: **Input:** (s$_t$, a$_t$, r$_t$, s$_{t+1}$ g$_{sub}$)
2: Q$^L_{nxt}$ = $\max_a$ Q$^L_{g_{sub},s_{t+1},a}$
3: Q$^L_{g_{sub},s_t,a_t}$ = Q$^L_{g_{sub},s_t,a_t} + \alpha * (r^L_t + \gamma Q^L_{nxt})$
4: **return** Q$^L$
```

level state depending on whether π_l managed to reach the sub-goal. To address this issue, we leverage the fact that the algorithm knows what a successful sub-episode is, i.e. it knows if the low-level agent managed to arrive at the desired sub-goal. Therefore, the algorithm only makes an update if the low-level agent is behaving reasonably well (i.e. solving the sub-goal). In this way, the algorithm filters all bad examples from the training set and the behaviour of P_h is more stable. Note however that the reward function of the high-level agent remains non-stationary. At first, sub-goals won't be solved optimally, incurring a small reward to the high-level agent, but as the low-level agent learns to solve sub-goals more efficiently the associated reward will increase.

As detailed in the function *LowLevelUpdate* in algorithm 2 the low-level agent simply performs Q-learning updates on the observed low-level transitions and rewards. The high-level agent also performs Q-learning updates, but only on successful transitions, as specified at line 15 of algorithm 1.

5. Experiments

We now empirically evaluate[2] the impact of the decomposition on a wide range of MDPs in order to validate the lower bound found in section 3 and evaluate the performance of our proposed SHQL algorithm. To satisfy the assumption of hierarchical structure, the environments considered are a generalization of the *four-room* problem with an arbitrary number of rooms. The entire maze is built by arranging an arbitrary number of rooms on a grid. The high-level task consists of learning the shortest sequence of rooms that lead the agent from the starting position (the top left room) to the goal room (the bottom right room). The low-level task is to learn how to navigate within each room and to reach the instructed hallway. To further modulate the difficulty of the task we vary the number of rooms (as depicted in figures 3 and 2, top rows) and the number of room profiles used (as depicted in the rightmost plot of figure 2).

The set of MDPs generated by these environments are the following:
The original MDP: This is a standard grid-world MDP, where the state space indicates the cell where the agent is located and the action space allows the agent to move one cell in any cardinal direction (North, South, East, West). To obtain stochastic environments, each action has a success probability of $p_{success} = 4/5$. In case of failure, the action will be chosen at random.
The high-level MDP: The high-level state space is restricted to the room where the agent is currently located, and the exact position of the agent within that room is abstracted away. The high-level actions consist of instructing the low-level to reach one of the available hallways. Note that not all rooms have access to the four hallways.
The low-level MDP: The low-level agent only observes the current location of the agent within a room and the goal instructed by the high-level agent (one of the reachable hallways). It then uses the primitive action space (the four cardinal directions) to reach the desired hallway.

5.1. Identical rooms

We first introduce the experimental setting in its simplest version. The environments considered in this subsection are mazes that are built by assembling identical rooms without any obstacles (i.e. the top room profile in Fig. 2). The learning curve of SHQL on the decomposed MDP and Q-learning on the original MDP is shown in Figure 2 (left plot). As expected for simple mazes (e.g. with 4 or 16 rooms) the hierarchical decomposition does not provide much improvement, but as the problems grow more difficult, the empirical evaluation suggests a significant improvement in sampling

[2]Experiments were run on a 12th Gen Intel Core i7 with 16GB of RAM, to train the agents on the largest maze considered takes $\sim$ 7 minutes.

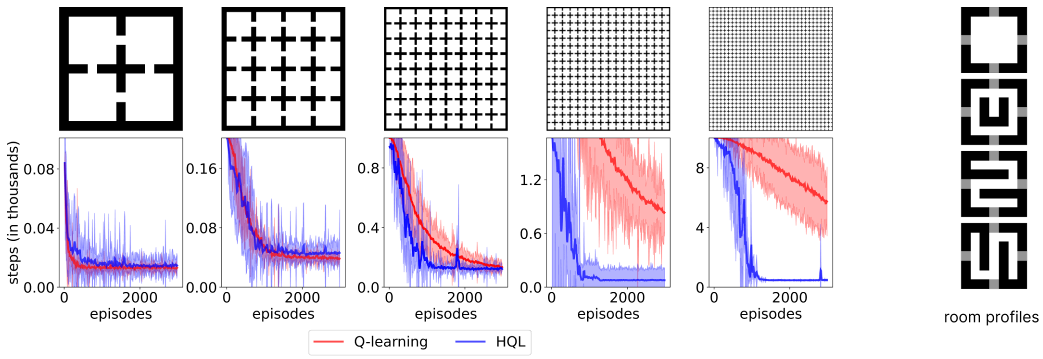

Figure 2. The grid of plots on the left-hand side depicts, on the top row, the mazes whose size ranges from 4 rooms to 1024 rooms. The bottom row shows the number of steps required for SHQL (in blue) and Q-learning (in red) to complete the maze. The standard deviation is obtained by running 10 different seeds. The right-hand side of the plot shows the different room profiles used to build the mazes.

efficiency. This is also confirmed by our bound (yellow curve on the rightmost plot of Fig. 3) which highlights that the efficiency gain of HRL is mostly achievable in complex MDPs. It is important to notice that in this experiment, the low-level decomposition remains constant for a given set of room profiles. This is the reason why the benefit of HRL increases with the number of rooms until a plateau is reached. Once the bound is dominated by the high-level MDP, the unchanging complexity of the low-level MDP causes the ratio between the RL bound (Eq. 4) and the high-level part of the HRL bound (Eq. 5), $\frac{|\mathcal{S}||\mathcal{A}|H}{|\mathcal{S}_H||\mathcal{A}_H|H_H}$, to remain constant (despite the fact that the number of rooms might still grow).

5.2. Different rooms

To make the task more challenging we now increase the number of room profiles used to construct the mazes. As depicted in the rightmost plot of figure 2 we considered four different room profiles, each one with a different obstacle in the room. The low-level agent has to learn to navigate through multiple types of rooms to reach the instructed sub-goal. The performance obtained by Q-learning and SHQL in this setting is shown in figure 3. The introduction of different room profiles allows to modulate the complexity of the low-level MDP, in contrast to varying the number of rooms which only affects the complexity of the high-level MDP. An increased number of room profiles results in a larger state space $\mathcal{S}_l$ but may also lead to a longer horizon H_l as the optimal trajectory might require more time to successfully navigate around the potential obstacles to reach the instructed hallway. While this added difficulty has very little effect on the standard Q-learning, it clearly postpones the efficiency gain of the hierarchical machinery, as seen in figure 3. The evolution of the bound ratio (HRL/RL) for the various MDPs considered is shown in the rightmost plot of figure 3. It shows that when the maze consists of a small number of rooms, the bound is dominated by the low-level agent. However, the curves clearly indicate that as the high-level MDP becomes more complex (i. e. balancing the complexity between the two levels of the hierarchy) the expected sample efficiency improves. This result is also supported by empirical evidence as illustrated in figures 2 (left plot), 3 (left plot), and figures4 and 5 in appendix A.2.

6. Conclusion

In this work, we analysed the sample efficiency of goal-conditioned HRL. To the best of our knowledge, it is the first result that provides an analysis of the intrinsic MDP decomposition induced by goal-conditioned HRL. In particular, our lower bound provides a useful tool for practitioners that illustrates whether they should consider an hierarchical decomposition for their problems. We also implemented a set of hierarchical tasks and designed a novel algorithm that could leverage the existing hierarchy to improve its sample efficiency. Our experimental setting further emphasizes the usefulness of the proposed bound since the empirical efficiency gains are supported by the theoretical analysis provided.

Although this paper has taken a significant first step in improving our understanding of the benefits of hierarchical decomposition, there is still scope for further work in this area. An immediate open question is whether our lower bound could be refined by explicitly accounting for the interactions between the low-level and the high-level agent. Moreover, the insights we proposed are framed in a tabular setting and do not yet apply in a continuous setting where function approximation could be leveraged to allow the low-level to generalise over sub-goals. Overcoming those limitations are interesting direction for future work.

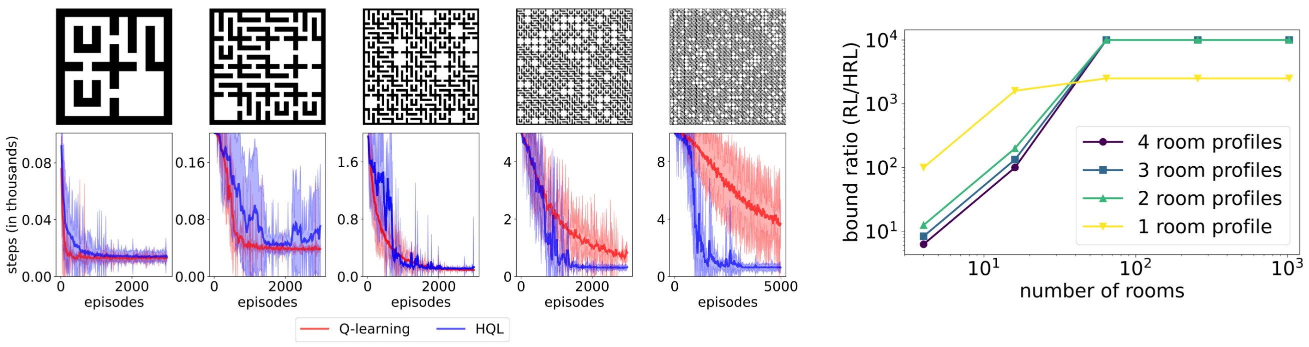

Figure 3. Left-hand plots are similar to figure 2, showing the performance obtained on mazes built from four different room layouts. The right-hand plot shows the evolution of the ratio between the RL bound Eq. (4) and the HRL bound Eq. (5) for various mazes and different room profiles. The curves are colour-coded such that a darker curve indicates more room profiles were considered.

Acknowledgements

AR is supported by an EPSRC CASE studentship sponsored by Shell. AAF is supported by a UKRI Turing AI Fellowship (EP/V025449/1).

References

Abramova, E., Dickens, L., Kuhn, D., and Faisal, A. Hierarchical, heterogeneous control of non-linear dynamical systems using reinforcement learning. In *European Workshop On Reinforcement Learning, at ICML*, volume 2012, 2012.

Auer, P. and Ortner, R. Online regret bounds for a new reinforcement learning algorithm. In *1st Austrian Cognitive Vision Workshop*, pp. 35–42. Österr. Computer-Ges., 2005.

Beyret, B., Shafti, A., and Faisal, A. A. Dot-to-dot: Explainable hierarchical reinforcement learning for robotic manipulation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5014–5019. IEEE, 2019.

Brunskill, E. and Li, L. Pac-inspired option discovery in lifelong reinforcement learning. In *International conference on machine learning*, pp. 316–324. PMLR, 2014.

Dann, C. *Strategic Exploration in Reinforcement Learning-New Algorithms and Learning Guarantees*. PhD thesis, Google, 2019.

Dann, C. and Brunskill, E. Sample complexity of episodic fixed-horizon reinforcement learning. *Advances in Neural Information Processing Systems*, 28, 2015.

Dayan, P. and Hinton, G. E. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

Dietterich, T. G. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

Dietterich, T. G. et al. The maxq method for hierarchical reinforcement learning. In *ICML*, volume 98, pp. 118–126, 1998.

Fruit, R. and Lazaric, A. Exploration-exploitation in mdps with options. In *Artificial intelligence and statistics*, pp. 576–584. PMLR, 2017.

Fruit, R., Pirotta, M., Lazaric, A., and Brunskill, E. Regret minimization in mdps with options without prior knowledge. *Advances in Neural Information Processing Systems*, 30, 2017.

Haarnoja, T., Hartikainen, K., Abbeel, P., and Levine, S. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 1851–1860. PMLR, 2018.

Hutsebaut-Buysse, M., Mets, K., and Latré, S. Hierarchical reinforcement learning: A survey and open research challenges. *Machine Learning and Knowledge Extraction*, 4 (1):172–221, 2022.

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. Is q-learning provably efficient? *Advances in neural information processing systems*, 31, 2018.

Jong, N. K., Hester, T., and Stone, P. The utility of temporal abstraction in reinforcement learning. In *AAMAS (1)*, pp. 299–306, 2008.

Li, G., Shi, L., Chen, Y., Chi, Y., and Wei, Y. Settling the sample complexity of model-based offline reinforcement learning. *arXiv preprint arXiv:2204.05275*, 2022.

McGovern, A., Sutton, R. S., and Fagg, A. H. Roles of macro-actions in accelerating reinforcement learning. In

Grace Hopper celebration of women in computing, volume 1317, pp. 15, 1997.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Nachum, O., Tang, H., Lu, X., Gu, S., Lee, H., and Levine, S. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019.

Pickett, M. and Barto, A. G. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *ICML*, volume 19, pp. 506–513, 2002.

Precup, D. and Sutton, R. S. Multi-time models for temporally abstract planning. *Advances in neural information processing systems*, 10, 1997.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999a.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999b.

Vezhnevets, A., Mnih, V., Osindero, S., Graves, A., Vinyals, O., Agapiou, J., et al. Strategic attentive writer for learning macro-actions. *Advances in neural information processing systems*, 29, 2016.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 3540–3549. PMLR, 2017.

Wen, Z., Precup, D., Ibrahimi, M., Barreto, A., Van Roy, B., and Singh, S. On efficiency in hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 33:6708–6718, 2020.

A. Appendix

A.1. Proof of Theorem 3.1

Theorem 3.1 states that there exist positive constants c_l, c_h and δ_0 such that for every $\delta \in (0, \delta_0)$ and for every algorithm A that satisfies a PAC guarantee for (ϵ, δ) and outputs a deterministic policy, there is a fixed horizon MDP such that A must collect

$$\mathbb{E}[N_e] = \Omega\Bigg(\max\Big(\frac{|\mathcal{S}_l||\mathcal{A}_h||\mathcal{A}|H_l^2}{\epsilon^2} \ln\big(\frac{1}{\delta + c_l}\big), \frac{|\mathcal{S}_h||\mathcal{A}_h|H_h^2}{\epsilon^2} \ln\big(\frac{1}{\delta + c_h}\big) \Big) \Bigg) \tag{7}$$

episodes until its policy is (ϵ, δ)-accurate.

Proof. An ϵ-accurate pair of policies (π_l, π_h) satisfies
$|V_o^{\pi_l^*,\pi_h^*} - V_o^{\pi_l,\pi_h}| \le \epsilon$. Note that by the triangle inequality, if $|V_o^{\pi_l^*,\pi_h^*} - V_o^{\pi_l^*,\pi_h}| + |V_o^{\pi_l^*,\pi_h} - V_o^{\pi_l,\pi_h}| \le \epsilon$, then we will have $|V_o^{\pi_l^*,\pi_h^*} - V_o^{\pi_l,\pi_h}| \le \epsilon$. We, therefore, focus on showing:

(i) the number of samples required to guarantee $|V_o^{\pi_l^*,\pi_h^*} - V_o^{\pi_l^*,\pi_h}| \le \epsilon/2$ is bounded by $\Omega\Big(\frac{|\mathcal{S}_h||\mathcal{A}_h|H_h^2}{\epsilon^2} \ln\big(\frac{1}{\delta+c_h}\big)\Big)$

(ii) the number of samples required to guarantee $|V_o^{\pi_l^*,\pi_h} - V_o^{\pi_l,\pi_h}| \le \epsilon/2$ is bounded by $\Omega\Big(\frac{|\mathcal{S}_l||\mathcal{A}_H||\mathcal{A}|H_l^2}{\epsilon^2} \ln\big(\frac{1}{\delta+c_l}\big)\Big)$

Then once we have both (i) and (ii), we know that after

$$\Omega\Bigg(\max\Big(\frac{|\mathcal{S}_L||\mathcal{A}_H||\mathcal{A}|H_L^2}{\epsilon^2} \ln\big(\frac{1}{\delta + c_l}\big), \frac{|\mathcal{S}_H||\mathcal{A}_H|H_H^2}{\epsilon^2} \ln\big(\frac{1}{\delta + c_h}\big) \Big) \Bigg)$$

episodes, we will have $|V_o^{\pi_l^*,\pi_h^*} - V_o^{\pi_l^*,\pi_h}| + |V_o^{\pi_l^*,\pi_h} - V_o^{\pi_l,\pi_h}| \le \epsilon$ and so $|V_o^{\pi_l^*,\pi_h^*} - V_o^{\pi_l,\pi_h}| \le \epsilon$.

Part (i) Note that only learning the high-level policy when the low-level policy is optimal, is equivalent to learning an ϵ-accurate high-level policy interacting with $\mathcal{M}_h$ with a stationary transition function (since the low-level behaviour is not evolving anymore). Hence we can bound the number of episodes N_h required to have: $|V_h^* - V_h^{\pi_l^*,\pi_h}| \le \epsilon$, by directly applying Eq. (4) to the high-level MDP to get

$$\mathbb{E}[N_h] = \Omega\Bigg(\frac{|\mathcal{S}_h||\mathcal{A}_h|H_h^2}{\epsilon^2} \ln\big(\frac{1}{\delta + c_h}\big) \Bigg)$$

To be able to use this result to construct the bound of interest, we need to make sure these results are valid under the original MDP: $|V_o^{\pi_l^*,\pi_h^*} - V_o^{\pi_l^*,\pi_h}| \le \epsilon$. In particular, the reward functions are not the same for $\mathcal{M}_o$ and $\mathcal{M}_h$. By decomposition, r_h includes the bonus (or the absence of penalty) the high-level gives to the low-level for completing the task. To compensate for that the low-level reward is re-scaled with a penalty twice larger per step. This ensure that $|V_o^{\pi_l^*,\pi_h^*} - V_o^{\pi_l^*,\pi_h}| \le 2|V_h^* - V_h^{\pi_l^*,\pi_h}|$. Hence after $\mathbb{E}[N_h]$ episodes, we have $|V_o^* - V_o^{\pi_l^*,\pi_h}| \le 2\epsilon$

Part (ii) By a similar argument to Part (i), we can bound the number of episodes in the low-level MDP required to obtain an ϵ-optimal low-level policy for a fixed high-level policy π_h. In particular, a lower bound on the number of episodes N_l required to have $|V_l^{\pi_h,\pi_l^*} - V_l^{\pi_l,\pi_h}| \le \epsilon$ can directly be obtained from Eq. (4):

$$\mathbb{E}[N_l] = \Omega\Bigg(\frac{|\mathcal{S}_l||\mathcal{A}_H||\mathcal{A}|H_l^2}{\epsilon^2} \ln\big(\frac{1}{\delta + c_l}\big) \Bigg).$$

We are interested in comparing the policies when they interact with the original MDP. The issue is that there is a difference of scale between $V_o^{\pi_l,\pi_h}$ and $V_l^{\pi_l,\pi_h}$. Episodes are shorter by a factor of H_h in the low-level MDP. So we need to ensure that $|V_l^{\pi_h,\pi_l^*} - V_l^{\pi_l,\pi_h}| \le \frac{\epsilon}{H_h}$. But by construction, this re-scaling is not necessary as a single episode in the original MDP corresponds to at most H_h episodes in the low-level MDP as a single episode in $\mathcal{M}_o$ with x sub-goals correspond to x episodes in $\mathcal{M}_l$.

This leads us to a lower bound on the number of episodes needed to obtain an ϵ-accurate pair of policies as the one stated in the theorem. □

A.2. Additional experiments

In the experimental section (Sec. 5) we used several room layouts. In the main paper, we only provide learning curves for mazes that are composed of rooms without any obstacles or mazes that are composed of all the possible room layouts depicted in the rightmost plot of figure 2. To complete our experiment we show below in (Fig. 4 and Fig. 5) the learning curves obtained when mazes are built from two or three different room layouts. Note also that those results were used to plot the evolution of the bound ratio in the rightmost plot of figure 3.

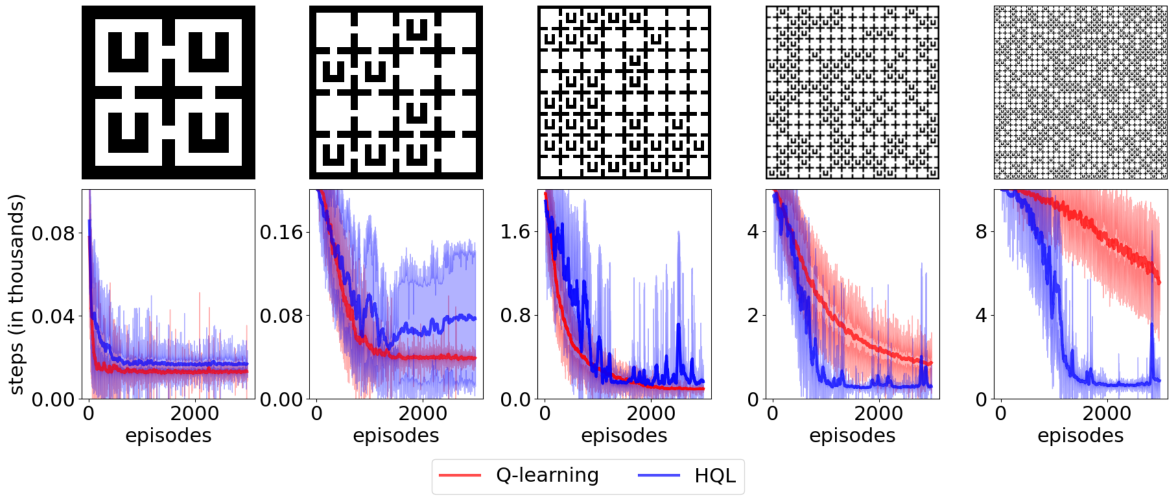

Figure 4. Shows learning curves on various maze sizes with two different room instances, either the room is empty or it has a U-shape obstacle in it. The performance of the agent is measured in the number of steps it requires to solve the task.

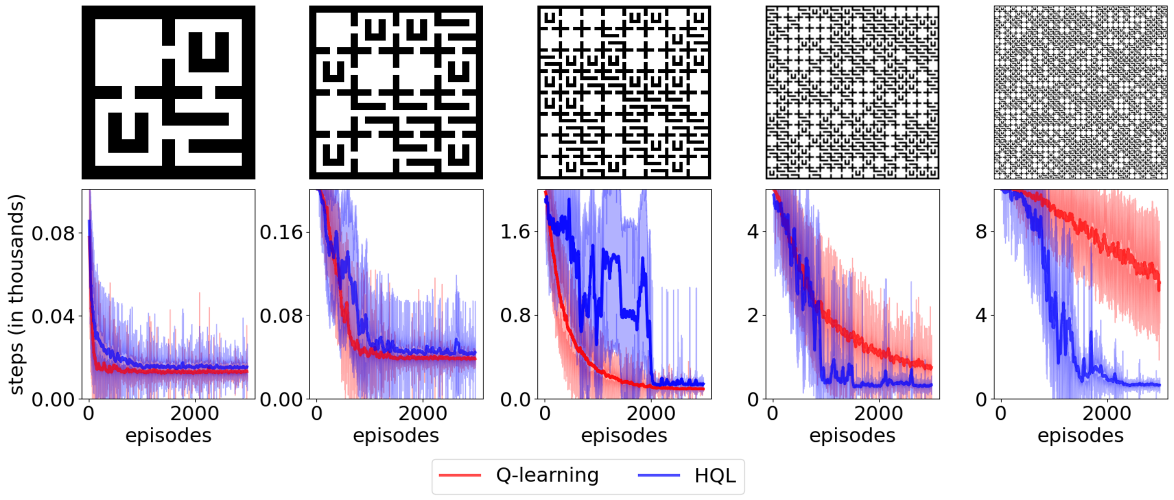

Figure 5. Shows learning curves on various maze sizes with three different room instances, either the room is empty or it has a U-shape obstacle or the room is stripped with horizontal walls. The performance of the agent is measured in the number of steps it requires to solve the task.