Direct then Diffuse: Incremental Unsupervised Skill Discovery for State Covering and Goal Reaching

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learning meaningful behaviors in the absence of a task-specific reward function is a challenging problem in reinforcement learning. A desirable unsupervised objective is to learn a set of diverse skills that provide a thorough coverage of the state space while being directed, i.e., reliably reaching distinct regions of the environment. At test time, an agent could then leverage these skills to solve sparse reward problems by performing efficient exploration and finding an effective goal-directed policy with little-to-no additional learning. Unfortunately, it is challenging to learn skills with such properties, as diffusing (e.g., stochastic policies performing good coverage) skills are not reliable in targeting specific states, whereas directed (e.g., goal-based policies) skills provide limited coverage. In this paper, inspired by the mutual information framework, we propose a novel algorithm designed to maximize coverage while ensuring a constraint on the directedness of each skill. In particular, we design skills with a decoupled policy structure, with a first part trained to be directed and a second diffusing part that ensures local coverage. Furthermore, we leverage the directedness constraint to adaptively add or remove skills as well as incrementally compose them along a tree that is grown to achieve a thorough coverage of the environment. We illustrate how our learned skills enables to efficiently solve sparse-reward downstream tasks in navigation and continuous control environments, where it compares favorably with existing baselines.

1 Introduction

Deep reinforcement learning (RL) algorithms have been shown to effectively solve a wide variety of complex problems [e.g., 30, 6, 39, 16, 2, 36]. However, they are often designed to solve one single task at a time and they need to restart the learning process from scratch for any new problem, even when it is defined on the very same environment (e.g., navigating to different locations in the same apartment). Recently, unsupervised RL (URL) has been proposed as an approach to address this limitation. In URL, the agent first interacts with the environment without any extrinsic reward signal. Afterward, the agent leverages the experience accumulated during the unsupervised learning phase to efficiently solve a variety of downstream tasks defined on the same environment.

In this paper, we consider the URL setting where the agent starts from an initial state s_0 and it resets to it every time the policy terminates. We focus on sparse-reward downstream tasks, which require effective exploration (i.e., via a thorough coverage of the state space) to find the goal as well as learning a policy reliably reaching the goal (i.e., a directed policy).

We build on the insight that *mutual information* (MI) effectively formalizes the dual objective of learning skills that both cover and navigate the environment efficiently [e.g., 15]. Specifically, given the state variable S and some variables Z on which the skill policies are conditioned, MI is defined as

$$\mathcal{I}(S;Z) = \underbrace{\mathcal{H}(S)}_{\texttt{coverage}} \underbrace{- \mathcal{H}(S|Z)}_{\texttt{directedness}} = \mathcal{H}(Z) - \mathcal{H}(Z|S), \tag{1}$$

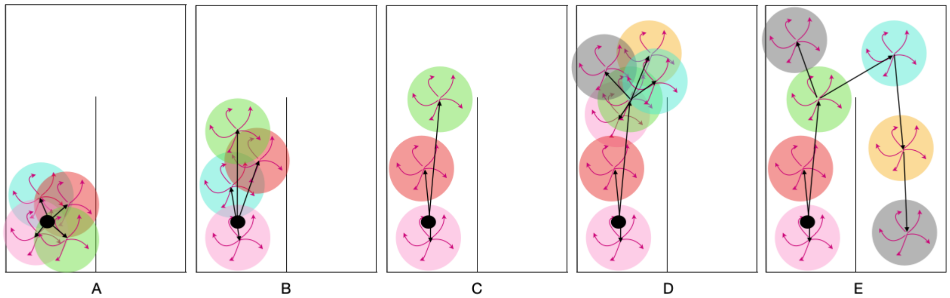


Figure 1: Overview of UPSIDE. The black dot corresponds to the initial state s_0. *(A)* A set of random skills is initialized, each skill being composed of a *directed* part (illustrated as a black arrow) and a *diffusing* part (red arrows), which induces a local coverage (colored circles). *(B)* The policies associated to the directed part of each skill are then updated to maximize the discriminability of the states reached by their diffusing part (Sect. 3.1). *(C)* The least discriminable skills are iteratively removed while the policies of the remaining skills are re-optimized. This is executed until the discriminability of each skill satisfies a given constraint (see Sect. 3.2). In this example three skills are kept. *(D)* One of these learned skill is then used as basis to add new skills, which are then optimized following the same procedure. For the "red" and "purple" skills, UPSIDE is not able to find sub-skills of sufficient quality and thus they are not expanded any further. *(E)* At the end of the process, UPSIDE has created a tree of directed skills covering the state space (Sect. 3.3). These covering skills can then be used to solve downstream tasks. Moreover, the discriminator learned together with the skills can be used to select the skill to reach any specific goal region, where the directed parts get close to the goal, while the diffusing part provides the local coverage to attain the goal. The complete algorithm is detailed in Sect. 3.4 and Appendix.

where $\mathcal{I}$ denotes the MI and $\mathcal{H}$ is the entropy function. The first expression, known as the forward form of MI, explicitly balances the two sought-after properties of *coverage* — captured by the entropy over the state space $\mathcal{H}(S)$ — and *directedness*, i.e., the ability to reach specific states S depending on Z — captured by the negative conditional entropy $-\mathcal{H}(S|Z)$. The second expression of (1), often easier to optimize and referred to as the reverse form, stipulates that the skills should be sampled as diversely as possible while being discriminable.

Maximizing (1) has been shown to be a powerful approach for encouraging exploration in RL [20, 32] and for unsupervised skill discovery [e.g., 15, 12, 1, 38, 10]. Nonetheless, learning skills that maximize the MI is a challenging optimization problem. Several approximations have been proposed to simplify the problem at the cost of possibly deviating from the original objective of coverage and directedness (see Sect. 4 for a review of related work). In this paper, we propose UPSIDE (*UnsuPervised Skills that dIrect then DiffusE*) to learn skills that can be effectively used to solve goal-based downstream tasks. Our solution builds on the following components (see Fig. 1 for an illustration of UPSIDE):

- *Skill structure.* In order to balance coverage and directedness, we design skills composed of two parts: **1)** a *directed* part that is trained to reach a distinct region of the environment, and **2)** a *diffusing* part that covers the states around the region attained by the first part.
- *Optimization.* We further strengthen the coverage and directedness properties of the skills by turning the MI objective into a constrained optimization problem designed to maximize coverage under the constraint that *each* skill achieves a minimum level of discriminability. This in turn enables UPSIDE to adaptively add skills to improve coverage, when all the initial skills meet the constraint, or remove those that violate the constraint to guarantee that each skill is directed and reaches a distinct region of the environment.
- *Tree structure.* When the agent starts from a fixed initial state, the skills' length is a crucial parameter, where short skills do not allow for proper coverage, and long skills are difficult to train. In UPSIDE we consider short skills to make the optimization easier, while composing them along a tree structure that ensures an adaptive and deep coverage of the environment.

We study how our learned skill structure enables to both perform efficient exploration and learn effective goal-reaching policies in a variety of navigation and continuous control environments (including MuJoCo's reacher) and we compare its performance to relevant baselines.

2 Setting

We consider the URL setting where the agent interacts with a Markov decision process (MDP) M with state space $\mathcal{S}$, action space $\mathcal{A}$, dynamics $p(s'|s,a)$, and **no reward**. The agent starts each episode from a designated initial state $s_0 \in \mathcal{S}$. Upon termination of the chosen policy, the agent is then reset to s_0. This setting is particularly challenging from an exploration point of view since the agent cannot rely on the initial distribution to cover the state space.

We recall the MI-based unsupervised skill discovery approach [see e.g., 15]. Denote by Z some (latent) variables on which the skills of length T are conditioned. There are three optimization variables: *(i)* the support of the skills denoted by $|Z|$ (we consider it to be discrete so $|Z|$ is the number of skills), *(ii)* the policy $\pi(z)$ associated to skill z, and *(iii)* the sampling rule ρ (i.e., $\rho(z)$ is the probability of sampling skill z at the beginning of the episode). Let the variable S_T be the random (final) state induced by sampling a skill z from ρ and executing the associated policy $\pi(z)$ from s_0 for an episode. We denote by $p_{\pi(z)}(s_T)$ the distribution over (final) states induced by executing the policy of skill z, by $p(z|s_T)$ the probability of z being the skill to induce state s_T, and let $\overline{p}(s_T) = \sum_{z\in Z} \rho(z) p_{\pi(z)}(s_T)$. Then maximizing the MI between Z and S_T can be written as

$$\begin{aligned}\max_{|Z|,\,\rho,\,\pi} \mathcal{I}(S_T;Z) &= \mathcal{H}(S_T) - \mathcal{H}(S_T|Z) = -\sum_{s_T} \overline{p}(s_T)\log\overline{p}(s_T) + \sum_{z\in Z}\rho(z)\mathbb{E}_{s_T}\left[\log p_{\pi(z)}(s_T)\right] \\ &= \mathcal{H}(Z) - \mathcal{H}(Z|S_T) = -\sum_{z\in Z}\rho(z)\log\rho(z) + \sum_{z\in Z}\rho(z)\mathbb{E}_{s_T}\left[\log p(z|s_T)\right], \end{aligned} \quad (1)$$

where in the expectations $s_T \sim p_{\pi(z)}(s_T)$. As discussed in Sect. 1, learning the optimal $|Z|$, ρ, and π is a challenging problem [see e.g., 15, 12, 10].

3 Algorithm Structure

UPSIDE is based on three main components: **a)** the skill learning corresponding to stage A and B of Fig. 1 and described in Sect. 3.1, **b)** a constrained optimization problem used to optimize the number of skills (stage C and Sect. 3.2) and **c)** a tree-building procedure (stage D and Sect. 3.3). Together, these components allow UPSIDE to discover skills that combine coverage and directedness.

3.1 Skill Structure and Optimization

As shown in e.g., [12, 38, 46], the level of stochasticity of each skill (e.g., induced via a regularization on the entropy over the actions) plays a key role in trading off coverage and directedness. In fact, while randomness promotes broader coverage, it may compromise the directedness of the skills. In fact, a highly stochastic skill tends to induce a distribution $p_{\pi(z)}(s_T)$ over final states with high entropy (thus decreasing $-\mathcal{H}(S_T|Z)$), which prevents the skill to be reusable in solving sparse-reward downstream tasks where the objective is to reliably reach a specific goal state of the environment. Determining *how much* stochasticity to inject to adequately balance both objectives and optimize (1) is a difficult problem.[1]

We propose to design skills with a *decoupled policy structure*:

- A *directed* part (of length T) with low stochasticity and trained to reach a specific region of the environment. It is responsible for increasing the $-\mathcal{H}(S|Z)$ term in (1).
- A *diffusing* part (of length H) with high stochasticity to promote local coverage of the states around the region reached by the directed part. It is responsible for increasing the $\mathcal{H}(S)$ term in (1).

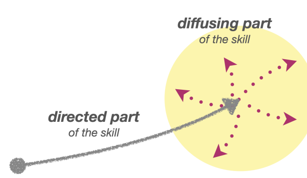


Figure 2: Directed and diffusing parts of the skill.

Similar to prior work [e.g., 15, 12], the policy associated to the directed part of skill z is trained to maximize an intrinsic reward $r_z(s) \approx p(z|s)$,[2] where $p(z|s)$ measures the "discriminability" of the skill z given the state s. More formally, $\pi(z)$ maximizes the cumulative reward $\mathbb{E}_{\pi(z)}\big[\sum_{t=T+1}^{T+H} r_z(s_t)\big]$ over the states traversed by the policy during the diffusing part. In practice, we also add a small entropy regularization $\mathcal{H}(\pi(\cdot|z,s_t))$ to the directed policy in order to ensure a minimum level of

[1] In RL, stochasticity is injected at "train time" to boost *exploration* or improve *robustness*, while the policy executed at "test time" is deterministic. Here we refer to stochasticity introduced to better optimize (1).

[2] Although [15, 12] employ rewards in the log domain, we find that using a reward that is a non-linear transformation into $[0, 1]$ works better in practice, as also observed in [42, 5]. Furthermore, in practice we replace $p(z|s)$ by the predictions of a learned discriminator $q_\phi(z|s)$ as explained in Sect. 3.4.

exploration and make the learning more robust. For the diffusing part, we rely on a simple random walk policy (i.e., a stochastic policy with uniform distribution over actions).

Intuitively, the diffusing part defines a cluster of states that is used as a goal for the directed part. This allows us to "ground" the latent variable representations of the skills Z to specific regions of the environment (i.e., the clusters). As a result, maximizing the MI over such skills can be seen as learning a set of "cluster-conditioned", and thus directed, policies.

3.2 Skill Support and Sampling Rule

The MI objective (1) crucially depends on the number of skills ($|Z|$) and the distribution $\rho(z)$. Unfortunately, it is been shown [e.g., 10] that solving (1) is particularly challenging. In order to simplify the optimization and the associated learning problem, we modify (1) in two ways.

First, coherently with the skill optimization detailed in Sect. 3.1, the random variable S in the conditional entropy is any state reached during the diffusing part of the skill and not just the terminal state. More formally, we denote by S_{diff} the random variable and its distribution for a specific skill z is $p_{\pi(z)}(s_{\text{diff}}) = 1/H \sum_{t=T+1}^{T+H} p_{\pi(z)}(s_t)$, i.e., the distribution over states obtained by averaging the distributions at any of the steps in the diffusing part. Similarly, $p(z|s_{\text{diff}})$ now denotes the probability of z being the skill to traverse s_{diff} during its diffusing part. As a result, training the skills to maximize MI naturally leads the diffusing parts to "push" the directed parts away so as to reach diverse regions of the environment. The combination of "global" coverage of the directed parts and "local" coverage of the diffusing part ensures that the whole environment is properly visited with $|Z| \ll S$ skills.[3]

Second, we introduce an alternative problem that simplifies the optimization while preserving the coverage and directedness properties of MI. This is achieved by introducing a stronger requirement on the discriminability. While the conditional entropy term $-\mathcal{H}(Z|S)$ in (1) promotes the discriminability of skills *on average*, we argue that a more suitable objective is to *constrain* each skill to achieve a *minimum* level of discriminability. First, we move from the average to the minimum over skills by lower bounding the conditional entropy as

$$-\mathcal{H}(Z|S_{\text{diff}}) = \sum_{z \in Z} \rho(z) \mathbb{E}_{s_{\text{diff}}} [\log p(z|s_{\text{diff}})] \geq \min_{z \in Z} \mathbb{E}_{s_{\text{diff}}} [\log p(z|s_{\text{diff}})], \tag{2}$$

which leads to the following optimization (assuming π is fixed for convenience)

$$\max_{|Z|=N,\rho} \Big\{ \mathcal{H}(Z) + \min_{z \in [N]} \mathbb{E}_{s_{\text{diff}}} [\log p(z|s_{\text{diff}})] \Big\}, \tag{3}$$

where with an abuse of notation we use $z \in [N]$ to denote all skills in a set Z with cardinality N. Since (3) is a lower bound to MI, it tends to promote the same type of covering and directed skills. Furthermore, (2) no longer depends on the distribution over skills and the entropy term $\mathcal{H}(Z)$ is maximized by setting ρ to the uniform distribution over N skills (i.e., $\max_\rho \mathcal{H}(Z) = \log(N)$), thus simplifying the optimization, which now only depends on N.

While optimizing (3) promotes a cardinality N such that all skills have good discriminability, a more convenient formulation is to explicitly set a minimum level of discriminability for all skills through the following constrained optimization problem:

$$\max_{N \geq 1} \; \log(N) \qquad \text{s.t.} \quad \min_{z \in [N]} \mathbb{E}_{s_{\text{diff}}} [\log p(z|s_{\text{diff}})] \geq \log \eta. \tag{4}$$

where η is a parameter that defines the discriminability threshold. A skill z is said to be η*-consolidated* if it satisfies the constraint. Crucially, let $P_N := \min_{z \in [N]} \mathbb{E}_{s_{\text{diff}}} [\log p(z|s_{\text{diff}})]$, then the sequence $(P_N)_{N \geq 1}$ is non-increasing with $P_1 = 0$ (i.e., the more skills the harder it is to meet the constraint). As a result, (4) can be optimized following a simple greedy strategy incrementally adding skills until the constraint is violated. The optimal N thus defines the *effective number* of η-consolidated skills and it corresponds to the largest number of skills that is guaranteed to display sufficient discriminability. Alternatively, we can interpret (4) as finding the largest number of clusters (i.e., the region reached by the directed part of a skill and covered by its associated diffusing part) with a minimum level of inter-cluster distance. This effect is qualitatively illustrated in Fig. 1, where the states attained by the directed part of the skills attain different regions that are locally covered by their diffusing parts.

[3] Notice that (1) is maximized by setting $|Z| = |S|$ (since $\max_Y \mathcal{I}(X,Y) = \mathcal{I}(X,X) = \mathcal{H}(X)$), i.e., where each skills is a goal-conditioned policy reaching a different state. This implies having as many policies as states, which makes the learning particularly challenging as the complexity of the environment increases.

Algorithm 1: `UPSIDE`

Initialize: Discriminability threshold $\eta \in (0,1)$, branching factor $N_0 \geq 1$, patience K
Initialize: Tree $\mathcal{T}$ initialized as a root node indexed by 0, queue of parent nodes $\mathcal{W} = \{0\}$.
while $\mathcal{W} \neq \emptyset$ **do** `// tree expansion`
  Dequeue a node/skill $w \in \mathcal{W}$ and expand $\mathcal{T}$ at w by adding a set $\mathcal{C}(w)$ of N_0 nodes/skills
  Create random policies $\pi_z,\ \forall z \in \mathcal{C}(w)$
  Initialize discriminator q_ϕ with $|\mathcal{T}|$ classes
  `Continue = true; Saturated = false`
  while `Continue` **do**
    for K iterations **do**
      Sample a skill z from $\mathcal{T}$ at random
      Extract the sequence of nodes $z_{(1)}, \dots, z$ in $\mathcal{T}$ leading to z
      Execute the composed (directed part) policy $(\pi_{z_{(1)}}, \dots, \pi_z)$ followed by the diffusing part
      Add states observed during the diffusion part to state buffer $\mathcal{B}_z$
      Update discriminator q_ϕ with SGD on $\mathcal{B}_z$ to predict label z
      if $z \in \mathcal{C}(w)$ **then** `// Update only new policies, other polices kept fixed`
        Update policy π_z using SAC to optimize the discriminator reward as in Sect. 3.1.
    Compute the skill-discriminability $d(z) = \widehat{q}_\phi^{(B)}(z) = \frac{1}{|\mathcal{B}_z|}\sum_{s \in \mathcal{B}_z} q_\phi(z|s)$ for all $z \in \mathcal{C}(w)$
    if $\min_{z \in \mathcal{C}(w)} d(z) < \eta$ **then** `// Node removal`
      Remove the node/skill $z = \arg\min_{z \in \mathcal{C}(w)} d(z)$ from $\mathcal{C}(w)$ and $\mathcal{T}$
      Set `Saturate = true`
    else if `not Saturated` **then**
      Add one new node/skill to $\mathcal{C}(w)$ and $\mathcal{T}$
    else
      Set `Continue = false`
  Enqueue in $\mathcal{W}$ the consolidated nodes $\mathcal{C}(w)$

3.3 Composing Skills in a Tree Structure

The MI optimization problem as well as our constrained variant (4) depend on the initial state s_0 and on the length of each skill. Although these quantities are usually predefined and only appear implicitly in the equations, they have a crucial impact on the obtained behavior. In fact, resetting after each skill execution unavoidably restricts the coverage to a radius of at most $T + H$ steps around s_0. This may suggest to set T and H to a large value. However, increasing the horizon makes the training of the skills more challenging, as learning π would require solving a difficult RL problem itself.

Instead, we propose to "extend" the length of the skills through composition. Indeed, the decoupled skill structure and the constraint in (4) entail that the directed part of each of the η-consolidated skills reliably reach a specific (and distinct) region of the environment and it is thus re-usable and amenable to composition. We propose to chain the directed part of the skills in order to reach further and further parts of the state space. Specifically, we build a growing tree, where the root is the initial state s_0, the edges represent the directed part of the skills, and the nodes represent the diffusing part of skills. As such, whenever a skill z is selected, the directed part of all the policies associated to its predecessor skills in the tree are executed first (see Fig. 1 for an illustration of the tree structure).

As a result, the agent naturally builds a curriculum on the episode lengths, which grow as the sequence $(iT + H)_{i \geq 1}$. As such, it does not require prior knowledge on an adequate horizon of the downstream goal-based task.[4] Here this knowledge is replaced by T and H which are more environment-agnostic and task-agnostic quantities, as their choice rather has an impact on the size and shape of the learned tree (e.g., the smaller T and H the bigger the tree).

3.4 The `UPSIDE` Algorithm

We are now ready to introduce `UPSIDE`, which provides a specific implementation of the components described before (see Fig. 1 for a qualitative illustration and Algorithm 2 for the detailed pseudo-code).

We perform standard approximations to make the constraint in (4) easier to estimate. We approximate the unknown posterior $p(z|s)$ with a learned discriminator $q_\phi(z|s)$ with parameters ϕ. We also

[4] See e.g., the discussion in [33] on the "importance of properly choosing the training horizon in accordance with the downstream-task horizon the policy will eventually face."

remove the logarithm from the constraint to have an estimation range of $[0,1]$ and thus lower variance[2]. Finally, we replace the expectation over s with an empirical estimate $\widehat{q}_\phi^{(B)}(z)$ averaging the value of the discriminator evaluated on the last B states observed while executing the diffusing part of z. Integrating these approximations in (4) leads to

$$\max_{N\geq 1,\pi} N \qquad \text{s.t.} \qquad \min_{z\in[N]} \widehat{q}_\phi^{(B)}(z) \geq \eta. \tag{5}$$

As discussed in Sect. 3.2, this problem can be conveniently optimized using a greedy strategy. We then integrate the optimization of (5) into an adaptive tree expansion strategy: **(Generating new skills)** Given a tree structure as described in Sect. 3.3, we expand the tree at a leaf w by adding N_0 new nodes/skills following a breadth-first-search approach (lines 1, 2). Then (**Skill Learning**) the new skills are optimized by: **i)** sampling random skills in the tree to update the discriminator (lines 7-11), and **ii)** by updating the policies to optimize the discriminability reward (Sect. 3.1) computed using the discriminator (lines 13). To speed-up convergence, we only update the policies that have be added to the tree structure, keeping all the previous policies fixed (line 12). Note that in the update of the discriminator we leverage the states observed in previous phases of the algorithm by maintaining a (small) replay buffer of states for each skill. **(Node Consolidation)** After a *patience* period (line 6), if all skills are η-consolidated, we tentatively add more skills to the leaf w (line 18). On the other hand, if any skill does not meet the discriminability threshold, we remove it and consolidate the remaining skills into the tree (lines 16, 17) and we repeat the process.

Model selection. A core aspect of any RL algorithm is *model selection*, i.e., finding the best configuration of hyperparameters. In URL with no prior knowledge of the downstream task(s), it is non-trivial to devise an adequate criterion for model selection and this aspect is rarely addressed, despite being crucial in practice. For instance, while the coverage of the state space may be a good proxy for the performance of a URL algorithm [see e.g., 10], it may be difficult to measure in continuous problems. Interestingly, our optimization problem directly provides a single, task-agnostic and environment-agnostic criterion for model selection, which is the number N of η-consolidated skills discovered by the agent. Indeed in all of our experiments we simply select the model (i.e., set of hyperparameters) that maximizes N. This is a significant advantage w.r.t. existing methods, such as `VIC` and `DIAYN`, for which no principled approach to model selection is provided.

4 Related work

Unsupervised Reinforcement Learning methods can be broadly decomposed according to the way they summarize the experience accumulated during the unsupervised phase into reusable knowledge to solve downstream tasks. This includes both off-policy model-free [e.g., 34] and model-based [e.g., 37] methods that seek to populate a representative replay buffer and build accurate value or model estimates, that are used to solve a given downstream task in a zero- or few-shot manner. The accumulated experience during train time can also be compressed into a low-dimensional representation for value functions as well as policies and to improve exploration [e.g., 45]. An alternative line of work focuses on the discovery of a set of skills in an unsupervised manner. Our approach falls in this category, on which we now focus our related work review.

Skill discovery based on MI maximization was first proposed in `VIC` [15], where only the final states of each trajectory are considered in the reverse form of (1) and where both the skills and their sampling rules are simultaneously learned (with a fixed support $|Z|$, i.e., a fixed number of skills). `DIAYN` [12] fixes the sampling rule to be uniform, and weighs the skills with an action-entropy coefficient (i.e., it additionally minimizes the MI between actions and skills given the state), so as to push the skills away from each other and enhance coverage. `DADS` [38] learns skills that are not only diverse but also predictable by learned dynamics models, by using a generative model over observations (rather than over skills) and optimizing a forward form of MI, namely $\mathcal{I}(s';z|s)$ between the next state s' and current skill z (with continuous latent) conditioned on the current state s. `EDL` [10] shows that existing skill discovery approaches can provide insufficient coverage, and instead proposes to rely on a fixed distribution over states $p(s)$ which is either provided by an oracle or learned. In `SMM` [24], the MI formalism is used to learn a policy for which the state marginal distribution matches a given target state distribution (e.g., uniform), which can be seen as a more scalable way of tackling the problem of maximum entropy over the state space [19], and as a way to encourage skills to go through unknown state regions. Other MI-based skill discovery methods include [13, 18, 31, 5, 43], as well as [44, 27] which investigate skill discovery in non-episodic settings.

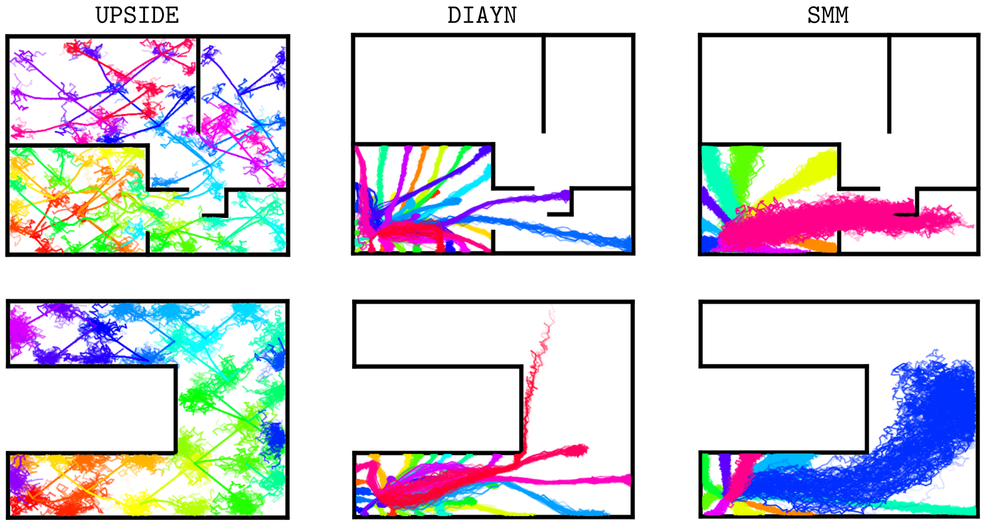


Figure 3: UPSIDE, DIAYN-curriculum and SMM-10 skills learned in a bottleneck maze *(Top)* and a U-maze *(Bottom)*. For both DIAYN and SMM we report the stochastic execution of the learned skills and for UPSIDE we report the deterministic directed parts (that are composed) followed by the (stochastic) diffusing part, which is the same protocol used to evaluate coverage.

Our approach shares a similar motivation to prior MI-based works of targeting skills that are both directed and state-covering. In particular, the decoupled structure introduced in Sect. 3.1 can be seen as a more suitable way to achieve the objective of improving the coverage of VIC as done in DIAYN and SMM, without compromising the directedness of the skills.

While most skill discovery approaches consider a fixed number of skills, a curriculum with increasing number of skills is studied in [1, 3]. Our discriminability constraint is what enables skills to be composed along a tree structure, which allows increases or decreases the support of available skills depending on the region of the state space.

Recently, [46] proposed a hierarchical RL method that discovers abstract and task-agnostic skills while jointly learning a higher-level policy which is trained to maximize environment reward. Our approach builds on a similar promise of composing skills instead of resetting to s_0 after each execution, yet we articulate the composition differently, by exploiting the direct-then-diffuse structure to ground learned skills to the state space instead of being abstract.

In addition, approaches such as DISCERN [42] and Skew-Fit [34] learn a goal-conditioned policy in an unsupervised way with an MI objective. As explained in [10, Sect. 5], this can be interpreted as a skill discovery approach with latent $Z = S$, i.e., where each goal state can define a different skill. Conditioning on either goal states or abstract latent skills forms two extremes of the spectrum of unsupervised RL. We target an intermediate approach, seeking to benefit from the groundedness of the latent skill Z and the states S (and thus amenability to composition) of goal-conditioned RL, and from the reduced search space and sampling ease of skill-based RL.

An alternative approach to skill discovery builds on "spectral" properties of the dynamics of the environment. This includes eigenoptions [28, 29] and covering options [22, 23], as well as the algorithm of [4] that builds a discrete graph representation which learns and composes spectral skills.

5 Experiments

In this section, we investigate the following questions: **i)** Can the adaptive tree structure of UPSIDE incrementally cover an unknown environment while preserving directedness of the skills? **ii)** Following the unsupervised phase, how can UPSIDE be leveraged to solve goal-based downstream tasks?

We report results on: **a)** Navigation problems in continuous mazes, where actions represent the desired shift in x and y coordinates; **b)** A difficult instance of CartPole, where the cart starts with zero speed and the pole is oriented downside; **c)** The Reacher [41] problem using the MuJoCo implementation in Gym [8]. In all environments, the per-dimension action space is in $[-1; +1]$.

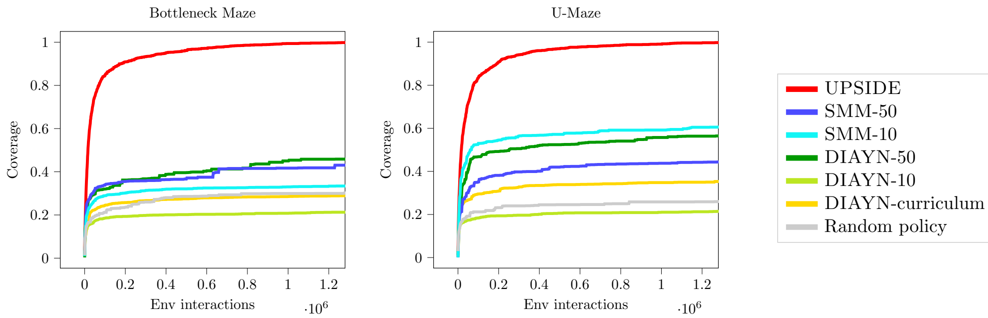


Figure 4: Normalized coverage in U-maze and bottleneck.

We compare to different baselines. `DIAYN`-K, where K is a fixed number of skills, is the original algorithm proposed in [12]. `DIAYN`-Curriculum is a variant where the number of skills is automatically tuned following the same procedure as in `UPSIDE` ensuring a good discriminability. We also compare to `SMM` [24], which is similar to `DIAYN`, but it includes an exploration bonus encouraging the policies to visit rarely encountered states. In our implementation, the exploration bonus is obtained by maintaining a multinomial distribution over "buckets of states" obtained by discretization, resulting in an computation-efficient and stable implementation that is more stable than the original VAE-based method. `UPSIDE` and all baselines are implemented with Soft-Actor Critic (SAC) [17].

Unsupervised Phase. We run all methods until convergence. We then do model selection according to the criterion of either the final number of skills for `UPSIDE` and `DIAYN`-curriculum and the final average discriminability for `DIAYN`-K and `SMM`. To compute the coverage, we perform rollouts by first sampling a skill uniformly at random and executing its associated policy until termination. We discretize states into buckets (50 interval per dimension for mazes and 10 for control environments) and report the proportion of buckets reached by each method as a function of the total number of steps executed in the environment over multiple rollouts. Since only a small portion of the discretized states can be reached, we normalize the coverage such that the best method obtains 1.

We consider two topologies of mazes with size (height and width) 50 such that exploration is non-trivial (i.e., a random policy is only able to cover a small part of the state space): a U-shaped maze and a Bottleneck maze (which is a harder version of the one in [10, Fig. 1] which is only of size 10 for the same action space). In Fig. 3 we show that `UPSIDE` succeeds in covering the near-entirety of the state space by creating a tree of directed skills. Moreover, `UPSIDE` created directed skills with a low entropy, while the two baselines tend to create skills that are more stochastic. This is particularly evident for SMM, due to the state-entropy exploration bonus, that while it encourages broader coverage makes skills less directed.

In Fig. 4 we report the coverage on the Bottleneck maze and U-Maze. For `UPSIDE`, executing a skill corresponds to executing the directed part of all the "parent" skills in the tree and concluding with the diffusion part of the skill. `SMM` achieves better coverage than `DIAYN` thanks to the increased level of stochasticity (diffusion) of its skills. `UPSIDE` outperforms both by reaching regions of the environment that are not be achieved by other methods. Here, we plot `UPSIDE` with $T = 10$ and $H = 10$, but we found `UPSIDE` to be robust to these parameters as shown in the supplementary.

Results are similar in the CartPole problem (see Fig. 5) where `UPSIDE` (with $T = 20$ and $H = 20$) obtains better coverage than baselines. On the other hand, in Reacher (see Fig. 5), `DIAYN-50` outperforms `UPSIDE` in terms of coverage. This can be explained by the fact that, in this environment, highly stochastic skills provide a good coverage. Nonetheless, this comes at the cost of very low discriminability (rightmost plot), which suggests `DIAYN-50` skills have poor directedness. On the other hand, `UPSIDE` (and `DIAYN`-curriculum) achieves much larger discriminability by removing redundant skills and favoring more directed policies.

Downstream Tasks. Following the unsupervised phase, `UPSIDE` has learned a tree of skills. We now investigate how these skills are used to tackle a downstream task. In that setting, we propose to use skill-based approaches (i.e `UPSIDE`, `DIAYN` and `SMM`) in the following way: a) (exploration) first we sample rollouts over the different skills. b) We then select the best skill based on the maximum cumulative reward collected and c) we fine-tune this skill to maximize the reward. We report results on mazes (additional results are provided in the supplementary). We consider a sparse positive reward

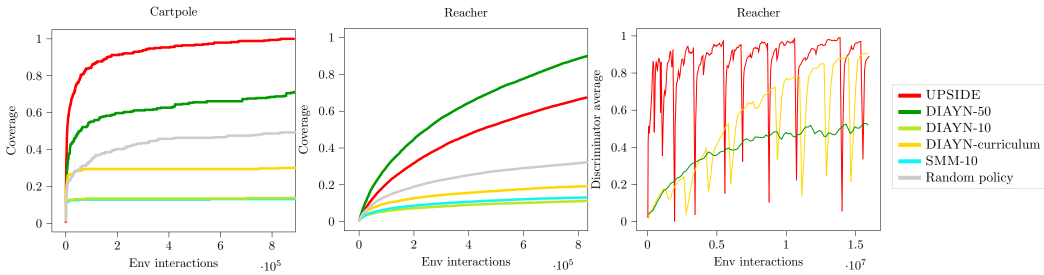


Figure 5: Normalized coverage in Cartpole *(Left)* and Reacher *(Middle)*. *(Right)* Average discriminability of the skills during training in Reacher.

when reaching a particular defined goal.[5] We consider goals at different distances from the initial state s_0, the further, the harder. Fig. 6 shows the learning curves obtained when fine-tuning the best skill for the different models and compare to a classical SAC algorithm where a single policy is learned from scratch. `DIAYN/SMM` means we use the best state-covering policies between `DIAYN` and `SMM`. For the "close" goal setting, both `UPSIDE` and `DIAYN/SMM` are able to learn to reach this goal efficiently while SAC solves the task only for some of the training runs. Note that we do not show `DIAYN` performance since it is lower than the `SMM` one. For the "far" goal setting, only `UPSIDE` learns to reach this goal. Obtained trajectories are illustrated in Fig. 6.

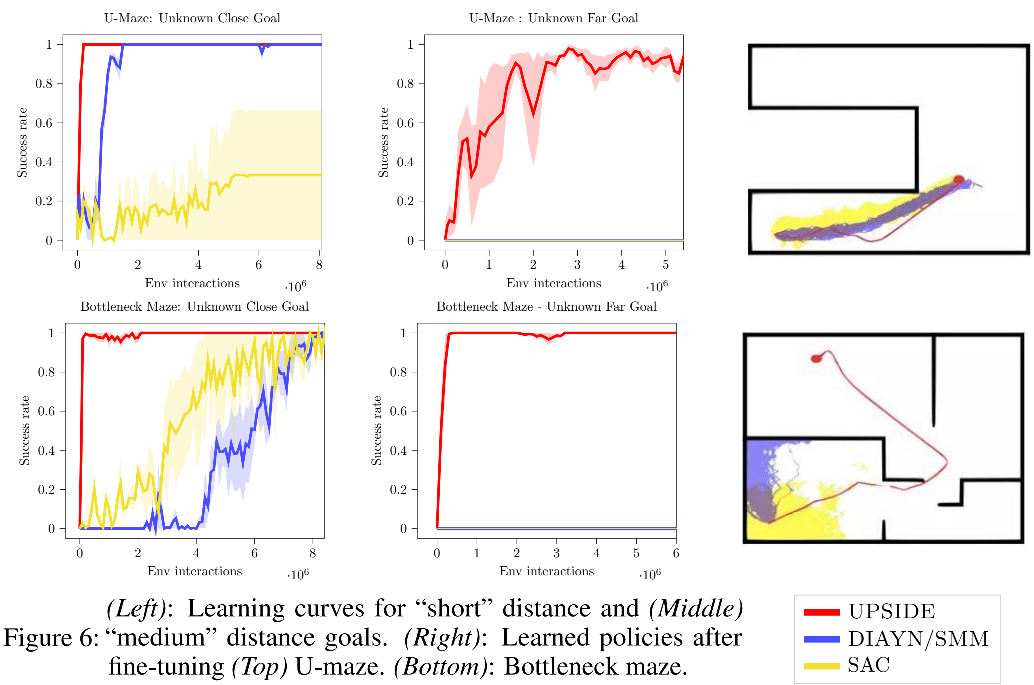


Figure 6: *(Left)*: Learning curves for "short" distance and *(Middle)* "medium" distance goals. *(Right)*: Learned policies after fine-tuning *(Top)* U-maze. *(Bottom)*: Bottleneck maze.

6 Conclusion

We introduced `UPSIDE`, a novel algorithm for unsupervised skill discovery designed to trade off between coverage and directedness and develop a tree of skills that can be used to both perform efficient exploration of the environment and learn effective goal-directed policies. Natural venues for future investigation are: **1)** The diffusing part of each skill could be explicitly trained to maximize local coverage; **2)** `UPSIDE` assumes a good representation of the state is provided as input, it would be interesting to pair `UPSIDE` with effective representation learning techniques to tackle problems with high-dimensional input (e.g., image-based RL); **3)** While `UPSIDE` is grounded on the solid principle of MI maximization, a more thorough theoretical investigation is needed to explicitly link the optimization problem and its approximations to the downstream performance.

[5] Notice that if the goal was known, the learned discriminator could be directly used to identify the most promising skill to fine-tune.

References

[1] J. Achiam, H. Edwards, D. Amodei, and P. Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[2] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. In *NIPS*, 2017.

[3] A. Aubret, L. Matignon, and S. Hassas. Elsim: End-to-end learning of reusable skills through intrinsic motivation. *arXiv preprint arXiv:2006.12903*, 2020.

[4] A. Bagaria, J. Crowley, J. W. N. Lim, and G. Konidaris. Skill discovery for exploration and planning using deep skill graphs. 2021.

[5] K. Baumli, D. Warde-Farley, S. Hansen, and V. Mnih. Relative variational intrinsic control. *arXiv preprint arXiv:2012.07827*, 2020.

[6] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[7] D. Bertsekas. *Dynamic programming and optimal control*, volume 2. 1995.

[8] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[9] A. Z. Broder and A. R. Karlin. Bounds on the cover time. *Journal of Theoretical Probability*, 2(1):101–120, 1989.

[10] V. Campos, A. Trott, C. Xiong, R. Socher, X. Giro-i Nieto, and J. Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, 2020.

[11] M. Diaz, L. Paull, and P. S. Castro. Loco: Adaptive exploration in reinforcement learning via local estimation of contraction coefficients. *ICLR Workshop SSL-RL*, 2021.

[12] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

[13] C. Florensa, Y. Duan, and P. Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

[14] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pages 1515–1528, 2018.

[15] K. Gregor, D. J. Rezende, and D. Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[16] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

[18] S. Hansen, W. Dabney, A. Barreto, D. Warde-Farley, T. Van de Wiele, and V. Mnih. Fast task inference with variational intrinsic successor features. In *International Conference on Learning Representations*, 2019.

[19] E. Hazan, S. Kakade, K. Singh, and A. Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691, 2019.

[20] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: variational information maximizing exploration. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1117–1125, 2016.

[21] T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

[22] Y. Jinnai, J. W. Park, D. Abel, and G. Konidaris. Discovering options for exploration by minimizing cover time. In *International Conference on Machine Learning*, pages 3130–3139. PMLR, 2019.

[23] Y. Jinnai, J. W. Park, M. C. Machado, and G. Konidaris. Exploration in reinforcement learning with deep covering options. In *International Conference on Learning Representations*, 2020.

[24] L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[25] S. H. Lim and P. Auer. Autonomous exploration for navigating in MDPs. In *Conference on Learning Theory*, pages 40–1, 2012.

[26] Y. Liu and E. Brunskill. When simple exploration is sample efficient: Identifying sufficient conditions for random exploration to yield pac rl algorithms. *arXiv preprint arXiv:1805.09045*, 2018.

[27] K. Lu, A. Grover, P. Abbeel, and I. Mordatch. Reset-free lifelong learning with skill-space planning. *arXiv preprint arXiv:2012.03548*, 2020.

[28] M. C. Machado, M. G. Bellemare, and M. Bowling. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*, pages 2295–2304. PMLR, 2017.

[29] M. C. Machado, C. Rosenbaum, X. Guo, M. Liu, G. Tesauro, and M. Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations*, 2018.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[31] N. Modhe, P. Chattopadhyay, M. Sharma, A. Das, D. Parikh, D. Batra, and R. Vedantam. Ir-vic: Unsupervised discovery of sub-goals for transfer in rl. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization, 2020.

[32] S. Mohamed and D. J. Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 2125–2133, 2015.

[33] M. Mutti, L. Pratissoli, and M. Restelli. A policy gradient method for task-agnostic exploration. *arXiv preprint arXiv:2007.04640*, 2020.

[34] V. H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine. Skew-fit: State-covering self-supervised reinforcement learning. In *International Conference on Machine Learning*, 2020.

[35] M. L. Puterman. *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[37] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

[38] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020.

[39] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[40] J. Tarbouriech, E. Garcelon, M. Valko, M. Pirotta, and A. Lazaric. No-regret exploration in goal-oriented reinforcement learning. In *International Conference on Machine Learning*, 2020.

[41] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 50pou26–5033, 2012.

[42] D. Warde-Farley, T. Van de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih. Unsupervised control through non-parametric discriminative rewards. In *International Conference on Learning Representations*, 2019.

[43] K. Xie, H. Bharadhwaj, D. Hafner, A. Garg, and F. Shkurti. Skill transfer via partially amortized hierarchical planning. In *International Conference on Learning Representations*, 2021.

[44] K. Xu, S. Verma, C. Finn, and S. Levine. Continual learning of control primitives: Skill discovery via reset-games. *arXiv preprint arXiv:2011.05286*, 2020.

[45] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Reinforcement learning with prototypical representations. *arXiv preprint arXiv:2102.11271*, 2021.

[46] J. Zhang, H. Yu, and W. Xu. Hierarchical reinforcement learning by discovering intrinsic options. In *International Conference on Learning Representations*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discuss the limitations and directions of further investigation in the conclusion.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We do not foresee any obvious negative societal impacts from our work, which focuses on the fundamentals of reinforcement learning and proposes a new algorithm for unsupervised skill discovery.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We plan to release our code upon acceptance of this work.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Yes when possible.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Sect. 5.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

Table of Contents

A UPSIDE Algorithm

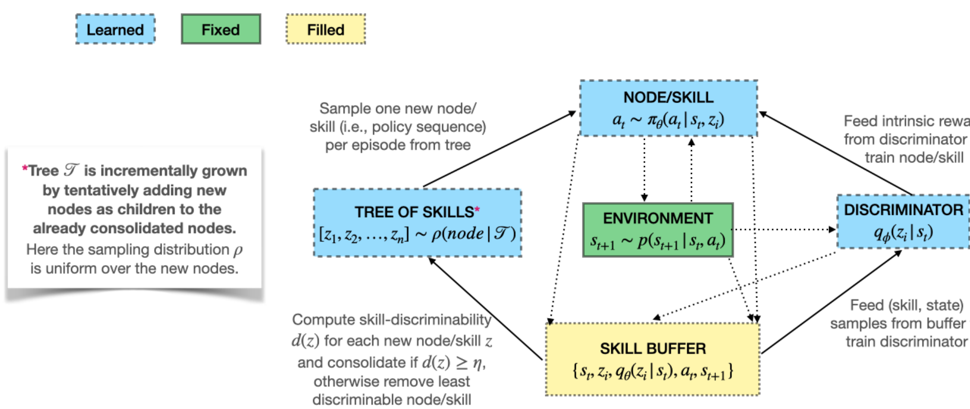


Figure 7: High-level approach of UPSIDE

We provide a diagram of the high-level approach of UPSIDE in Fig. 7 and a detailed pseudo-code in Alg. 2. UPSIDE initializes a tree structure $\mathcal{T}$ with root node 0 and queue of parent nodes $\mathcal{W} = \{0\}$. As long as the queue is not empty, the following steps are performed:

- **(Generating new skills)** We expand the tree at a leaf $w \in \mathcal{W}$ by adding N_0 new nodes/skills denoted by $\mathcal{C}(w)$ (lines 1, 2). We initialize a discriminator with $|\mathcal{T}|$ classes to account for the newly created nodes (line 3).
- (**Skill Learning**) Then the new skills and discriminator are optimized as follows:

  - We sample (uniformly) and rollout the new skills $z \in \mathcal{C}(w)$ and add the states of their diffusing parts in their corresponding buffers $\mathcal{B}_z$ (lines 8 to 11).

  - We update the discriminator by leveraging the states and skill labels in the buffers (lines 12 to 17). In particular, the ratio μ signifies that we train more often the discriminator on the previously consolidated skills/classes than on the new skills/classes in $\mathcal{C}(w)$, i.e., we sample pairs (label z, state in $\mathcal{B}_z$) with probability $(\mathbb{I}[z \in |\mathcal{C}(w)|] + \mu\mathbb{I}[z \notin |\mathcal{C}(w)|])/((1-\mu)|\mathcal{C}(w)| + \mu|\mathcal{T}|)$. We give slightly more weight to the already consolidated skills in the discriminator training because the

Algorithm 2: `UPSIDE`

Initialize: Discriminability threshold $\eta \in (0, 1)$, branching factor $N_0 \geq 1$ (to be adapted at each node), (optional) maximum branching factor $N_{\max} \geq N_0$, patience K, window $\mathcal{V}$, number of trajectory rollouts R per update of discriminator and policies, batch size of N_{discr} to train the discriminator, ratio μ of probabilities between consolidated classes and new classes to train discriminator
Initialize: Tree $\mathcal{T}$ initialized as a root node indexed by 0, queue of parent nodes $\mathcal{W} = \{0\}$.
while $\mathcal{W} \neq \emptyset$ **do** `// tree expansion`
1 Dequeue a node/skill $w \in \mathcal{W}$ and expand $\mathcal{T}$ at w by adding a set $\mathcal{C}(w)$ of N_0 nodes/skills
2 Create random policies π_z and buffers $\mathcal{B}_z$, $\forall z \in \mathcal{C}(w)$
3 Initialize discriminator q_ϕ with $|\mathcal{T}|$ classes
4 `Continue = true; Saturated = false`
5 **while** `Continue` **do**
6 **for** K iterations **do**
7 **for** $r \in [\![1, R]\!]$ **do** `// collect` R `trajectories`
8 Sample a skill z from $\mathcal{C}(w)$ at random
9 Extract the sequence of nodes $z_{(1)}, \ldots, z$ in $\mathcal{T}$ leading to z
10 Execute the composed (directed part) policy $(\pi_{z_{(1)}}, \ldots, \pi_z)$ followed by the diffusing part
11 Add states observed during the diffusing part to $\mathcal{B}_z$
12 $B = \{\}$ `// Initialize batch to update the discriminator`
13 **while** $|B| < N_{discr}$ **do**
14 Sample a skill z from $\mathcal{T}$ w.p. $(\mathbb{I}[z \in |\mathcal{C}(w)|] + \mu \mathbb{I}[z \notin |\mathcal{C}(w)|])/((1-\mu)|\mathcal{C}(w)| + \mu|\mathcal{T}|)$
15 Sample a state s from the last $\mathcal{V}$ states of $\mathcal{B}_z$
16 Add (s, z) to B
17 Update discriminator q_ϕ with SGD on B to predict label z
18 **for** $z \in \mathcal{C}(w)$ **do**
19 Update policy π_z using SAC to optimize the discriminator reward as in Sect. 3.1.
20 Compute the skill-discriminability $d(z) = \widehat{q}_\phi^{(B)}(z) = \frac{1}{|B|} \sum_{(s,z) \in B} q_\phi(z|s)$ for all $z \in \mathcal{C}(w)$
21 **if** $\min_{z \in \mathcal{C}(w)} d(z) < \eta$ **then** `// Node removal`
22 Remove the node/skill $z = \arg\min_{z \in \mathcal{C}(w)} d(z)$ from $\mathcal{C}(w)$ and $\mathcal{T}$
23 Set `Saturate = true`
24 **else if** `not Saturated` **then**
25 Add one new node/skill to $\mathcal{C}(w)$ and $\mathcal{T}$
26 **if** $|\mathcal{C}(w)| = N_{\max}$ **then**
27 Set `Saturate = true`
28 **else**
29 Set `Continue = false`
30 Enqueue in $\mathcal{W}$ the consolidated nodes $\mathcal{C}(w)$

discriminator is reinitialized whenever new classes (i.e., nodes) are added, thus we seek to avoid the new classes from invading the territory of the older classes that were previously correctly learned. In addition, we only update the discriminator on recent batches of data from the buffers via the window $\mathcal{V}$ (which considers only the last $\mathcal{V}$ states in each skill buffer), which is more sample efficient than doing the discriminator update in a fully on-policy manner (e.g., [12]), especially in our setting where the discriminator changes over training as new skill-nodes (i.e., classes) are added.

- We update the policies of the new skills/nodes in $\mathcal{C}(w)$ with SAC to optimize the intrinsic reward of the discriminator predictions as explained in Sect. 3.1 (line 19). Note that we keep fixed the policies of the previously consolidated nodes/skills, which makes the learning of the tree more stable.

- **(Node Consolidation)** After a *patience* period characterized by K iterations of training (line 6), if all skills are η-consolidated (i.e., the constraint of problem (5) is verified), we tentatively add more skills to the leaf w (line 25). On the other hand, if any skill does not meet the discriminability threshold, we remove it and seek to consolidate the remaining skills into the tree (line 22). The role of the `Saturated` and `Continue` booleans is to ensure that the node addition operation cannot be performed if a node removal operation has already been performed in the training of the set $\mathcal{C}(w)$. Recall that the function is monotone, so if a skill is removed, the optimum cannot be larger. The (optional) $N_{\max}$ value represents the maximum branching factor (i.e., number of children nodes) imposed at each node of the tree.

B Environment Details

Continuous mazes. We consider mazes with height and width 50. The state space is continuous, and there are some horizontal and verticals walls of width 1. The agent observes its current (x, y) Cartesian position (i.e., it does not observe the walls) and it outputs actions $[dx, dy]$ that control its location. The actions dx and dy are constrained to be in $[-1, +1]$. The movement of the agent is affected by collisions with walls: when the agent collides with a wall, it stays in its original position.

CartPole. We modify slightly the simulator from OpenAI Gym [8] to make exploration more difficult and thus to make it more challenging to learn diverse behaviors: the agent moves along the x horizontal position between -2.4 and 2.4 and the pole starts in the reverse position at $x = 0$. When the agent goes out of the x interval, it is teleported back to its initial position (but there is no reset). Observations are $(x, \dot{x}, \theta, \dot{\theta})$ where x is the horizontal position, θ is the angle of the pole to the x-axis, and $\dot{x}, \dot{\theta}$ are their respective velocities.

Reacher. We use the standard MuJoCo implementation of Reacher [41], which is a two-joint robotic arm where the action space ($[-1, +1]$) is the torque applied to both joints with gear 30.

C Experimental Details

C.1 Baselines

For all methods, we augment the state space with the current time-step because horizons are finite.

DIAYN-K. This corresponds to the original `DIAYN` algorithm [12] where K is the number of skills to be learned. In order to make the architecture more similar to `UPSIDE`, we use distinct policies for each skill, i.e. they do not share weights as opposed to [12]. While this may come at the price of sample efficiency, it may also help put lesser constraint on the model (e.g. gradient interference).

DIAYN-Curriculum. We augment `DIAYN` with a curriculum that enables to be less dependant on an adequate tuning of the hyperparameter of the number of skills of `DIAYN`. We consider the curriculum of `UPSIDE` where we start from either a large or small number N_0 of skills, learn skills during a period of time/number of interactions. If the configuration satisfies the discriminablity threshold η, a skill is added, otherwise a skill is removed or learning stopped (as in Alg. A, lines 20-29). Note that the increasing version of this curriculum is similar to the one proposed in `VALOR` [1, Sect. 3.3].

SMM. We used SMM [24] as it is state-of-art in terms of coverage, at least on long-horizon control problems, although [10] reported poor performance in hard-to-explore bottleneck mazes. We tested the regular SMM version, i.e. learning a state density model with a VAE, yet we failed to make it work on the Mazes domain. As we use the cartesian xy positions in maze domains, learning the identity function on two-dimensional input data is too easy with a VAE, thus preventing the benefits of using a density model to drive exploration. Thus we considered a more straightforward implementation of SMM by using the “real” state distribution through counting. Specifically, we maintain a discretized state distribution by counting states in buckets (similar to the way we compute the achieved coverage). The distribution is just computed by dividing by the sum over buckets. We did not use a moving average so counts are not forgotten: the state distribution is over all policies encountered since the beginning of training (whereas the state distribution is “online” in [24]).

C.2 Architecture and Hyperparameters

The architecture of the different methods remains the same in all our experiments, except that the number of hidden units changes across considered environments. We consider decoupled actor and critic in SAC, they both have the same (but unshared weights) state processing architectures. The observation and the step are passed through non-linear MLP with 1 hidden layer with units h, then are concatenated. The concatenation is then mapped to an embedding. For the actor, this embedding is mapped to a mean and variance embedding, then passes through a Squashed Gaussian as explained in [17]. For the critic, the embedding is concatenated with a non-linear (1 hidden layer) embedding of the action, then passed through a final non-linear MLP (1 layer) to a one-dimensional value.

The discriminator is a two-hidden layer model with output size the number of skills in the tree.

Environment-specific hyperparameters. Mazes: $h = \{16, 64\}$ hidden units per layer for policy, and $h = 128$ hidden units per layer for discriminator. Continuous control domains: $h = 256$ hidden units per layer for both policy and discriminator.

Common (for methods and environments) optimization hyperparameters. (See App. A for meaning of each hyperparameter)

- SAC entropy: $\{0.1, 0.01, 0.001\}$
- discount factor: $\gamma = 0.99$
- Q-function soft updates $\tau = 0.005$
- learning rates $lr_{\text{policy}} = 0.001,\ lr_{\text{discriminator}} = \{0.0001, 0.001\}$
- discriminator batch size $B = 1024$
- $\mu = \{2, 5\}$
- $\mathcal{V} = 100$
- Replay buffer size: $1e6$

Note that hyperparameters are kept fixed for the downstream tasks too.

For `UPSIDE` and `DIAYN`-curriculum, we set the patience to be a time-limit instead of a number of iterations. We tried both 300 and 600 seconds to avoid the running time getting too high if the tree grows large.

The total running time for `DIAYN`-K and `SMM` is the same than the maximum running time of `UPSIDE`.

C.3 Model selection

We train all methods with a grid search over the set of hyperparameters described in App. C.2, for multiple seeds, which we call *unsupervised seeds*, to evaluate robustness over both the initialization of model weights and randomness of the algorithm. For each unsupervised seed, we select the set of hyperparameters that has maximum value for the criterion of number of skills for `UPSIDE`, `DIAYN`-curriculum and for the criterion of average discriminability for `DIAYN`-K and `SMM`.

With this set of hyperparameters per seeds, we can then report some measurement, e.g. coverage, averaged over unsupervised seeds.

C.4 Evaluation protocol

1. We train the method in its unsupervised phase.
2. We then do model selection as explained in App. C.3, which gives a model per method per unsupervised seed.
3. We rollout N episodes per model and compute coverage as explained in the main paper in Sect. 5. Coverage is averaged over unsupervised seeds.
4. For each model (associated to a method) and unsupervised seed, we run the downstream tasks (as explained in App. C.5), with the same grid search over hyperparameters, with additional seeds, which we call *downstream seeds*.
5. For each method and unsupervised seed, we do model selection over downstream seeds on the criterion of reward.
6. We plot the reward averaged over unsupervised and downstream seeds, with error bars for each method.

C.5 Downstream task scenario in Mazes

We consider the downstream task of quickly finding and then reliably reaching an unknown goal, summarized in Alg. 3. There exists a goal region $\mathcal{G}$ with unknown coordinates $(x_{\mathcal{G}}, y_{\mathcal{G}})$ that can be identified only once it is reached. The unknown nature of the goal and its sparse identification signal (i.e., reward $r_{\mathcal{G}}(s) = \mathbb{1}[s \in \mathcal{G}]$) makes the problem challenging, as the agent must perform "blind" and exhaustive exploration so as to encounter the goal as quickly as possible. UPSIDE's clustering of the state space with its ability to navigate efficiently to any given cluster is a desirable property to tackle this problem. In Alg. 3, we uniformly sample the nodes of the tree (i.e., execute the diffusing part of each skill) until the goal is found. Note that we use a budget of K iterations (which could be either environment interactions or time) for UPSIDE to find the goal with the tree, otherwise we train a policy with SAC on the reward.

Once the goal is identified, this becomes a standard goal-oriented task, where no distance-to-goal is available, i.e., the reward signal is *sparse*, which makes the learning problem more difficult. The design of UPSIDE enables to identify the closest skill to the goal according to the learned discriminator, and we then fine-tune its diffusing part into a goal-oriented policy, as shown in Alg. 4.

The same approach is used for DIAYN and SMM. For SAC, a plain policy is trained directly on the reward signal.

We thus see that this task calls for a dual property of coverage and directedness.

Goals g were sampled uniformly in the available state-space, but for the sake of simplicity, we only show in Section 5 two representative goal positions, a moderately close goal and a far goal. The goal region is a circle with radius 1, thus the agent gets rewarded 1, when $\|s - g\|_2^2 < 1$.

Algorithm 3: Unknown goal

Input: Unknown goal region $\mathcal{G}$, Budget K .

for K iterations **do** `// Find` $\mathcal{G}$
- Sample z in $\mathcal{T}$ at random
- Extract the sequence of nodes $z_{(1)}, \ldots, z$ in $\mathcal{T}$ leading to z
- Execute the composed (directed part) policy $(\pi_{z_{(1)}}, \ldots, \pi_z)$ followed by the diffusing part of z
- Stop if $\mathcal{G}$ is reached.

if $\mathcal{G}$ was found **then**
- Run Alg. 4 with goal $\mathcal{G}$

else
- Train SAC policy on the reward

Algorithm 4: Known goal

Input: Known goal region $\mathcal{G}$.

Compute skill-node

$$z^* = \arg\max_{z \in \mathcal{T}} \sum_{g \in \mathcal{G}} q_\phi(z|g).$$

Fine-tune the diffusing part of skill-node z^* via RL with reward $r_{\mathcal{G}}(s) = \mathbb{1}[s \in \mathcal{G}]$.

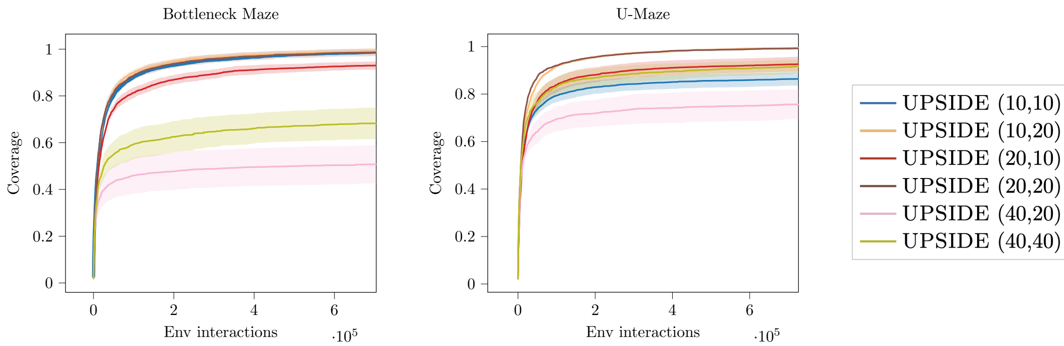


Figure 8: Ablation on the values of T and H for `UPSIDE` on the bottleneck and U-maze.

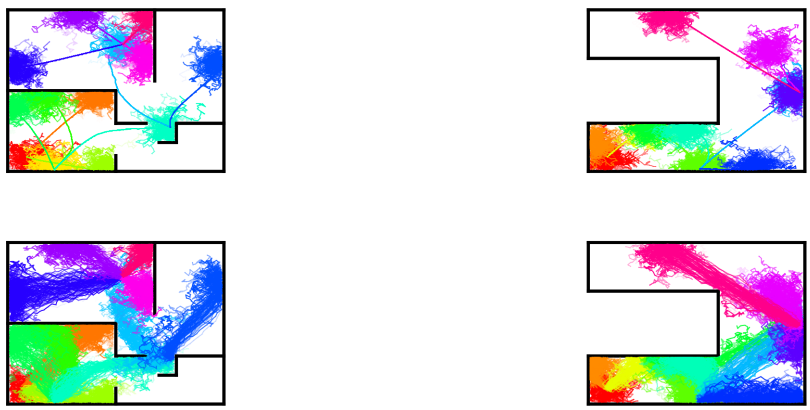


Figure 9: Difficulty of `UPSIDE` to cover the mazes if the hyperparameters T, H are set too large w.r.t. the environment size (here, $T = H = 40$, and we recall that the mazes are of size 50×50). Top (resp. bottom) row corresponds to the stochastic (resp. deterministic) executions of the policies of the directed parts of the skills.

D Additional Experiments

In this section, we report additional experiments. We ran all methods with 3 unsupervised seeds for each set of hyperparameters. All plots are generated according to the evaluation protocol explained in App. C.4.

D.1 Ablation on the skill lengths T and H

We investigate the sensitiveness of `UPSIDE` w.r.t. T and H, the lengths of the directed and diffusing part of the skill, respectively. Fig. 8 shows that the method is quite robust to reasonable choices of T and H, although there exists configurations where `UPSIDE` does not achieve full coverage, in particular in the bottleneck maze when T and H are too large (e.g., $T = 40, H = 20$), see also Fig. 9. This makes sense as the environments require "narrow" exploration (e.g., the bottleneck region that the agent must "escape" from is quite small), thus composing disproportionately long skills may hinder the coverage. Moreover, increasing T and H makes the RL training longer and more challenging (e.g., the reward is more delayed).

D.2 Visual example how the tree learned by `UPSIDE` fits the environment

We investigate the adaptivity (w.r.t. the input branching factor) of the tree structure of `UPSIDE` and illustrate that it can properly fit the unknown environment. As demonstrated in 10, `UPSIDE` successfully covers a large part of the tree maze, which is quite hard to explore given its narrow corridors. Here $T = 5$ and $H = 10$, and the branching factor N_0 is set to 3. In the terminal region

Figure 10: *(Left)* Unbalanced tree-shaped maze and *(Right)* the tree structure learned by UPSIDE. We see that it can successfully *map* the underlying structure of the unknown environment.

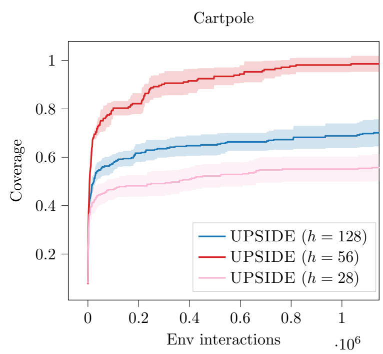


Figure 11: Ablation on h, the number of hidden units per layer of the discriminator of UPSIDE

of skill 1 (yellow), it is crucial to consolidate two skills 2 and 3 so that the tree can grow in both directions. While the tree may have expanded two skills 4 and 5 straight from 3, we see that the skill 4 (blue) overlaps with the intersection of the two small corridors, thus it is the only one sufficiently discriminable at this tree level, and UPSIDE covers the bottom right corridor in the subsequent level (i.e., from skill 4 to skill 5 in purple).

D.3 Ablation on the capacity of the discriminator

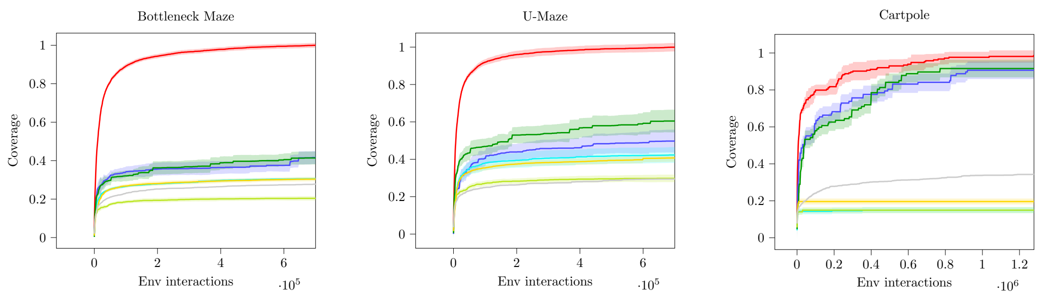
On CartPole, we found that it was quite easy for the discriminator to separate skills, though they had close behaviors "visually". This can be explained by the fact that high-dimensional states are easier to discriminate. By reducing the capacity of the discriminator, skills would be naturally forced to be more "diverse" and avoid overfitting to certain state space regions. To verify this claim, we perform an ablation on the number of hidden units per layer of the discriminator (Fig. 11), which reveals that there is a sweet spot of hidden size where coverage is the best. When the hidden size h is too big (128 or 256 in the main paper), many skills (more) are consolidated, but not diverse in their behavior, thus the coverage is not that large. On the other hand, when h is too small, it is too hard to discriminate between skills.

D.4 Results with more unsupervised seeds

In Fig. 12 we add results with 3 new unsupervised seeds per method and set of hyper-parameters. This complements Fig. 4 and 5 from the main paper by adding error bars. Compared to the main paper, training time was increased, thus explaining the slight differences in performance (e.g., UPSIDE, DIAYN-50, SMM-50 improve compared to the random policy thanks to training time).

D.5 Average discriminator performance on Mazes and Cartpole

Fig. 13 reports the average discriminability of the skills (UPSIDE, DIAYN-curriculum and DIAYN-50) during training in Bottleneck maze, U-maze and Cartpole. We make the same observation as for Reacher (see rightmost plot of Fig 5). The DIAYN-50 skills (green) suffer from low discriminability,



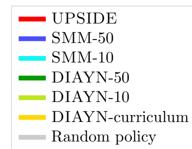


Figure 12: Normalized coverage in U-maze and bottleneck, averaged over 3 unsupervised seeds.

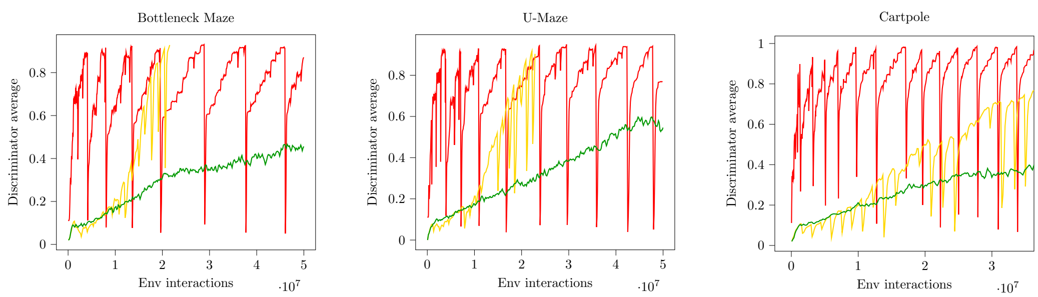


Figure 13: Average discriminability of the skills during training in Bottleneck maze, U-maze and Cartpole

while UPSIDE (red) (as well as DIAYN-curriculum in yellow) achieves much higher discriminability by removing redundant skills.

E An interpretation of our optimization problem

In this section we provide a theoretically grounded interpretation of the optimization problem solved by UPSIDE in Sect. 3 and the extent to which it allows to tackle two downstream scenarios: known goal (Alg. 4) or unknown goal (Alg. 3). Throughout App. E, we consider that the MDP M is finite-state, finite-action and communicating [35] (i.e., for every pair of states (s, s'), there exists a deterministic stationary policy under which s' is accessible from s in finite time with non-zero probability).

E.1 Preliminaries

First we review some concepts and define notation. For any stationary policy π and pair of states (s, s'), we denote by $\tau_\pi(s, s')$ the (possibly infinite) random variable of the hitting time of state s' starting from state s following policy π. We then define the *distance* d_π as the expected hitting time of policy π, i.e.,

$$\tau_\pi(s, s') := \inf\{t \geq 0 : s_{t+1} = s' \mid s_1 = s, \pi\}, \qquad d_\pi(s, s') := \mathbb{E}[\tau_\pi(s, s')],$$

where the expectation is w.r.t. the random sequence of states generated by executing π starting from state s. In addition, given a starting state s_0 and distance d, we define the *Max-Distance* as

$$D\big(d, M, s_0\big) := \max_{s \in \mathcal{S}} d(s_0, s).$$

We can instantiate two commonly considered distances.

- First, the *random-walk distance* is $d_{\mathtt{RW}}(s, s') := d_{\pi_{\mathtt{RW}}}(s, s')$, where $\pi_{\mathtt{RW}}$ denotes a uniformly stochastic policy (i.e., whose executed actions are uniformly distributed, independently of the state of the MDP). Note that for $d \leftarrow d_{\mathtt{RW}}$, D corresponds to the *cover time* starting from s_0. This notion of complexity measures how hard it is, in expectation, to cover the entire state space of the MDP following a uniformly stochastic policy starting from s_0. It was studied in e.g., [26, 22, 11], leveraging graph theory [9].
- Second, the *shortest-path distance* is $d_{\mathtt{SP}}(s, s') := \min_\pi d_\pi(s, s')$ (the minimum can be taken over the set of stationary deterministic policies [7]). Note that for $d \leftarrow d_{\mathtt{SP}}$, D corresponds to the *diameter* of the MDP [21, 40] and characterizes the complexity of navigating the state space starting from s_0 following the set of shortest-path policies. Note that for any state s', $d_{\mathtt{SP}}(\cdot, s')$ is the optimal value function under (undiscounted) reward function $r(s) = -\mathbb{1}[s \neq s']$. As such, numerous methods in goal-conditioned RL (explicitly or implicitly) target the set of policies that minimize $d_{\mathtt{SP}}$ (or variants of it, such as discounted or horizon-truncated) [e.g., 2, 34].

E.2 Interpretation

An interpretation of our approach is that it performs **clustering** over the state space based on **two different distance functions**:

- ***A tight and difficult-to-deploy distance*** $d_\star$**.** The tightest metric possible is to consider $d_\star \leftarrow d_{\mathtt{SP}}$, the shortest-path distance.
- ***A coarse and cheap-to-deploy distance*** d_+**.** For example we can consider $d_+ \leftarrow d_{\mathtt{RW}}$, the random-walk distance.

Specifically, our approach can be interpreted as seeking to minimize the intra-cluster distance d_+ while maximizing the inter-cluster distance $d_\star$. Although d_+ is coarser than $d_\star$, it had the advantage of being easier to execute the policy to which it corresponds (e.g., a random policy). The implicit assumption that we make is that d_+ is a *decent enough proxy* for $d_\star$ for *small horizon*, although it degrades sharply as the horizon increases.

In App. E.3 we analyze a simplified structure of our algorithm UPSIDE which allows us to theoretically analyze the extent to which UPSIDE can tackle the two downstream scenarios explained in App. C.5, depending on the environment's properties. For simplicity we will consider the "flat case" of UPSIDE (see Rmk. 2 for a discussion on the extension to the tree case). Before analyzing the downstream scenarios (App. E.4.1), we begin by analyzing the properties of the directed part (App. E.3.1) and the diffusing part (App. E.4) of each UPSIDE skill.

E.3 An analysis of two downstream scenarios tackled by UPSIDE

E.3.1 Directed part of each UPSIDE skill

Structure/Assumptions.

- The directed part of each skill k is characterized by a pair (π_k, c_k) where the policy π_k is of length T and aims to attain a goal state $c_k \in \mathcal{S}$ (chosen by the skill).
- From the optimization problem of UPSIDE, each skill is η-consolidated according to the discriminator. We consider that the latter discriminates between the goals $\{c_k\}_{k\in[K]}$ given the current state. We then have that $\min_{k\in[K]} q_\phi(c_k|s_T) \geq \eta$.
- Finally, we assume that the predictions of the discriminator can serve as $\varepsilon_{\text{discr}}$-accurate approximations of the probability of π_k reaching the centroid c_k within its length of T steps, where $0 \leq \varepsilon_{\text{discr}} < \eta$. This implicitly assumes that we can connect the discriminability property and the directedness property (respectively appearing in the reverse and forward forms of MI).

We first notice that the directed part π_k has an intrinsic reward signal that approximately targets a goal-directed behavior. Indeed, as argued in [34, App. E], having an intrinsic reward signal of $r_z(s)$ scaling as $p(c_k|s)$ would amount to learning a goal-oriented policy with goal c_k. In particular, the optimal non-episodic policy $\pi^\dagger$ that minimizes $\mathbb{E}\big[\sum_{t=1}^{+\infty}(1+\beta\mathcal{H}(\pi(\cdot|s_t)))\mathbb{1}[s_t \neq c_k]\big]$ induces a distance-to-goal of $d_{\pi^\dagger}(s, c_k) \leq (1+\beta \log A) d_{\texttt{SP}}(s, c_k)$, i.e., it targets the shortest path up to an entropy bias. However, algorithmically, the directed parts are of length T and the UPSIDE skills reset every $T+H$ steps. This episodic nature introduces a bias w.r.t. the optimal shortest-path behavior that is non-trivial to analyze and bound.

We now show that thanks to the constraint in the optimization problem of UPSIDE and by our assumption on the connection between the discriminability property and the directeness property, we can recover goal-directed properties for each first part of skills output by UPSIDE.

Lemma 1. *Any pair* (π_k, c_k) *output by* UPSIDE *verifies*

$$d_{\pi_k}(s_0, c_k) \leq \frac{T+H+1-\eta+\varepsilon_{\text{discr}}}{\eta-\varepsilon_{\text{discr}}}.$$

Proof. Recall that the skill k is episodic of length $T+H$, i.e., it resets to s_0 every $T+H$ time steps. We denote by $d_\pi^{(T+H)}$ the total number of steps before reaching either the skill's centroid or $T+H$ steps, and by $f_\pi^{(T+H)}$ the probability of failure to reach the centroid within $T+H$ steps. Then

$$d_{\pi_k}(s_0, c_k) \overset{\text{(i)}}{=} \frac{d_{\pi_k}^{(T+H)}(s_0, c_k) + f_{\pi_k}^{(T+H)}(s_0, c_k)}{1 - f_{\pi_k}^{(T+H)}(s_0, c_k)} \overset{\text{(ii)}}{\leq} \frac{T+H+f_{\pi_k}^{(T)}(s_0, c_k)}{1-f_{\pi_k}^{(T)}(s_0, c_k)},$$

where (i) comes from [25, App. B.3], (ii) uses that $d^{(T+H)} \in [0, T+H]$ and that $f_{\pi_k}^{(T+H)}(s_0, c_k) \leq f_{\pi_k}^{(T)}(s_0, c_k)$. We now approximate the probability of failure of reaching the centroid by using the predictions of the discriminator (the more expressive the discriminator, the better the approximation): $|1 - f_{\pi_k}^{(T)}(s_0, c_k) - q_\phi(c_k|s_T)| \leq \varepsilon_{\text{discr}}$. The constraint of our optimization problem ensures that the pair (π_k, c_k) output by UPSIDE satisfies $q_\phi(c_k|s_T) \geq \eta$. Therefore, it holds that

$$\frac{T+H+f_{\pi_k}^{(T)}(s_0, c_k)}{1-f_{\pi_k}^{(T)}(s_0, c_k)} \leq \frac{T+H+1-q_\phi(c_k|s_T)+\varepsilon_{\text{discr}}}{q_\phi(c_k|s_T)-\varepsilon_{\text{discr}}} \leq \frac{T+H+1-\eta+\varepsilon_{\text{discr}}}{\eta-\varepsilon_{\text{discr}}}. \quad (6)$$

□

Note that given any goal state g, having a policy π with bounded $d_\pi(\cdot, g)$ is non-trivial, since it implies that it reaches the goal with probability 1 (i.e., that it is proper [7]). Also note that the "worst-case" discriminability property in the constraint (i.e., $q_\phi(c_k|s_T) \geq \eta$) is crucial to obtain Lem. 1, since it may not be possible to guarantee it given a discriminability property verified on average (e.g., via a conditional entropy term in the MI).

E.4 Diffusing part of each UPSIDE skill

Structure/Assumptions.

- The diffusing part of skill k is of length H and is composed of a set of states radiating around c_k, which acts as a centroid for the cluster of states generated by the diffusing part. Formally, we consider that there exists $\delta > 0$ such that

$$\texttt{DIFF}(k) := \{y_k : \mathbb{P}(\tau_+(c_k, y_k) \leq H) \geq \delta\}, \tag{7}$$

where $\tau_+(s, s')$ denotes the hitting time following the policy that minimizes $d_+(s, s')$.

- According to the optimization problem solved by UPSIDE, the *clusters* associated to the K skills *saturate* the state space, i.e., we cannot consolidate an additional cluster. We propose to write this condition as

$$\forall s \in \mathcal{S},\ \exists k \in [K],\ \exists y_k \in \texttt{DIFF}(k),\ \mathbb{P}\big(\tau_+(s, y_k) \leq H\big) \geq \delta, \tag{8}$$

otherwise from (7) it would be possible to consolidate an additional cluster with centroid s.

- Finally, we spell out an assumption on the environment that we make throughout App. E.4:

Assumption 1. *There exists* $\Theta \geq 0$ *such that*

$$\forall (s, s'), \quad \mathbb{P}(\tau_+(s, s') \leq H) \geq \delta \implies d_\star(s, s') \leq H + \Theta.$$

This formalizes the assumption commonly made in goal-conditioned deep RL — either implicitly or explicitly [e.g., 14, Sect. 3.3] — that if a goal is reachable, then there exists a policy that does so reliably. Note that in the special case of a deterministic MDP we have $\Theta = 0$.

Definition 1. *We define the following "local" quantities:*

- *For any* $s \in \mathcal{S}$ *and any* $k \in [K]$, *define* $\Delta_+(s; k) := \max_{y_k \in \texttt{DIFF}(k)} |d_+(s, y_k) - d_+(y_k, s)|$.
- *For any* $s \in \mathcal{S}$ *and any* $k \in [K]$, *define* $\Delta_\star(s; k) := \max_{y_k \in \texttt{DIFF}(k)} |d_\star(s, y_k) - d_\star(y_k, s)|$.

Note that under the communicating MDP assumption, both quantities are always bounded. They measure the level of "reversibility" of the MDP w.r.t. the d_+ *and* $d_\star$ *distance, respectively. Moreover, in the special case of an MDP with locally symmetric actions, the distance* $d_\star$ *is symmetric so* $\Delta_\star = 0$.

We first derive two lemmas and then position their statements w.r.t. the two downstream objectives of UPSIDE.

Lemma 2. *It holds that*

$$\forall s \in \mathcal{S},\ \exists k \in [K]: \quad \mathbb{P}\Big(\tau_+(c_k, s) \leq H + \frac{H + \Delta_+(s; k) + \Theta}{1 - \delta}\Big) \geq \delta^2.$$

Lemma 3. *It holds that*

$$\forall s \in \mathcal{S},\ \exists k \in [K]: \quad d_\star(c_k, s) \leq 2(H + \Theta) + \Delta_\star(s; k),$$

Proof of Lem. 2. We prove the result by contradiction. Assume the contrary of Lem. 2; then there exists a state $s \in \mathcal{S}$ such that for every $k \in [K]$, $\mathbb{P}(\tau_+(c_k, s) \leq H + Z_k) < \delta^2$, with $Z_k := (H + \Delta_+(s; k) + \Theta)/(1 - \delta)$. We now use that the diffusing part of each skill k radiating around its centroid c_k is composed of states $\{y_k : \mathbb{P}(\tau_+(c_k, y_k) \leq H) \geq \delta\}$. This means that for every $k \in [K]$ and $y_k \in \texttt{DIFF}(k)$, $\mathbb{P}(\tau_+(y_k, s) \leq Z_k) < \delta$. Noticing that $d_+(y_k, s) = \mathbb{E}[\tau_+(y_k, s)]$ by definition, we get $d_+(y_k, s) > (1 - \delta) Z_k \geq H + \Theta + d_+(y_k, s) - d_+(s, y_k)$, where the last inequality comes from the definition of $\Delta_+(s; k)$. Therefore, $d_+(s, y_k) > H + \Theta$. So by contraposition of Asm. 1, $\mathbb{P}(\tau_+(s, y_k) \leq H) < \delta$. Since this is true for all $k \in [K]$ and $y_k \in \texttt{DIFF}(k)$, we get a contradiction on condition (8). □

Proof of Lem. 3. Take any state $s \in \mathcal{S}$. Case 1: $\exists k \in [K], s \in \texttt{DIFF}(k)$. Then $\mathbb{P}(\tau_+(c_k, s) \leq H) \geq \delta$. From Asm. 1 this means $d_\star(c_k, s) \leq H + \Theta$. Case 2: $\forall k \in [K], s \notin \texttt{DIFF}(k)$. Then from condition (8), there exists $k \in [K]$ and $y_k \in \texttt{DIFF}(k)$ such that $\mathbb{P}\big(\tau_+(s, y_k) \leq H\big) \geq \delta$, which implies that $d_\star(s, y_k) \leq H + \Theta$ from Asm. 1. By definition of $\Delta_\star(s; k)$, it holds that $d_\star(s_k, y) \leq H + \Theta + \Delta_\star(s; k)$. Furthermore, y_k verifies $\mathbb{P}(\tau_+(c_k, y_k) \leq H) \geq \delta$, which means from Asm. 1 that $d_\star(c_k, y_k) \leq H + \Theta$. We conclude by the triangle inequality that $d_\star(c_k, y) \leq d_\star(c_k, s_k) + d_\star(s_k, y) \leq 2(H + \Theta) + \Delta_\star(s; k)$. □

E.4.1 Analysis of two downstream scenarios tackled by `UPSIDE`

We consider the two downstream tasks detailed in App. C.5: ① finding an unknown goal (Alg. 3) and ② reliably reaching a known goal (Alg. 4).

These downstream scenarios require the ability to efficiently traverse from s_0 to any state s of the MDP. Ideally we would deploy the policy associated to $d_\star(s_0, s)$, i.e., the shortest-path policy, yet it is difficult to compute. On the other extreme, deploying the random-walk strategy is very easy yet much more inefficient, since $d_\star(s_0, s) \ll d_+(s_0, s)$. Our approach targets the following intermediate approach.

First, we upper bound using the triangle inequality

$$\max_{s \in \mathcal{S}} d_\star(s_0, s) \;\leq\; \max_{s \in \mathcal{S}} \left\{ \min_{k \in [K]} d_\star(s_0, c_k) + d_\star(c_k, s) \right\}. \tag{9}$$

Under a **zero-shot downstream set-up**, the training objective of `UPSIDE` seeks to control the following upper bound of (9)

$$\max_{s \in \mathcal{S}} \left\{ \min_{k \in [K]} d_\star(s_0, c_k) + d_+(c_k, s) \right\}. \tag{10}$$

Under a **few-shot downstream set-up**, `UPSIDE` fine-tunes the diffusing part of the skill to reach the desired goal state. As such, it targets (9).

We now distinguish between the two types of downstream scenarios.

① The unknown-goal downstream task.

From Lem. 2, whatever the unknown goal state s, there exists a skill k whose diffusing part starting from its centroid c_k can reach s with strictly positive probability, as long as it is executed long enough (with length depending in particular on the *local* quantity $\Delta_+(s; k)$).

As such, Lem. 1 and Lem. 2 prescribe the following *algorithmic strategy*: in a round-robin fashion over $k \in [K]$, execute the directed part of skill k plus its diffusing part for *increasing* lengths (i.e., starting from H and then gradually increasing it). The unknown goal should then be discovered at some point, specifically within

$$O\left(\frac{1}{\delta^2} \left(\frac{T + H + 1 - \eta + \varepsilon_{\text{discr}}}{\eta - \varepsilon_{\text{discr}}} + \frac{H + \Delta_+(s; k) + \Theta}{1 - \delta} \right) \right)$$

time steps, by combining Lem. 1 and 2.

② The known-goal downstream task.

From Lem. 3, for any known goal state $s \in \mathcal{S}$, there exists a skill $k \in [K]$ from which learning to reach the goal s can be facilitated. Indeed, the shortest-path distance from its centroid c_k to the goal s depends on the *local* quantity $\Delta_+(s; k)$ (as well as H and Θ).

As such, Lem. 1 (with (6)) and Lem. 3 prescribe the following *algorithmic strategy*: first reach the centroid c_k for which $k \in \arg\max q_\phi(c_k | s)$ (i.e., execute the directed part of skill k), and second learn to reach s from c_k (by fine-tuning the diffusing part of skill k).

Remark 1. Inspecting the quantities in Def. 1 and in Asm. 1 allows to characterize the complexity of the environment in tackling the two types of downstream tasks. In particular, we see that the complexity is reduced in environments that are close to deterministic (i.e., smaller Θ in Asm. 1) and that exhibit a "balanced / symmetric" behavior, with the least bottlenecks possible (i.e., smaller quantities in Def. 1). In addition, the size of the state space $\mathcal{S}$ and the diameter of the MDP implicitly play a role in the value of the number of clusters K required and in the choice of T, which must be large enough to ensure in Lem. 1 that $\eta > 0$ holds in the discriminator predictions.

Remark 2. In the tree case of `UPSIDE`, T does not have to be large enough (as needed in the flat case) since the state space may be covered by sequentially composing the directed parts of the skills of length T. The equations from the flat case would look the same as in the flat case, yet two quantities would be replaced: the probability of success of reaching the centroid of each cluster of skill-node n would go from η to $\eta^{d(n)}$ where $d(n)$ is the depth of skill-node n, and the length of the skill would go from $T + H$ to $d(n)T + H$.