# Sample Complexity of Goal-Conditioned Hierarchical Reinforcement Learning

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Hierarchical Reinforcement Learning (HRL) algorithms can perform planning at multiple levels of abstraction. Empirical results have shown that state or temporal abstractions might significantly improve the sample efficiency of algorithms. Yet, we still do not have a complete understanding of the basis of those efficiency gains, nor any theoretically-grounded design rules. In this paper, we derive a lower bound on the sample complexity for the proposed class of goal-conditioned HRL algorithms (e.g. Dot-2-Dot) that lead us to a novel Q-learning algorithm and clearly establishes 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. We build upon this to formulate a simple Q-learning-type algorithm that leverages goal hierarchical decomposition. We empirically validate our theoretical findings by investigating the sample complexity of the proposed hierarchical algorithm on a spectrum of tasks. The tasks were designed to allow us to dial up or down their complexity over multiple orders of magnitude. Our theory and algorithmic findings provide a step towards answering the foundational question of quantifying the improvement hierarchical decomposition offers over monolithic solutions in reinforcement learning.

## 1 Motivation

Hierarchical Reinforcement Learning (HRL) [25, 7, 8, 3] 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 [18, 19, 27, 7], there are also cases where temporal abstraction worsens the empirical sample complexity [15]. Therefore, a natural question to ask is when does HRL lead to improved sample complexity and how much of an improvement can it provide?

Theoretical work on sample-complexity bounds in Machine Learning has been integral to the development of the field. Moreover, theoretical results (e.g. [6, 16, 2, 14, 24]) often uncover interesting principles that are useful for improving algorithm design. For example, the Q-learning algorithm analysed in [14] improved our understanding of exploration strategies in model-free RL and the policy gradient theorem [24] 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 [7, 9], time [21], or action [26, 20, 1].

To address this gap in the literature, we consider a tabular version of the goal-based approach to HRL [18, 3] and we analyze the induced MDP decomposition to derive a lower bound on the sample complexity of this specific HRL framework. This lower bound allows us to understand when a hierarchical decomposition is beneficial and motivates a new hierarchical Q-learning algorithm that

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

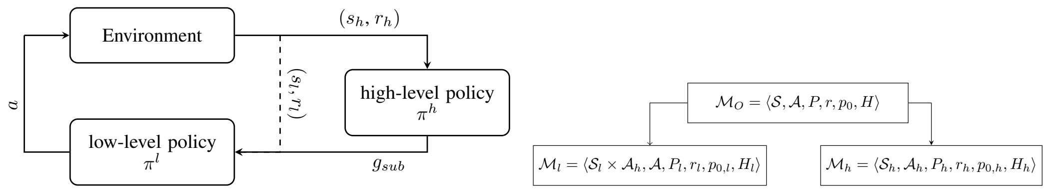


Figure 1: The leftmost diagram depicts the interaction between the different components of a goal-conditioned hierarchical agent. The pair $(s_h, r_h)$ denotes the high-level state and reward, while $g_{sub}, r_l$ denotes the sub-goal chosen by the high-level policy as well as the reward associated with 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 rightmost diagram illustrates the MDP decomposition considered.

can leverage the hierarchical structure to improve its 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, left 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 provide a lower bound on the sample complexity associated with the hierarchical decomposition (see Sec. 3). This lower bound allows practitioners to quantify the efficiency gain they might obtain from decomposing their task.
- We proposes a simple, yet novel, Q-learning-type algorithm for goal-hierarchical RL, inspired by the type of decomposition considered (see Sec. 4).
- We empirically validate the theoretical findings using a synthetic task with hierarchical properties that can be scaled in complexity (see Sec. 5). This evidence confirms 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 [23] algorithms aim to learn an optimal policy (i.e. a policy that maximizes the reward accumulated) only through interactions with the environment. When a task is too complex, the number of interactions required to learn a near-optimal policy becomes prohibitive. The task complexity typically depends on the difficulty of temporal credit assignment (which is directly related to the episode length) and the size of the state space [17]. To address this complexity, HRL leverages temporal abstractions [25] and state abstractions [7] to improve sample efficiency when learning an optimal policy. There exists a wide range of HRL frameworks, see [13] for a survey. In this paper, we focus on the goal-conditioned HRL framework [18, 3]. Of the other HRL frameworks, only the options framework [25] and the resulting semi-Markov Decision Process [11, 28, 4, 10] are supported by a well-developed theory. However, in practice, the goal-conditioned hierarchical framework presented in figure 1 is often preferred. Unlike the options framework, the goal-conditioned HRL framework requires no prior knowledge about the task [13] and introduces the opportunity to generalize over the goal space, leading to significant performance gains in benchmark tasks [27, 18, 12]. 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 [10, 11, 28] to our goal-conditioned HRL setting, where we consider state abstraction, action abstraction, time abstraction and jointly learn all level of the hierarchy through interaction with the environment.

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

### 2.1 Episodic finite-horizon Markov Decision Process

An episodic finite-horizon Markov Decision Process (MDP) is defined 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 determine in which state an episode starts. The learner interacts with the MDP in episodes of at most H time steps. The starting state $s_0 \sim p_0$ of the episode is drawn from the initial state distribution. In each time step $t = 0, \ldots, H-1$ 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 reaches the terminal state, or because it interacted with the environment for H time-steps.

The objective of the agent is to select actions that maximize the expected return over the duration of an episode. We typically assume actions are chosen according to a policy, $a_t \sim \pi(s_t)$, where $\pi$ is a function that maps each state and time step pair to a distribution over actions $\pi : \mathcal{S} \times [H-1] \rightarrow \Delta_{\mathcal{A}}$, and $\Delta_{\mathcal{A}}$ is the set of all probability distributions over $\mathcal{A}$. The agents aim is to select a policy $\pi$ 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 usually the case for finite-horizon MDPs that the policy also depends on the current time step, however to simplify notation we do not make this relation explicit.

For a given policy $\pi$, we define the value function, $V_\tau^\pi(s)$, and the Q-function, $Q_\tau^\pi(s, a)$, at time step $\tau \in [H-1]$ as follows:

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

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

where $s \in \mathcal{S}$ denotes the state, $a \in \mathcal{A}$ is the action and the notation $a_{\tau:H-1} \sim \pi$ is used to specify that actions between time step $\tau$ and time step $H-1$ were selected using $\pi$. The optimal policy $\pi^*$ is the policy with the highest value function for every time step and every state, $V_\tau^{\pi^*}(s) = V_\tau^*(s) = \max_\pi V_\tau^\pi(s) \, \forall \tau \in [H-1], \forall s \in \mathcal{S}$. There is always a deterministic Markov policy that maximizes the total expected reward in a finite-horizon MDP [22].

In this article, we assess the quality of a policy by its expected value at the beginning of an episode. To lighten the notation, we define $V^\pi = \mathbb{E}_{s_0 \sim p_0}[V_0^\pi(s)]$ 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 illustrated on right diagram of figure 1. To avoid any ambiguity, when necessary, we use the following notation: the subscript $o$ denotes the original MDP, while subscripts $l$ and $h$ 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 require that these MDPs satisfy the following set of conditions:

**Action space:** The low-level action space consists of the set of primitive actions that the agent can use to interact with the environment. It is equivalent to the original MDP action space $\mathcal{A}$. The high-level action space $\mathcal{A}_h$ is the set of the 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 $s_h$. To simplify our notation we do not make this relationship explicit.

**State spaces:** The low-level state $s_l$ and the high-level state $s_h$ contain all 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 characteristic. 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 consist 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 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 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 $\pi_l$. Since $P_h$ depends on the low-level policy it is non-stationary, making the learning task more challenging.

**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 find 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 this decomposition is valid, it is not necessarily useful. Here, our goal is to identify when a given decomposition is useful, specifically in terms of improvements in the sample efficiency.

We denote by $\pi_l$ a policy interacting with the low-level MDP $\mathcal{M}_l$, and $\pi_h$ a policy interacting with the high-level MDP $\mathcal{M}_h$. In goal-conditioned HRL, the low-level policy maps a (low-level state, sub-goal) pair 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 evaluated 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 $\pi_l^* = \mathrm{argmax}_{\pi_l} V_l^{\pi_l}$ for the low-level policy and $\pi_l^* = \mathrm{argmax}_{\pi_h} V_h^{\pi_h}$ for the high-level policy. Moreover, as we show below, every pair of policies $(\pi_l, \pi_h)$ can be combined to produce a policy $\pi$ that interacts with the original MDP $\mathcal{M}_o$.

**Definition 2.1.** A hierarchical policy consists of a pair $(\pi_l, \pi_h)$ that can be mapped to a policy $\pi$ 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 merging $(\pi_l^*, \pi_h^*)$. It is important to note that not all policies $\pi$ in the original MDP have a corresponding decomposition $(\pi_l, \pi_h)$, and in particular, there is no guarantee that the optimal policy in the original MDP can be decomposed.

Our goal is to understand when a hierarchical decomposition of the MDP allows us to learn a near-optimal policy faster. Therefore, we are interested in evaluating the performance of the combination of $\pi_l$ and $\pi_h$ 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 $(\pi_l, \pi_h)$ and their associated policy in the original MDP, $\pi$, 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 on the original MDP $\mathcal{M}_o$.

When learning in a decomposed MDP, the learner has to learn two policies, the high-level policy, $\pi_h$, and the low-level policy, $\pi_l$. This is done in an episodic setting where an episode unfolds as follows. Firstly, the learner 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 steps. 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. Often, each agent will try to 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 $(\pi_l, \pi_h)$ which have a near-optimal value. To formalize this, we introduce the Probably-Approximately Correct (PAC) RL notion. We denote by $\Delta_k$ the sub-optimality gap, that is the difference between the optimal (non-hierarchical) policy $\pi^*$ and the current hierarchical policy $(\pi_l^k, \pi_h^k)$: $\Delta_k = V_o^* - V_o^{\pi_l^k, \pi_h^k}$. Note that both policies are evaluated on the original MDP $\mathcal{M}_o$. The PAC guarantee in this paper follows the definition in [5].

**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 $\epsilon$-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 the Section 3, we will 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 $\epsilon$-optimal with probability at least $1 - \delta$.

### 2.4 Running Example

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 $g \in \mathcal{S}$ is the goal state, and $\mathbb{1}$ is 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 specified 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, 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 [6] that, for any RL algorithm, the number of sample episodes necessary to obtain an $(\epsilon, \delta)$-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 notice that even the best hierarchical policy (as constructed in Eq. (3)) might be sub-optimal. This a direct consequence of the goal-conditioned architecture. If while executing a sub-episode it appears that another sub-goal becomes more valuable the architecture proposed do not allow interruptions. The agent will first have to complete the current sub-episode before being able to adapt 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 lower bound the number of episodes required to learn a pair of policies $(\pi_l, \pi_h)$ which are $\epsilon$-accurate with respect to the optimal hierarchical policy $(\pi_l^*, \pi_h^*)$. By the above argument, this will also be a lower bound on the number of episodes necessary to learn an $\epsilon$-accurate policy with respect to the optimal policy $\pi^*$.

**Theorem 3.1.** *There exist positive constants $c_l$, $c_h$ and $\delta_0$ such that for every $\delta \in (0, \delta_0)$ and for every algorithm A that satisfies a PAC guarantee for $(\epsilon, \delta)$ 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 $(\epsilon, \delta)$-accurate.*

The complete proof is given in Appendix A.1. In the following we highlight the main steps.

**Sketch of the proof:** An $\epsilon$-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 $\epsilon$-accurate pair of policies $(\pi_l, \pi_h)$ we used the following steps:

(i) We decompose the objective using 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 $\Omega\Bigg(\frac{|\mathcal{S}_h||\mathcal{A}_h|H_h^2}{\epsilon^2} \ln\big(\frac{1}{\delta+c_h}\big)\Bigg)$

(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 $\Omega\Bigg(\frac{|\mathcal{S}_l||\mathcal{A}_H||\mathcal{A}|H_l^2}{\epsilon^2} \ln\big(\frac{1}{\delta+c_l}\big)\Bigg)$

Combining these three steps together gives us the result in Theorem 3.1, see A.1 for more details.

**Interpretation of the sample complexity bound:** By comparing this lower bound[1] to that in the original MDP, we can clearly identify the problem characteristics 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 suggest that an efficient decomposition must separate the original state space as evenly as possible between the two level of the hierarchy. Another phenomena 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). We rewrite the lower bound 5 in terms of the re-usability index $\kappa = \frac{|\mathcal{S}|}{|\mathcal{S}_l|}$.

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

Equation 6 clearly highlights that a large re-usability index improve the sample efficiency.

**Temporal abstraction:** Similarly, only one of the two time horizons will dominate the bound,

[1]Note that this is a lower bound - we still do not know if there exist algorithms which achieve this lower bound.

again suggesting a fair repartition of the load. 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 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 side 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.

The above discussion highlights properties of the heirachical decomposition that could improve sample complexity. Note however, 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 exploit this. In this section, we propose the *Stationary Hierarchical Q-learning* algorithm (SHQL) for this purpose.

**Algorithm 1:** Stationary Hierarchical Q-learning (SHQL)

**Input:** $Q^L_{:,:,:} = 0$, $Q^H_{:,:} = 0$, done$_H = False$, $t = k = 0$

1 **while** *not done*$_H$ *and* $k < K$ **do**
2 Observe $s^H_k$, $s^L_t$
3 **while** *not done*$_L$ *and* $t < T$ **do**
4 $a^L_t = \pi^L(s^L_t)$
5 Observe $s^L_{t+1}$, $r^L_t$
6 *LowLevelUpdate*(($s^L_t$, $a^L_t$, $r^L_t$, $s^L_{t+1}$, $g_{sub}$))
7 $s_t = s_{t+1}$
8 $t = t + 1$
9 Observe $s^H_{k+1}$ $r^H_k$
10 **if** *done*$_L$ **then**
11 $Q^H_{nxt} = \max_a Q^H_{s^H_{k+1},a}$
12 $Q^L_{s_k,a_k} = Q^H_{s_k,a_k} + \alpha * (r^H_k + \gamma Q^H_{nxt})$
13 **Function** `LowLevelUpdate(`$s_t$, $a_t$, $r_t$, $s_{t+1}$ $g_{sub}$`):`
14 $Q^L_{nxt} = \max_a Q^L_{g_{sub},s_{t+1},a}$
15 $Q^L_{g_{sub},s_t,a_t} = Q^L_{g_{sub},s_t,a_t} + \alpha * (r^L_t + \gamma Q^L_{nxt})$
16 **return** $Q^L$

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, $\pi_l$ with the next high-level state depending on whether $\pi_l$ 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 1 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.

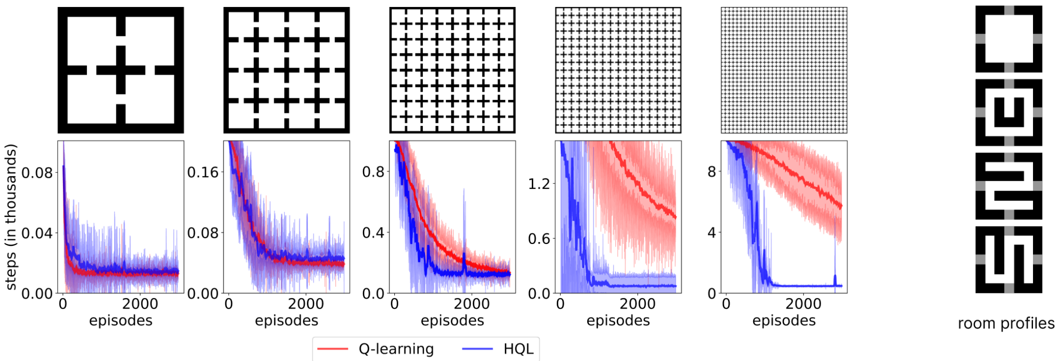


Figure 2: The grid of plots on the left-hand side depicts, on the top row, the mazes whose size ranges from 4 rooms to 1024 rooms. The bottom row shows the number of steps required for SHQL (in blue) and Q-learning (in red) to complete the maze. The standard deviation is obtained by running 10 different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

# 5 Experiments

We now empirically evaluate[2] the impact of the decomposition on various 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 would consist 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 (in addition to the maze size), we vary the room profiles used, as depicted in the right most 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 form. 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). Figure 2 illustrates the empirical performance of our SHQL algorithm against Q-learning in the original MDP. 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 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 remain 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

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

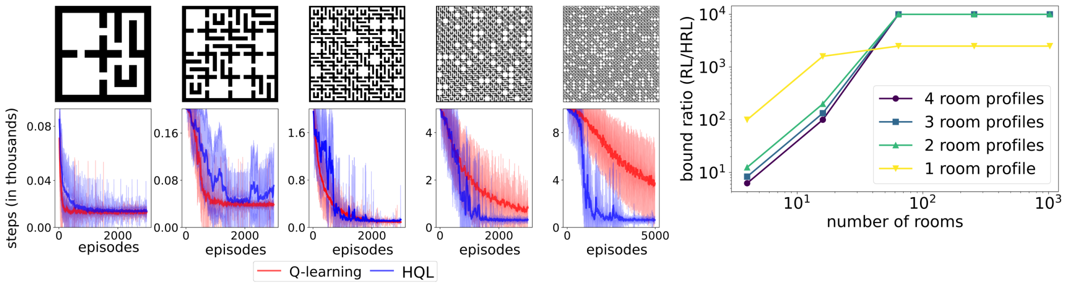


Figure 3: Left-hand plots are similar to figure 2, showing the performance obtained on mazes built from four different room layouts. The right-hand plot shows the evolution of the ratio between the RL bound Eq. (4) and the HRL bound Eq. (5) for various mazes and different room profiles. The curves are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

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 number of room might still grow).

### 5.2 Different rooms

To make the task more challenging we next 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 must now learn to navigate through multiple types of room to reach the sub-goal instructed by the high-level agent. The performance of the algorithms with different rooms is shown in figure 3. The introduction of different room profiles allows us 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. This additional complexity results in a larger state space $\mathcal{S}_l$ but may also result in a longer horizon $H_l$ as the optimal trajectory might require more time to successfully navigate around obstacles to reach the instructed hallway. While it has very little effect on the standard Q-learning, this added difficulty 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 level of the hierarchy) the expected sample efficiency improve. 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 complexity of goal-conditioned HRL. To the best of our knowledge it is the first result that provide an analysis of the intrinsic 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 hierarchy to improve its sample efficiency. This experimental setting further emphasizes the usefulness of the proposed bound since empirical efficiency gains are supported by our theoretical findings.

Although this paper has taken a significant first step in bettering 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 does not yet apply in a continuous setting were function approximation could be leveraged to allow the low-level to generalise over sub-goals. Overcoming those limitations are interesting direction for future work.

## References


[1] Ekaterina Abramova, Luke Dickens, Daniel Kuhn, and Aldo Faisal. Hierarchical, heterogeneous control of non-linear dynamical systems using reinforcement learning. In *European Workshop On Reinforcement Learning, at ICML*, volume 2012, 2012.

[2] Peter Auer and Ronald Ortner. Online regret bounds for a new reinforcement learning algorithm. In *1st Austrian Cognitive Vision Workshop*, pages 35–42. Österr. Computer-Ges., 2005.

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

[4] Emma Brunskill and Lihong Li. Pac-inspired option discovery in lifelong reinforcement learning. In *International conference on machine learning*, pages 316–324. PMLR, 2014.

[5] Christoph Dann. *Strategic Exploration in Reinforcement Learning-New Algorithms and Learning Guarantees*. PhD thesis, Google, 2019.

[6] Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. *Advances in Neural Information Processing Systems*, 28, 2015.

[7] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

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

[9] Thomas G Dietterich et al. The maxq method for hierarchical reinforcement learning. In *ICML*, volume 98, pages 118–126, 1998.

[10] Ronan Fruit and Alessandro Lazaric. Exploration-exploitation in mdps with options. In *Artificial intelligence and statistics*, pages 576–584. PMLR, 2017.

[11] Ronan Fruit, Matteo Pirotta, Alessandro Lazaric, and Emma Brunskill. Regret minimization in mdps with options without prior knowledge. *Advances in Neural Information Processing Systems*, 30, 2017.

[12] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 1851–1860. PMLR, 2018.

[13] Matthias Hutsebaut-Buysse, Kevin Mets, and Steven Latré. Hierarchical reinforcement learning: A survey and open research challenges. *Machine Learning and Knowledge Extraction*, 4(1):172–221, 2022.

[14] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *Advances in neural information processing systems*, 31, 2018.

[15] Nicholas K Jong, Todd Hester, and Peter Stone. The utility of temporal abstraction in reinforcement learning. In *AAMAS (1)*, pages 299–306, 2008.

[16] Gen Li, Laixi Shi, Yuxin Chen, Yuejie Chi, and Yuting Wei. Settling the sample complexity of model-based offline reinforcement learning. *arXiv preprint arXiv:2204.05275*, 2022.

[17] Amy McGovern, Richard S Sutton, and Andrew H Fagg. Roles of macro-actions in accelerating reinforcement learning. In *Grace Hopper celebration of women in computing*, volume 1317, page 15, 1997.

[18] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[19] Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning? *arXiv preprint arXiv:1909.10618*, 2019.

[20] Marc Pickett and Andrew G Barto. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *ICML*, volume 19, pages 506–513, 2002.

[21] Doina Precup and Richard S Sutton. Multi-time models for temporally abstract planning. *Advances in neural information processing systems*, 10, 1997.

[22] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[23] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[24] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[25] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[26] Alexander Vezhnevets, Volodymyr Mnih, Simon Osindero, Alex Graves, Oriol Vinyals, John Agapiou, et al. Strategic attentive writer for learning macro-actions. *Advances in neural information processing systems*, 29, 2016.

[27] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

[28] Zheng Wen, Doina Precup, Morteza Ibrahimi, Andre Barreto, Benjamin Van Roy, and Satinder Singh. On efficiency in hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 33:6708–6718, 2020.