Efficient Exploitation of Hierarchical Structure in Sparse Reward Reinforcement Learning

Gianluca Drappo[*,†]
gianluca.drappo@polimi.it
Politecnico Di Milano

Arnaud Robert[*]
a.robert20@imperial.ac.uk
Imperial College London

Marcello Restelli
marcello.restelli@polimi.it
Politecnico Di Milano

Aldo A. Faisal
a.faisal@imperial.ac.uk
Imperial College London
University of Bayreuth

Alberto Maria Metelli
albertomaria.metelli@polimi.it
Politecnico Di Milano

Ciara Pike-Burke
c.pike-burke@imperial.ac.uk
Imperial College London

Abstract

We study goal-conditioned Hierarchical Reinforcement Learning (HRL), where a high-level agent instructs sub-goals to a low-level agent. Under the assumption of a sparse reward function and known hierarchical decomposition, we propose a new algorithm to learn optimal hierarchical policies. Our algorithm takes a low-level policy as input and is flexible enough to work with a wide range of low-level policies. We show that when the low-level policy is optimistic and provably efficient, our HRL algorithm enjoys a regret bound which represents a significant improvement compared to previous results for HRL. Importantly, our regret upper bound highlights key characteristics of the hierarchical decomposition that guarantee that our hierarchical algorithm is more efficient than the best monolithic approach. We support our theoretical findings with experiments that underscore that our method consistently outperforms algorithms that ignore the hierarchical structure.

1 INTRODUCTION

This work focuses on finding efficient solutions to a family of finite-horizon Markov Decision Processes (MDPs) with a sparse reward function, which only rewards the agent when it successfully reaches the pre-specified goal state. While the sparse reward assumption may appear limitating, it is present in various problems. Sparse reward functions are commonly used in practice for goal-based tasks such as learning robot skills and motion planning, where the learner is only rewarded when the goal is reached (Mülling et al., 2013; Andrychowicz et al., 2017; Qureshi et al., 2019). These reward functions are common and easy to design, making them of great practical interest. In this work, we study how this model can be used to understand one of the most challenging aspects of Hierarchical Reinforcement Learning (HRL): jointly learning the high-level and low-level policies efficiently.

Many goal-based tasks can be hierarchically decomposed into a sequence of individually solvable sub-tasks. To tackle such problems, we propose a goal-conditioned Hierarchical Reinforcement Learning (HRL) approach (Pateria et al., 2021) that leverages the structured decomposition of the task into sub-tasks. Our method learns two different policies: a high-level policy instructing sub-goals and a low-level policy which aims to solve them. We will see that it suffices to use a standard algorithm to compute the low-level policy. To learn the high-level policy, our method uses the low-level value function as a proxy for the transition function. With a sparse reward function, the value function of a policy instructed to solve a sub-task is exactly the probability with which the policy will complete the sub-task and, consequently, move on to solving the next sub-task.

Our main result is to derive an upper bound on the regret incurred by the proposed algorithm (see Theorem 4.1). The exact regret will depend on the choice of the low-level algorithm. Many low-level methods can be used within our algorithm, the only require-

[*]Equal contribution. [†]While visiting Imperial College. Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

ment of our analysis is that the low-level algorithm is optimistic (see Def. 2). When a provably efficient method is used, such as UCBVI (Azar et al., 2017), our regret bound (see Corollary 4.2) is an improvement on the other results in the HRL literature. Moreover, our algorithm outperforms any standard RL method as long as the considered MDP exhibits a specific hierarchical structure. We also empirically validate our findings in several maze environments and show that the proposed algorithm consistently outperforms its monolithic counterpart when the environment exhibits the expected hierarchical structure.

2 RELATED WORK

Despite a growing literature of provably efficient algorithms for standard Reinforcement Learning (RL) (also referred to as *monolithic* approaches due to the lack of a hierarchical structure, e.g. Auer et al. (2008); Dann and Brunskill (2015); Azar et al. (2017)), theoretical guarantees for HRL are still lacking. Most theoretical guarantees are formulated in the options framework (Sutton et al., 1999) and rely on the assumption that the option policies are known in advance (Fruit et al., 2017; Fruit and Lazaric, 2017). The only exceptions are the analyses of Wen et al. (2020); Drappo et al. (2023, 2024), which provide upper bounds for jointly learning both levels of policies. However, Wen et al. (2020) only provides guarantees on the Bayesian regret and requires additional assumptions on the reward function to allow the propagation of the relative value of each sub-task. Drappo et al. (2023) provides sub-optimal guarantees using an Explore-then-Commit (Lattimore and Szepesvári, 2020) based approach. Lastly, Drappo et al. (2024) provides an improved regret bound, but this bound involves constants that depend on the learning behaviour which can grow arbitrarily large. Robert et al. (2024) provides a lower bound on the sample complexity of goal-conditioned HRL. Note that the lower bound of Robert et al. (2024) does not necessarily apply to the setting considered in this paper since we assume the reward is sparse and that we have access to a deterministic function that maps each possible high-level state and sub-goal pair to the next high-level state in the case of successful completion of the sub-goal[1].

3 PROBLEM SETTING

This section introduces the problem setting, necessary notations[2] and key assumptions. Ultimately, we are interested in solving a goal-based finite-horizon MDP; we first recall the definition of this specific framework (Sec. 3.1). Then, we define the hierarchical decomposition considered in this work (Sec. 3.2), showing how we can reconstruct the original MDP from the decomposition (Sec. 3.2.3). We conclude by stating the learning objective and the assumptions made (Sec. 3.3).

3.1 Goal-based Finite-Horizon Markov Decision Process with Sparse Reward

A goal-based finite-horizon MDP is a tuple $M_o = \langle \mathcal{S}_o, \mathcal{A}_o, R_o, P_o, H_o, s_{o,1}, g\rangle$. In general for a finite set $\mathcal{X}$, $X = |\mathcal{X}|$ defines its cardinality, and for any integer $I \in \mathbb{R}$, $[I] = \{1, \cdots, I\}$. We refer to this MDP as the original MDP. Hence, we index each of its components with the subscript o. The state space $\mathcal{S}_o$ and action space $\mathcal{A}_o$ are finite. The state space also contains the goal state $g \in \mathcal{S}_o$, which is an absorbing state. The goal state is used to characterise the sparse reward function $R_o : \mathcal{S}_o \times [H] \to [0, 1]$, which depends on the state and the time step, and is equal to 0 except if in the last step H the current state is the goal state, in which case the reward is equal to 1, that is $R_o(s, h) = \mathbb{I}\{s = g \cap h = H\}$.

The transition model $P_o : \mathcal{S}_o \times \mathcal{A}_o \times \mathcal{S}_o \to [0, 1]$ provides the probability of reaching a specific state s' after executing an action a in state s, $P_o(s'|s, a)$. The horizon H_o determines the number of interactions with the environment the agent can perform within a single episode. Each episode starts from an initial state $s_{o,1} \in \mathcal{S}_o$; then, for the following $H_o - 1$ steps, the agent interacts with the MDP by choosing an action a depending on the observed state $s \in \mathcal{S}_o$. The result of this action is immediately observed as a new state $s' \sim P_o(\cdot|s, a)$ and an immediate reward $R_o(s', h)$, where h denotes the current timestep. After K episodes, the algorithm has interacted with the environment for $T = KH_o$ times.

The behaviour of an agent is modelled by a policy $\pi : \mathcal{S}_o \to \mathcal{A}_o$. We measure the quality of a policy with its value function; for any $s \in \mathcal{S}_o$ and $t_o \in [H_o]$

$$V^{\pi}_{o,t_o}(s) = \mathbb{E}\Bigg[\sum_{i=t_o}^{H_o} R_o(s_i, i)\Bigg| a_i = \pi(s_i), s_{t_o} = s, \\ s_{i+1} \sim P_o(\cdot|s_i, a_i)\Bigg].$$

The aim of RL algorithms is to find an optimal policy π^*, within the set of Markov deterministic policies Π, with maximal value function, $\pi^* = \operatorname{argmax}_{\pi\in\Pi} V^{\pi}_{o,1}(s_{o,1})$.

3.2 Hierarchical MDP

We now introduce the concepts of hierarchical decomposition of an MDP, and the associated notation. We decompose an MDP into high- and low-level

[1]Refer to Appendix B for a more detailed discussion.

[2]We refer to Appendix A for a table summarising all the notation used

MDPs. In particular, the hierarchical decomposition of an original finite horizon MDP M_o is expressed as $M_{l \times h} = (\{M_{l,i}\}_{i=1}^{L},\ M_h)$, where $\{M_{l,i}\}_{i=1}^{L}$ is the set of L distinct low-level MDPs and M_h is the high-level MDP. It is known that such a decomposition can always be constructed (Robert et al., 2024). The algorithm and the theoretical guarantees presented below assume that the decomposition is known. We now define the low- and high-level MDPs and then show the connection between the decomposition and the original MDP. We finally highlight how executing a hierarchical policy in the decomposed MDP corresponds to executing the corresponding policy in the original MDP.

3.2.1 Low-level MDPs

The low-level MDPs are a set of L unique MDPs $\{M_{l,i}\}_{i=1}^{L}$; they are similar to the concept of *sub-MDP* introduced by Wen et al. (2020). One low-level MDP is formally defined as $M_{l,i} = \langle \mathcal{S}_{l,i} \cup \mathcal{E}_i, \mathcal{A}_o, R_{l,i}, P_{l,i}, H_l, \rho_{l,i} \rangle$. Here, $\mathcal{S}_{l,i}$ is the low-level state space of the i^{th} low-level MDP. $\mathcal{E}_i$ is the set of reachable sub-goals; these sub-goals are represented as additional artificial absorbing states (note that the goal state g is the only sub-goal that is not artificial). We use the notation $\mathcal{S}_{l,i}^{+}$ to represent the extended state space, $\mathcal{S}_{l,i}^{+} = \mathcal{S}_{l,i} \cup \mathcal{E}_i$. Note that one of the main benefits of hierarchical decomposition is its ability to capture repetitive patterns in the original MDP. In particular, the decomposition is useful if the size of all the low-level state spaces is smaller than the original state space, $\sum_{i=1}^{L} S_{l,i} < S_o$. Another benefit of the decomposition is that it breaks down the length of the episodes. A single low-level episode lasts for H_l steps, with the low-level horizon $H_l \leq H_o$. Note that the low-level action space consists of the original action space $\mathcal{A}_o$. The low-level reward function is an artificial construction, $R_{l,i} : \mathcal{S}_{l,i}^{+} \times \mathcal{E}_i \times [H_l] \to [0,1]$, which is defined as, $R_{l,i}(s, e, t_l) = \mathbb{I}\{s = e, t_l = H_l\}$ for state $s \in \mathcal{S}_{l,i}^{+}$ and the goal $e \in \mathcal{E}_i$ at time step $t_l \in [H_l]$.

The learning agent interacts in each low-level MDP for H_l steps using a deterministic policy $\nu : \cup_{i=1}^{L} \mathcal{S}_{l,i}^{+} \to \mathcal{A}_o$. Let e be the sub-goal state visited at the end of the previous low-level episode. The initial state in the next low-level MDP $M_{l,i}$ is determined by the map $\rho_{l,i}(e) = s_1$, with $s_1 \in \mathcal{S}_{l,i}$. The low-level transition function, $P_{l,i} : \mathcal{S}_{l,i}^{+} \times \mathcal{A}_o \times \mathcal{S}_{l,i}^{+} \to [0,1]$ encodes the dynamics of P_o on the subset of states whose low-level description belong to $S_{l,i}$. Note that transitions from $\mathcal{E}_i$ are trivial as every sub-goal state is absorbing. So, we focus only on transitions from $S_{l,i}$. Each original state $s_o \in \mathcal{S}_o$ has a corresponding low-level state $s_l \in \mathcal{S}_{l,i}$. There are two distinct types of low-level transitions that we need to consider: transitions from $s_l \in \mathcal{S}_{l,i}$ to $s_l' \in \mathcal{S}_{l,i}$ and the ones from $s_l \in \mathcal{S}_{l,i}$ to $e \in \mathcal{E}_i$. First, given $s_o, s_o' \in \mathcal{S}_o$ and if their corresponding low level description $s_l, s_l' \in \mathcal{S}_{l,i}$, the low-level transition can be written as $P_{l,i}(s_l'|s_l, a) = P_o(s_o'|s_o, a)$ for all $a \in \mathcal{A}_o$. Second, given two low-level MDPs $M_{l,i}$ and $M_{l,j}$, let's consider that $M_{l,i}$ is connected to $M_{l,j}$ via the sub-goal $e \in \mathcal{E}_i$, which means that when the agent successfully completes e while in MDP $M_{l,i}$, it will start the next low-level episode in MDP $M_{l,j}$. The probability of transitioning to the artificial sub-goal state e from a connected low-level state $s_l \in \mathcal{S}_{l,i}$ corresponds to the transition probability in the original MDP to go from the original state associated to s_l, $s \in \mathcal{S}_o$, to s', the state corresponding to the initial state in the next MDP $s_l' = \rho_{l,j}(e)$. So we have $P_{l,i}(e|s_l, a) = P_o(s'|s, a)$ for all $a \in \mathcal{A}_o$. Because the sub-goal states are absorbing, we define the associated deterministic transition to be $P_{l,i}(e|e, a) = 1$ for any $e \in \mathcal{E}_i, a \in \mathcal{A}_o$ and 0 otherwise. Note that when actions are selected by a low-level policy ν , we compactly write the transition function vector as $P_{l,i}^{\nu}(\cdot|s_l)$, which contains the probabilities $P_{l,i}(s_l'|s_l, \nu(s_l))$ for all s_l', $s_l \in \mathcal{S}_{l,i}^{+}$.

We define the low-level value function of the policy ν, in low-level state $s \in \mathcal{S}_{l,i}$, when tasked to solve the sub-goal $e \in \mathcal{E}_i$ in the low-level MDP $M_{l,i}$, $V_{l,i,t_l}^{\nu}(s, e)$, at low-level time step $t_l \in \{1, \ldots H_l\}$, as:

$$V_{l,i,t_l}^{\nu}(s, e) = R_{l,i}(s, e, t_l) + P_{l,i}^{\nu}(\cdot|s)^T V_{l,i,t_l+1}^{\nu}(\cdot, e), \quad (1)$$

with $V_{l,i,H_l}^{\nu}(s, e) = R_{l,i}(s, e, H_l)$ for all $s \in \mathcal{S}_{l,i}^{+}$ and $e \in \mathcal{E}_i$. Essentially, the low-level value function consists of the expected total low-level reward. The optimal low-level policy ν^* is the policy that maximises the value function:

$$\nu^*(s, e) = \operatorname*{argmax}_{\nu \in \mathcal{N}} V_{l,i,1}^{\nu}(s, e), \quad (2)$$

for all $i \in [L]$, $e \in [|\mathcal{E}_i|]$, and $s \in \mathcal{S}_{l,i}$ and with $\mathcal{N}$ being the set of deterministic low-level policies.

Recall that the low-level reward function only rewards the agent if it reaches the sub-goal within the H_l steps. In this setting, low-level policies with the highest value function have the highest probability of reaching the sub-goal. The value function of a low-level policy can be directly related to its probability of reaching the specific sub-goal. For this, we define an event describing a successful sub-goal completion by a policy ν.

Definition 1. *We define $\text{end}_\nu(i, e, s_1)$ as the event that the policy ν successfully reached sub-goal $e \in \mathcal{E}_i$ given that it starts its trajectory in state $s_1 \in \mathcal{S}_{l,i}$ in low-level MDP $M_{l,i}$, that is $\text{end}_\nu(i, e) = \{s_{H_l} = e\}$ where $\tau = s_1, \ldots, s_{H_l}$ is a trajectory produced using the policy ν.*

The low-level value function can then be written as $V_{l,i,1}^{\nu}(s, e) = \mathbb{P}(\text{end}_\nu(i, e, s))$, for any low-level MDP $M_{l,i}$, sub-goal $e \in \mathcal{E}_i$ and starting state $s_1 \in \mathcal{S}_{l,i}$.

3.2.2 The High-Level MDP

The high-level MDP is defined by the following tuple $M_h = \langle \mathcal{S}_h, \mathcal{A}_h, R_h, P_h^\nu, H_h, s_{h,1}, \phi, \psi \rangle$. The high-level MDP operates at a higher temporal resolution and essentially navigates between low-level MDPs. The hierarchical structure suggests that there is an associated low-level MDP for each high-level state. The relationship between the high-level state $s_h \in \mathcal{S}_h$ and its corresponding low-level MDP $M_{l,i}$ is encoded by the map $\phi : \mathcal{S}_h \to [L]$. If $\phi(s) = i$, the low-level MDP corresponding to the high-level state $s \in \mathcal{S}_h$ is $M_{l,i}$. The high-level MDP actions are sub-goals that are instructed to the low-level policy. Specifically, the high-level action space in the high-level state s_h comprises the sub-goals of the current low-level MDP $M_{\phi(s_h),l}$, denoted by $\mathcal{E}_{\phi(s_h)}$. The complete high-level action space $\mathcal{A}_h = \cup_{i=1}^{L} \mathcal{E}_i$ consists of the union of the sub-goals $\mathcal{E}_i$ in the unique sub-MDPs $M_{l,i}$. The hierarchical decomposition also includes a structure map, ψ, that associates to each high-level state and sub-goal pair the next high-level state that would be reached if the sub-goal was completed successfully, $\psi(s_h, e) = s'_h$, for all $s_h \in \mathcal{S}_h$, $e \in \mathcal{E}_{\phi(s_h)}$. We assume this structure map is known; in the case this map is unknown, it would have a very limited impact on the algorithmic complexity as a single successful completion of every sub-goal would be sufficient to learn the map.

Each high-level episode starts in a unique starting state $s_{h,1}$, which corresponds to the high-level description of the original MDP initial state, $s_{o,1}$, and lasts $H_h - 1$ decisions — H_h states are visited. The end goal g is the only original state to have both a high-level representation s_h^g and a low-level representation s_l^g. As g is absorbing, the corresponding high- and low-level representations are absorbing as well. With this in mind, we define the high-level reward function:

$$R_h(s_h, t_h) = \begin{cases} 1, & \text{if } s_h = s_h^g \quad \text{and} \quad t_h = H_h \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

for all high-level states $s_h \in \mathcal{S}_h$ and high-level steps $t_h \in \{1, \cdots, H_h\}$. There is only one high-level goal state; thus, even the high-level reward function R_h is sparse and assigns 0 to all high-level states except for the high-level state associated with the unique low-level MDP that contains the goal state. The high-level transition function P_h^ν encodes the probability of reaching a high-level state $s'_h \in \mathcal{S}_h$ given that in high-level state $s_h \in \mathcal{S}_h$ the low-level policy ν was instructed to reach the sub-goal $e \in \mathcal{E}_{\phi(s_h)}$, $P_h^\nu(s'_h | s_h, e)$. Learning P_h^ν is generally challenging as its dependence on the low-level policy, ν, makes it non-stationary.

The high-level policy is defined as $\mu : \mathcal{S}_h \to \mathcal{A}_h$. When sub-goals are selected by a policy μ and a low-level policy ν is used to solve each sub-goal, we define the high-level value function, $V_{h,t_h}^{(\nu,\mu)}$, of the policy μ, at high-level time step $t_h \in \{1, \dots H_h\}$ as:

$$V_{h,t_h}^{(\nu,\mu)}(s) = R_h(s, t_h) + \sum_{s' \in \mathcal{S}_h} P_h^\nu(s' | s, \mu(s)) V_{h,t_h+1}^{(\nu,\mu)}(s'),$$

for all $s \in \mathcal{S}_h$ and with $V_{h,H}^{(\nu,\mu)}(s) = R_h(s, H_h)$ for all $s \in \mathcal{S}_h$. The high-level value depends on the low-level policy ν since ν selects primitive actions in $\mathcal{A}_o$. In particular, we define the optimal pair of policies (ν^*, μ^*) as the pair that maximises the value function:

$$(\nu^*, \mu^*) = \operatorname*{argmax}_{(\nu,\mu)} V_{h,1}^{(\nu,\mu)}.$$

3.2.3 Equivalence between the original and the hierarchical MDPs

The hierarchical algorithm we consider will learn an optimal policy in the decomposed MDP $M_{l\times h}$. This section highlights the connection between the hierarchical task and the original task; in particular, we show that every trajectory in the decomposed MDP $M_{l\times h}$ has a corresponding trajectory in the original MDP M_o. We illustrate this for a small grid world example in Fig. 1 in the next subsection. In what follows, we show the equivalence between each element of the decomposed MDP and the original MDP.

State Space: A state in the original MDP $s_o \in \mathcal{S}_o$ can be factorized into a pair of states $(s_{l,i}, s_h)$ where $s_{l,i} \in \mathcal{S}_{l,i}$ is a state in the i^{th} low-level MDP and $s_h \in \mathcal{S}_h$ is a state in the high-level MDP. Specifically, the low-level state, $s_{l,i}$, and the high-level state, s_h, contain all the necessary information to reconstruct the corresponding state, s_o, in the original MDP. As each high-level state corresponds to a low-level MDP, there are $|\mathcal{S}_h|$ low-level MDPs, with $L \leq |\mathcal{S}_h|$ unique low-level MDPs. The size of the original state space is the sum of the size of all the low-level state spaces $S_o = \sum_{s \in \mathcal{S}_h} S_{l,\phi(s)}$.

Action Space: Interactions with the environment are only possible through the original action space $\mathcal{A}_o$; hence the action space of the low-level MDPs is exactly $\mathcal{A}_o$. The high-level action space encodes all the sub-goals the high-level agent wants to instruct; this corresponds to a set of *artificial* low-level states and the goal state; we assume this set to be given by the decomposition. In each high-level state $s \in \mathcal{S}_h$, the agent instructs the low-level agent to reach a sub-goal $e \in \mathcal{E}_{\phi(s)}$. Then the low-level agent selects actions $a \in \mathcal{A}_o$ to reach e.

Transition: Most of the complexity of the transition dynamics is encoded in the low-level MDP dynamics $P_{l,i}$. Essentially, the low-level and the original transition functions are almost identical for all the states

within a sub-MDP $\mathcal{S}_{l,i}$. The only difference is that the original transition function operates over the full state description $\mathcal{S}_o$. The low-level transition function operates over $\{\mathcal{S}_{l,i}^{+}\}_{i=1}^{L}$, which considers only the low-level description of the states, and the transition to and from the artificial sub-goal states $\mathcal{E}_i$. To reconstruct the original transition function from the decomposed transition function, we consider two different scenarios.
(i) First, transitions from s_o to s_o' where the low-level description of both states lies in the same low-level MDP $s_l, s_l' \in \mathcal{S}_{l,i}$. Then, the low-level transition function is equal to a restriction of the original transition in the appropriate low-level state space $P_o(s_o'|s_o,a) = P_{l,i}(s_l'|s_l,a)\ \forall a \in \mathcal{A}_o$.
(ii) Second, we consider the transition between the low-level states $s_l \in \mathcal{S}_{l,\phi(s_h)}$ and $s_l' \in \mathcal{S}_{l,\phi(s_h')}$ where the associated high-level states $s_h, s_h' \in \mathcal{S}_h$ are different. We consider all sub-goal states, $e \in \mathcal{E}_{\phi(s_h)}$, supporting such a transition. Formally, we consider the set $Z = \{e : \exists \nu \in \mathcal{N}\ \text{ s.t. } P_h^{\nu}(s_h'|s_h,e) > 0 \text{ and } \rho_{l,\phi(s_h')}(e) = s_l'\}$, where $e \in \mathcal{E}_{\phi(s_h)}$ are sub-goals, $s_h, s_h' \in \mathcal{S}_h$ are the high-level state description of the current and the next state, and $\rho_{l,\phi(s_h')}(e)$ denotes the initial state in low-level MDP $M_{l,\phi(s_h')}$ when the low-level agent reaches the sub-goal $e \in \mathcal{E}_{\phi}(s_h)$ in the previous low-level episode. The original transition is the probability of going via any of the sub-goal states in Z,

$$P_o(s_o'|s_o,a) = \sum_{e \in Z} P_{l,i}(e|s_l,a) \quad \forall a \in \mathcal{A}_o,$$

where $s_l \in \mathcal{S}_{l,i}$ is the low-level state description associated with the state in the original MDP $s_o \in \mathcal{S}_o$.

Reward There is a direct equivalence between the original reward function and the reward functions in the decomposed MDPs. Since the high-level reward function only rewards the agent when it ends the episode in the absorbing high-level goal state, it ignores all the bonuses received when the low-level policy reaches a sub-goal $e \in \mathcal{A}_h \setminus g$. Hence, the only high-level reward signal preserved is when that agent reaches the goal state g; hence, the high-level reward function is a restriction of the original reward function on $\mathcal{S}_h$.

Horizon The two levels of the hierarchy operate at two different time scales, with the low-level episode $t_l \in \{1, \dots H_l\}$ happening completely within one high-level time step. The high-level agent observes H_h high-level states and instructs $H_h - 1$ sub-goals in an episode. Hence, the total number of decisions in both MDPs is equivalent: $H_o - 1 = (H_l - 1)(H_h - 1)$. Note that the control returns to the high-level policy exactly after H_l steps; even if the low-level policy reaches the sub-goal in fewer steps, it waits in the absorbing low-level MDP sub-goal state. Each time the high-level recovers the control, it observes the current high-level state s_h and selects the next sub-goal.

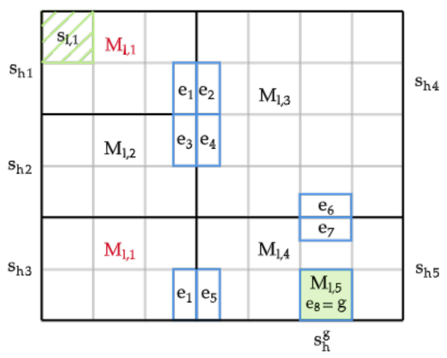

Figure 1: An illustration of a grid world and its associated hierarchical decomposition.

3.2.4 A simple example

To illustrate the notions introduced so far, we consider the simple grid world environment depicted in Fig. 1. This grid world is composed of rooms that are delimited by solid black lines the goal of the agent is to navigate from the initial state (the striped green square on the top left corner) to the goal state g, depicted as a green square. This problem can naturally be modeled as an MDP. Each grey cell represents a state, and an action consists of moving from one cell to a neighboring cell (in one of the four cardinal directions). The goal of the agent is to find, within some horizon H, a path that connects the initial state to the goal state g. Alternatively, one can notice that there is more structure to this problem. In particular there are bottleneck states,i.e. hallways, which are mandatory steps in order to go from one room to the next and there are rooms which share the same structure as well. The hierarchical decomposition takes advantage of such structures. The high level states consist of the different rooms (depicted with solid black lines); inside each room, the low-level states are illustrated with grey squares. One can observe that the low-level MDP $M_{1,l}$ is used twice (in the top and bottom left rooms). The blue rectangle represents the artificial exit states that connect two low-level MDPs, they correspond to the high-level actions. Lastly, the original goal $g \in \mathcal{S}_l$ and $g \in \mathcal{S}_o$, is illustrated with a green cell in the bottom right room, and it is the only sub-goal state that is not an artificial construction and that belongs to the low-level state space $\mathcal{S}_{l,i}$. Additionally, the green cell also defines the artificial high-level state s_h^g and its corresponding unique low-level MDP $M_{l,5}$, in which the agent can wait the end of the episode once reached.

3.3 Learning in a Hierarchical MDP

The hierarchical decomposition described in section 3.2.3 contains additional information about the structure of the problem, like the structure map ψ and the sub-goal spaces $\{\mathcal{E}_i\}_{i=1}^{L}$. We assume that this additional information is known to the learner. We now focus on developing an algorithm to solve such hierarchical problems efficiently, assuming that the hierarchical structure and additional information is known. We evaluate the efficiency of our algorithm using the notion of regret (Auer et al., 2008; Azar et al., 2017)

Regret The algorithms proceed in episodes $k = 1, \dots, K$; at the beginning of each high-level episode, the algorithm computes a hierarchical pair of policies $(\nu, \mu)_k$. The regret is defined as the cumulative sum of differences between the value of the optimal pair of hierarchical policies (ν^*, μ^*) and the policies played in each episode. Formally the regret suffered by a hierarchical algorithm $\mathfrak{A}$, which computes in each episode, k, the pair of policies $(\nu, \mu)_k$, over K episodes is:

$$R^h_{\mathfrak{A}}(M_{l\times h}, K) \coloneqq \sum_{k=1}^{K} V_{h,1}^{(\nu^*,\mu^*)}(s_1) - V_{h,1}^{(\nu,\mu)_k}(s_1), \quad (4)$$

with $s_1 \in \mathcal{S}_h$ the initial state in the high-level MDP.

Note that in this regret formulation, we ignore a linear term $K\Delta_{struct}(M_{l\times h})$, that accounts for the sub-optimality gap, $\Delta_{struct}(M_{l\times h}) = V_{o,1}^{\pi^*_1} - V_{h,1}^{\nu^*,\mu^*}$, induced by the hierarchical decomposition. This occurs since the value of the optimal hierarchical policy, $V_{h,1}^{\nu^*,\mu^*}$, which is the target of our algorithm (see the definition of regret in Eq. 4), might be smaller than the value of the optimal policy in the original MDP $V_{o,1}^{\pi^*_1}$. The goal is to minimize the regret in Eq. (4). Before formally presenting our algorithm in Section 4, we conclude this section by highlighting some important assumptions that we make.

Assumption 1: Every low-level MDP is communicating. This ensures that for every low-level MDP $M_{l,i}$, with $i \in [L]$ and any subgoal $e \in \mathcal{E}_i$, there exists a low-level policy ν able to reach the sub-goal state e within H_l steps.
Assumption 2: The structure map ψ is such that there always exists a sequence of $H_h - 1$ (or less) subgoals $e_1, \cdots, e_{H_h-1}$ that starts in the high-level initial state s_1 and ends in the high-level goal state s^g_h, with $s_{h+1} = \psi(s_h, e_h)$ for $h = 1 \cdots, H_h$.
Assumption 3: The original MDP M_o is time-homogeneous. This is a common assumption in RL as it simplifies the setting and the notation.

It is also helpful to consider the regret suffered by a low-level algorithm $\mathfrak{B}$ which, in each low-level episode $n = 1, \dots, N$, selects the policy ν_n to solve sub-goal $e \in \mathcal{E}_i$ in MDP $M_{l,i}$.

$$R^l_{\mathfrak{B}}((M_{l,i}, e), N) \coloneqq \sum_{n=1}^{N} V_{l,i,1}^{\nu^*}(s_n, e) - V_{l,i,1}^{\nu_n}(s_n, e). \quad (5)$$

Here, $V_{l,i,1}^{\nu^*}(s_n, e)$ and $V_{l,i,1}^{\nu_n}(s_n, e)$ are respectively the value functions of the optimal policy, ν^*, and the current policy, ν_n, when tasked solve sub-goal e in MDP $M_{l,i}$ from s_n— the low-level state in which the low-level episode n started.

4 MAIN RESULT

We propose an algorithm that directly leverages the known hierarchical decomposition to efficiently solve the original MDP. In this section, we first describe the general framework and define the set of low-level algorithms that can be used. We then state the main theorem, which provides a general upper bound on the worst-case regret in terms of the regret of the low-level algorithm. We conclude the section with a corollary bounding the total regret when the low-level algorithm is UCBVI (Azar et al., 2017).

We present the Goal-conditioned Hierarchical Reinforcement Learning algorithm (GHRL) in Alg.1. Intuitively, GHRL exploits the fact that the low-level value functions $V_{l,i}^{\nu}(s, e)$ of a policy ν is equivalent to the probability that ν successfully complete sub-goal e in $M_{l,i}$. Specifically, the computed optimistic low-level value functions are used to construct an optimistic approximation of the high-level MDP $\tilde{M}_h$. Using backward induction in $\tilde{M}_h$, GHRL computes an optimistic high-level plan. GHRL takes as input the number of high-level episodes, K, the high-level structure map ψ, the high-level reward function R_h, and the choice of the low-level regret minimiser $\mathfrak{B}$. To highlight that a specific low-level algorithm $\mathfrak{B}$ was used, we call the resulting hierarchical algorithm GHRL$_{\mathfrak{B}}$. Our theoretical guarantee requires that GHRL$_{\mathfrak{B}}$ receives as input a low-level algorithm that belongs to the set optimistic algorithms, $\mathfrak{B} \in \Xi$, where we define an optimistic algorithm as follows.

Definition 2. *Given a confidence value $\beta \geq 0$, an algorithm is said to be $(1-\beta)$-optimistic if, in each round k, it computes an optimistic value function $\tilde{V}_{l,i,1}^{\nu_k}$, such that $\tilde{V}_{l,i,1}^{\nu_k}(s_l) \geq V_{l,i,1}^{*}(s_l)$, with probability $1-\beta$, for every state $s_l \in \mathcal{S}_{l,i}$.*

For example algorithms like UCBVI Azar et al. (2017), UCRL2 Auer et al. (2008) or Q-UCB Jin et al. (2018) belong to Ξ. The optimism of the low-level algorithm is required to ensure sufficient exploration for the high-level policy, which guarantees all low-level MDP sub-goal pairs are visited sufficiently often.

The algorithm maintains a dataset D^l_k containing all the low-level information collected up to the k^{th} episode. In each episode k, the optimistic low-level algorithm $\mathfrak{B} \in \Xi$ uses D^l_k to compute an optimistic low-level policy ν_k, and an optimistic value function $\tilde{V}_{l,i}^{\nu_k}$ for each low-level MDP, $M_{l,i}$ and sub-goal $e \in \mathcal{E}_i$ pair.

The algorithm's key innovation is to compute an optimistic approximation of the high-level transition function $P_h^{\nu_k}$ under the low-level policy ν_k. For this, we first recall that the structure map ψ indicates the intended high-level next state s'_h, given that in high-level state s_h, the instructed sub-goal is e, $\psi(s_h, e) = s'_h$. Additionally, in the sparse reward setting, the value function of the low-level policy, $V_{l,i,1}^{\nu_k}(s, e)$ is equivalent to the probability that policy ν_k reaches the sub-goal e while starting in state $s \in \mathcal{S}_{l,i}$. Consequently, the optimistic value function $\tilde{V}_{l,i,1}^{\nu_k}(s, e)$ upper bounds the probability that the current low-level policy reaches the instructed goal. Then, an optimistic approximation of the high-level transition function can be computed as follows:

$$\tilde{P}_h^{\nu_k}(s'_h|s_h, e) = \begin{cases} \min(\tilde{V}_{l,i,1}^{\nu_k}(s_l, e), 1), & \text{if } s'_h = \psi(s_h, e) \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where $i = \phi(s_h)$ is the low-level MDP associated with the current high-level state. Note that $\tilde{P}_h^{\nu_k}$ is an approximation of a transition function as it does not necessarily sum to 1, that is, $\sum_{s' \in \mathcal{S}_h} \tilde{P}_h^{\nu_k}(s'|s, e) \leq 1$.

For each episode k, the algorithm computes the high-level policy μ_k as the optimal policy of the optimistic high-level MDP $\tilde{M}_h = \langle \mathcal{S}_h, \mathcal{A}_h, R_h, \tilde{P}_h^{\nu_k}, H_h, s_{h,1}, \phi, \psi \rangle$, where the high-level transition function $\tilde{P}_h^{\nu_k}$ is the optimistic approximation of the true transition function $P_h^{\nu_k}$ under the current low-level policy ν_k. The optimistic high-level Q-function, $\tilde{Q}_h^{(\nu,\mu)_k}$, is computed via backward induction (see Alg. 2 in the Appendix) using the optimistic approximate transition function $\tilde{P}_h^{\nu}$ and the high-level reward R_h.

The current hierarchical policy pair $(\nu, \mu)_k$ is used to collect the low-level samples that are stored in the dataset $\mathcal{D}_l^k$. The high-level policy selects a sub-goal e in the initial state. The low-level policy ν_k is executed in the MDP $M_{l,i}$ with $i = \phi(s_h)$. The low-level agent aims to solve the sub-goals instructed by μ_k in the H_l steps. Whenever a low-level episode ends, either the sub-goal is reached, and the next sub-goal in the high-level plan is instructed, or the sub-goal is not reached. In cases where the sub-goal is not reached, the algorithm stops collecting samples for that episode.

Our main result is an upper bound on the regret suffered by GHRL (Alg. 1), which depends on the choice of the low-level algorithm, $\mathfrak{B}$.

Theorem 4.1. *Given $\delta \in [0, 1]$ and $\beta = \frac{\delta}{KA_h}$, for any hierarchical decomposition $M_{l\times h}$, and any $(1-\beta)$-optimistic low-level regret minimiser $\mathfrak{B} \in \Xi$, the regret of GHRL$_{\mathfrak{B}}$ is bounded, with probability $1-\delta$, by:*

$$R^h_{GHRL}(M_{l\times h}, K) \leq \sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} \Big(R^l_{\mathfrak{B}}\big((M_{l,i}, e), N(i, e)\big) + 2N(i, e) \max_{s_l \in \mathcal{S}_{l,i,1}} \mathbb{P}(\neg end_{\nu^*}(i, e, s)) \Big) \tag{7}$$

where $N(i, e)$, $\forall i \in [L]$ and $e \in \mathcal{E}_i$, is the number of times in which sub-goal e has been chosen in the state s_h, with $\phi(s_h) = i$ and $e = \mu_k(s_h)$. $R^l_{\mathfrak{B}}\big((M_{l,i}, e), N(i, e)\big)$ is the regret suffered by $\mathfrak{B}$ while learning to solve goal e in low-level MDP $M_{l,i}$ for $N(i, e)$ low-level episodes. $\mathcal{S}_{l,i,1}$ is the set of potential initial states in the MDP $M_{l,i}$, $\mathcal{S}_{l,i,1} = \bigcup_{e \in \mathcal{A}_h} \{s_1 = \rho_{l,i}(e) : s_1 \in \mathcal{S}_{l,i}\}$.

The proof of Theorem 4.1 follows from combining properties of the sparse reward function and the optimism of the low-level algorithm. Interestingly, the optimistic high-level plan, computed with $\tilde{Q}_h^{(\nu,\mu)_k}$, explores sub-goals considering the uncertainty of the low-level MDPs' transition dynamics, which is encoded into $\tilde{P}_k^{\nu}$. Thus, the regret of GHRL can be upper bounded by the probability of failing each instructed sub-goal, which directly relates to the low-level regret. A detailed proof of Theorem 4.1 is in Section E.

GHRL's theoretical and computational guarantees are then tightly linked to $\mathfrak{B}$'s guarantees and to the failure probability of the optimal policy, $\mathbb{P}(\neg\text{end}_{\nu^*}(i, e))$. The regret consists of the sum of the regret suffered by the low-level policy at each possible low-level MDP sub-goal pair. The approximate high-level MDP constructed does not model transitions after a failed sub-goal. Hence, our algorithm discards all samples that follow a failed sub-goal. The optimal hierarchical policy (ν^*, μ^*) does not suffer from this limitation; hence, the second term of Eq. (7) reflects the sub-optimality induced by this limitation. Additionally, this regret bound highlights the impact of the hierarchical structure $M_{l\times h}$ on the algorithm's performance, suggesting an improvement in problems with highly frequent patterns (i.e., $L \ll S_h$).

We obtain the following upper bound if we choose UCBVI Azar et al. (2017) as the low-level algorithm.

Corollary 4.2. *Let $\delta \in [0, 1]$ and $\beta = \frac{\delta}{KA_h}$ be the failure probability of the low-level algorithm, UCBVI. For any hierarchical decomposition $M_{l\times h}$ the regret suffered by GHRL$_{UCBVI}$ for K high-level episodes is bounded, with probability $1-\delta$, by:*

$$R^h_{GHRL}(M_{l\times h}, K) \leq \sqrt{A_o A_h \max_i \{S^+_{l,i}\} H_l K H_h} + \Delta \tag{8}$$

with

$$\Delta = \sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} 2N(i,e) \max_{s_l \in \mathcal{S}_{l,i,1}} \mathbb{P}(\neg end_{\nu^*}(i,e,s)).$$

A proof of Corollary 4.2 is in Appendix E.

Algorithm 1 Goal-conditioned HRL (GHRL)

```
1: Input: $K$ episodes, high-level structure map $\psi$, high-level reward function $R_h$, low-level regret minimizer $\mathfrak{B}$ and failure tolerance $\delta \in (0,1]$.
2: Output: with probability $1-\delta$ an optimal hierarchical policy.
3: Arbitrarily initialize $\mu_1$ and $\nu_1$, and $\mathcal{D}_1^l \leftarrow \{\}$
4: for $k = 1, \dots, K$ do
5:     $\nu_{i,k}, \tilde{V}_{l,i}^{\nu_k} = \mathfrak{B}(\mathcal{D}_k^l, M_{i,l}, e) \quad \forall i \in [L], e \in \mathcal{E}_i$
6:     $\tilde{V}_l^{\nu_k} = [\tilde{V}_{l,i}^{\nu_k}, \cdots \tilde{V}_{l,L}^{\nu_k}]$
7:     $\tilde{P}_h^{\nu_k}(s'_h|s_h, e)$ with Eq. (6) $\forall s_h, s'_h \in \mathcal{S}_h, e \in \mathcal{E}_{\phi(s_h)}$
8:     $\tilde{Q}_h^{(\nu,\mu)_k} = \text{BackwardInduction}(\mathcal{S}_h, \mathcal{A}_h, R_h, \tilde{P}_h^{\nu_k})$.
9:     $\mu_k(s_h) = \text{argmax}_{a_h} \tilde{Q}_{h,t_h}^{(\nu,\mu)_k}(s_h, a_h) \quad \forall s_h \in \mathcal{S}_h$;
10:    $s = s_{o,1} = (s_{l,i,1}, s_{h,1})$ with $i = \phi(s_{h,1})$
11:    $s_h = s_{h,1}, s_{l,i} = s_{l,i,1}$
12:    for $t_h = 1, \dots, H_h$ do
13:        $e = \mu_k(s_h)$
14:        for $t_l = 1, \dots, H_l$ do
15:            play action $a = \nu_k(s_{l,i}, e)$, in $M_{l,i}$ with $i = \phi(s_h)$
16:            observe $s'_{l,i}$, update $\mathcal{D}_{k+1}^l = \mathcal{D}_k^l \cup \{(s_{l,i}, a, s'_{l,i})\}$
17:            $s_{l,i} = s'_{l,i}$
18:        end for
19:        if $s_{l,i} \neq e$ then
20:            discard the rest of the sample and set $t_h = H_h, s_h = s_{h,1}, s_{l,i} = s_{l,i,1}$
21:        else
22:            $s_h = \psi(s_h, e)$
23:        end if
24:    end for
25: end for
```

5 DISCUSSION

If we consider hierarchical structures $M_{l\times h}$ where a hierarchical policy is optimal in the original MDP, i.e. where $\Delta_{struct}(M_{l\times h}) = 0$, we can compare the result in Corollary 4.2 with the performance of the best monolithic approach. Osband and Van Roy (2016) provide a lower bound on the regret paid by any algorithm in the original MDP M_o:

$$R^o(M_o, K) = \Omega(\sqrt{H_o A_o S_o K}) \tag{9}$$

where R^o is the standard regret as defined in Osband and Van Roy (2016), Note that the sparse reward assumption allows us to replace the dependency on H_o^2 with H_o since the maximum achievable value is 1.

The ratio between the regret of GHRL$_{UCBVI}$ and the lower bound in Eq. (9), provides insights into when to prefer GHRL over any monolithic approach. The ratio up to constants can be written as:

$$\begin{aligned}\frac{R^h_{\text{GHRL}}(M_{l\times h}, K)}{R^o(M_o, K)} &\leq \frac{A_o A_h \max_i(S^+_{l,i}) H_l K H_h + \Delta}{A_o S_o H_o K} \\ &\leq \frac{A_h \max_i(S^+_{l,i})}{S_o} + \frac{2\mathbb{P}_{\max}}{A_o S_o H_o}\end{aligned} \tag{10}$$

where Eq. (10) comes from upper bounding Δ with $2\mathbb{P}_{\max}K$, where $\sum_{i=1}^L \sum_{e\in\mathcal{E}_i} N(i,e) = K$ and $\mathbb{P}_{\max} = \max_{i,e,s} \mathbb{P}(\neg\text{end}_{\nu^*}(i,e,s))$ is the largest probability that the optimal policy fails. In general, GHRL$_{\text{UCBVI}}$ is more efficient than any regret minimiser operating in the original MDP when the ratio in Eq. (10) is smaller than 1. First, we realise that $\mathbb{P}_{\max}$ is a probability as such is at most 1. Hence, for most MDPs, the second term of Eq. (10) will be negligible. We can then focus on the first term of Eq. 10. We first recall that $A_h = \sum_{i=1}^L |\mathcal{E}_i|$ and can be upper bounded by the number of low-level MDPs multiplied by the largest set of sub-goals $A_h \leq L \max_i(|\mathcal{E}_i|)$. This perspective allows us to clarify that we expect GHRL$_{\text{UCBVI}}$ to outperform any RL algorithm when $L \max_i(S^+_{l,i}) \max_i(|\mathcal{E}_i|) < S_o$. This condition highlights an essential property of the hierarchical structure that is required. There is a need for repetitive patterns inside the MDP, suggesting that $L \max_i(S^+_{l,i}) \ll S_o$. This source of efficiency gains should compensate for the low-level policy's added difficulty in solving multiple sub-goals. In particular, improvements are realisable if the number of sub-goals is constrained to:

$$\max_i(|\mathcal{E}_i|) \leq \frac{S}{L \max_i(S^+_{l,i})}. \tag{11}$$

6 EXPERIMENTS

We now empirically validate the GHRL algorithm (see Alg. 1) and show that the proposed method outperforms standard RL algorithms when the MDP exhibits a hierarchical structure[3]. The GHRL framework presented in Alg. 1 is flexible enough to accommodate a wide range of regret minimiser choices for the low-level algorithm. To empirically validate our framework, we consider Q-UCB Jin et al. (2018) to compute the low-level policy. We propose to evaluate this specific instance of GHRL on a path-like grid world composed of several rooms. An illustration of the environment considered is available in Appendix F (see Fig. 3). The goal is to go through the entire path from the leftmost room to the rightmost room, and the agent can only move from one cell of the maze to a neighbouring cell. Using rooms as the building block of our path leads to an immediate hierarchical structure. The high-level

[3]All experiments were run on a 12th Gen Intel Core i7 with 16GB of RAM

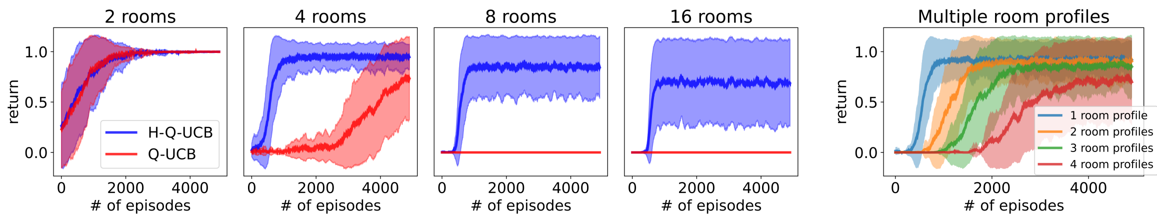

Figure 2: The four leftmost plots show the learning curves of GHRL with Q-UCB (in blue) and standard Q-UCB (in red). We compare the performance of the two algorithms on a navigation environment that describes a path composed of several rooms. We see that as we increase the number of identical rooms, the efficiency gain of GHRL becomes more evident. Additionally, the rightmost plot shows the impact of an increasing number of low-level MDPs on the learning curve. We ran the experiment for each plot with ten different random seeds; the solid line is the mean performance, and the shaded area covers two standard deviations.

agent navigates from room to room, and the low-level agent navigates within a room. We defined the low-level horizon as $H_l = 20$, the high-level horizon as equal to the number of rooms N, $H_h = N$, and the horizon in the monolithic problem is $H_o = 20 * N$. As illustrated on the four leftmost plots of Fig. 2, the performance of GHRL$_{Q\text{-}UCB}$[4] and Q-UCB are similar if there is only one room to navigate and the goal room (i.e. two rooms), as the hierarchy cannot be leveraged. However, as we increase the number of rooms, the benefit of the hierarchical framework becomes more visible. Note that for more than four rooms, Q-UCB cannot find the goal within the allowed number of interactions.

In the previous experiment, all the room profiles were identical, so there was only a single low-level MDP in the hierarchical decomposition. This benefits the hierarchical algorithm considerably. We repeat this experiment, but this time, we add different room profiles to challenge the low-level to learn to navigate more than a single low-level MDP. In the rightmost plot of Fig. 2, we show the learning curve of our algorithm as we increase the number of room profiles. As expected, as we increase the number of room profiles, the problem becomes more challenging, empirically showing what Corollary 4.2 highlighted.

7 CONCLUSION

This work addresses an important open problem in HRL, which is understanding to what extent jointly learning the high- and low-level policies impacts the efficiency gain of hierarchical approaches. To do so, we focus on the goal-based sparse reward setting. We propose an algorithm that leverages the optimism of the low-level agent to design an optimistic high-level plan and show that the regret of learning this joint policy is dominated by the regret of the low-level policy. Our method works for any choice of optimistic low-level regret minimiser, and we show how choosing a good low-level algorithm can lead to improved performance compared to monolithic RL approaches for particular structures. This improvement is largest when the MDP contains repeated patterns that an HRL algorithm can leverage. Our result improves upon existing regret upper bounds in HRL (Drappo et al., 2024; Wen et al., 2020). The sparse reward setting allowed us to highlight that problems exist for which an improved regret bound is achievable. Nevertheless, extending our results to more general reward functions is an interesting perspective for future work.

[4]Q-UCB's parameters are the failure probability, β and a constant $c > 0$. We set $\beta = 0.1$ and $c = 1 \times 10^{-6}$.

Acknowledgments

This paper is supported by FAIR (Future Artificial Intelligence Research) project, funded by the NextGenerationEU program within the PNRR-PE-AI scheme (M4C2, Investment 1.3, Line on Artificial Intelligence). AR was supported by an EPSRC CASE studentship supported by Shell, and AAF was supported by a UKRI Turing AI Fellowship (EP/V025449/1). The authors acknowledge AI4REALNET that has received funding from European Union's Horizon Europe Research and Innovation programme under the Grant Agreement No 101119527. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. (2017). Hindsight experience replay. *Advances in Neural Information Processing Systems*, 30.

Auer, P., Jaksch, T., and Ortner, R. (2008). Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21.

Azar, M. G., Osband, I., and Munos, R. (2017). Minimax regret bounds for reinforcement learning. In *International conference on machine learning*, pages 263–272. PMLR.

Baykal-Gürsoy, M. (2010). Semi-markov decision processes. *Wiley Encyclopedia of Operations Research and Management Science*.

Cinlar, E. (2013). *Introduction to stochastic processes*. Courier Corporation.

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

Dann, C., Lattimore, T., and Brunskill, E. (2017). Unifying pac and regret: Uniform pac bounds for episodic reinforcement learning. *Advances in Neural Information Processing Systems*, 30.

Drappo, G., Metelli, A., and Restelli, M. (2024). A provably efficient option-based algorithm for both high-level and low-level learning. In *Reinforcement Learning Conference*, pages 1–21.

Drappo, G., Metelli, A. M., and Restelli, M. (2023). An option-dependent analysis of regret minimization algorithms in finite-horizon semi-MDP. *Transactions on Machine Learning Research*.

Fruit, R. and Lazaric, A. (2017). Exploration-exploitation in mdps with options. In *Artificial Intelligence and Statistics*, pages 576–584. PMLR.

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

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

Lattimore, T. and Szepesvári, C. (2020). *Bandit algorithms*. Cambridge University Press.

Mülling, K., Kober, J., Kroemer, O., and Peters, J. (2013). Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279.

Osband, I. and Van Roy, B. (2016). On lower bounds for regret in reinforcement learning. *arXiv preprint arXiv:1608.02732*.

Pateria, S., Subagdja, B., Tan, A.-h., and Quek, C. (2021). Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35.

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

Qureshi, A. H., Simeonov, A., Bency, M. J., and Yip, M. C. (2019). Motion planning networks. In *2019 International Conference on Robotics and Automation*, pages 2118–2124. IEEE.

Robert, A., Pike-Burke, C., and Faisal, A. A. (2024). Sample complexity of goal-conditioned hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

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

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

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes**. The mathematical setting is described in Section 3, and Alg. 1 details the considered algorithm.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes**. We provide an upper bound on the regret suffered by the proposed algorithm.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **Yes**. It is part of the supplementary material.
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. **Yes**. All assumptions are listed at the end of Section 3.
   (b) Complete proofs of all theoretical results. **Yes**. Proofs are included in the Appendix.
   (c) Clear explanations of any assumptions. **Yes**
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **Yes**. It is part of the supplementary material.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **Yes**. See Section 6.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **Yes**. See caption of Fig. 2.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **Yes**.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. **Not Applicable**.
   (b) The license information of the assets, if applicable. **Not Applicable**.
   (c) New assets either in the supplemental material or as a URL, if applicable. **Not Applicable**.
   (d) Information about consent from data providers/curators. **Not Applicable**.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **Not Applicable**.
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. **Not Applicable**
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **Not Applicable**
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **Not Applicable**

A NOTATION

Since the paper contains a lot of notations, we provide a table with a summary of them here.

	Original MDP	Low-level MDP $M_{l,i}$	High-level MDP M_h
State space	$\mathcal{S}_o$	$\mathcal{S}_{l,i}$	$\mathcal{S}_h$
Sub-goal states	$\emptyset$	$\{\mathcal{E}_i\}_{i=1}^L$	$\emptyset$
Action space	$\mathcal{A}_o$	$\mathcal{A}_o$	$\mathcal{A}_h = \cup_{i=1}^L \mathcal{E}_i$
Transition	P_o	$P_{l,i}$	P_h
Reward	R_o	R_l	R_h
Horizon	$H_o = H_l * H_h$	H_l	H_h
Initial state	$s_{o,1}$	$s_{l,1}$	$s_{h,1}$
Timestep	t_o	t_l	t_h
Value function	$V_{o,t_o}^{\pi}(s)$	$V_{l,i,t_l}^{\nu}(s)$	$V_{h,t_h}^{(\nu,\mu)}(s)$
Optimistic value	$\tilde{V}_{o,t_o}^{\pi_k}(s)$	$\tilde{V}_{l,i,t_l}^{\nu_k}(s)$	$\tilde{V}_{h,t_h}^{(\nu,\mu)_k}(s)$
Optimistic Q-function	$\tilde{Q}_{o,t_o}^{\pi_k}(s,a)$	$\tilde{Q}_{l,i,t_l}^{\nu_k}(s,a)$	$\tilde{Q}_{h,t_h}^{(\nu,\mu)_k}(s,e)$
Policy (optimal)	π (π^*)	ν (ν^*)	μ (μ^*)

B RELATED WORK

In recent years, the Hierarchical Reinforcement Learning literature has focused on empirically demonstrating that HRL methods can be more efficient than standard ones in particular problems and applications Mülling et al. (2013); Andrychowicz et al. (2017); Qureshi et al. (2019).

Most existing theoretical work focuses on a related but different framework, the *options* framework. Options correspond to the low-level policies and are generally *pre-trained*; as such, most studies in the options framework ignore the cost of learning the options. Fruit and Lazaric (2017) and Fruit et al. (2017) studied the impact of *pre-trained options* (i.e. low-level policies) in Average Reward MDPs (Puterman, 2014), showing a benefit in term of more efficient exploration. Fruit and Lazaric (2017) proposed an algorithm based on UCRL2 (Auer et al., 2008) for Semi-Markov Decision Processes (Baykal-Gürsoy, 2010; Cinlar, 2013), and compared its upper bound to the regret obtained with UCRL2 with a monolithic approach. Fruit et al. (2017) remove the assumption of having particular knowledge of the distributions of cumulative reward and duration of each option. In our setting, while we assume that the low-level episode has a fixed and known length and the low-level reward to be sparse, we consider the much more challenging setting where the high-level policy and the low-level policy (or options) are jointly learned.

There exist several interesting expectations in the literature that consider jointly learning the options and the policy over options. In particular, Wen et al. (2020) highlights how patterns and substructures within MDPs can be leveraged for both planning and learning. In particular, they propose a posterior sampling-based algorithm that exploits these structural properties. Their analysis shows how similarities between substructures, as well as the complexity of sub-problems, influence the regret of their algorithm. However, they study Bayesian regret, which offers weaker guarantees than frequentist regret. Their algorithm relies on an additional reward structure, placed at sub-goals and termed the *exit profile*. The *exit profiles* are then used to inform the current options about the relative value of the goal that can be reached. This reward, contrary to the artificial reward that we add to the low-level MDP, influenced the option's policy learning. The low-level policy is conditioned on the sub-goal. Which sub-goal to prefer is chosen by the high-level policy. On the other hand, they have an option for sub-MDP, defined by a single policy not parametric to the sub-goal; hence, the decision over which exit or sub-goal to prioritise is conditioned by the exit profile – the higher the reward on a sub-goal, the more likely the option in that MDP would lead the agent to it. Consequently, for effective low-level learning, they required strong guarantees on the performance induced by the exit profiles, resulting in a rather strong assumption. Apart from this assumption, their regret bound shows similar dependencies on the hierarchy characteristics:

$$\text{BayesRegret}_{PSHRL}^{h}(M_{l\times h}, K) \leq \Delta A_h T + \tilde{O}(H_o^{3/2} \max_i \{S_{l,i}^{+}\} \sqrt{L A_o K}).$$

The first linear component corresponds to the sub-optimality gap induced by the structure, Δ_{struct}. They also show dependencies on $\sqrt{A_o K}$, while their result is sub-optimal in terms of $H_o^{3/2} \max_i \{S_{l,i}^{+}\}$. Note that a factor $\sqrt{H_o}$ is associated with the density of the reward, and in the case of sparse reward, it would be 1. However, it is worth mentioning that their algorithm is not affected by the optimal low-level policy performance and by $\sqrt{A_h}$, the number of possible sub-goals, but by $\sqrt{L}$, the number of distinct MDPs, which can be smaller than $\sqrt{A_h}$.

Drappo et al. (2023) and later Drappo et al. (2024) tried to develop approaches that jointly learn the low-level policy (*options*) and the high-level policy. The main challenge when jointly learning both levels of the hierarchy is the non-stationarity of the high-level dynamics. The high-level dynamics depend on the behaviour of the low-level policy, which changes over time as the agent learns it. Drappo et al. (2023) proposed a naive approach to mitigate this problem. Their 'explore-then-commit' algorithm first learns each sub-task policy and then exploits them to learn a higher-level policy in the same fashion as Fruit and Lazaric (2017). However, it is well known that Explore-then-Commit approaches suffer sub-optimal regret Lattimore and Szepesvári (2020). Therefore, the later work (Drappo et al., 2024) provides an alternative and more refined solution, which, instead of separating the learning process into two phases, divides it into multiple phases, and then the learning is alternated between the high-level problem of learning the meta-policy selecting among options, and the low-level problem consisting of learning the options' policies. This leads to an improved upper bound on the regret R^h:

$$R^h_{HLML}(M_o, \mathcal{O}, K) \leq \tilde{O}\Bigg(C^L \underbrace{H_o\sqrt{S_oOKd}}_{\text{High-Level Regret}} + C^H \underbrace{H_l\sqrt{OS_oA_oKH_o}}_{\text{Low-Level Regret}}\Bigg),$$

where $\mathcal{O}$ is the set of options, with $|\mathcal{O}| = O$, and d represents the average number of options played per episode so that $d \approx H_h$. Compared to our result, this upper bound clearly highlights the regret associated with high-level learning, which does not appear in our case because we are able to exploit the map ψ without needing to learn the high-level dynamics. Apart from this, the low-level regret is similar to ours: O is comparable to our A_h, the additional $\sqrt{H_o}$ is due to the time-inhomogeneous transition model, and $\sqrt{H_l}$ serves as an upper bound on $\|V\|$ when we consider a more general reward function (i.e. each reward is in $[0, 1]$). However, the biggest drawback we avoid is that this algorithm suffers from the influence of the constants C^H and C^L, which depend on how well each learned policy covers the optimal one during the learning process. In this work, we focus on a more specific family of problems, goal-based with sparse reward, to show that when there is this particular hierarchical structure, the regret only depends on the decomposition characteristics and has improved performance up to the failing probability of the optimal low-level policy.[5]

The lower bound on the sample efficiency of goal-based HRL provided by Robert et al. (2024) is not directly comparable to this analysis. First, it is a lower bound on the sample complexity, not the regret. More importantly, it considers a slightly different hierarchical decomposition. In particular, in Robert et al. (2024), the structure map ψ is not known, and the reward function is more general than the sparse reward considered here. Consequently, the lower bound presented in Robert et al. (2024) considers the case where both levels of the hierarchy must be learned. Interestingly, the lower bound of Robert et al. (2024) highlights that jointly learning both levels of the hierarchy is at least as difficult as the most difficult of the two problems. Since, in our setting, we consider we can compute the high-level dynamics (i.e. we do not need to learn them), the most complex part of our problem is necessarily learning the low-level dynamics.

C USEFUL DEFINITIONS

First of all, let's define the high-level plan. At the beginning of each episode, k, the high-level agent computes the exact trajectory it would like to complete. We call such a trajectory the high-level plan $Plan_k$.

Definition 3. *A plan, $Plan_k$, is the sequence of high-level states the high-level policy μ_k would like to follow in episode k, in accordance with the map ψ. It consists of the pairs of high-level states encoding the expected trajectory:*

$$[s_1, s_2, s_3, \cdots s_{n-1}, s_n] = Plan_k \tag{12}$$

describes a plan consisting of n high-level states, where $s_i \in \mathcal{S}_h$ for all $i \in [n]$, $s_{i+1} = \psi(s_i, \mu_k(s_i))$, and $n \leq H_h$. Note that given all the samples collected up to episode k and the current low-level policy ν_k, the plan, $Plan_k$, is deterministic.

[5] Note that both algorithms suffer from the sub-optimality gap induced by their respective structures – ours is defined by the two MDPs, and theirs by the options. However, the structure of the options allows for greater flexibility, as the options are less constraining.

We now state a proposition that helps to relate the regret of the low-level learning problem with the regret of the high-level learning problem.

The low-level MDP reward is a denser version of the original reward. It is enriched with a value of 1 given when the agent reaches the sub-goal state. This reward model allows us to state the following definition:

Definition 4. *Consider the low-level MDP $\mathcal{M}_{l,i}$ with a reward that returns one when the agent reaches the sub-goal state. For this reward model, the low-level value function $V^{\nu_k}_{l,i,1}(s,e)$ of the policy ν_k associated with the sub-goal $e \in \mathcal{E}_i$ and state $s \in \mathcal{S}_{l,i}$, corresponds to the probability of successful completion of the sub-goal (see definition 1):*

$$\mathbb{E}\left[\mathbb{I}\{end_{\nu_k}(i,e,s)\}\right] = \mathbb{P}(end_{\nu_k}(i,e,s)) = V^{\nu_k}_{l,i,1}(s,e) \quad \forall s \in \mathcal{S}_{l,i}. \tag{13}$$

This definition is valid for any low-level policy and, in particular, for the optimal low-level policy ν^*:

$$\mathbb{E}\left[\mathbb{I}\{\mathrm{end}_{\nu^*}(i,e,s)\}\right] = \mathbb{P}(\mathrm{end}_{\nu^*}(i,e,s)) = V^{\nu^*}_{l,i,1}(s,e) \quad \forall s \in \mathcal{S}_{l,i}. \tag{14}$$

D OPTIMISM

This section presents the high-level optimism necessary to prove Theorem 4.1. In particular, this section shows that the low-level regret minimiser propagates its optimism to the high level.

Since our algorithm only considers trajectories that follow the instructed high-level plan and stops collecting samples after the first missed sub-goal, it has an inherent sub-optimality gap as the optimal hierarchical policy (ν^*, μ^*) does not have this limitation. The following lemma guarantees that the plan computed by the high-level policy is optimistic up to this specific sub-optimality gap; that is, the value of the plan in the optimistic MDP plus the sub-optimality gap is greater or equal to the value of the optimal policy pair in the true high-level MDP.

We measure the value associated with the optimistic plan using the following value function:

$$\tilde{V}^{(\nu,\mu)}_{h,t_h}(s) = R_h(s,t_h) + \sum_{s' \in \mathcal{S}_h} \tilde{P}^{\nu}_h(s'|s,\mu(s))\tilde{V}^{(\nu,\mu)}_{h,t_h+1}(s')$$

for all $s \in \mathcal{S}_h$, any $t_h \in [H_h - 1]$ and with $\tilde{V}^{(\nu,\mu)}_{h,H_h}(s) = R_h(s,H_h)$ for all $s \in \mathcal{S}_h$. The approximation of the high-level transition function $\tilde{P}_h$ is defined in Eq. 6. Because $\tilde{P}_h$ is only an approximation of a transition function, it only models the probability of reaching the next intended state; all other transitions have no probability mass. It has a sub-optimality gap, upper bounded by the probability that the optimal policy pair fails each sub-goal of the current plan,

$$\Delta_{t_h} \leq \sum_{s \in \mathrm{Plan}[t_h:H_h]} \mathbb{P}(\neg \mathrm{end}_{\nu^*}(\phi(s),\mu(s),s_{l,t_h})). \tag{15}$$

where s_{l,t_h} is the first low-level state visited in the low-level episode corresponding to the stage t_h of $Plan_k$.

Lemma D.1. *Let (ν^*, μ^*) be the optimal policies on $M_{l\times h}$, and $(\nu,\mu)_k$ be the policy pair compute by Alg. 1 at the beginning of any episode k. Given optimistic low-level value functions, we guarantee that the high-level value function constructed in our algorithm and the sub-optimality gap upper bound the value function of the optimal hierarchical policy:*

$$\tilde{V}^{(\nu,\mu)_k}_{h,t_h}(s) + \sum_{s' \in Plan_k[t_h:H_h]} \mathbb{P}(\neg end_{\nu^*}(\phi(s'),\mu_k(s'),s_{l,t_h})) \geq V^{(\nu^*,\mu^*)}_{h,t_h}(s) \quad \textit{with } s = Plan_k[t_h]. \tag{16}$$

where $\mathbb{P}(\neg end_{\nu^*}(\phi(s'),\mu_k(s'),s_l))$ is the probability for the optimal low-level policy to fail to reach sub-goal $e \in \mathcal{E}_{\phi(s)}$ while starting a low-level episode in MDP $i = phi(s)$ in the initial low-level state $s_l \in \mathcal{S}_{l,i}$ (see Eq. 14).

Note that the sub-optimality gap consists of the failure probability only for the remaining states in the plan, Plan_k. We make this explicit using the following notation: at time step t_h, we denote the remaining part of the plan, $Plan_k$, using vector notation $Plan_k[t_h : H_h]$.

Proof. We need the following observations on the construction of the optimistic high-level MDP.
First, we note that the regret minimiser $\mathfrak{A}$ used by $\mathrm{GHRL}_{\mathfrak{A}}$ computes at every episode k an optimistic value

function for every low-level MDP $M_{l,i}$ and sub-goal $e \in \mathcal{E}_i$ pair.
Second, we notice that in this goal-conditioned setting with sparse reward, the low-level value function $V^{\nu}_{l,i,1}(s,e)$ of a policy ν for any state $s \in \mathcal{S}_{l,i}$ and sub-goal $e \in \mathcal{E}_i$ is equal to the probability that the policy ν reaches the goal e while starting the low-level episode in state s, $V^{\nu}_{l,i,1}(s,e) = \mathbb{P}(\mathrm{end}_{\nu}(i,e,s))$, as explained in Definition 4. From the high-level perspective, this corresponds to the probability under the low-level policy ν to transition to the instructed next high-level state $\psi(s,e)$, that is, $P^{\nu}_{h}(\psi(s,e)|s,e) = V^{\nu}_{l,\phi(s),1}(s,e)$. Combining these observations with the map ψ, we obtain an approximate optimistic high-level transition function $\tilde{P}^{\nu}_{h}$ presented in Eq. (6).

Now, let's define the optimistic high-level Q-function as the expected return the pair of policies $(\nu,\mu)_k$ can collect after executing sub-goal e from high-level state s,

$$\tilde{Q}^{(\nu,\mu)_k}_{h,t_h}(s,e) = R_h(s,t_h) + \sum_{s' \in \mathcal{S}_h} \tilde{P}^{\nu_k}_{h}(s'|s,e)\tilde{V}^{(\nu,\mu)_k}_{h,t_h}(s'). \tag{17}$$

We now proceed with a proof by induction. We note that Lemma D.1 trivially holds for the last step of the high-level MDP H_h since we reached the end of the episode and the agent does not have the opportunity to fail a sub-goal the sub-optimality gap $\Delta_{H_h} = 0$. Hence the optimistic high-level value function and the optimal value function at time step H_h are equal,

$$\tilde{V}^{(\nu,\mu)_k}_{h,H_h}(s) = V^{(\nu^*,\mu^*)}_{h,H_h}(s) \quad \forall s \in \mathcal{S}_h.$$

Let's now consider any step $t_h < H_h$ and prove that the lemma still holds. We use the inductive hypothesis that the lemma holds for the step $t_h + 1$, that is,

$$\tilde{V}^{(\nu,\mu)_k}_{h,t_h+1}(s) + \sum_{s' \in Plan_k[t_h+1:H_h]} \mathbb{P}(\neg\mathrm{end}_{\nu^*}(\phi(s'),\mu(s'),s'_{l,t_h})) \geq V^{(\nu^*,\mu^*)}_{h,t_h+1}(s). \tag{18}$$

with $s \in \mathcal{S}_h$ being the high-level state in step t_h+1 of $Plan_k$.

We now show that Eq. (18) holds for time step t_h for an arbitrary state $s \in \mathcal{S}_h$ and sub-goal $e \in \mathcal{E}_{\phi(s)}$. Note that the plan, $Plan_k$, computed by μ_k from state $\psi(s,e)$ is the sequence of high-level states with the largest probability to reach the goal state, according to the high-level transition $\tilde{P}^{\nu_k}_{h}$.

In what follows, we denote the instructed next high-level state with $s_\psi = \psi(s,e)$ to lighten the notation.

$$\begin{aligned}
&\tilde{Q}^{(\nu,\mu)_k}_{h,t_h}(s,e) - Q^{(\nu^*,\mu^*)}_{h,t_h}(s,e) && (19)\\
&\quad = R_h(s,t_h) - R_h(s,t_h) + \sum_{s' \in \mathcal{S}_h}\Big(\tilde{P}^{\nu_k}_{h}(s'|s,e)\tilde{V}^{(\nu,\mu)_k}_{h,t_h+1}(s') - P^{*}_{h}(s'|s,e)V^{(\nu^*,\mu^*)}_{h,t_h+1}(s')\Big) && (20)\\
&\quad = \sum_{s' \in \mathcal{S}_h}\Big(\tilde{P}^{\nu_k}_{h}(s'|s,e)\tilde{V}^{(\nu,\mu)_k}_{h,t_h+1}(s') - P^{*}_{h}(s'|s,e)V^{(\nu^*,\mu^*)}_{h,t_h+1}(s')\Big) && (21)\\
&\quad = \tilde{P}^{\nu_k}_{h}(s_\psi|s,e)\tilde{V}^{(\nu,\mu)_k}_{h,t_h+1}(s_\psi) - P^{*}_{h}(s_\psi|s,e)V^{(\nu^*,\mu^*)}_{h,t_h+1}(s_\psi)\\
&\qquad + \sum_{s' \in \mathcal{S}_h\setminus\{s_\psi\}}\Big(\tilde{P}^{\nu_k}_{h}(s'|s,e)\tilde{V}^{(\nu,\mu)_k}_{h,t_h+1}(s') - P^{*}_{h}(s'|s,e)V^{(\nu^*,\mu^*)}_{h,t_h+1}(s')\Big) && (22)\\
&\quad = \tilde{P}^{\nu_k}_{h}(s_\psi|s,e)\tilde{V}^{(\nu,\mu)_k}_{h,t_h+1}(s_\psi) - P^{*}_{h}(s_\psi|s,e)V^{(\nu^*,\mu^*)}_{h,t_h+1}(s_\psi) - \sum_{s' \in \mathcal{S}_h\setminus\{s_\psi\}} P^{*}_{h}(s'|s,e)V^{(\nu^*,\mu^*)}_{h,t_h+1}(s') && (23)\\
&\quad \geq \tilde{P}^{\nu_k}_{h}(s_\psi|s,e)\tilde{V}^{(\nu,\mu)_k}_{h,t_h+1}(s_\psi) - P^{*}_{h}(s_\psi|s,e)V^{(\nu^*,\mu^*)}_{h,t_h+1}(s_\psi) - \mathbb{P}(\neg\mathrm{end}_{\nu^*}(\phi(s),e,s_{l,t_h})) && (24)\\
&\quad \geq \big(\tilde{P}^{\nu_k}_{h} - P^{*}_{h}\big)(s_\psi|s,e)V^{(\nu^*,\mu^*)}_{h,t_h+1}(s_\psi) - \mathbb{P}(\neg\mathrm{end}_{\nu^*}(\phi(s),e,s_{l,t_h}))\\
&\qquad - \tilde{P}^{\nu_k}_{h}(s_\psi|s,e)\sum_{s' \in Plan_k[t_h+1:H_h]} \mathbb{P}(\neg\mathrm{end}_{\nu^*}(\phi(s'),\mu_k(s'),s'_{l,t_h+1})) && (25)\\
&\quad \geq \big(\tilde{P}^{\nu_k}_{h} - P^{*}_{h}\big)(s_\psi|s,e)V^{(\nu^*,\mu^*)}_{h,t_h+1}(s_\psi) - \sum_{s' \in Plan_k[t_h:H_h]} \mathbb{P}(\neg\mathrm{end}_{\nu^*}(\phi(s'),\mu_k(s'),s'_{l,t_h})) && (26)
\end{aligned}$$

$$= \big(\tilde{V}^{\nu_k}_{l,\phi(s),1}(s_\psi) - V^*_{l,\phi(s),1}(s_\psi)\big)V^{(\nu^*,\mu^*)}_{h,t_h+1}(s_\psi) - \sum_{s' \in Plan_k[t_h:H_h]} \mathbb{P}(\neg\text{end}_{\nu^*}(\phi(s'), \mu_k(s'), s'_{l,t_h})) \tag{27}$$

$$\geq - \sum_{s' \in Plan_k[t_h:H_h]} \mathbb{P}(\neg\text{end}_{\nu^*}(\phi(s'), \mu_k(s'), s'_{l,t_h})) \tag{28}$$

Eq. (20) is simply the definition of the optimistic Q-function and the optimal Q-function. Eq. (21) follows from the fact that the two high-level rewards are identical. Eq. 22 divides the summation over all possible next states in two; the first term considers the next state defined by the map $\psi(s,e)$, $s_\psi = \psi(s,e)$, and the other term considers all the other possible next states. Eq. 23 holds since $\tilde{P}^{\nu_k}_h$ is zero for all the potential next high-level states not supported by the map $s' \neq \psi(s,e)$ (as defined in Eq. 6). Eq. 24 uses the fact the $V^{(\nu^*,\mu^*)}_{h,t_h+1}(s)$ can be upper bounded by one for all $s \in \mathcal{S}_h$ and the sum of the probabilities to reach any other states than s_ψ, is the failure event $\mathbb{P}(\neg\text{end}_{\nu^*}(\phi(s), e, s_{l,t_h}))$. Eq. (25) follows from the induction hypothesis defined in Eq. 18, note that the induction hypothesis is applied to $\tilde{P}^{\nu_k}_h(s_\psi|s,e)\tilde{V}^{(\nu,\mu)_k}_{h,t_h+1}(s_\psi)$. Eq. (26) upper bounds $\tilde{P}^{\nu_k}_h(s_\psi|s,e)$ with 1. As the failure event $\mathbb{P}(\neg\text{end}_{\nu^*}(\phi(s), e, s_{l,t_h}))$ consists of failing the t_h^{th} step of the plan, $s = Plan_k[t_h]$, started the low-level episode in the corresponding low-level state s_l, and $\psi(s,e) = Plan_k[t_h+1]$. We then extend the sum over the sub-plan $Plan_k[t_h : H_h]$ to account for this additional failure event. Eq. (27) follows directly from the equivalence between the high-level transition function and the low-level value function (see Eq. 6). Finally, Eq. (28) is obtained by the optimistic guarantee of the value function computed by the low-level algorithm $\mathfrak{A} \in \Xi$ and because the optimal value function is necessarily positive, $V^{(\nu^*,\mu^*)} \geq 0$. □

E PROOF OF THEOREM 4.1 AND COROLLARY 4.2

We first prove Theorem 4.1 that we restate below for clarity.

Theorem 4.1. *Given $\delta \in [0,1]$ and $\beta = \frac{\delta}{KA_h}$, for any hierarchical decomposition $M_{l\times h}$, and any $(1-\beta)$-optimistic low-level regret minimiser $\mathfrak{B} \in \Xi$, the regret of GHRL$_\mathfrak{B}$ is bounded, with probability $1-\delta$, by:*

$$R^h_{GHRL}(M_{l\times h}, K) \leq \sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} \Big(R^l_{\mathfrak{B}}\big((M_{l,i}, e), N(i,e)\big) + 2N(i,e) \max_{s_l \in \mathcal{S}_{l,i,1}} \mathbb{P}(\neg end_{\nu^*}(i,e,s))\Big) \tag{7}$$

where $N(i,e)$, $\forall i \in [L]$ and $e \in \mathcal{E}_i$, is the number of times in which sub-goal e has been chosen in the state s_h, with $\phi(s_h) = i$ and $e = \mu_k(s_h)$. $R^l_{\mathfrak{B}}\big((M_{l,i}, e), N(i,e)\big)$ is the regret suffered by $\mathfrak{B}$ while learning to solve goal e in low-level MDP $M_{l,i}$ for $N(i,e)$ low-level episodes. $\mathcal{S}_{l,i,1}$ is the set of potential initial states in the MDP $M_{l,i}$, $\mathcal{S}_{l,i,1} = \bigcup_{e\in\mathcal{A}_h} \{s_1 = \rho_{l,i}(e) : s_1 \in \mathcal{S}_{l,i}\}$.

Proof. The total regret suffered consists of the regret suffered while learning policies to solve every sub-goal of each low-level MDP. Let's first recall how the GHRL (Alg. 1) works. In every episode k, GHRL updates the low-level policy with an optimistic regret minimiser $\mathfrak{B}$ using the samples collected in the low-level MDPs. Note that the collected samples are modified with an artificial reward, i.e. the low-level reward function R_l, which gives an additional reward of 1 when the agent fulfils the sub-goal. Because the low-level reward function is sparse and only rewards the agent when it completes the low-level episode in the sub-goal state, the low-level optimistic value function (computed by $\mathfrak{B}$) can be used to approximate the transition function of the high-level MDP $\tilde{P}^{\nu_k}_h$ (defined in Eq. 6). Equipped with this approximation of the high-level transition function and the known high-level reward function, we compute the high-level policy, μ_k, using backward induction in the high-level MDP. The algorithm collects new samples using ν_k to interact with the environment when trying to complete sub-goals selected by μ_k. The sequence of sub-goals considered for a given episode k is computed by the current high-level policy μ_k at the beginning of episode k. The high-level trajectory induced by this sequence of sub-goals is stored in the high-level trajectory Plan$_k$. Note that each time the low-level policy fails to achieve its sub-goal, the algorithm stops collecting samples until the end of the episode.

The regret of the algorithm GHRL, on the hierarchical structure $M_{l\times h} = (\{M_{l,i}\}_{i\in[L]}, M_h)$, is defined as the sum over K episodes of differences between the value function of the optimal hierarchical policy (ν^*, μ^*) and the learned policies at time step k, $(\nu,\mu)_k$, evaluated at the first state in the episode $s_1 \in \mathcal{S}_h$:

$$R^h_{\text{GHRL}}(M_{l\times h},K) = \sum_{k=1}^{K} V^{(\nu^*,\mu^*)}_{h,1}(s_1) - V^{(\nu,\mu)_k}_{h,1}(s_1) \tag{29}$$

$$\leq \underbrace{\sum_{k=1}^{K} \tilde{V}^{(\nu,\mu)_k}_{h,1}(s_1) - V^{(\nu,\mu)_k}_{h,1}(s_1)}_{(a)} + \underbrace{\sum_{k=1}^{K}\sum_{s\in Plan_k} \mathbb{P}(\neg\text{end}_{\nu^*}(s,\mu_k(s),s_l))}_{(b)} \tag{30}$$

Eq. (30) is obtained through the optimism guarantee of Lemma D.1 and decomposes the regret into two terms (a) and (b).

It is important to observe that the above theorem only holds if the optimism holds for every low-level MDP, sub-goal pair (i,e), and episode k, which is formally defined as the following event:

$$A = \cap_{k=1}^{K} \cap_{i=1}^{L} \cap_{e\in\mathcal{E}_i} \{\tilde{V}^{\nu_k}_{l,i,1}(s,e) \geq V^*_{l,i,1}(s,e)\} \tag{31}$$

where the low-level optimism, i.e. $\{\tilde{V}^{\nu_k}_{l,i,1}(s,e) \geq V^*_{l,i,1}(s,e)\}$, is guarantee to hold with probability $1-\beta$ for all $i\in[L]$, $e\in\mathcal{E}_i$ and $s=\mathcal{S}_{l,i}$.

Theorem 4.1 requires that the event A holds with high probability, $\mathbb{P}(A) > 1-\delta$. To ensure this, we need to set the low-level optimism failure probability to $\beta = \frac{\delta}{A_h K}$. This can be obtained by observing that we need to guarantee $\mathbb{P}(A^c) \leq \delta$. Where A^c is the complement of A:

$$A^c = \cup_{k=1}^{K} \cup_{i=1}^{L} \cup_{e\in\mathcal{E}_i} \{\tilde{V}^{\nu_k}_{l,i,1}(s,e) < V^*_{l,i,1}(s,e)\}. \tag{32}$$

The probability $\mathbb{P}(A^c)$ consists of the union of independent failure of low-level optimism guarantee for every episode $k\in[K]$, low-level MDP $i\in[L]$, and sub-goal $e\in\mathcal{E}_i$. Since each optimism guarantee is independent, $\mathbb{P}(A^c)$ can be written as the sum over all the failure probabilities, which gives us the following upper bound on *delta*:

$$\delta \leq \sum_{k=1}^{K}\sum_{i=1}^{L}\sum_{e\in\mathcal{E}_i} \beta = KA_h\beta \tag{33}$$

where we recall that $\sum_{i=1}^{L}\sum_{e\in\mathcal{E}_i} 1 = A_h$. This implies that $\beta = \frac{\delta}{KA_h}$

We first analyse the term (a) in Eq. (30):

$$(a) \leq \sum_{k=1}^{K}\sum_{j=1}^{H_h}\left[(R_h(s^k_j,j) - R_h(s^k_j,j)) + \sum_{s'\in\mathcal{S}_h}\left(\tilde{P}^{\nu_k}_h(s'|s^k_j,\mu_k(s^k_j)) - P^{\nu_k}_h(s'|s^k_j,\mu_k(s^k_j))\right)\tilde{V}^{(\nu,\mu)_k}_{h,j+1}(s')\right] \tag{34}$$

$$\leq \sum_{k=1}^{K}\sum_{s^k_j\in Plan_k}\left[\left(\tilde{P}^{\nu_k}_h(s^k_{j+1}|s^k_j,\mu_k(s^k_j)) - P^{\nu_k}_h(s^k_{j+1}|s^k_j,\mu_k(s^k_j))\right)\tilde{V}^{(\nu,\mu)_k}_{h,j+1}(s^k_{j+1})\right] \tag{35}$$

$$\leq \sum_{k=1}^{K}\sum_{s^k_j\in Plan_k}\left[\left(\tilde{P}^{\nu_k}_h(s^k_{j+1}|s^k_j,\mu_k(s^k_j)) - P^{\nu_k}_h(s^k_{j+1}|s^k_j,\mu_k(s^k_j))\right)\right] \tag{36}$$

$$\leq \sum_{k=1}^{K}\sum_{s^k_j\in Plan_k}\left(1 - P^{\nu_k}_h(s^k_{j+1}|s^k_j,\mu_k(s^k_j))\right) \tag{37}$$

$$= \sum_{k=1}^{K}\sum_{s^k_j\in Plan_k}\mathbb{P}(\neg\text{end}_{\nu_k}(\phi(s^k_j),\mu_k(s^k_j),s^k_l)) \quad \text{with } s^k_l = \rho_{l,\phi(s^k_{j-1})}(\mu_k(s^k_{j-1})) \tag{38}$$

$$= \sum_{i=1}^{L}\sum_{e\in\mathcal{E}_i}\sum_{n=1}^{N(i,e)}\mathbb{P}(\neg\text{end}_{\nu_n}(i,e,s_n)) \tag{39}$$

Eq. 34 uses the performance difference lemma (see Lemmma E.15 Dann et al. (2017)). Then, Eq. 35 reformulates the regret as the difference in performance following in each episode a predefined high-level trajectory. These trajectories correspond to the high-level plan, $Plan_k$, which is deterministically constructed with the policy μ_k and the map ψ (see Definition 3). The state s^k_{j+1} corresponds to the next state considered in the plan, $Plan_k$. Then considering only the steps anticipated by $Plan_k$ is possible since the optimistic high-level transition 0 for transitions to a high-level state do not correspond to the instructed subgoal, that is, $\tilde{P}^{\nu_k}_h(s'|s,e) = 0$ for all $s' \neq \psi(s,e)$. We obtain Eq. (36) by upper bounding the optimistic value function $\tilde{V}^{(\nu,\mu)_k}_h$ with 1, this yields a valid upper bound as the optimism property of the low-level algorithm guarantees $\tilde{P}^{\nu_k}_h \geq P^{\nu_k}_h$ with high-probability. Then in Eq. (37) we upper bound the optimistic transition function $\tilde{P}^{\nu_k}_h$ with 1. Finally, we rewrite Eq. (37) in terms of the probability that the current low-level policy ν_k does not reach the instructed sub-goal $\mu_k(s_j)$ to obtain Eq. (38). Eq. (39) is obtained by taking the sum over $N(i,e)$ instead of K, with $N(i,e)$ being a counter of the number times in which the pair (i,e) has been visited, considering the counter to be updated until the execution follows the plan – when we discard samples we are not updating the counter $N(i,e)$. The state s_n is the low-level state from which the low-level episode played by the n^{th} policy started.

This allows us to analyse the pairs (i,e) separately, explicitly indicating the regret suffered in the single problem using the fact that the sum of failing probability can be related to the low-level regret.

To bound $\sum_{i=1}^{L}\sum_{e\in\mathcal{E}_i}\sum_{n=1}^{N(i,e)}\mathbb{P}(\neg\text{end}_{\nu_n}(i,e,s_n))$, we use the following relationship between the low-level regret of an optimistic algorithm and its failure probability. For any N number of steps, the regret suffered by any optimistic algorithm $\mathfrak{B} \in \Xi$ to learn an optimal policy to solve a sub-goal e in a low-level MDP $M_{l,i}$ is defined as:

$$\begin{aligned} R^l_{\mathfrak{B}}((M_{l,i},e),N) &= \sum_{n=1}^{N} V^*_{l,i,1}(s_n,e) - V^{\nu_n}_{l,i,1}(s_n,e) \\ &= \sum_{n=1}^{N} V^*_{l,i,1}(s_n,e) - 1 + 1 - V^{\nu_n}_{l,i,1}(s_n,e) \\ &= \sum_{n=1}^{N} \mathbb{P}(\text{end}_{\nu^*}(i,e,s_n)) - 1 + \mathbb{P}(\neg\text{end}_{\nu_n}(i,e,s_n)), \end{aligned} \tag{40}$$

where we use the notation $(M_{l,i},e)$ to explicitly mention that we compute the regret suffered on MDP $M_{l,i}$ while trying to solve the sub-goal e. The state s_n is the low-level state in which the low-level episode n started.

Eq. 40 directly relates the expected probability of failing of a policy ν_n to the regret paid to learn the sub-goal e. Rearranging, we obtain:

$$\sum_{n=1}^{N}\mathbb{P}(\neg\text{end}_{\nu_n}(i,e,s_n)) \leq R^l_{\mathfrak{B}}((M_{l,i},e),N) + N \max_{s\in\mathcal{S}_{l,i,1}}\big(\mathbb{P}(\neg\text{end}_{\nu^*}(i,e,s))\big). \tag{41}$$

where $\mathcal{S}_{l,i,1}$ is the set of potential initial states in the MDP $M_{l,i}$, $\mathcal{S}_{l,i,1} = \bigcup_{e\in\mathcal{A}_h}\{s_1 = \rho_{l,i}(e) : s_1 \in \mathcal{S}_{l,i}\}$. Hence, we use this in (39) to show that,

$$(a) \leq \sum_{i=1}^{L}\sum_{e\in\mathcal{E}_i}\sum_{n=1}^{N(i,e)}\mathbb{P}(\neg\text{end}_{\nu_n}(i,e,s_n)) \leq \sum_{i=1}^{L}\sum_{e\in\mathcal{E}_i}\Big(R^l_{\mathfrak{B}}((M_{l,i},e),N(i,e)) + N(i,e)\max_{s\in\mathcal{S}_{l,i,1}}\big(\mathbb{P}(\neg\text{end}_{\nu^*}(i,e,s_n))\big)\Big). \tag{42}$$

It is important to notice that the algorithm $\mathfrak{B}$ can be any optimistic algorithm in Ξ. Optimism is required with high probability (i.e. $1-\beta$) to guarantee that the high-level policy considers the uncertainty of the low-level transition model while planning; refer to section D for further details.

Let's now focus on analysing the term (b). With similar algebraic modifications, we can relate it to:

$$(b) = \sum_{k=1}^{K}\sum_{s\in Plan_k}\mathbb{P}(\neg\text{end}_{\nu^*}(s,\mu_k(s),s_l)) \tag{43}$$

$$= \sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} \sum_{n=1}^{N(i,e)} \mathbb{P}(\neg \mathrm{end}_{\nu^*}(i,e,s_n)) \tag{44}$$

$$\leq \sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} N(i,e) \max_{s \in \mathcal{S}_{l,i,1}} \big(\mathbb{P}(\neg \mathrm{end}_{\nu^*}(i,e,s))\big) \tag{45}$$

In Eq. 44, we sum over all possible low-level MDP, sub-goal pair (i, e) and the corresponding number of visits $N(i, e)$. As the probability that the optimal policy ν^* fails is independent of the number of visits, we directly obtain Eq. 45.

Then, by summing (a) and (b), we see that the regret can be upper bounded by:

$$R^h_{GHRL}(M_{l\times h}, K) \leq \sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} \Big(R^l_{\mathfrak{B}}((M_{l,i}, e), N(i,e)) + 2N(i,e) \max_{s_l \in \mathcal{S}_{l,i,1}} \mathbb{P}(\neg \mathrm{end}_{\nu^*}(i,e,s)) \Big). \tag{46}$$

which concludes the proof. □

We now focus on proving Corollary 4.2. For clarity, we first restate the result.

Corollary E.1. *Let $\delta \in [0, 1]$ and $\beta = \frac{\delta}{KA_h}$ be the failure probability of the low-level algorithm, UCBVI. For any hierarchical decomposition $M_{l\times h}$ the regret suffered by GHRL$_{UCBVI}$ for K high-level episodes is bounded, with probability $1 - \delta$, by:*

$$R^h_{GHRL}(M_{l\times h}, K) \leq \sqrt{A_o A_h \max_i \{S^+_{l,i}\} H_l K H_h} + \Delta \tag{8}$$

with

$$\Delta = \sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} 2N(i,e) \max_{s_l \in \mathcal{S}_{l,i,1}} \mathbb{P}(\neg end_{\nu^*}(i,e,s)).$$

Proof. Let's now consider the first term in Theorem 4.1 (i.e. Eq. (7)) with UCBVI as a low-level regret minimiser. The regret of UCBVI is upper bounded by $\tilde{O}(\sqrt{A_o H_l S_{l,i} N(i,e)})$ (Theorem 2, Azar et al., 2017) – note that the upper bound differs for a term H_l, for the sparse reward setting, the infinity norm of the value function is bounded by one and not by H_l, $\|V^{\nu_k}\|_\infty \leq 1$.

$$\begin{aligned}
\sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} R^l_{\mathrm{UCBVI}}((M_{l,i}, e), N(i,e)) &\leq \sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} \sqrt{A_o H_l S_{l,i} N(i,e)} \\
&\leq \sqrt{A_o H_l} \sqrt{\sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} 1} \sqrt{\sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} S_{l,i} N(i,e)} \qquad (47) \\
&\leq \sqrt{A_o A_h \max_i \{S_{l,i}\} H_l K H_h}. \qquad (48)
\end{aligned}$$

Eq. (47) follows by applying Cauchy Schwarz inequality, while the last inequality follows from:

$$\sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} S_{l,i} N(i,e) \leq \max_i \{S_{l,i}\} \sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} N(i,e) \leq \max_i \{S^+_{l,i}\} K H_h. \tag{49}$$

where $\mathcal{A}_h$ is the set of all possible sub-goals the high-level can instruct in all low-level MDPs, that is $\sum_{i=1}^{L} \sum_{e \in \mathcal{E}_i} 1 = |\mathcal{A}_h|$, and the last inequality in Eq.(49), follow from the fact that by discarding the samples and not considering them to update $N(i, e)$, in the episodes we could have visited less than H_h times one pair (i, e).

□

F ADDITIONAL EXPERIMENTS

In this section, we first illustrate the path-like environment used in Section 6. As shown in Fig. 3, the environment consists of a sequence of rooms. The agent starts in the leftmost room, and its goal is to reach the right room; the goal state corresponds to the first cell of the rightmost room. In the experiment conducted in Section 6 (Fig. 2, four leftmost plots), we consider paths of varying length, and on the rightmost plot of Fig. 2, we run the algorithm on the four corridor environments depicted in Fig. 3. From top to bottom, we add each time a different room profile, making it more challenging for the low-level, who needs to learn how to navigate in a new room. Then, in Fig. 4, we show that GHRL with Q-UCB to compute the low-level policy is outperforming Q-UCB on a slightly different maze task where, this time, the rooms are arranged on a grid instead of a path.

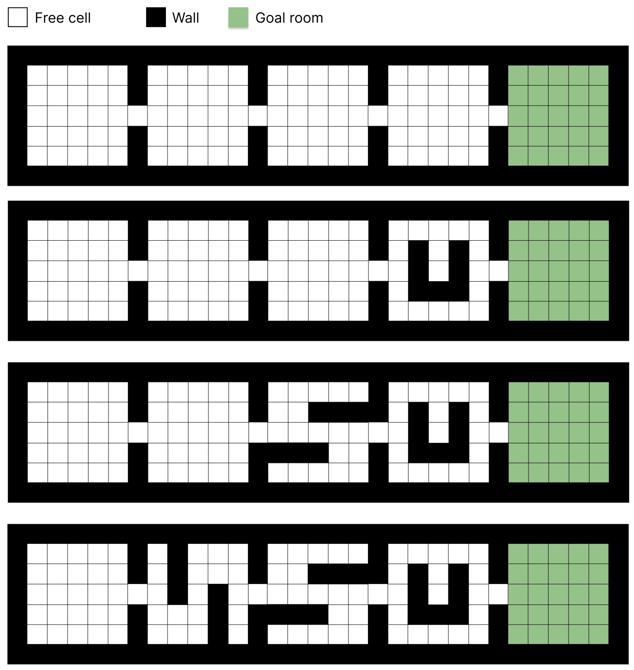

Figure 3: An illustration of the four path-like MDPs considered. All path MDPs shown in this figure are composed of five rooms. The plot at the top shows an MDP with identical rooms. In each row, we change the room profile of a room; that is, we add a unique obstacle inside the room. We repeat this until all rooms have a unique room profile, as shown in the bottom plot.

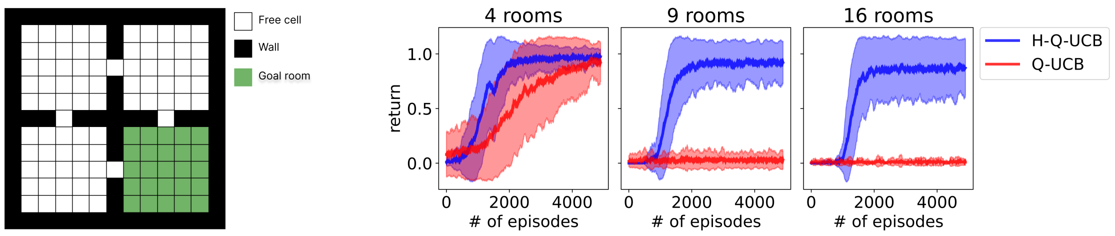

Figure 4: An illustration of the results obtained on the n-room environment. The leftmost plot shows an example of a maze composed of 4 rooms, while the plots on the right show the performance obtained by our framework (in blue) against Q-UCB (in red). As expected, as we increase the number of rooms, the efficiency gains of our method become more evident.

G BACKWARD INDUCTION

For completeness, we present the Backward Induction algorithm used to perform low-level planning in Alg. 1.

Algorithm 3 Planning with Backward Induction

```
 1: Input: $\{P_h\}_{h=1}^H$, $\{r_h\}_{h=1}^H$, $\mathcal{S}$, $\mathcal{A}$
 2: Output: $V$, $\pi$
 3: for $s \in \mathcal{S}$ do
 4:     $V_H(s) = \max_{a \in \mathcal{A}} r_H(s,a)$
 5: end for
 6: for $t \in \{H-1, \cdots, 1\}$ do
 7:     for $s \in \mathcal{S}$ do
 8:         $V_t(s) = \max_{a \in \mathcal{A}} r_t(s,a) + \sum_{s' \in \mathcal{S}} P_t(s'|s,a) V_{t+1}(s')$
 9:         $\pi(s,t) = \operatorname{argmax}_{a \in \mathcal{A}} r_t(s,a) + \sum_{s' \in \mathcal{S}} P_t(s'|s,a) V_{t+1}(s')$
10:     end for
11: end for
```