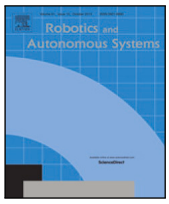
Robotics and Autonomous Systems 193 (2025) 105116

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Exploiting robot abstractions in episodic RL via reward shaping and heuristics

Roberto Cipollone [a], Marco Favorito [c], Flavio Maiorana [a], Giuseppe De Giacomo [a,b], Luca Iocchi [a], Fabio Patrizi [a,*]

[a] DIAG, Sapienza University of Rome, Italy
[b] Department of Computer Science, University of Oxford, UK
[c] Bank of Italy, Italy

ARTICLE INFO

Keywords:
Hierarchical reinforcement learning
Abstraction
Planning
Reward shaping
Exploration heuristics
Robotics

ABSTRACT

One major limitation to the applicability of Reinforcement Learning (RL) to many domains of practical relevance, in particular in robotic applications, is the large number of samples required to learn an optimal policy. To address this problem and improve learning efficiency, we consider a linear hierarchy of abstraction layers of the Markov Decision Process (MDP) underlying the target domain. Each layer is an MDP representing a coarser model of the one immediately below in the hierarchy. In this work, we propose novel techniques to automatically define Reward Shaping and Reward Heuristic functions that are based on the solution obtained at a higher level of abstraction and provide rewards to the finer (possibly the concrete) MDP at the lower level, thus inducing an exploration heuristic that can effectively guide the learning process in the more complex domain. In contrast with other works in Hierarchical RL, our technique imposes fewer requirements on the design of the abstract models and is tolerant to modeling errors, thus making the proposed approach practical. We formally analyze the relationship between the abstract models and the exploration heuristic induced in the lower-level domain, we prove that the method guarantees optimal convergence, and finally demonstrate its effectiveness experimentally in several complex robotic domains.

1. Introduction

In Reinforcement Learning (RL), agents have no complete model available to predict the outcomes of their actions. Since coming up with a complete and faithful model of the world is generally difficult, this allows for the wide applicability of RL algorithms. Nonetheless, such a lack of knowledge also demands a significant number of interactions with the environment, before a (near-)optimal policy can be estimated. As a result, most of the successes of RL come from the digital world (e.g., video games, simulated environments), especially those in which a large number of samples can be easily generated, while applications to real environments, such as robotic scenarios, are still rare.

Many RL tasks are *goal-oriented*, which implies that when the goal states are *sparse*, so are the rewards. This is a well-known challenging scenario for RL, which further increases the amount of required samples. Unfortunately, sparse goal states are very common, as they may arise in simple tasks, such as reaching specific configurations in large state spaces, and even in modest environments, for complex target behaviors, such as the successful completion of a sequence of smaller tasks [1,2]. In order to improve sample efficiency, *Hierarchical* RL approaches [3] have been proposed to decompose a complex problem into subtasks that are easier to solve individually. In this context, an *abstraction* is a simplified, coarser model that reflects the decomposition induced over the *ground*, more complex environment [4].

In this paper, we consider a linear hierarchy of abstractions of the Markov Decision Process (MDP) underlying the target domain. Each layer in the hierarchy is a simplified model, still represented as an MDP, of the one immediately below. A simple example is that of an agent moving in a map. The states of the ground MDP may capture the real pose of the agent, as continuous coordinates and orientation. The states of its abstraction, instead, may provide a coarser description, obtained by coordinate discretization, semantic map labeling (i.e., associating metric poses to semantic labels), or by projecting out state variables. Ultimately, such a compression corresponds to partitioning the ground state space into abstract states, implicitly defining a mapping from the former to the latter. The action spaces of the two models can also differ, in general, as including the actions that are best appropriate for each representation. Simulators are commonly used in RL and robotics. Often, simply through a different configuration of the same software,

* Corresponding author.
E-mail address: patrizi@diag.uniroma1.it (F. Patrizi).

https://doi.org/10.1016/j.robot.2025.105116

Available online 27 June 2025

such as noiseless or ideal movement actions, a simplified environment acting as an abstraction can be obtained. Importantly, this simplified model also applies to a variety of tasks that may be defined over the same domain.

Taking advantage of the abstraction hierarchy, we devise an approach that allows RL algorithms to efficiently explore the ground environment, while guaranteeing optimal convergence. The core intuition is that the value function $\bar{V}^*$ of the abstract MDP $\bar{\mathcal{M}}$ can be exploited to guide learning on the lower model $\mathcal{M}$. Technically, we adopt Reward Shaping (RS) and Reward Heuristics (RH) techniques based on $\bar{V}^*$. In this way, when learning in $\mathcal{M}$, the agent is *biased* to visit first the states that correspond in $\bar{\mathcal{M}}$ to those that are preferred by the abstract policy, thus trying, in a sense, to replicate its behavior in $\mathcal{M}$. In order to guarantee the effectiveness of the exploration bias, it is essential that the transitions of $\bar{\mathcal{M}}$ are good proxies for the dynamics of $\mathcal{M}$. We characterize this relationship by identifying conditions under which the abstraction induces an exploration policy that is consistent with the ground domain.

The proposed approach is also tailored to complex robot control problems in presence of obstacles. Multi-joint robots controlled with force/torque inputs and acting in environments with obstacles are abstracted with discrete position control problems, whose solutions are exploited to compute the solution of the original problem. In this context, planning techniques are used to find solutions in the abstract layer, such solutions are then used to automatically generate reward shaping and reward heuristic functions, and finally such additional reward functions are used to effectively drive the exploration of the RL agent in the more concrete layer.

The contributions of this work include: (i) the definition of a novel RS schema that allows for transferring the acquired experience from coarser models to the more concrete domains in the abstraction hierarchy; (ii) the derivation of a relationship between each abstraction and the exploration policy that is induced in the lower MDP; (iii) the identification of *abstract value approximation*, as a new condition to evaluate when an abstraction can be a good representative of ground MDP values; (iv) an effective abstraction for robot control problems allowing for integration of planning and RL techniques; (v) an extensive experimental analysis showing that our approach significantly improves sample-efficiency and that, in complex robot control problems, planning and learning integration and reward heuristics significantly outperform baseline approaches.

2. Related work

The field of Hierarchical Reinforcement Learning specifically studies how to improve the learning efficiency in presence of MDP abstractions. In this area, many of the early works can be categorized as instances of "state abstractions", since they primarily focus on how states are grouped together into abstract states. In this group we recall classic approaches, such as MAXQ [5] and HAM [6]. State compressions have been later studied by [4], who proposed a first systematic categorization of state mapping functions. This work does not discuss learning algorithms, but it listed the main properties of state abstractions. Together with [4,7] has been one of the early works to formally link the abstract decision process to the ground domain in terms of the respective transition probabilities. Namely, [7] developed MDP Homomorphisms and some approximated variants. Differently to our work, these MDP Homomorphisms can only capture spatial symmetries in the transition and reward functions. Instead, we are interested in coarse partitioning of neighboring states, which cannot be approximated with a single value. This challenge, namely, the impossibility of approximating groups of states with a single value $\bar{V}^*(\bar{s})$, also appeared in [8], where it was treated as a generic form of non-Markovianity. Instead, we do not incur in the same issue, thanks to our use of reward shaping.

Our notion of abstraction is inspired from the state abstractions listed above and the classic works on options [9]. However, instead of simply augmenting the ground MDP with options, which would result in a semi-MDP with non-Markovian effects, we only use them as formalizations of partial policies that appears in the formal proofs. The specific options do not need to be explicitly defined and linked between the various MDP levels. Our abstractions are closely related to those described in [10,11], in which both the state and the action spaces differ between the ground and the abstract decision process. Apart from the elegant formalization, these two works do not exploit, as in our work, explicit abstract MDPs, since they only learn in one ground model. Specifically, these papers do not consider any abstract dynamics and reward associated to the abstract states.

One of the earliest works that discussed the use of Reward Shaping in Hierarchical RL is [12]. This work studied how to learn an abstract potential function from experience and how to use it in potential-based RS. This insightful work introduced some important principles in this topic. However, it did not discuss how mapping functions should be defined between the two representations and how the non-Markovianity of HRL should be handled. As a later work, we recall [13], who also studied the application of a form of reward shaping in context of HRL. This algorithm was specifically developed for the MAXQ algorithm. Although not strictly related to HRL, [14] proposed a new form of biased RS for goal MDPs. The form of shaping they employ is designed to be effective for goal-based MDP, but has more relaxed guarantees with respect to classic Potential-Based RS. In a different work, [15], the authors propose a translation from formal tasks specifications to PBRS. Here, what composes the hierarchy is the prioritization of the various formal objectives to satisfy. Therefore, we do not regard this work as being part of the HRL literature. More recently, following [16], the work [17] investigated the impact of Potential-Based RS and derived an interesting connection with the baselines of policy-gradient algorithms. Moreover, they showed that the undesired effects of PBRS for the episodic case can be avoided by considering a longer horizon time, for which they also provided an upper bound. This result contributes to the theoretical understanding of PBRS. Nevertheless, our off-policy scheme is more suitable when the environment is relatively slow and shorter episodes are preferred.

With a different objective with respect to this paper, various works consider how abstractions may be learnt instead of being pre-defined, including [12,18,19]. This is an interesting direction to follow and the models that are obtained with such techniques may be still exploited with our method.

Hierarchical RL has also been used for complex robotics tasks. [20, 21] both propose hierarchical learning architectures for complex robotic manipulation, which involves decomposing long-horizon tasks into smaller, reusable skills or primitives. The former introduces a Universal Option Framework for multistep robotic manipulation, utilizing goal-conditioned universal policies at different hierarchical levels, while the latter introduces a hierarchical hybrid learning system for contact-rich robotic assembly that combines a low-level parameterized skill library with an imitation-learned high-level policy. So, these approaches basically reiterate on the options framework. [22] introduces a framework for quadrupedal robots to manipulate objects in cluttered environments. The framework utilizes a two-level control structure: a high-level policy determines target motions based on task objectives and obstacle avoidance, while a low-level policy translates these into joint commands for execution at a higher frequency. Our approach, while also more general in its methodology, in the end produces a single policy containing useful information about the coarser representations, instead of using two separate and interconnected controllers feeding each other.

3. Preliminaries

Notation. Any total function $f : \mathcal{X} \to \mathcal{Y}$ induces a partition on its domain $\mathcal{X}$, such that two elements are in the same block iff $f(x) = f(x')$. We denote blocks of the partition by the elements of $\mathcal{Y}$, thus writing $x \in y$ instead of $x \in \{x' \mid x' \in \mathcal{X}, f(x') = y\}$. With $\Delta(\mathcal{Y})$, we denote the class of probability distributions over a set $\mathcal{Y}$. Also, $f : \mathcal{X} \to \Delta(\mathcal{Y})$ is a function returning a probability distribution (i.e., $f : \mathcal{X}, \mathcal{Y} \to [0, 1]$, with $\sum_{y \in \mathcal{Y}} f(x, y) = 1$, for all $x \in \mathcal{X}$).

Markov decision processes (MDPs). A Markov Decision Process (MDP) $\mathcal{M}$ is a tuple $\langle S, \mathcal{A}, T, R, \gamma \rangle$, where S is a set of states, $\mathcal{A}$ is a set of actions, $T : S \times \mathcal{A} \to \Delta(S)$ is the probabilistic transition function, $R : S \times \mathcal{A} \times S \to \mathcal{R}$ is the reward function (for $\mathcal{R} := [r_-, r_+] \subset \mathbb{R}$), and $0 < \gamma < 1$ is the discount factor. A deterministic policy is a function $\rho : S \to \mathcal{A}$. We follow the standard definitions for the value of a policy $V^\rho(s)$, as the expected sum of discounted returns, the value of an action $Q^\rho(s, a)$, and their respective optima $V^* = \max_{\rho \in P} V^\rho(s)$ and $Q^*(s, a)$ [23]. We may also write $Q(s, \rho)$ to denote $V^\rho(s)$. Reinforcement Learning (RL) is the task of learning an optimal policy in an MDP with unknown T and R.

Definition 1. A RL algorithm is *off-policy* if, for any MDP $\mathcal{M}$ and experience $(\mathcal{ARS})^t$ generated by a stochastic exploration policy $\rho_e : S, \mathbb{N} \to \Delta(\mathcal{A})$ such that $\forall s, t, a : \rho_e(s, t, a) > 0$, the algorithm converges to the optimal policy of $\mathcal{M}$, as $t \to \infty$.

Reward shaping (RS). Reward Shaping (RS) is a technique for learning in MDPs with sparse rewards, i.e., those occurring rarely during exploration. The purpose of RS is to guide the agent by exploiting some prior knowledge in the form of additional rewards: $R^s(s, a, s') := R(s, a, s') + F(s, a, s')$, where F is the *shaping* function. In the classic approach, called *Potential-Based RS* [24] (simply called Reward Shaping from now on), the shaping function is defined in terms of a potential function, $\Phi : S \to \mathbb{R}$, as:

$$F(s, a, s') := \gamma \Phi(s') - \Phi(s)$$

If an infinite horizon is considered, this definition and its variants [25,26] guarantee that the set of optimal policies of $\mathcal{M}$ and $\mathcal{M}^s := \langle S, \mathcal{A}, T, R^s, \gamma \rangle$ coincide. Indeed, as shown by [27], the Q-learning algorithm over $\mathcal{M}^s$ performs the same updates as Q-learning over $\mathcal{M}$ with the modified Q-table initialization: $Q'_0(s, a) := Q_0(s, a) + \Phi(s)$.

Options. An option [9], for an MDP $\mathcal{M}$, is a temporally-extended action, defined as $o = \langle \mathcal{I}_o, \rho_o, \beta_o \rangle$, where $\mathcal{I}_o \subseteq S$ is an initiation set, $\rho_o : S \to \mathcal{A}$ is the policy to execute, and $\beta_o : S \to \{0, 1\}$ is a termination condition that, from the current state, computes whether the option should terminate. We write $Q^*(s, o)$ to denote the expected return of executing o until termination and following the optimal policy afterwards.

ϕ-Relative options. [11] Given an MDP $\mathcal{M}$ and a function $\phi : S \to \bar{S}$, an option $o = \langle \mathcal{I}_o, \rho_o, \beta_o \rangle$ of $\mathcal{M}$ is said to be *ϕ-relative* iff there is some $\bar{s} \in \bar{S}$ such that:

$$\mathcal{I}_o = \{s \mid s \in \bar{s}\}, \quad \beta_o(s) = \mathbb{I}(s \notin \bar{s}), \quad \rho_o \in P_{\bar{s}}$$

where $\mathbb{I}(\xi)$ equals 1 if ξ is true, 0 otherwise, and $P_{\bar{s}}$ is the set of policies of the form $\rho_{\bar{s}} : \{s \mid s \in \bar{s}\} \to \mathcal{A}$. In other words, policies of ϕ-relative options are defined within some block $\bar{s}$ in the partition induced by ϕ and they must terminate exactly when the agent leaves the block where it was initiated.

4. Framework

Consider an environment where experience is costly to obtain, such as a complex simulation or an actual environment where a physical robot operates. We take this as the *ground* MDP $\mathcal{M}_0$, which we aim to solve while limiting, as much as possible, the number of interactions with the environment. To this end, instead of learning directly on the ground MDP, over which collecting experience is costly, we consider a simplified version of it, called the *abstract* MDP, which defines a simpler, related problem. Obviously, this approach can be iterated over and over, by considering the obtained abstract MDP as the ground one, producing a simplified version of it, ad so on. Indeed, we consider a hierarchy of related MDPs $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_n$, of decreasing complexity, where the experience acquired by an expert acting in $\mathcal{M}_i$ can be exploited to speed up learning in $\mathcal{M}_{i-1}$.

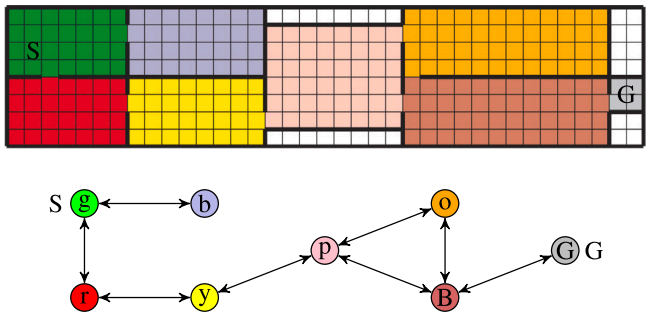


Fig. 1. A grid world domain (top) and an abstraction (bottom). The colors encode the mapping function, G is the goal. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In this work, we assume that the abstract MDPs are given. More precisely, we assume that a mapping function is provided, which defines how an abstract MDP can be obtained starting from the MDP in the next lower level. In this article, we focus on how the algorithm can use such abstractions and do not discuss how a mapping function defining effective abstractions can actually be obtained.

For each MDP abstraction $\mathcal{M}_i$, we assume a *mapping* function $\phi_i : S_i \to S_{i+1}$ projecting the states of $\mathcal{M}_i$ to those of the direct higher-level MDP $\mathcal{M}_{i+1}$. This is a classic assumption in Hierarchical RL [7,10,11], and the function ϕ_i can be easily obtained together with the abstract MDP. We call the pair $\langle \mathcal{M}_{i+1}, \phi_i \rangle$ an *abstraction* of $\mathcal{M}_i$. We will often omit ϕ_i and refer to $\mathcal{M}_{i+1}$ as the abstraction of $\mathcal{M}_{i+1}$, implicitly assuming the existence of ϕ_i. It is immediate to see that each ϕ_i induces a partition over S_i, where each block contains all the states that are mapped, through ϕ_i, to the same state in $\mathcal{M}_{i+1}$.

As for actions, unlike other works, we do not require any mapping between the action spaces of $\mathcal{M}_i$ and $\mathcal{M}_{i+1}$, thus leaving the underlying relationship implicit and providing the designer with great flexibility in defining the hierarchy.

An abstract model is therefore a suitable relaxation of the environment dynamics. For example, in a navigation scenario, an abstraction could contain high-level actions that allow to just "leave the room", instead of navigating through the space using low-level controls. In Section 6, we formalize this intuition by deriving a measure that quantifies how accurate an abstraction is with respect to the lower-level domain. As a simple example, consider the grid-world domain, the abstract MDP and the mapping in Fig. 1. Using abstraction, we can easily constrain the agent to avoid exploring "room" (b), and only learn options for moving to and within the other rooms. The same does not necessarily hold for the orange block (o), as the optimal path depends on the specific transition probabilities in each arc.

4.1. Exploiting knowledge

Consider a hierarchy of abstractions $\mathcal{M}_0, \ldots, \mathcal{M}_n$, with mappings $\phi_0, \ldots, \phi_{n-1}$. The learning process proceeds incrementally along the hierarchy, from the highest (easiest) to the lowest (hardest) MDP. When learning on $\mathcal{M}_i$, we can take advantage of the knowledge acquired on $\mathcal{M}_{i+1}$ using a Reward-Shaping function defined starting from the

solution obtained for $\mathcal{M}_{i+1}$. Indeed, we observed that the optimal value function V^*_{i+1} of $\mathcal{M}_{i+1}$ acts as a helpful heuristic to estimate the desirability of a state s of $\mathcal{M}_i$, as the value of the state $s' = \phi_i(s')$ of $\mathcal{M}_{i+1}$. This is formalized by the following definition.

Definition 2. Let $\mathcal{M}_i$ be an MDP and $\langle \mathcal{M}_{i+1}, \phi_i \rangle$ its abstraction. The *biased MDP* of $\mathcal{M}_i$ with respect to $\langle \mathcal{M}_{i+1}, \phi_i \rangle$ is the model $\mathcal{M}^b_i$, resulting from the application of Reward Shaping to $\mathcal{M}_i$, using the potential function:

$$\Phi(s) := V^*_{i+1}(\phi_i(s)),$$

where V^*_{i+1} is the optimal value function of $\mathcal{M}_{i+1}$.

With $\mathcal{M}^b_i$ at hand, we can assess how desirable each state s of $\mathcal{M}_i$ is by looking at the value of $\phi_i(s)$. This can be very beneficial to the learning process, as high potentials yield high Q-function value initializations [27] and then lead, in turn, to a reduced number of iterations for value-based algorithms, such as Value Iteration, or the choice of optimal baselines in gradient-based algorithms [17]. In fact, in [24] it was first noticed that an MDP's optimal value function is a very natural potential for RS. In this paper, we push this idea further by using, instead of the MDP's own value function, that of the abstract MDP, which can be computed more efficiently.

5. Reward shaping for episodic RL

Potential-Based RS has been explicitly designed not to alter the optimal policies. Specifically, regardless of the potential function adopted, in infinite-horizon settings, or where episodes terminate in a zero-potential absorbing state, the optimal policies obtained using the potential function match those obtained without [24]. However, this is no longer true when episodes may terminate in states with arbitrary potential, as common in RL, where the agent's experiences are diversified by breaking up exploration into episodes of finite length. In these cases, the optimal policies obtained with and without potential function may differ [28]. To see this, consider an episode $\pi = s_0 a_1 r_1 s_1 \dots s_n$ from an MDP $\mathcal{M}$ and the associated episode $\pi' = s_0 a_1 r'_1 s_1 \dots s_n$, where rewards have been modified via RS. The returns of the two sequences are related as follows [28]:

$$G(\pi') := \sum_{t=0}^{n-1} \gamma^t r'_{t+1} = G(\pi) + \gamma^n \Phi(s_n) - \Phi(s_0) \tag{1}$$

As it can be seen, the term $\Phi(s_0)$ does not affect, alone, the optimal policies in the two cases, as its contribution is uniform over all episodes. As for the term $\gamma^n \Phi(s_n)$, instead, its contribution varies depending on the length of the episode and the final state s_n, thus overall affecting in a different way the return of each episode, which leads to different optimal policies.

To avoid the above, [28] proposes assigning the null potential $\Phi(s_n) = 0$ to every terminal state, thus preserving the original returns (modulo a subtractive constant). However, while correct, having matching optimal policies is not the only possible solution nor necessarily the most desirable. One might be interested in relaxing the guarantee of identical policies in favor of a stronger impact on learning speed. This has also been discussed and addressed with different solutions in [14]. As an example, consider an MDP with a null reward function on every transition, except on those reaching a specific goal state. As a consequence of Eq. (1) and assignment $\Phi(s_n) = 0$, all finite trajectories containing no goal state have the same return. Since the agent cannot estimate the distance to the goal through the differences in return, the return-invariant RS of [28] does not provide the agent with a *persistent* exploration bias.

The form of RS adopted in this paper and formalized in Definition 2 is different from that above, in that it does not assign null potentials to terminal states; for this reason, we call it *non* return-invariant RS. Since, as a consequence of using a non return-invariant RS, the optimal policies of $\mathcal{M}^b_i$ and $\mathcal{M}_i$ in Definition 2 do not necessarily match, we refer to $\mathcal{M}^b_i$ as a *biased* MDP.

In principle, it is reasonable to expect that using a biased MDP may potentially lead to non-optimal solutions. This issue is addressed in the next section, where we report a learning procedure which recovers optimal convergence in this setting.

Algorithm 1: Main algorithm

Input: Off-policy learning algorithm $\mathcal{L}$
Input: $\mathcal{M}_0, \dots, \mathcal{M}_n$, $\phi_0, \dots, \phi_{n-1}$
Output: $\hat{\rho}^*_0$, ground MDP estimated optimum

1 $\hat{V}^*_{n+1} : s \mapsto 0$;
2 $\phi_n : s \mapsto s$;
3 **foreach** $i \in \{n, \dots, 0\}$ **do**
4 $\quad F_i \leftarrow \texttt{Shaping}(\gamma_i, \phi_i, \hat{V}^*_{i+1})$;
5 $\quad \text{Learner}_i \leftarrow \mathcal{L}(\mathcal{M}_i)$;
6 $\quad \text{Learner}^b_i \leftarrow \mathcal{L}(\mathcal{M}_i)$;
7 $\quad$ **while** *not* $\text{Learner}_i.\texttt{Stop}()$ **do**
8 $\quad\quad s \leftarrow \mathcal{M}_i.\texttt{State}()$;
9 $\quad\quad a \leftarrow \text{Learner}^b_i.\texttt{Action}(s)$;
10 $\quad\quad r, s' \leftarrow \mathcal{M}_i.\texttt{Act}(a)$;
11 $\quad\quad r^b \leftarrow r + F_i(s, a, s')$;
12 $\quad\quad \text{Learner}^b_i.\texttt{Update}(s, a, r^b, s')$;
13 $\quad\quad \text{Learner}_i.\texttt{Update}(s, a, r, s')$;
14 $\quad$ **end**
15 $\quad \hat{\rho}^*_i \leftarrow \text{Learner}_i.\texttt{Output}()$;
16 $\quad \hat{V}^*_i \leftarrow \texttt{ComputeValue}(\hat{\rho}^*_i, \mathcal{M}_i)$;
17 **end**

5.1. The algorithm

Since we deliberately adopt a form of RS which is not return invariant in the episodic setting, we devised a technique to recover optimality in the original MDP, when coupled with any off-policy algorithm. The procedure is presented in detail in Algorithm 1. Learning proceeds sequentially, training from the most abstract model to the ground domain. When learning on the ith MDP, the estimated optimal value function $\hat{V}^*_{i+1}$ of the previous model is used to obtain a reward shaping function (line 4). This implicitly defines the biased MDP $\mathcal{M}^b_i$ of Definition 2. Experience is collected by sampling actions according to a stochastic exploration policy, as determined by the specific learning algorithm $\mathcal{L}$. Such policy may be derived from the current optimal policy estimate for $\mathcal{M}^b_i$, such as an ϵ-greedy exploration policy in $\hat{\rho}^{b*}_i$ (line 9). Finally, the outputs of each learning phase are the estimates $\hat{\rho}^*_i, \hat{V}^*_i$ for the original MDP $\mathcal{M}_i$. This allows iterating the process with an unbiased value estimate or concluding the procedure with the final learning objective $\hat{\rho}^*_0$.

Proposition 1. *Consider MDPs $\mathcal{M}_0, \dots, \mathcal{M}_n$ and their associated mapping functions $\phi_0, \dots, \phi_{n-1}$. If $\mathcal{L}$ is an off-policy learning algorithm, then, in every ith iteration of Algorithm 1, $\hat{\rho}^*_i$ converges to ρ^*_i, as the number of environment interactions increases.*

Proof. At any iteration $i \in \{n, \dots, 0\}$ of Algorithm 1, we are given $\mathcal{M}_i$, ϕ_i and $\hat{V}^*_{i+1}$. By construction, the two instantiations of $\mathcal{L}$, Learner_i and Learner^b_i, perform updates from transitions generated from $\mathcal{M}_i$ and $\mathcal{M}^b_i$, respectively. Actions are selected according to Learner^b_i which follows some exploration policy ${\rho_e}^b$. Since $\mathcal{M}_i$ and $\mathcal{M}^b_i$ share the same state and action spaces, ${\rho_e}^b$ is also an exploration policy for $\mathcal{M}_i$, in the sense of Definition 1. Therefore, Learner_i also converges to ρ^*_i, as the number of environment interactions $t \to \infty$. □

6. Abstraction quality

In principle, our approach can be used with any abstract model. However, for a ground MDP, not all abstractions are equally helpful. In this section, we discuss the properties that abstractions should have in order to effectively support the RL process. We stress that we assume the abstraction given, e.g., manually designed by a modeler based on its expertise, and we do not propose a technique to automatically generate it. Although being of theoretical importance, the definitions we introduce in the following do not offer practical guidelines to construct abstraction functions that guarantee improving sample efficiency.

As we can see from Algorithm 1, abstractions are used to construct effective exploration policies (line 9). Therefore, they should induce a biased MDP that assigns higher rewards to regions of the state space from which the optimal policy of the original problem can be easily estimated. This intuition is formalized by the notion of *exploration loss* of an abstraction, provided in Definition 4.

Although our method is applicable to any MDP, our analysis focuses on *goal MDPs*, a specific class that captures a broad set of tasks of practical interest.

Definition 3. An MDP $\mathcal{M} = \langle S, \mathcal{A}, T, R, \gamma \rangle$ is a *goal MDP* iff there exists a set $\mathcal{G} \subseteq S$ of *goal states* such that:

$$R(s,a,s') = \begin{cases} 1, \text{ if } s \notin \mathcal{G} \text{ and } s' \in \mathcal{G} \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

$$V^*(s) = 0, \forall s \in \mathcal{G} \tag{3}$$

Requirement (2) imposes that a unitary reward be offered whenever the MDP enters a goal state from a non-goal one, while no reward is offered in any other case. Requirement (3) implies that once a goal state is entered, no additional reward can be collected anymore (not even if the goal state is left for a non-goal one and then a goal state is reached again). Goal MDPs define tasks in a very straightforward way while remaining sufficiently general to capture a large number of practical scenarios, as will be seen in the experimental section.

We make the following assumptions about the relationship between an MDP and its abstraction.

Assumption 1. The ground MDP $\mathcal{M}_0$ is a goal MDP.

Assumption 2. If $\mathcal{M}_i$ is a goal MDP with goal states $\mathcal{G}_i$ and a corresponding abstraction $\langle \mathcal{M}_{i+1}, \phi_i \rangle$, also $\mathcal{M}_{i+1}$ is a goal MDP, such that:

$$\mathcal{G}_{i+1} = \cup_{s \in \mathcal{G}_i} \phi(s) \tag{4}$$

In other words, all and only the goal states of an MDP map to all and only the goal states of its direct abstraction. In the example of Fig. 1, the gray cells in the ground MDP map to the abstract goal G.

We start our analysis with two observations. First, by the way the framework is designed, convergence on some $\mathcal{M}_i$ depends only on its abstraction $\mathcal{M}_{i+1}$, and not on the other models. Thus, when discussing convergence properties, it will suffice to consider a generic MDP $\mathcal{M} = \langle S, \mathcal{A}, T, R, \gamma \rangle$ and its direct abstraction, $\bar{\mathcal{M}} = \langle \bar{S}, \bar{\mathcal{A}}, \bar{T}, \bar{R}, \bar{\gamma} \rangle$. We denote the relevant mapping as $\phi : S \to \bar{S}$.

Second, while a goal MDP $\mathcal{M}$ has sparse rewards, the reward function of the biased MDP $\mathcal{M}^b$ is no longer sparse. Depending on the abstraction $\langle \bar{\mathcal{M}}, \phi \rangle$, from which it is defined, the rewards of $\mathcal{M}^b$ can be as dense as needed. As confirmed empirically, this allows faster convergence on the biased MDP. Clearly, in addition to this, we also want the optimal policy ρ^{b*} to be a good exploration policy for the original domain, i.e., as similar as possible to an optimal policy ρ^* of $\mathcal{M}$. We measure such similarity as follows.

Definition 4. Given an MDP $\mathcal{M}$, the *exploration loss* of an abstraction $\langle \bar{\mathcal{M}}, \phi \rangle$ of $\mathcal{M}$ is the expected value loss of executing on $\mathcal{M}$ the optimal policy ρ^{b*} of $\mathcal{M}^b$, instead of some optimal policy ρ^* of $\mathcal{M}$:

$$L(\mathcal{M}, \langle \bar{\mathcal{M}}, \phi \rangle) := \max_{s \in S} |V^*(s) - Q(s, \rho^{b*})|$$

In order to limit this quantity, we relate abstract states $\bar{s} \in \bar{S}$ to sets of states $\phi^{-1}(\bar{s}) \subseteq S$ in the ground MDP. Similarly, actions $\bar{a} \in \bar{\mathcal{A}}$ correspond to non-interruptible policies that only terminate when leaving the current block. So, a more appropriate correspondence can be identified between abstract actions and ϕ-relative options in $\mathcal{M}$. We start by deriving, in Eq. (5), the multi-step value of a ϕ-relative option in goal MDPs.

Multi-step value of options. By combining the classic multi-step return of options [9], ϕ-relative options from [11] and goal MDPs of Definition 3, we obtain the following result.

Lemma 1. *Given a goal MDP $\mathcal{M}$ and $\phi : S \to \bar{S}$, for any $s \in S$ and ϕ-relative option o, the optimal value of o is:*

$$Q^*(s,o) = \sum_{k=0}^{\infty} \gamma^k \sum_{s_{1:k} \in \phi(s)^k} \sum_{s' \notin \phi(s)} p(s_{1:k} s' \mid s, \rho_o) \big(\mathbb{I}(s' \in G) + \gamma V^*(s') \big) \tag{5}$$

Proof. By assumption $\mathcal{M}$ is a goal MDP over some $\mathcal{G} \subseteq S$. For $s \in \mathcal{G}$, we know $Q^*(s,o) = 0$. We consider $s \notin \mathcal{G}$. Since the MDP $\mathcal{M}$ is clear from the context, to uniform the notation, we use $p(s' \mid s, a)$ instead of $T(s,a,s')$. Following a similar procedure as [11], for our definition of goal MDPs:

$$\begin{aligned} Q^*(s,o) = \; & \\ & \mathbb{E}_{s'|s,o}[R(s,\rho_o(s),s') + \gamma(\mathbb{I}(s' \in \phi(s)) Q^*(s',o) + \mathbb{I}(s' \notin \phi(s)) V^*(s'))] \\ & = \sum_{s' \in \phi(s)} p(s' \mid s, \rho_o(s))[\cdot] + \sum_{s' \notin \phi(s)} p(s' \mid s, \rho_o(s))[\cdot] \\ & = \sum_{s' \in \phi(s)} p(s' \mid s, \rho_o(s)) \gamma Q^*(s',o) \\ & \quad + \sum_{s' \notin \phi(s)} p(s' \mid s, \rho_o(s)) \big(\mathbb{I}(s' \in \mathcal{G}) + \gamma V^*(s') \big) \end{aligned} \tag{6}$$

We abbreviate the second term of (6) with Ψ and let $s_0 = s$. Then, similarly to the classic multi-step value of options [9], we can expand over time.

$$\begin{aligned} Q^*(s,o) &= \sum_{s' \in \phi(s)} p(s' \mid s, \rho_o(s)) \gamma Q^*(s',o) + \Psi(s,o) \\ &= \Psi(s,o) + \gamma \sum_{s' \in \phi(s)} p(s' \mid s, \rho_o(s)) \Psi(s',o) + \\ &\gamma^2 \sum_{s', s'' \in \phi(s)^2} p(s's'' \mid s, \rho_o(s)\, \rho_o(s')) Q^*(s'',o) \\ &= \sum_{k=0}^{\infty} \gamma^k \sum_{s_{1:k} \in \phi(s)^k} p(s_{1:k} \mid s, \rho_o) \Psi(s_k, o) \\ &= \sum_{k=0}^{\infty} \gamma^k \sum_{s_{1:k} \in \phi(s)^k} \sum_{s' \notin \phi(s)} p(s_{1:k} s' \mid s, \rho_o) \big(\mathbb{I}(s' \in \mathcal{G}) + \gamma V^*(s') \big) \end{aligned}$$

which is the expression in Lemma 1. □

With a similar procedure, we can also show that the multi-step value of a ϕ-relative option o in a biased MDP $\mathcal{M}^b$ with respect to $\langle \bar{\mathcal{M}}, \phi \rangle$ is as follows:

$$\begin{aligned} Q^{b*}(s,o) = & \sum_{k=0}^{\infty} \gamma^k \sum_{s_{1:k} \in \phi(s)^k} \sum_{s' \notin \phi(s)} \\ & p(s_{1:k} s' \mid s, \rho_o) \big(\mathbb{I}(s' \in \mathcal{G}) + \gamma \bar{V}^*(\phi(s')) - \bar{V}^*(\phi(s)) + \gamma V^*(s') \big) \end{aligned}$$

$Q^*(s,o)$ sums over any sequence of states $s_1 \dots s_k$ remaining within $\phi(s)$ and leaving the block after k steps at s'. A similar result was derived in [11] for a slightly different definition of goal MDPs. However, Eq. (5) is not an expression about abstract states, yet, because

it depends on the specific ground state s' that is reached at the end of the option. Therefore, in the following definition, we introduce a parameter ν that quantifies how much the reachable states s' in each block are dissimilar in value. This allows to jointly talk about the value of each group of states as a whole. We define a function $W_\nu : \bar{S}\times\bar{S} \to \mathbb{R}$, that, given a pair of abstract states $\bar{s}, \bar{s}'$, predicts, with ν-approximation error, the value of any successor ground state $s' \in \bar{s}'$ that can be reached from some $s \in \bar{s}$.

Definition 5. Consider an MDP $\mathcal{M}$ and an abstraction $\langle \bar{\mathcal{M}}, \phi \rangle$. We define the *abstract value approximation* as the smallest $\nu \geq 0$ such that there exists a function $W_\nu : \bar{S} \times \bar{S} \to \mathbb{R}$ which, for all $\bar{s}, \bar{s}' \in \bar{S}$, satisfies:

$$\forall s \in \bar{s}, \forall s' \in \bar{s}', \forall a \in \mathcal{A} T(s,a,s') > 0 \Rightarrow |W_\nu(\bar{s},\bar{s}') - V^*(s')| \leq \nu \tag{7}$$

According to this definition, the frontier separating any two sets of states in the partition induced by ϕ must lie in ground states that can be approximated with the same optimal value, with a maximum error ν. Thus, any small ν puts a constraint on the mapping function ϕ. In the example of Fig. 1, each room is connected to each other through a single location, so this condition is simply satisfied for $\nu = 0$. However, this definition allows one to apply our results in the general case, provided that the ground states between two neighboring regions can be approximated to be equally close to the goal, with a maximum error of ν.

Thanks to Definition 5, the value of options can be bounded, by only considering future abstract states. This is formalized in the forthcoming Lemma 2. For this purpose, when starting from some $s \in \mathcal{S}$, we use $p(\bar{s}', k \mid s)$ to denote the probability of the event of remaining for k steps in the same block as s, then reaching any $s' \in \bar{s}'$ at the next transition.

Lemma 2. *Let $\mathcal{M}$ be an MDP and $\langle \bar{\mathcal{M}}, \phi \rangle$ its abstraction, satisfying Assumptions 1 and 2. The value of any ϕ-relative option o in $\mathcal{M}$ admits the following lower bound:*

$$Q^*(s,o) \geq \sum_{\bar{s}' \in \bar{S}\setminus\{\phi(s)\}} \sum_{k=0}^{\infty} \gamma^k \, p(\bar{s}', k \mid s, \rho_o) \left(\mathbb{I}(\bar{s}' \in \bar{\mathcal{G}}) + \gamma \left(W_\nu(\phi(s), \bar{s}') - \nu \right) \right)$$

at any $s \in \mathcal{S}$, where, ν and W_ν follow Definition 5.

Proof. We recall that $p(\bar{s}', k \mid s, \rho)$ denotes the probability of the event of remaining for k steps within $\phi(s)$, then reaching $\bar{s}'$ at the next transition, when following policy ρ, starting from s. Similarly, we use $p(s', k \mid s, \rho)$ to represent the probability of remaining k steps within $\phi(s)$ then reaching a specific ground state $s' \in \mathcal{S} \setminus \phi(s)$.

To obtain the result, we marginalize the probabilities appearing in Lemma 1 over all possible trajectories $s_{1:k}$:

$$Q^*(s,o) = \sum_{s' \in \mathcal{S}\setminus\phi(s)} \sum_{k=0}^{\infty} \gamma^k \, p(s', k \mid s, \rho_o) \left(\mathbb{I}(s' \in \mathcal{G}) + \gamma \, V^*(s') \right)$$

Now, for all s', k such that $p(s', k \mid s, \rho_o) > 0$, there is one state $s_k \in \phi(s)$, reachable in k steps from s under ρ_o, from which $T(s_k, \rho_o(s_k), s') > 0$. From Definition 5, we know $|W_\nu(\phi(s_k), \phi(s')) - V^*(s')| \leq \nu$. Therefore, we can provide a lower bound for each term $V^*(s')$ in the sum above:

$$Q^*(s,o) \geq \sum_{s' \in \mathcal{S}\setminus\phi(s)} \sum_{k=0}^{\infty} \gamma^k p(s', k \mid s, \rho_o) \left(\mathbb{I}(s' \in \mathcal{G}) + \gamma (W_\nu(\phi(s), \phi(s')) - \nu) \right)$$

because $\phi(s_k) = \phi(s)$. It is now possible to split the sum $\sum_{s' \in \mathcal{S}\setminus\phi(s)}$ into $|\bar{S}| - 1$ sums over future blocks and marginalize among them to obtain:

$$Q^*(s,o) \geq \sum_{\bar{s}' \in \bar{S}\setminus\{\phi(s)\}} \sum_{k=0}^{\infty} \gamma^k \, p(\bar{s}', k \mid s, \rho_o) \left(\mathbb{I}(\bar{s}' \in \bar{\mathcal{G}}) + \gamma (W_\nu(\phi(s), \bar{s}') - \nu) \right)$$

since $\mathbb{I}(s \in \mathcal{G}) = \mathbb{I}(\phi(s) \in \bar{\mathcal{G}})$. This proves the lemma. With the same procedure, we also obtain the upper bound:

$$Q^*(s,o) \leq \sum_{\bar{s}' \in \bar{S}\setminus\{\phi(s)\}} \sum_{k=0}^{\infty} \gamma^k p(\bar{s}', k \mid s, \rho_o) \left(\mathbb{I}(\bar{s}' \in \bar{\mathcal{G}}) + \gamma (W_\nu(\phi(s), \bar{s}') + \nu) \right) \quad \square$$

This lemma provides a different characterization of options, in terms of abstract states, so that it can be exploited to obtain Theorem 1.

Exploration loss of abstractions. Based on the results of previous section, we can now derive a bound for the exploration loss of Definition 4, for any abstraction. We expand the results of [11] to limit this quantity.

First, we observe that any policy can be regarded as a finite set of ϕ-relative options whose initiation sets are partitioned according to ϕ. Then, by Lemma 2, we know that an approximation for the value of options only depends on the k-step transition probability to each abstract state. So, we assume this quantity is bounded by some ϵ:

Definition 6. Given an MDP $\mathcal{M}$, a function $\phi : \mathcal{S} \to \bar{\mathcal{S}}$ and two policies ρ_1, ρ_2, we say that ρ_1 and ρ_2 have *abstract similarity* ϵ, if:

$$\forall s \in \mathcal{S}, \forall \bar{s}' \in \bar{\mathcal{S}} \setminus \{\phi(s)\}, \forall k \in \mathbb{N}, |p(\bar{s}', k \mid s, \rho_1) - p(\bar{s}', k \mid s, \rho_2)| \leq \epsilon$$

Intuitively, abstract similarity measures the difference between the two abstract actions described by each policy, as it only depends on the probability of the next abstract state that is reached, regardless of the single trajectories and the specific final ground state. In the running example of Fig. 1, two policies with low ϵ, after the same number of steps, would reach the same adjacent room with similar probability. It is now finally possible to state our result.

Theorem 1. *Let $\mathcal{M}$ and $\langle \bar{\mathcal{M}}, \phi \rangle$ be, respectively, an MDP and an abstraction satisfying Assumptions 1 and 2, and let $\mathcal{M}^b$ be the biased MDP. If ϵ is the abstract similarity of ρ^* and ρ^{b*}, and the abstract value approximation is ν, then, the exploration loss of $\langle \bar{\mathcal{M}}, \phi \rangle$ satisfies the following:*

$$L(\mathcal{M}, \langle \bar{\mathcal{M}}, \phi \rangle) \leq \frac{2|\bar{S}|(\epsilon + \gamma\nu)}{(1-\gamma)^2}$$

Proof. Any policy ρ can be represented as a set $\mathcal{O}$, composed of ϕ-relative options whose initiation sets are partitioned according to ϕ and $\forall s \in \mathcal{S}, \exists o \in \mathcal{O} : \rho_o(s) = \rho(s)$. Let $\mathcal{O}^*$ and $\mathcal{O}^{b*}$ be the ϕ-relative options associated to ρ^* and ρ^{b*}. We now compute what is the difference in value between executing ρ^* and ρ^{b*}, for one option each, then following ρ^* afterwards. For any $s \in \mathcal{S}\setminus\mathcal{G}$, let o^* and o^{b*} be the relevant options in $\mathcal{O}^*$ and O^{b*}, respectively. We bound the following difference in value:

$$|Q^*(s,o^*) - Q^*(s,o^{b*})| = Q^*(s,o^*) - Q^*(s,o^{b*})$$

From an application of the upper and lower bound of Lemma 2,

$$|Q^*(s,o^*) - Q^*(s,o^{b*})| \leq$$

$$\sum_{\bar{s}' \in \bar{S}\setminus\{\phi(s)\}} \sum_{k=0}^{\infty} \gamma^k p(\bar{s}', k \mid s, \rho_{o^*}) \left(\mathbb{I}(\bar{s}' \in \bar{\mathcal{G}}) + \gamma (W_\nu(\phi(s), \bar{s}') + \nu) \right) -$$

$$\sum_{\bar{s}' \in \bar{S}\setminus\{\phi(s)\}} \sum_{k=0}^{\infty} \gamma^k p(\bar{s}', k \mid s, \rho_{o^{b*}}) \left(\mathbb{I}(\bar{s}' \in \bar{\mathcal{G}}) + \gamma (W_\nu(\phi(s), \bar{s}') - \nu) \right)$$

$$= \sum_{\bar{s}' \in \bar{S}\setminus\{\phi(s)\}} \left(\mathbb{I}(\bar{s}' \in \bar{\mathcal{G}}) + \gamma W_\nu(\phi(s), \bar{s}') \right) \sum_{k=0}^{\infty} \gamma^k \left(p(\bar{s}', k \mid s, \rho_{o^*}) - p(\bar{s}', k \mid s, \rho_{o^{b*}}) \right) +$$

$$\sum_{\bar{s}' \in \bar{S}\setminus\{\phi(s)\}} \sum_{k=0}^{\infty} \gamma^k \left(p(\bar{s}', k \mid s, \rho_{o^*}) + p(\bar{s}', k \mid s, \rho_{o^{b*}}) \right) \gamma\nu$$

Now we apply Definition 6 of abstract similarity and bound

$$|Q^*(s,o^*) - Q^*(s,o^{b*})| \leq$$

$$\sum_{\bar{s}' \in \bar{S}\setminus\{\phi(s)\}} \left(\mathbb{I}(\bar{s}' \in \bar{\mathcal{G}}) + \gamma W_\nu(\phi(s), \bar{s}') \right) \sum_{k=0}^{\infty} \gamma^k \epsilon + \sum_{\bar{s}' \in \bar{S}\setminus\{\phi(s)\}} \sum_{k=0}^{\infty} \gamma^{k+1} 2\nu$$

$$= \sum_{\bar{s}' \in \bar{S}\setminus\{\phi(s)\}} \left(\left(\mathbb{I}(\bar{s}' \in \bar{\mathcal{G}}) + \gamma W_\nu(\phi(s), \bar{s}') \right) \frac{\epsilon}{1-\gamma} + \frac{2\gamma\nu}{1-\gamma} \right)$$

Since in a goal MDP the maximum value is 1, we know $W_\nu(\phi(s), \bar{s}') \leq (1+\nu)$, for all $s \in \mathcal{S}, \bar{s}' \in \bar{\mathcal{S}}$. Moreover, if W_ν satisfies condition (7), the function $W_\nu^{\text{clip}}(\bar{s}, \bar{s}') := \min\{1, W_\nu(\bar{s}, \bar{s}')\}$ also satisfies it. Concluding,

$$|Q^*(s,o^*) - Q^*(s,o^{b*})| \leq \sum_{\bar{s}' \in \bar{S}\setminus\{\phi(s)\}} \left((1+\gamma) \frac{\epsilon}{1-\gamma} + \frac{2\gamma\nu}{1-\gamma} \right) \leq$$

$$|\bar{S}|\Big(\frac{2\epsilon}{1-\gamma}+\frac{2\gamma\nu}{1-\gamma}\Big)=\frac{2|\bar{S}|(\epsilon+\gamma\nu)}{1-\gamma}$$

This is a bound on the value loss of executing a single option from the set $\mathcal{O}^{b*}$. To this option set the results from [11] apply and Eq. (3), in particular. So we obtain the final result:

$$L(\mathcal{M},\langle\bar{\mathcal{M}},\phi\rangle)\leq\frac{2|\bar{S}|(\epsilon+\gamma\nu)}{(1-\gamma)^2}\quad\square$$

This result shows under which conditions an abstraction induces an exploration policy that is similar to some optimal policy of the original domain. Observe that optimal convergence is guaranteed regardless of the abstraction quality, because the stochastic exploration policy satisfies the mild conditions imposed by the adopted off-policy learning algorithm.

Apart from our application to the biased MDP policy, the theorem also has a more general impact. It shows, indeed, that, provided that the abstraction induces a partition of states whose frontiers have some homogeneity in value (Definition 5), it is possible to reason in terms of abstract transitions. Only for a $\nu = 0$, this bound has similarities with inequality n. 5 in [11]. Notice, though, that the one stated here is expressed in terms of size of the abstract state space, which usually can be assumed to be $|\bar{S}| \ll |S|$.

7. Abstraction for robot control

In this section, we describe a suitable abstraction for robot control that is based on abstracting a continuous force/torque control problem in a discrete position control problem. More specifically, the MDP $\mathcal{M}$ models continuous states and actions to control a multi-joint robot with force/torque inputs, while the abstract MDP $\bar{\mathcal{M}}$ includes discrete states and actions corresponding to the Cartesian movement of the robot end-effector. The mapping function $\phi : S \to \bar{S}$ maps robot joint states to a discretized position of the end-effector (using forward kinematics or assuming end-effector position observations in $\mathcal{M}$). ϕ-relative options correspond to sequences of torque control actions allowing the robot end-effector to move between discrete positions (i.e., between $\bar{\mathcal{M}}$ blocks $\bar{s}$)

More specifically, we consider robot control problems in which the robot has to reach a goal configuration of its end-effector, i.e., a final state $s_f \in \{s; \|s - g\| < \delta\}$, where g is the target goal and δ is an error tolerance. The robot is controlled with force/torque actions τ on its joints and the kinematics and the dynamic models are assumed to be unknown. Moreover, we consider the presence of static obstacles in the environment that must be avoided along the trajectory that reaches the target goal.

The MDP used to learn the actual continuous robot behaviors is defined as $\mathcal{M} = \langle S, \mathcal{A}, T, R\rangle$, with $S = \{\langle\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \mathbf{x}, \mathbf{q}, \mathbf{t}, \mathbf{O}\rangle\}$, in which $\boldsymbol{\theta}$ are the joints' angles, $\dot{\boldsymbol{\theta}}$ the joints' velocities, $\mathbf{x}$, $\mathbf{q}$ is the pose (position and orientation) of the end effector, $\mathbf{t}$ is the position of the target, and $\mathbf{O}$ are the positions of the obstacles, $\mathcal{A} = \{\boldsymbol{\tau}\}$, in which $\boldsymbol{\tau}$ is the torque control for the joints, T is an unknown transition function, and R is set to reach the target goal (more details in the following).

The abstraction considered in this article for the robot control problem illustrated above is given by a deterministic discrete MDP $\bar{\mathcal{M}} = \langle\bar{S}, \bar{\mathcal{A}}, \bar{T}, \bar{R}\rangle$, defined as follows. The mapping function ϕ maps the positions of the end effector and of the target in a discrete space: i.e., $\phi(\langle\ldots, \mathbf{x}, \mathbf{t}, \ldots\rangle) = \langle\bar{\mathbf{x}}, \bar{\mathbf{t}}\rangle$, where $\bar{\mathbf{x}}$ is the discrete position of the end-effector (i.e., a discrete value for $\mathbf{x}$) and $\bar{\mathbf{t}}$ is the discrete position of the target (i.e., a discrete value for $\mathbf{t}$). Consequently, $\bar{S} = \{\langle\bar{\mathbf{x}}, \bar{\mathbf{t}}\rangle\}$. $\bar{\mathcal{A}}$ is the set of actions connecting discrete states $\bar{\mathbf{x}}$: in a 2D environment $\bar{\mathcal{A}}$ is a set of 4 actions, representing the 4 connections between adjacent cells in the discrete 2D space of the positions of the end-effector.

$\bar{T}$ is a deterministic transition function describing the discrete position movements of the end effector between two cells of the discrete space. $\bar{T}(\langle\bar{\mathbf{x}}, \bar{\mathbf{t}}\rangle, \bar{a}) = \langle\bar{\mathbf{x}}', \bar{\mathbf{t}}\rangle$ denotes the transition between discrete end-effector positions $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$ through the abstract action $\bar{a}$. We refer to $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$ as *adjacent* cells. If a cell contains an obstacle, then all the transitions to that cell are disabled (i.e., no state–action pair leads to that cell). The obstacle cells are called *unreachable* and all the cells that can be reached from the initial state are defined as *reachable*. Finally, $\bar{R}$ is a reward function to reach the target in the discrete space: $\bar{R}(\langle\bar{\mathbf{x}}, \bar{\mathbf{t}}\rangle) = \mathbb{I}(\bar{\mathbf{x}} = \bar{\mathbf{t}})$.

Notice that the abstract MDP $\bar{\mathcal{M}}$ can be built from the knowledge of the initial state of the original MDP $\mathcal{M}$. Indeed, since the target goal and the obstacles are static, the abstract transition function can be computed with the knowledge of the target and obstacle positions that are known at the start of each episode.

Given the full knowledge of the abstract MDP $\bar{\mathcal{M}}$, the value function $\bar{V}$ is computed using a value iteration algorithm. In practice, $\bar{V}$ can be computed when receiving the first observation of the concrete MDP $\mathcal{M}$. This abstraction allows thus to extend the scope of Algorithm 1 presented in this article to cases in which the abstract MDP $\bar{\mathcal{M}}$ becomes fully known and can thus be solved with planning techniques, thus implementing an interesting and effective integration of planning and learning techniques for robot control.

With the construction of the abstract MDP described above, we can adapt Algorithm 1 described in the previous section to Algorithm 2 tailored to the considered robot control problems. In particular, the following changes have been applied: (1) we use only two layers of abstraction; (2) the `Shaping` function is extended to be a general reward heuristic function `Heuristic`; (3) with the goal of reaching the target, it is not necessary to explicitly consider the biased MDP; (4) the learning algorithm $\mathcal{L}$ used in Algorithm 1 to compute the solution of the abstract MDP is generalized with any method able to solve the abstract MDP, including planning based on the known abstract transition function $\bar{T}$; (5) a Deep Reinforcement Learning (DRL) approach is used to solve the continuous MDP.

It is important to notice that the abstraction described here does not require an abstract simulator to learn the abstract policy (or value function). Observe also that the theoretical result stated in Theorem 1 also applies to the abstraction described in this section, and it establishes that, under the specified hypotheses and when $\mathcal{L}$ computes an optimal policy (with respect to the biased MDP), the policy $\hat{\rho}^*$ returned by Algorithm 2 is near-optimal.

In practice, in the considered robot control problems (modeled as Goal MDPs), the main challenge is to compute a policy achieving the goal with high sample efficiency, rather than finding an optimal one. To this end, the experiments reported in the next sections measure the capability of the proposed method to efficiently learn a goal-reaching policy and compare different approaches measuring the number of training steps needed to find one such policy.

Algorithm 2: Robot control variant of Algorithm 1

Input: MDP discrete planning/learning algorithm $\mathcal{P}$
Input: MDP continuous DRL algorithm $\mathcal{L}$
Input: $\mathcal{M}, \bar{\mathcal{M}}, \phi$
Output: $\hat{\rho}^*$ ground MDP goal policy

```
V̄* ← P(M̄) ;
F ← Heuristic(γ, φ, V̄*);
Learner ← L(M);
while not Learner.Stop() do
    s ← M.State();
    a ← Learner.Action(s);
    r, s' ← M.Act(a);
    r_h ← r + F(s, a, s');
    Learner.Update(s, a, r_h, s');
end
ρ̂* ← Learner.Output();
```

In the experimental results reported in the next section, we observe that an on-policy DRL algorithm converges with high sample efficiency to a goal-reaching policy. This observation provides the interesting intuition that the loss function induced by the proposed method is convex. Indeed, we know that by construction $\bar{V}^*$ has the following properties: (1) there exists exactly one abstract state $\bar{g} \in \bar{S}$ that contains goal states of the concrete MDP and for which $\bar{V}^*(\bar{g})$ is maximum; (2) for every other reachable state $\bar{s} \in \bar{S}, \bar{s} \neq \bar{g}$, there exists a reachable adjacent state $\bar{s}'$ such that $\bar{V}^*(s') > \bar{V}^*(s)$. Moreover, we consider and compare increasing monotonic reward heuristic functions as `Heuristic` in Algorithm 2. We believe that the relationship among these findings deserves future investigation.

8. Validation

The validation of the proposed method has been carried out in two use cases: one in which we assume the availability of multiple simulators at different levels of abstraction and use the Hierarchical Reinforcement Learning approach described in Algorithm 1 (Section 8.1), the other in which we build an abstraction for robot control as described in Section 7 and integrate planning and learning as described in Algorithm 2 (Section 8.2).

8.1. Hierarchical RL in a robot navigation scenario

We initially consider a navigation scenario where some locations in a map are selected as goal states. In this scenario, we are interested in solving the control problem for a mobile robot controlled by velocity inputs, moving in the 2D (concrete) environment, while using an abstraction where the robot is controlled in position.

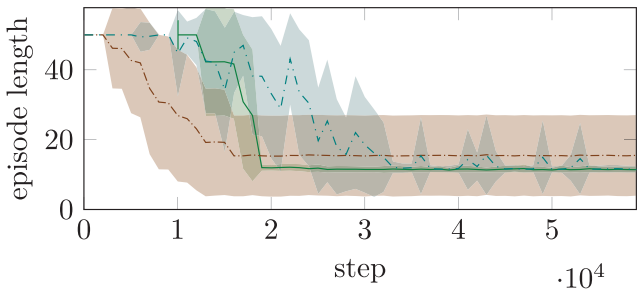
Environments. We start with two levels of abstraction, $\mathcal{M}_1$ and $\mathcal{M}_2$. The ground MDP $\mathcal{M}_1$ consists of a finite state space S_1, containing a set of locations, and a finite set of actions $\mathcal{A}_1$ that allow to move among neighboring states, with some small failure probability. Following the idea of Fig. 1, we also define an abstract MDP $\mathcal{M}_2$, whose states correspond to contiguous regions of S_1. Actions $\mathcal{A}_2$ allow to move, with high probability, from one region to the other only if there is a direct connection between the two in $\mathcal{M}_1$. We instantiate this idea in two domains. In the first, we consider a map as the one in the classic 4-rooms environment from [9]. The second is the "8-rooms" environment shown in Fig. 1.[1]

Training results. In the plots of Figs. 2(a) and 2(b), for each of the two ground MDPs, we compare the performance of the following algorithms:

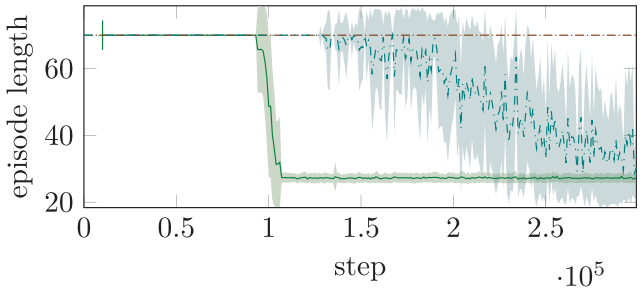
- Q-learning [29];
- Delayed Q-learning [30];
- Algorithm 1 (our approach) with Q-learning.

Each episode is terminated after a fixed timeout or when the agent reaches a goal state. Therefore, shorter episode lengths are associated with higher cumulative returns. The horizontal axis spans the number of sampled transitions. The plots report average value and standard deviation of the evaluations of 10 different runs. The solid green line of our approach is shifted to the right, in order to account for the number of time steps that were spent in training the abstraction. As we can see in Fig. 2(a), all algorithms converge relatively easily in the smaller 4-rooms domain. In Fig. 2(b), as the state space increases and becomes more difficult to explore, a naïve exploration policy does not allow Q-learning to converge in a reasonable time. Our agent, on the other hand, steadily converges to optimum, even faster than Delayed Q-learning, which has polynomial time guarantees.

[1] Code and reproducibility details are available at https://github.com/cipollone/multinav2.

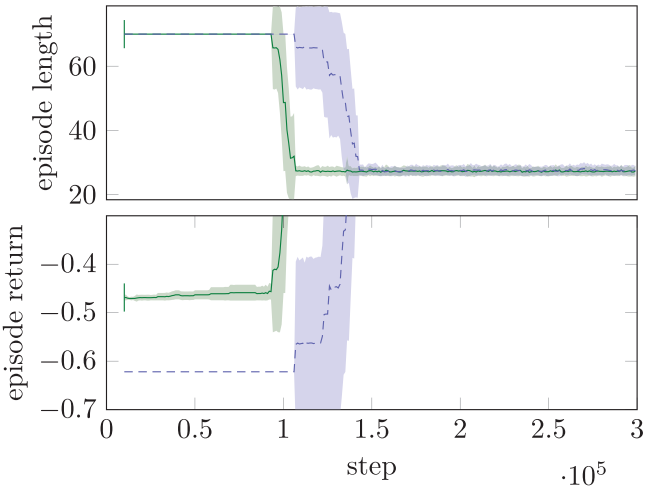


(a) Navigation task in the 4-rooms domain.

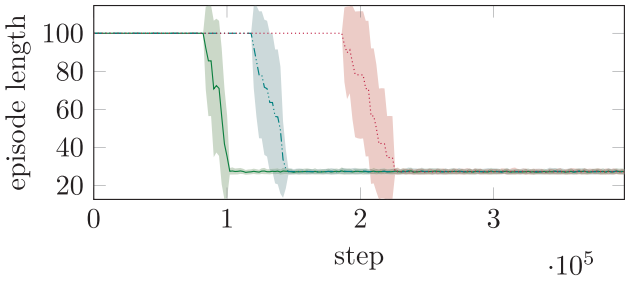


(b) Navigation task in the 8-rooms domain.

Fig. 2. Results on the navigation tasks.



(a) Return-invariant RS and our approach.



(b) Training in presence of errors.

Fig. 3. Results on the navigation tasks.

Return-invariant shaping. As discussed in Section 5, when applying RS in the episodic setting, there is a technical but delicate distinction to make between:

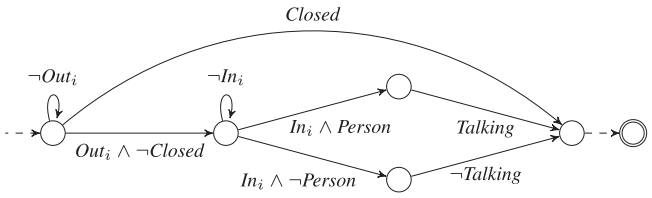


Fig. 4. A temporally-extended task, repeated for $i = 1, 2$. The missing transitions go to a failure sink state.

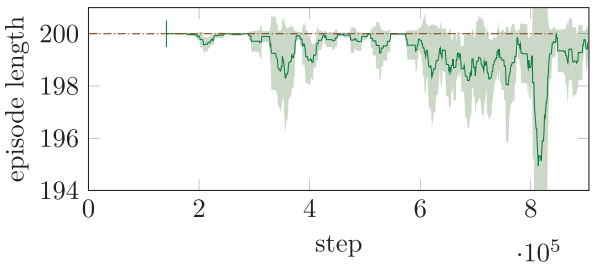


Fig. 5. Dueling DQN algorithm with and without our RS. Training episode lengths, averaged over 5 runs.

- Return-invariant RS (null potentials at terminal states);
- Non return-invariant RS (our approach).

In Fig. 3(a) (top), we compare the two variants on the 8-rooms domain.

Although both agents receive RS from the same potential, the minor modification in the invariance of RS suffices to produce a noticeable difference. The reason lies in the returns the two agents observe (bottom). Although they are incomparable in magnitude, in the early learning phase, we see that only our RS variant is able to reward each episode differently, depending on the estimated distance to the goal.

Robustness to modeling errors. We also considered the effect of significant modeling errors in the abstraction. In Fig. 3(b), we report the performance of our agent on the 8-rooms domain, when driven by three different abstractions:

- $\mathcal{M}_2$: is the same abstraction used in Fig. 2(b);
- $\mathcal{M}_2^{(b)}$: is $\mathcal{M}_2$ with an additional transition from the pink states (p) to the goal (G), not achievable in $\mathcal{M}_1$.
- $\mathcal{M}_2^{(c)}$: is $\mathcal{M}_2^{(b)}$ with an additional transition from the blue (b) to the pink region (p), not achievable in $\mathcal{M}_1$.

Clearly, abstractions with larger differences with respect to the underlying domain slow down the learning process. However, with any of these, Q-learning converges to the desired policy and performances degrade gracefully. Interestingly, even in the presence of severe modeling errors, the abstraction still provides useful information with respect to uninformed exploration.

Interaction task. In this section, we show that the proposed method applies to a wide range of algorithms, dynamics and tasks. With respect to variability in tasks, we emphasize that goal MDPs can capture many interesting problems. For this purpose, instead of reaching a location, we consider a complex temporally-extended behavior such as: "reach the entrance of the two rooms in sequence and, if each door is open, enter and interact with the person inside, if present". This task is summarized by the deterministic automaton $\mathcal{D}$ of Fig. 4. Note that there is a single accepting state, and arcs are associated to environment events.

Regarding the environment dynamics, instead, we define $\mathcal{M}_{2,d}$ and $\mathcal{M}_{1,d}$, respectively the abstract and grid transition dynamics seen so far. In addition, we consider a ground MDP $\mathcal{M}_{0,d}$ at which the robot movements are modeled using continuous features. The state space $\mathcal{S}_0$ now contains continuous vectors (x, y, θ, v), representing pose and velocity of agent's mobile base on the plane. The discrete set of actions $\mathcal{A}_0$ allows to accelerate, decelerate, rotate, and a special action denotes the initiation of an interaction.

There exists goal MDPs, $\mathcal{M}_2, \mathcal{M}_1, \mathcal{M}_0$ that capture both the dynamics and the task defined above, which can be obtained through a suitable composition of each $\mathcal{M}_{i,d}$ and $\mathcal{D}$ [1,2]. Therefore, we can still apply our technique to the composed goal MDP. Since $\mathcal{M}_0$ now includes continuous features we adopt Dueling DQN [31], a Deep RL algorithm. The plot in Fig. 5 shows a training comparison between the Dueling DQN agent alone (dot-dashed brown), and Dueling DQN receiving rewards from the grid abstraction (green). As we can see, our method allows to provide useful exploration bias even in case of extremely sparse goal states, as in this case.

8.2. Integration of planning and learning for multi-joint robot control

The second set of experiments was designed to evaluate performance in multi-joint robot control. We defined three environments (see Figs. 11, 12, 13) in which robots with different degrees of freedom (2, 3, and 6 respectively) are controlled with force/torque inputs to reach a target position (the red dot) avoiding obstacles (the cyan cylinders). The concrete MDPs for robot control are implemented as Mujoco environments[2] modeling the domains described above. The state and action spaces as described in Section 7. The transition function is implemented by the Mujoco simulator, while the basic reward function returns a positive reward 1 when the goal is reached within a predefined tolerance δ, i.e., $\|\mathbf{x} - \mathbf{t}\| < \delta$ (with $\mathbf{x}$ the end-effector's and $\mathbf{t}$ the desired position) and a negative reward -1 when an obstacle is hit.

The next subsections report two groups of experiments: (i) comparison between DRL approaches and model-based optimization approaches using the knowledge of the model in simple environments; (ii) comparison among different reward heuristic functions applied to DRL in more complex environments.

The first group aims to compare the quality of the solutions computed by DRL with trajectory optimization, an optimal method that takes advantage of full knowledge of the environment. As expected, the solutions computed by the trajectory optimization method reach the goal much faster than those obtained with DRL, and are also optimal in terms of energy consumption (which is minimized), while DRL solutions are slower and produce suboptimal trajectories wrt energy consumption. However, trajectory optimization scales very poorly when the complexity of the scenario grows (in particular, for increasing degrees of freedom and number of obstacles) and requires, in some cases, a difficult parameter tuning; on the other hand, DRL approaches scale much better to complex situations and are less sensitive to hyperparameters, although the number of training steps to converge is highly affected by the exploration strategy. Since, for more complex scenarios, trajectory optimization was unable to return a solution within the time limit, the second group of experiments has been performed only using DRL, with up to 6 DOF and 3 obstacles.

8.2.1. Experiment setup

We adopted Stable Baselines 3 (version 2.3.2) for the algorithm implementations (mainly PPO) and Gymnasium (version 0.29.1) for the environment definition. Most of the parameters of PPO are the default ones. The most important ones or those that were changed from the default settings are listed below:

- Learning Rate: 1×10^{-3}
- Batch Size: 64
- γ (Discount Factor): 0.99
- λ (GAE parameter): 0.95

[2] Code and reproducibility details are available at https://github.com/neverorfrog/reacher-obstacles.

Table 1
Reacher3 — TO vs PPO with Reward Heuristics.

Env	RL mean error	TO mean error	RL energy	TO energy
1	0.11	0.083	0.11	0.10
2	0.15	0.083	0.40	0.10
3	0.16	–	0.32	–
4	0.17	0.17	0.85	0.18
5	0.11	0.11	0.63	0.32

- Number of Epochs per Update: 10

Further details on the remaining hyperparameters can be found in the Stable Baselines 3 documentation for PPO.

8.2.2. Comparison with trajectory optimization

We compared DRL approaches against a fully model-based method, namely trajectory optimization (TO), on Reacher3 under 5 different configurations of initial state, goal state and obstacle position. Figs. 6–10 report the trajectory and the torque profile obtained with DRL and TO, for all the environments. Table 1 reports the corresponding mean error of the fingertip wrt the goal position, and the energy spent to reach the final position

In TO, the state space, the action space and the transition function are the same as in the Mujoco environment used for RL, but they are known and are instantiated using Pinocchio [32]. Moreover, there is no explicit concept of reward as in RL; instead, there must be an explicit definition of the cost-to-go (for every time instant of the trajectory) which needs to be minimized by an appropriate solver. We used CasADi [33] for that purpose. Specifically to this task, cost-to-go is proportional to the squared error between the fingertip and the goal positions, while we added constraints to the optimization problem to deal with obstacles. There are some obvious advantages in using RL instead of TO:

- Parameter tuning and programming in general is easier with RL. Tuning the weights of the cost function is a tedious job, while defining a reward heuristic can be less painful.
- RL is not limited by some numerical solver, thus more flexible. Sometimes, TO fails to find a solution at all.
- RL scales better, since it does the training offline and does not take into account the model in the online execution. TO, instead, solves the problem with the model at hand: if the model becomes too complicated, computational resources become a problem.

Nevertheless, having a model embedded directly into the optimization process also brings its advantages. Explainability in the first place, and also the possibility to explicitly add constraints (to enforce stability) in the optimization problem. Computational efficiency cannot be considered as an absolute parameter: TO takes significantly less to train (less than 1s vs 50s), while RL learns offline but takes less once the policy is deployed. Besides, other factors regarding the model and the environment should be considered to adequately assess computational performance.

One substantial difference between the two approaches is the produced torque profile, which is much smoother for TO, with a consequent reduction in the energy spent, as illustrated in Table 1. Energy was computed as the sum of torques: $E_\tau = \int_0^T \sum_i \tau_i^2 \, dt$. This behavior is particulary evident in environments 2 and 4, where TO chooses a completely different (and more energy-efficient) trajectory (see Figs. 7 and 9).

Notably, TO was not able, even with some tentative parameter tuning, to solve environment 3, while RL solved it successfully. More in general, RL seems to be much more robust with respect to the obstacle configuration, while TO is very sensitive to both the obstacles and the cost function weights.

We observe that both responsiveness and energy consumption are crucial aspects in robotic scenarios, a favoring factor for trajectory optimization over DRL. The main goal of this group of experiments was to measure the quality of the solutions obtained by our HRL approach, which uses minimal knowledge about the system dynamics approach, with respect to the optimal solutions obtained with other approaches that require full knowledge of the dynamics of the environment. An exhaustive comparison with other approximate control-based techniques is beyond the scope of this article. This work is positioned at an earlier stage, that is, exploring the potentiality of abstraction techniques in improving the efficiency of DRL-based solution approaches in solving robotics tasks.

8.2.3. Comparison among reward heuristics

The environments used for the DRL experiments are configured with episodes terminating after a maximum number of steps. Reaching the target goal or touching obstacles do not terminate the episode, thus the agent is rewarded to reach the goal state (while avoiding the obstacles) as soon as possible and to remain in the goal state until the end of the episode.

We use DRL algorithms based on policy gradient as solution methods. With the basic sparse reward of the Goal MDP, policy gradient methods generally fail to converge to a solution given the lack of guidance during training. To solve the problems, we compare four different ways of defining a dense reward, i.e., to implement the `Heuristic` function of Algorithm 2: a baseline method just using the distance to the goal, a reward shaping mechanism, and two reward heuristic functions, whose details are described below.

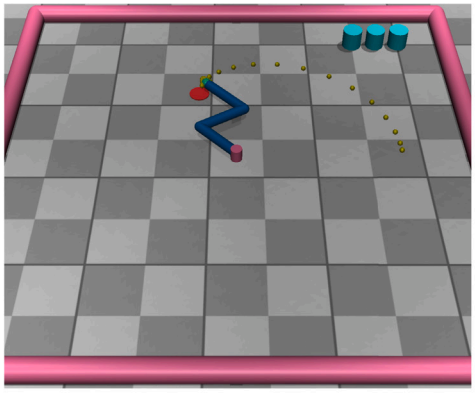
1. **Baseline**. Reward heuristic based on distance to the goal: $F(s,a,s') = \|\mathbf{x}' - \mathbf{t}\|$, with $\mathbf{x}'$ being the end-effector position in s'.
2. **RS**. Reward shaping function: $F(s,a,s') = \gamma \bar{V}^*(\phi(s')) - \bar{V}^*(\phi(s))$.
3. **RHV**. Reward heuristic based on $\bar{V}^*$: $F(s,a,s') = \bar{V}^*(\phi(s'))$.
4. **RHd**. Reward heuristic based on distance map computed from the abstract transition function $\bar{T} \in \bar{\mathcal{M}}$, $F(s,a,s') = d_{MAP}(\phi(s'), \phi(g))$.

To compare the different approaches, we defined the following *episode success* criterion: an episode is successful when the robot remains in the goal state ($\|\mathbf{x} - \mathbf{t}\| < \delta$) for at least all the last $T/10$ steps, with T being the maximum number of steps allowed for the episode. During training, we periodically test the current policy (disabling exploration) and stop the training process when such a policy determines an *episode success*. In this way, we can measure the number of training steps needed to compute a policy that achieves episode success. To account for the variability of DRL algorithms, we repeated each experiment many times with different random seeds for action exploration of DRL (at least 10 in all the experimental results provided in this section) and computed a global score as follows. Given an environment configuration c for a goal MDP $\mathcal{M}$, a DRL algorithm $\mathcal{L}$, a maximum number of training steps ts_{max}, and a number of repetitions n_r, the overall score of training the environment c with $\mathcal{L}$ is computed with these steps: (1) train environment c for n_r runs using different random seeds; (2) collect $success\_rate$ as the percentage of runs in which a successful policy is found within the maximum training steps ts_{max}; (3) collect ts_{avg} and ts_{std} as, respectively, average and standard deviation of the number of training steps needed to compute a successful policy for all the successful runs; (4) compute the *sample efficiency score* as $eff\_score = (ts_{max} - ts_{avg})/ts_{max}$; (5) finally compute the overall score as

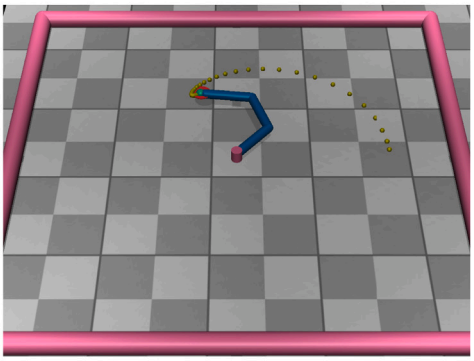
$$overall\_score = \frac{2 \cdot success\_rate \cdot eff\_score}{success\_rate + eff\_score}$$

Notice that the metrics $success\_rate$ and eff_score and $overall\_score$ are normalized in $[0,1]$, with 1 being the desired best performance.

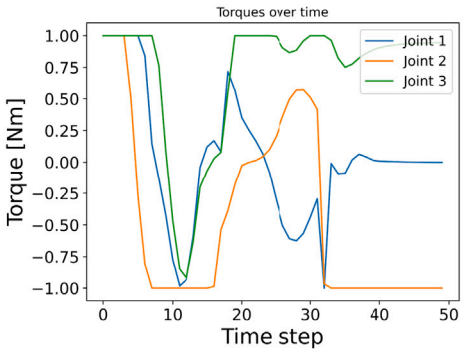
Three scenarios are used for the evaluation in which different kinds of robots have to reach a specific goal area in the Cartesian space.



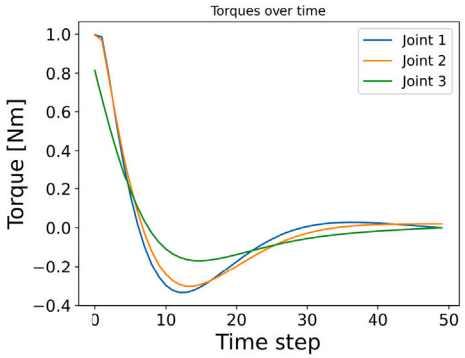
(a) RL Trajectory



(b) TO Trajectory

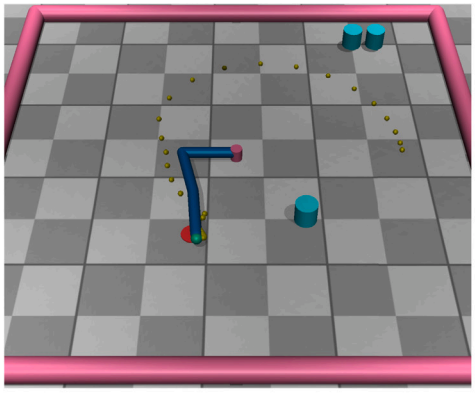


(c) RL Torque History

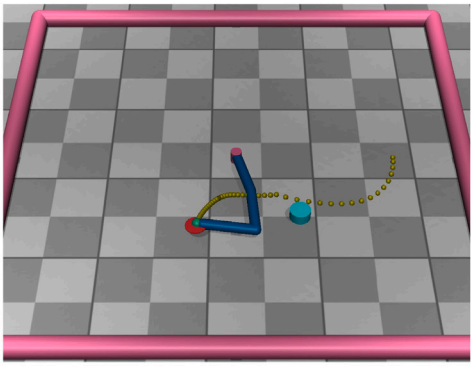


(d) TO Torque History

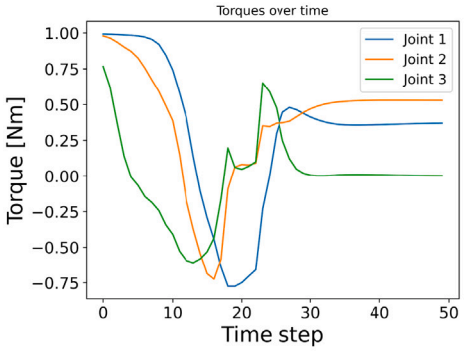
Fig. 6. Comparison of RL and TO on environment 1.



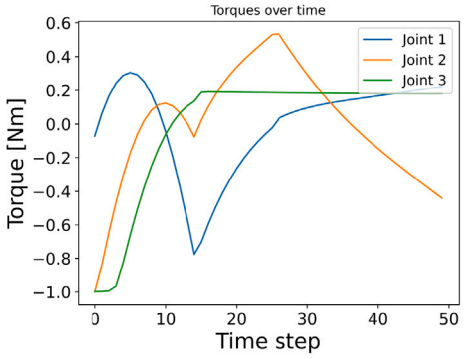
(a) RL Trajectory



(b) TO Trajectory

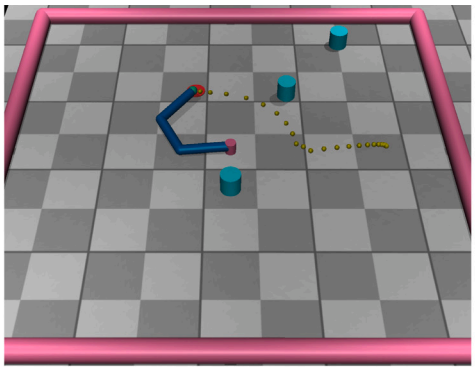


(c) RL Torque History

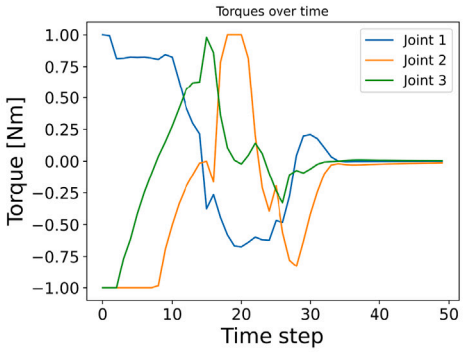


(d) TO Torque History

Fig. 7. Comparison of RL and TO on environment 2.

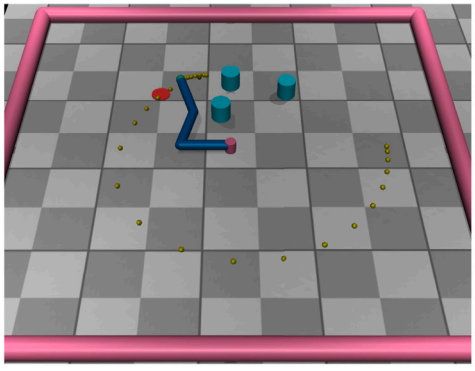


(a) RL Trajectory

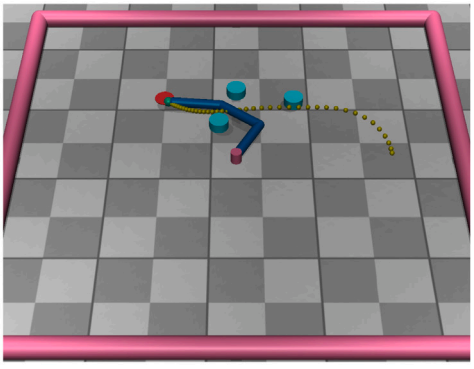


(b) RL Torques

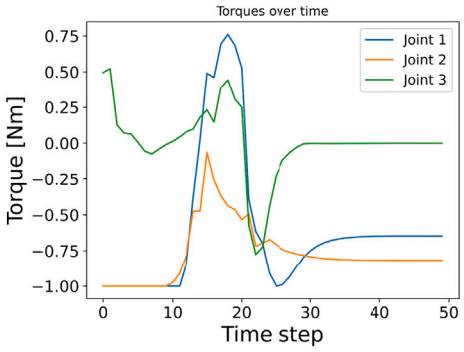
Fig. 8. RL Trajectory on environment 3.



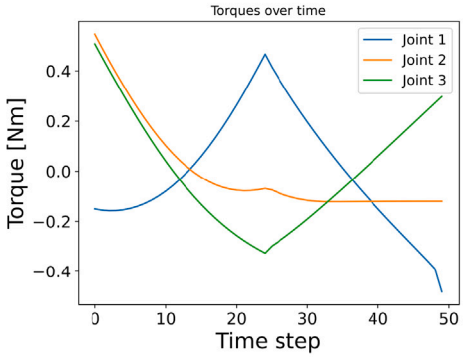
(a) RL Trajectory



(b) TO Trajectory



(c) RL Torque History



(d) TO Torque History

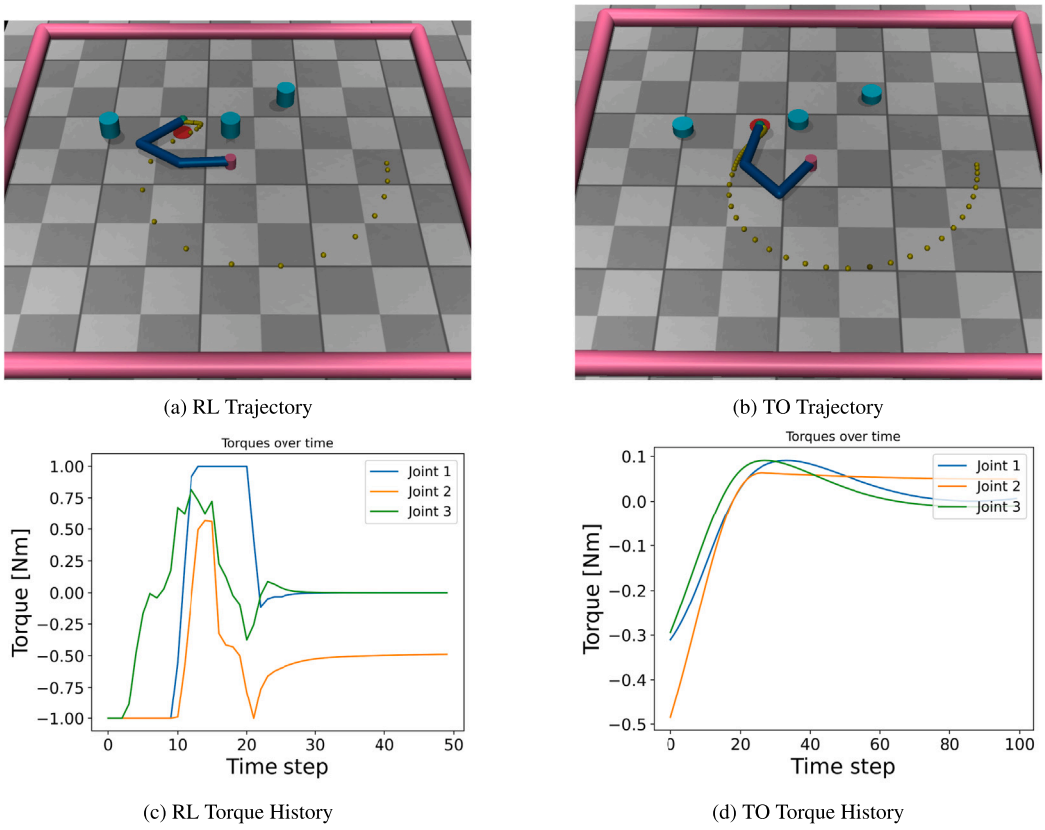
Fig. 9. Comparison of RL and TO on environment 4.

BallReacher is an environment in which a 2 dof ball is controlled with 2D force inputs to reach a target goal. Reacher3 is an environment in which a 3 dof planar robot arm is controlled with torque inputs to reach with its end-effector a target goal. MARRtinoArm6 in which a 6 dof robot arm is controlled with torque inputs to reach with its end-effector a target goal. While in the first two environments, the ball and the end-effector make 2D planar movements, in the third environment the end-effector moves in the 3D space. For each environment, we defined a set of configurations with different obstacles (10 for BallReancher, 5 for Reacher3, and 5 for MARRtinoArm6). The 20 configurations used in the experiments are shown in Figs. 11, 12, 13 with the red circle indicating the target goal and the blue cylinders used as obstacles. Every configuration has been tested with the four reward heuristics described above and for 10 different random seeds. The evaluation benchmark is thus composed of a total of 800 runs of experiments.

Tables 2, 3 4, 5 6, 7 show the comparative results of the four reward mechanisms in the 20 configurations of the three environments, reporting the described performance metrics: $success\_rate$, ts_{avg}, ts_{std}, and $overall\_score$.

As shown in the tables, the **Baseline** approach has the lowest success rates and the worst sample efficiency (highest training samples to reach the goal). It is clear that, in scenarios with obstacles within the robot and the target (such as most of the configurations in BallReacher), the reward heuristic of the **Baseline** approach induces a loss function for the DRL algorithm with local maxima corresponding to non-goal states. While, the other reward mechanisms have significantly higher success rate showing that the proposed approach actually allows driving the DRL algorithm towards a goal-reaching policy.

The reward shaping function **RS** exhibits generally worse performance than the reward heuristics. This is explained by the fact that the potential function (obtained from a discrete function) used for reward



(a) RL Trajectory

(b) TO Trajectory

(c) RL Torque History

(d) TO Torque History

Fig. 10. Comparison of RL and TO on environment 5.

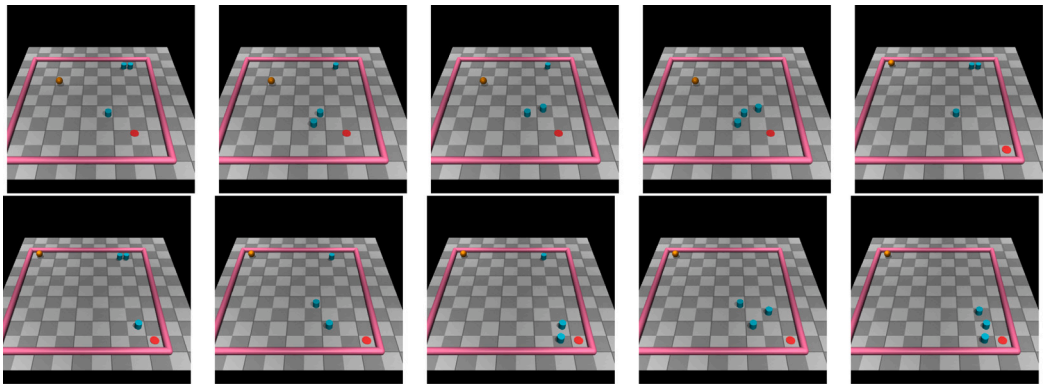


Fig. 11. BallReacher environment: ten configurations tested: initial state.

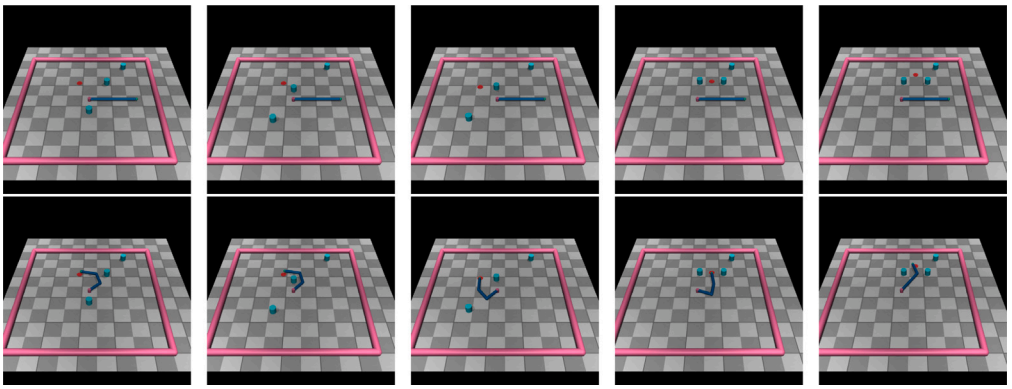


Fig. 12. Reacher3 environment: five configurations tested: (top) initial state, (bottom) reached state.

Table 2
BallReacher: results with PPO $\gamma = 0.9, \lambda = 0.001$ in form $success\_rate$, $ts_{avg}(ts_{std})$.

Env	Baseline	RS	RHV	RHd
FTaO1a	0.20, 378.9 (32.8)	1.00, 156.9 (72.1)	1.00, 116.3 (59.4)	1.00, 56.5 (19.6)
FTaO2a	0.00, 0.0 (0.0)	0.90, 127.2 (52.0)	1.00, 168.8 (96.4)	1.00, 155.2 (59.4)
FTaO2b	0.10, 460.8 (0.0)	1.00, 194.2 (62.5)	1.00, 123.1 (51.5)	1.00, 78.4 (37.5)
FTaO3a	0.00, 0.0 (0.0)	0.60, 184.3 (42.5)	1.00, 177.6 (60.2)	0.90, 174.1 (73.8)
FTbO1a	0.00, 0.0 (0.0)	0.90, 259.2 (121.6)	0.50, 313.8 (132.9)	1.00, 63.7 (22.5)
FTbO1b	0.00, 0.0 (0.0)	1.00, 349.4 (63.8)	1.00, 255.0 (71.7)	0.90, 150.2 (68.3)
FTbO2a	0.10, 315.4 (0.0)	0.90, 250.3 (65.9)	0.80, 205.8 (59.1)	1.00, 67.4 (18.6)
FTbO2b	0.00, 0.0 (0.0)	1.00, 412.5 (65.0)	1.00, 335.5 (74.8)	0.90, 204.3 (117.2)
FTbO3a	0.00, 0.0 (0.0)	0.90, 311.3 (59.0)	0.90, 271.0 (67.0)	1.00, 300.0 (115.6)
FTbO3b	0.00, 0.0 (0.0)	0.90, 377.3 (55.1)	0.90, 334.3 (71.5)	0.80, 240.9 (112.1)

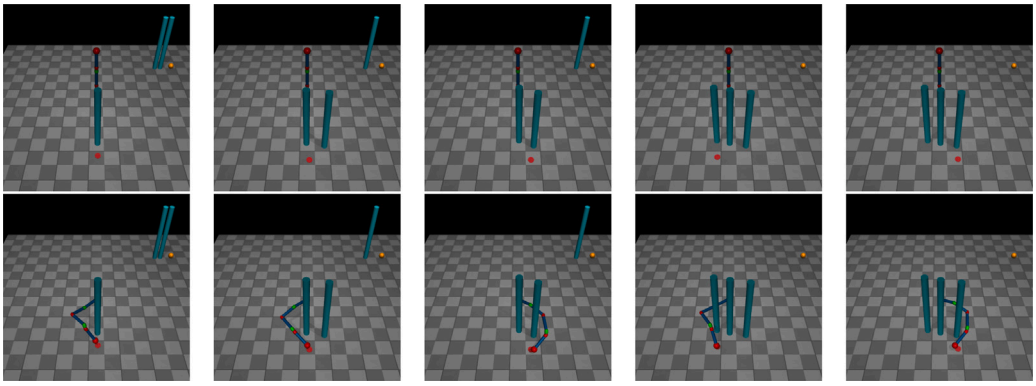


Fig. 13. MARRtinoArm6 environment: five configurations tested: (top) initial state, (bottom) reached state.

Table 3
BallReacher: overall score with PPO $\gamma = 0.9, \lambda = 0.001$.

Env	Baseline	RS	RHV	RHd
FTaO1a	0.219	0.814	0.868	0.940
FTaO2a	0.000	0.816	0.797	0.816
FTaO2b	0.088	0.759	0.860	0.915
FTaO3a	0.000	0.615	0.784	0.756
FTbO1a	0.000	0.627	0.427	0.932
FTbO1b	0.000	0.463	0.658	0.787
FTbO2a	0.157	0.642	0.678	0.928
FTbO2b	0.000	0.298	0.495	0.714
FTbO3a	0.000	0.532	0.607	0.571
FTbO3b	0.000	0.386	0.484	0.629

shaping has many flat regions, thus the additional reward signal is non-zero only in the transitions between abstract states.

Between the reward heuristic functions **RHV** and **RHd** we can observe a general trend of **RHV** having higher success rate and **RHd** having higher sample efficiency. Overall, considering all 800 runs of experiments and using the *overall_score* metric, we can state that the approach based on the integration of planning and learning generating the **RHd** reward heuristic generally performs better than other reward heuristics and reward shaping, and it performs significantly better than a baseline dense approach that uses only the distance to the target goal.

9. Discussion and conclusion

In this work, we have presented an approach to increase sample efficiency of RL algorithms, based on a linear hierarchy of abstract simulators and new forms of reward shaping and reward heuristics. In the linear hierarchy, the ground MDP accurately captures the environment dynamics, while higher-level models represent increasingly coarser abstractions of it. We have described the properties of our method under different abstractions and have shown its effectiveness in practice. Importantly, our approach is very general, as it makes no assumptions on the RL algorithm used and has minimal requirements in terms of mapping between the abstraction layers.

The proposed solution also has some limitations that will drive future research in the field. The main limitation of our general approach is obviously the need for a manual construction of the abstraction function. This requires the designer to have some understanding of both the environment and the task, in order to guarantee that the proposed abstraction is meaningful for the task. If this is not the case, the approach can even be detrimental to sample efficiency; indeed, the learned higher-level policy might induce an exploration heuristic that misleads the lower-level search, ultimately requiring a higher number of samples than those required by the search without heuristic, to learn the same (or comparable) low-level policy.

Similarly, even when the designer has a good understanding of the task and the abstraction function is meaningful, the design choices made (e.g., modeling a too coarse abstraction of the environment) might be ineffective in improving sample efficiency, ultimately slowing down the overall learning process, as requiring an additional learning phase (the one carried out in the higher-level abstraction).

In general, the effectiveness of the approach depends on the experience of the designer. Although we believe that in many scenarios, such as those arising in robotics contexts, "natural" abstractions exist that make our approach beneficial to the learning process (such as modeling physical spaces as grids), and that the designer is likely to use such abstractions, this is not true in general, and even when it is, the effectiveness of the abstractions still depends on a number of parameters that require to be carefully chosen (e.g., number of cells in the grid). Although this work provides a theoretical framework for measuring the quality of abstractions, the definitions provided are of a semantical nature, and thus do not help to obtain an automated procedure for building the abstractions.

Table 4
Reacher3: results with PPO $\gamma = 0.9, \lambda = 0.001$ in form $success\_rate$, $ts_{avg}(ts_{std})$.

Env	Baseline	RS	RHV	RHd
FTO2a	0.50, 299.8 (68.1)	0.40, 141.3 (107.1)	1.00, 231.6 (120.4)	1.00, 156.3 (35.5)
FTO2b	0.70, 187.0 (84.7)	1.00, 125.7 (36.9)	1.00, 82.5 (18.2)	1.00, 44.6 (5.8)
FTO2c	0.00, 0.0 (0.0)	0.60, 74.8 (16.7)	0.80, 70.1 (14.5)	1.00, 53.7 (14.9)
FTO2d	0.90, 142.0 (64.0)	0.80, 73.0 (8.3)	1.00, 62.9 (11.1)	1.00, 60.2 (11.3)
FTO2e	0.20, 465.9 (5.1)	0.50, 362.1 (157.6)	1.00, 256.8 (90.4)	0.80, 233.2 (70.0)

Table 5
Reacher3: overall score with PPO $\gamma = 0.9, \lambda = 0.001$.

Env	Baseline	RS	RHV	RHd
FTO2a	0.445	0.514	0.699	0.815
FTO2b	0.661	0.856	0.910	0.953
FTO2c	0.000	0.704	0.829	0.943
FTO2d	0.798	0.826	0.933	0.936
FTO2e	0.102	0.356	0.654	0.640

Table 6
MARRtinoArm6: results with PPO $\gamma = 0.99, \lambda = 0.001$ in form $success\_rate$, $ts_{avg}(ts_{std})$.

Env	Baseline	RS	RHV	RHd
TO1a	0.50, 615.2 (217.1)	0.00, 0.0 (0.0)	0.70, 295.2 (204.6)	0.60, 286.0 (155.1)
TO2a	0.10, 450.6 (0.0)	0.10, 120.8 (0.0)	0.40, 391.2 (319.9)	0.40, 303.6 (64.3)
TO2b	0.30, 504.5 (64.0)	0.20, 541.7 (369.7)	0.60, 424.6 (307.9)	0.80, 381.2 (188.1)
TO3a	0.10, 251.9 (0.0)	0.00, 0.0 (0.0)	0.50, 469.0 (353.6)	0.70, 287.9 (63.5)
TO3b	0.10, 880.6 (0.0)	0.10, 180.2 (0.0)	0.60, 402.8 (306.4)	0.80, 467.2 (251.7)

Table 7
MARRtinoArm6: overall score with PPO $\gamma = 0.99, \lambda = 0.001$.

Env	Baseline	RS	RHV	RHd
TO1a	0.435	0.000	0.702	0.652
TO2a	0.169	0.180	0.483	0.508
TO2b	0.374	0.278	0.587	0.698
TO3a	0.176	0.000	0.515	0.706
TO3b	0.109	0.178	0.599	0.640

The above mentioned limitations inspire future work to further study the properties of abstractions in Hierarchical RL. Moreover, a benchmark of challenging robot control environments can be tailored to compare HRL techniques. Finally, we plan to study and evaluate further ways of integrating planning and RL exploiting hierarchical models and to evaluate them in other robotic application scenarios.

CRediT authorship contribution statement

Roberto Cipollone: Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Marco Favorito:** Writing – review & editing, Writing – original draft, Validation, Investigation, Formal analysis, Conceptualization. **Flavio Maiorana:** Validation, Software. **Giuseppe De Giacomo:** Writing – review & editing, Writing – original draft, Validation, Supervision, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Luca Iocchi:** Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Funding acquisition, Formal analysis, Conceptualization. **Fabio Patrizi:** Writing – review & editing, Writing – original draft, Supervision, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Giuseppe De Giacomo reports article publishing charges, equipment, drugs, or supplies, and travel were provided by Italian Ministry of Research and Education. Fabio Patrizi reports article publishing charges, equipment, drugs, or supplies, and travel were provided by Italian Ministry of Research and Education. Luca Iocchi reports financial support, article publishing charges, equipment, drugs, or supplies, and travel were provided by Italian Ministry of Research and Education. co-Guest Editor for the Special Issue to which this manuscript is being submitted - F.P. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by the PRIN project RIPER (No. 20203FFYLK), the PNRR MUR project PE0000013-FAIR, and the Sapienza project MARLeN.

Data availability

Links to data/code are provided in the paper.

References

[1] R.I. Brafman, G. De Giacomo, F. Patrizi, LTLf / LDLf non-Markovian rewards, in: AAAI 2018, 2018, pp. 1771–1778.
[2] R.T. Icarte, T. Klassen, R. Valenzano, S. McIlraith, Using reward machines for high-level task specification and decomposition in reinforcement learning, in: ICML, vol. 80, PMLR, 2018, pp. 2107–2116.
[3] M. Hutsebaut-Buysse, K. Mets, S. Latré, Hierarchical reinforcement learning: A survey and open research challenges, Mach. Learn. Knowl. Extr. 4 (1) (2022) 172–221, http://dx.doi.org/10.3390/make4010009.
[4] L. Li, T.J. Walsh, M.L. Littman, Towards a unified theory of state abstraction for MDPs, in: ISAIM 2006, 2006, pp. 531–539.
[5] T.G. Dietterich, Hierarchical reinforcement learning with the MAXQ value function decomposition, JAIR 13 (2000) 227–303.
[6] R. Parr, S. Russell, Reinforcement learning with hierarchies of machines, Adv. Neural Inf. Process. Syst. (1998) 1043–1049.
[7] B. Ravindran, A.G. Barto, Model minimization in hierarchical reinforcement learning, in: SARA 2002, vol. 2371, Springer, 2002, pp. 196–211, http://dx.doi.org/10.1007/3-540-45622-8_15.
[8] K. Jothimurugan, O. Bastani, R. Alur, Abstract value iteration for hierarchical reinforcement learning, in: AISTATS, vol. 130, PMLR, 2021, pp. 1162–1170.

[9] R.S. Sutton, D. Precup, S. Singh, Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning, Artificial Intelligence 112 (1–2) (1999) 181–211.
[10] D. Abel, D. Hershkowitz, M. Littman, Near optimal behavior via approximate state abstraction, in: ICML 2016, vol. 48, PMLR, 2016, pp. 2915–2923.
[11] D. Abel, N. Umbanhowar, K. Khetarpal, D. Arumugam, D. Precup, M. Littman, Value preserving state-action abstractions, in: AISTATS, vol. 108, PMLR, 2020, pp. 1639–1650.
[12] B. Marthi, Automatic shaping and decomposition of reward functions, in: ICML 2007, vol. 227, ACM, 2007, pp. 601–608, http://dx.doi.org/10.1145/1273496.1273572.
[13] Y. Gao, F. Toni, Potential based reward shaping for hierarchical reinforcement learning, in: IJCAI 2015, AAAI Press, 2015, pp. 3504–3510.
[14] I. Schubert, O.S. Oguz, M. Toussaint, Plan-based relaxed reward shaping for goal-directed tasks, in: ICLR 2021, OpenReview.net, 2021.
[15] E.A. Aguilar, L. Berducci, A. Brunnbauer, R. Grosu, D. Nickovic, From STL rulebooks to rewards, 2021, CoRR abs/2110.02792.
[16] R. Cipollone, G.D. Giacomo, M. Favorito, L. Iocchi, F. Patrizi, Exploiting multiple abstractions in episodic RL via reward shaping, in: AAAI, AAAI Press, 2023, pp. 7227–7234.
[17] G. Canonaco, L. Ardon, A. Pozanco, D. Borrajo, On the sample efficiency of abstractions and potential-based reward shaping in reinforcement learning, 2024, CoRR abs/2404.07826.
[18] M. Grzes, D. Kudenko, Multigrid reinforcement learning with reward shaping, in: ICANN 2008, Part I, vol. 5163, Springer, 2008, pp. 357–366.
[19] L. Steccanella, S. Totaro, A. Jonsson, Hierarchical representation learning for Markov decision processes, 2021, CoRR abs/2106.01655. arXiv:2106.01655.
[20] X. Yang, Z. Ji, J. Wu, Y.-K. Lai, C. Wei, G. Liu, R. Setchi, Hierarchical reinforcement learning with universal policies for multistep robotic manipulation, IEEE Trans. Neural Netw. Learn. Syst. 32 (11) (2021) 5174–5184.
[21] J. Sun, A. Curtis, Y. You, Y. Xu, M. Koehle, L. Guibas, S. Chitta, M. Schwager, H. Li, Hierarchical hybrid learning for long-horizon contact-rich robotic assembly, 2024, arXiv preprint arXiv:2409.16451.
[22] D. Azimi, R. Hoseinnezhad, Hierarchical reinforcement learning for quadrupedal robots: Efficient object manipulation in constrained environments, Sensors 25 (5) (2025) http://dx.doi.org/10.3390/s25051565, URL https://www.mdpi.com/1424-8220/25/5/1565.
[23] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley, 1994, http://dx.doi.org/10.1002/9780470316887.
[24] A.Y. Ng, D. Harada, S.J. Russell, Policy invariance under reward transformations: theory and application to reward shaping, in: ICML 1999, Morgan Kaufmann, 1999, pp. 278–287.
[25] E. Wiewiora, G.W. Cottrell, C. Elkan, Principled methods for advising reinforcement learning agents, in: ICML 2003, AAAI Press, 2003, pp. 792–799.
[26] S. Devlin, D. Kudenko, Dynamic potential-based reward shaping, in: AAMAS 2012, IFAAMAS, 2012, pp. 433–440.
[27] E. Wiewiora, Potential-based shaping and Q-value initialization are equivalent, J. Artificial Intelligence Res. 19 (2003) 205–208, http://dx.doi.org/10.1613/jair.1190.
[28] M. Grzes, Reward shaping in episodic reinforcement learning, in: AAMAS 2017, ACM, 2017, pp. 565–573.
[29] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.
[30] A.L. Strehl, L. Li, E. Wiewiora, J. Langford, M.L. Littman, PAC model-free reinforcement learning, in: ICML 2006, 2006, pp. 881–888.
[31] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, N. de Freitas, Dueling network architectures for deep reinforcement learning, in: ICML, vol. 48, JMLR.org, 2016, pp. 1995–2003.
[32] J. Carpentier, J. Mirabel, N. Mansard, the Pinocchio Development Team, Pinocchio: fast forward and inverse dynamics for poly-articulated systems, 2015–2018, https://stack-of-tasks.github.io/pinocchio.
[33] J.A.E. Andersson, J. Gillis, G. Horn, J.B. Rawlings, M. Diehl, CasADi – A software framework for nonlinear optimization and optimal control, Math. Program. Comput. 11 (1) (2019) 1–36, http://dx.doi.org/10.1007/s12532-018-0139-4.