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

Pierre-Alexandre Kamienny [*1 2] Jean Tarbouriech [*1 3] Alessandro Lazaric [1] Ludovic Denoyer [1]

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 enable to efficiently solve sparse-reward downstream tasks in navigation environments, comparing favorably with existing baselines.

[*]Equal contribution [1]Facebook AI Research [2]Sorbonne Université, LIP6 [3]Inria, Scool team. Correspondence to: Pierre-Alexandre Kamienny <pakamienny@fb.com>.

ICML 2021 Workshop on Unsupervised Reinforcement Learning. Copyright 2021 by the author(s).

1. Introduction

Deep reinforcement learning (RL) algorithms have been shown to effectively solve a wide variety of complex problems (e.g., Mnih et al., 2015; Bellemare et al., 2013; Silver et al., 2017; Gu et al., 2017; Andrychowicz et al., 2017; Schulman et al., 2017). 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., Gregor et al., 2016). Specifically, given the state variable S and some variables Z on which the skill policies are conditioned, MI is defined as

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

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 reliably 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

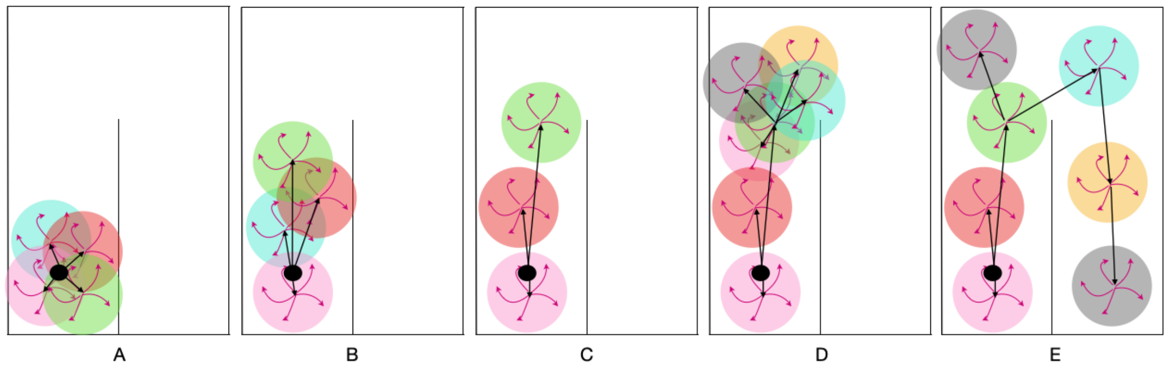

Figure 1: Overview of `UPSIDE`. The black dot corresponds to the initial state s_0. *(A)* A set of random skills is initialized, each skill being composed of a *directed* part (illustrated as a black arrow) and a *diffusing* part (red arrows) which induces a local coverage (colored circles). *(B)* The 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 App. A.

while being discriminable.

Maximizing (1) has been shown to be a powerful approach for encouraging exploration in RL (Houthooft et al., 2016; Mohamed & Rezende, 2015) and for unsupervised skill discovery (e.g., Gregor et al., 2016; Eysenbach et al., 2019; Achiam et al., 2018; Sharma et al., 2020; Campos et al., 2020). 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 navigation environments 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., Gregor et al., 2016). 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

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

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., Gregor et al., 2016; Eysenbach et al., 2019; Campos et al., 2020).

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., (Eysenbach et al., 2019; Sharma et al., 2020; Zhang et al., 2021), 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. Indeed 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).

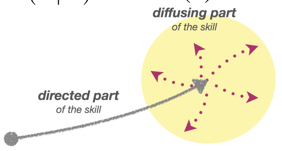

Figure 2: Directed and diffusing parts of the skill.

Similar to prior work (e.g., Gregor et al., 2016; Eysenbach et al., 2019), 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 $\mathbb{E}_{\pi(z)}\left[\sum_{t=T+1}^{T+H} r_z(s_t)\right]$ 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 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 has been shown (e.g., Campos et al., 2020) 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

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

[2] Although (Gregor et al., 2016; Eysenbach et al., 2019) 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 (Warde-Farley et al., 2019; Baumli et al., 2021). 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.

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 $q_\phi$ 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 $z_{(1)}, \ldots, z$ in $\mathcal{T}$ leading to $z$
            Execute the composed (directed part) policy $(\pi_{z_{(1)}}, \ldots, \pi_z)$ followed by the diffusing part
            Add states observed during the diffusion part to state buffer $\mathcal{B}_z$
            Update discriminator $q_\phi$ with SGD on $\mathcal{B}_z$ to predict label $z$
            if $z \in \mathcal{C}(w)$ then // Update only new policies, other policies kept fixed
                Update policy $\pi_z$ using SAC to optimize the discriminator reward as in Sect. 3.1.
        For all $z \in \mathcal{C}(w)$ compute the skill-discriminability $d(z) = \widehat{q}_\phi^{(\mathcal{B})}(z) = \frac{1}{|\mathcal{B}_z|}\sum_{s \in \mathcal{B}_z} q_\phi(z|s)$.
        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 $\mathcal{C}(w)$
```

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

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

$$\geq \min_{z \in Z} \mathbb{E}_{s_{\text{diff}}} [\log p(z|s_{\text{diff}})], \quad (3)$$

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

$$\max_{|Z|=N,\rho} \left\{ \mathcal{H}(Z) + \min_{z \in [N]} \mathbb{E}_{s_{\text{diff}}} [\log p(z|s_{\text{diff}})] \right\}, \quad (4)$$

where with an abuse of notation we use $z \in [N]$ to denote all skills in a set Z with cardinality N. Since (4) 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 (4) 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:

$$\max_{N \geq 1} \log(N) \;\; \text{s.t.} \;\; \min_{z \in [N]} \mathbb{E}_{s_{\text{diff}}} [\log p(z|s_{\text{diff}})] \geq \log \eta. \quad (5)$$

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, (5) 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 are guaranteed to display sufficient discriminability. Alternatively, we can interpret (5) 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

[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.

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.

3.3. Composing Skills in a Tree Structure

The MI optimization problem as well as our constrained variant (5) 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 requires 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 (5) 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 Alg. 1 for the detailed pseudo-code).

We perform standard approximations to make the constraint in (5) easier to estimate. We approximate the unknown posterior $p(z|s)$ with a learned discriminator $q_\phi(z|s)$ with parameters ϕ. We also 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 (5) leads to

$$\max_{N\geq 1,\pi} N \qquad \text{s.t.} \quad \min_{z\in[N]} \widehat{q}_\phi^{(B)}(z) \geq \eta. \tag{6}$$

As discussed in Sect. 3.2, this problem can be conveniently optimized using a greedy strategy. We then integrate the optimization of (6) 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 been 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 19). 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., Campos et al., 2020), 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

[4]See e.g., the discussion in (Mutti et al., 2021) on the "importance of properly choosing the training horizon in accordance with the downstream-task horizon the policy will eventually face."

into reusable knowledge to solve downstream tasks. This includes both off-policy model-free (e.g., Pong et al., 2020) and model-based (e.g., Sekar et al., 2020) methods that seek to populate a representative replay buffer and build accurate value or model estimates, which 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., Yarats et al., 2021). 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` (Gregor et al., 2016), 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` (Eysenbach et al., 2019) 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` (Sharma et al., 2020) 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` (Campos et al., 2020) 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` (Lee et al., 2019), 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 (Hazan et al., 2019), and as a way to encourage skills to go through unknown state regions. Other MI-based skill discovery methods include (Florensa et al., 2017; Hansen et al., 2019; Modhe et al., 2020; Baumli et al., 2021; Xie et al., 2021), as well as (Xu et al., 2020; Lu et al., 2020) which investigate skill discovery in non-episodic settings.

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 (Achiam et al., 2018; Aubret et al., 2020). We consider a similar discriminability criterion and cast it as a proxy to measure the skills' amenability to be composed. By enforcing a discriminability constraint in our optimization problem, we maintain skills that can be readily composed along a tree structure. The latter can either increase or decrease the support of available skills depending on the region of the state space.

Recently, (Zhang et al., 2021) propose a hierarchical RL method that discovers abstract and task-agnostic skills while jointly learning a higher-level policy that 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` (Warde-Farley et al., 2019) and `Skew-Fit` (Pong et al., 2020) learn a goal-conditioned policy in an unsupervised way with an MI objective. As explained in (Campos et al., 2020, 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 MDP. This includes eigenoptions (Machado et al., 2017; 2018) and covering options (Jinnai et al., 2019; 2020), as well as the algorithm of (Bagaria et al., 2021) 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 perform our experiments based on the RLStructures framework (Denoyer et al., 2021). We consider navigation problems in fully continuous 2D mazes, where the agent observes its current position and outputs actions that control its location, which is affected by collisions with walls.

We compare to different baselines. `DIAYN-K`, where K is

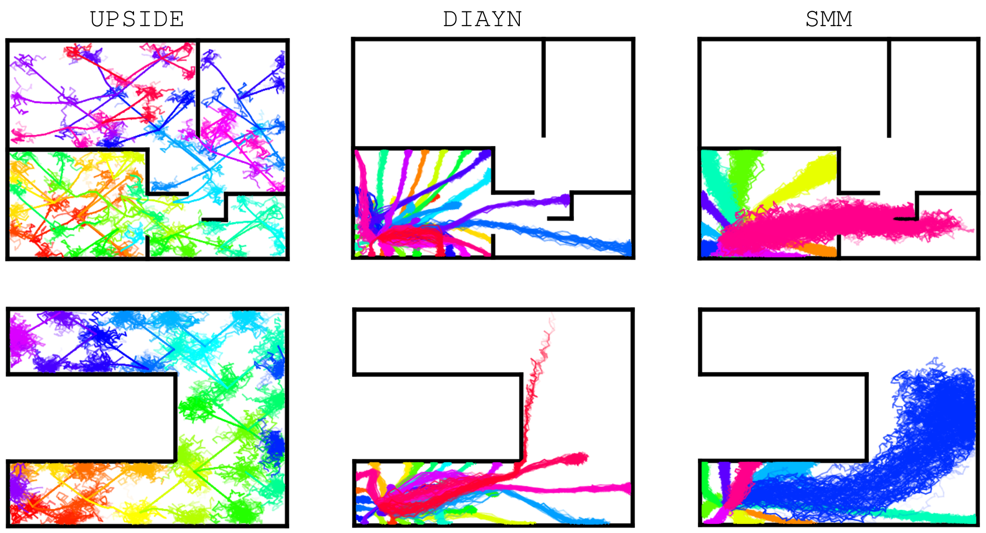

Figure 3: UPSIDE, DIAYN-curriculum and SMM-10 skills learned in a bottleneck maze *(Top)* and a U-maze *(Bottom)*. For both DIAYN and SMM we report the stochastic execution of the learned skills and for UPSIDE we report the deterministic directed parts (that are composed) followed by the (stochastic)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

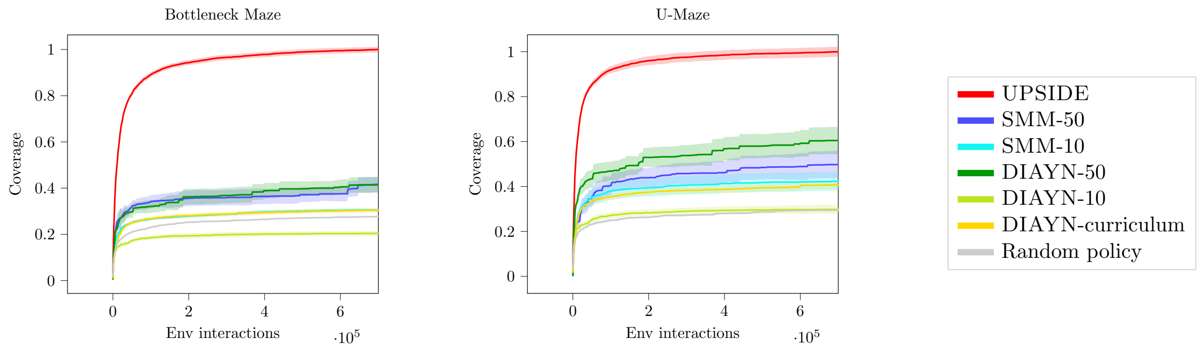

Figure 4: Normalized coverage in U-maze and bottleneck, averaged over 3 seeds.

a fixed number of skills, is the original algorithm proposed in (Eysenbach et al., 2019). 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 (Lee et al., 2019), 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, which is more computationally efficient and stable than the original VAE-based method in the environments that we consider. UPSIDE and all baselines are implemented with Soft-Actor Critic (SAC) (Haarnoja et al., 2018).

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, or 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) 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) 50×50 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 Campos et al., 2020, 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-

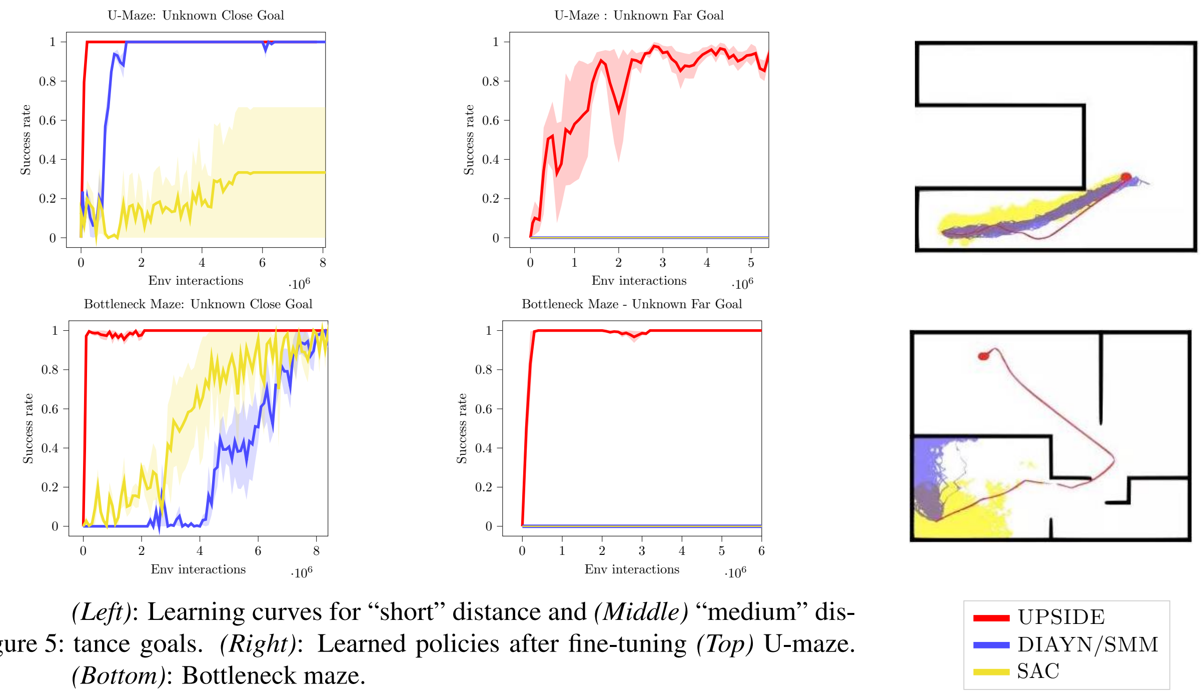

Figure 5: *(Left)*: Learning curves for "short" distance and *(Middle)* "medium" distance goals. *(Right)*: Learned policies after fine-tuning *(Top)* U-maze. *(Bottom)*: Bottleneck maze.

entirety of the state space by creating a tree of directed skills. Moreover, UPSIDE creates directed skills with a low entropy, while the two baselines tend to create skills that are more stochastic. This is particularly evident for SMM, whose state-entropy exploration bonus encourages broader coverage yet 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 quite robust to these parameters as shown in App. D.

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 consider a sparse positive reward when reaching a particular defined goal.[5] We consider goals at different distances from the initial state s_0, the further, the harder. Fig. 5 shows the learning curves obtained when fine-tuning the best skill for the different models and compares to a classical SAC algorithm where a single policy is learned from scratch. DIAYN/SMM signifies that 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. 5.

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

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.

References

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

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

Aubret, A., Matignon, L., and Hassas, S. Elsim: End-to-end learning of reusable skills through intrinsic motivation. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2020.

Bagaria, A., Senthil, J. K., and Konidaris, G. Skill discovery for exploration and planning using deep skill graphs. In *International Conference on Machine Learning*, pp. 521–531. PMLR, 2021.

Baumli, K., Warde-Farley, D., Hansen, S., and Mnih, V. Relative variational intrinsic control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6732–6740, 2021.

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

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

Denoyer, L., Rothermel, D., and Martinet, X. RLStructures - A simple library for RL research. `https://gitHub.com/facebookresearch/rlstructures`, 2021.

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

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

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

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

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

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

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

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

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

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

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

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

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

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

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

Modhe, N., Chattopadhyay, P., Sharma, M., Das, A., Parikh, D., Batra, D., and Vedantam, R. 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*, 2020.

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

Mutti, M., Pratissoli, L., and Restelli, M. Task-agnostic exploration via policy gradient of a non-parametric state entropy estimate. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9028–9036, 2021.

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

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

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

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

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

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

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

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

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Reinforcement learning with prototypical representations. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 11920–11931. PMLR, 2021.

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

A. UPSIDE Algorithm

We provide a diagram of the high-level approach of UPSIDE in Fig. 6 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 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., Eysenbach et al., 2019), 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 (6) 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.

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 (Eysenbach et al., 2019) 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 (Eysenbach et al., 2019). 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. 1, lines 20-29). Note that the increasing version of this curriculum is similar to the one proposed in VALOR (Achiam et al., 2018, Sect. 3.3).

SMM. We used SMM (Lee et al., 2019) as it is state-of-art in terms of coverage, at least on long-horizon control problems, although (Campos et al., 2020) 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 maze

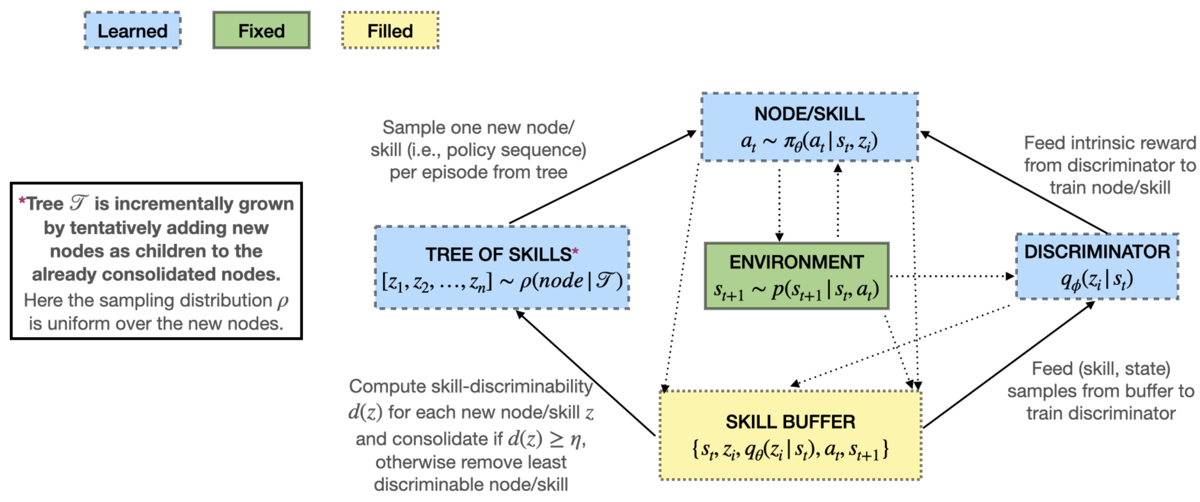
Learned
Fixed
Filled
*Tree $\mathscr{T}$ is incrementally grown by tentatively adding new nodes as children to the already consolidated nodes.
Here the sampling distribution ρ is uniform over the new nodes.
Sample one new node/skill (i.e., policy sequence) per episode from tree
TREE OF SKILLS*
$[z_1, z_2, \ldots, z_n] \sim \rho(node \mid \mathscr{T})$
NODE/SKILL
$a_t \sim \pi_\theta(a_t \mid s_t, z_i)$
ENVIRONMENT
$s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$
DISCRIMINATOR
$q_\phi(z_i \mid s_t)$
Feed intrinsic reward from discriminator to train node/skill
Compute skill-discriminability $d(z)$ for each new node/skill z and consolidate if $d(z) \geq \eta$, otherwise remove least discriminable node/skill
SKILL BUFFER
$\{s_t, z_i, q_\theta(z_i \mid s_t), a_t, s_{t+1}\}$
Feed (skill, state) samples from buffer to train discriminator

Figure 6: High-level approach of UPSIDE

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 $N_{\text{discr}}$ to train the discriminator, ratio $\mu$ 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
    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 $\pi_z$ and buffers $\mathcal{B}_z$, $\forall z \in \mathcal{C}(w)$
    Initialize discriminator $q_\phi$ with $|\mathcal{T}|$ classes
    Continue = true; Saturated = false
    while Continue do
        for $K$ iterations do
            for $r \in [\![1, R]\!]$ do // collect $R$ trajectories
                Sample a skill $z$ from $\mathcal{C}(w)$ 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
                Add states observed during the diffusing part to $\mathcal{B}_z$
            $B = \{\}$ // Initialize batch to update the discriminator
            while $|B| < N_{discr}$ do
                Sample a skill $z$ from $\mathcal{T}$ w.p. $(\mathbb{I}[z \in |\mathcal{C}(w)|] + \mu\mathbb{I}[z \notin |\mathcal{C}(w)|])/((1-\mu)|\mathcal{C}(w)| + \mu|\mathcal{T}|)$
                Sample a state $s$ from the last $\mathcal{V}$ states of $\mathcal{B}_z$
                Add $(s, z)$ to $B$
            Update discriminator $q_\phi$ with SGD on $B$ to predict label $z$
            for $z \in \mathcal{C}(w)$ do
                Update policy $\pi_z$ using SAC to optimize the discriminator reward as in Sect. 3.1.
        Compute the skill-discriminability $d(z) = \widehat{q}_\phi^{(B)}(z) = \frac{1}{|B|}\sum_{(s,z)\in B} q_\phi(z|s)$ 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 $\mathcal{T}$
            if $|\mathcal{C}(w)| = N_{\max}$ then
                Set Saturate = true
        else
            Set Continue = false
    Enqueue in $\mathcal{W}$ the consolidated nodes $\mathcal{C}(w)$
```

Algorithm 3: Unknown goal

```
Input: Unknown goal region $\mathcal{G}$, Budget $\kappa$.
for $\kappa$ iterations do // Find $\mathcal{G}$
    Sample $z$ in $\mathcal{T}$ 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 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
```

domains that we consider. As we use the cartesian (x, y) 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 Lee et al., 2019).

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 (Haarnoja et al., 2018). 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.

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: $\gamma = 0.99$

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 $r_{\mathcal{G}}(s) = \mathbb{1}[s \in \mathcal{G}]$.
```

- 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: $1e6$
- $h = \{16, 64\}$ hidden units per layer for policy, and $h = 128$ hidden units per layer for discriminator

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. Downstream task scenario

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

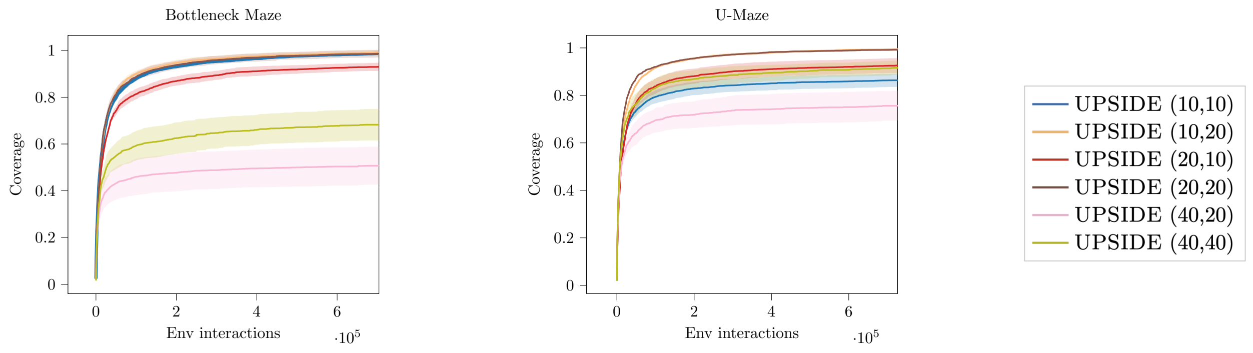

Figure 7: Ablation on the values of T and H for `UPSIDE` on the bottleneck and U-maze.

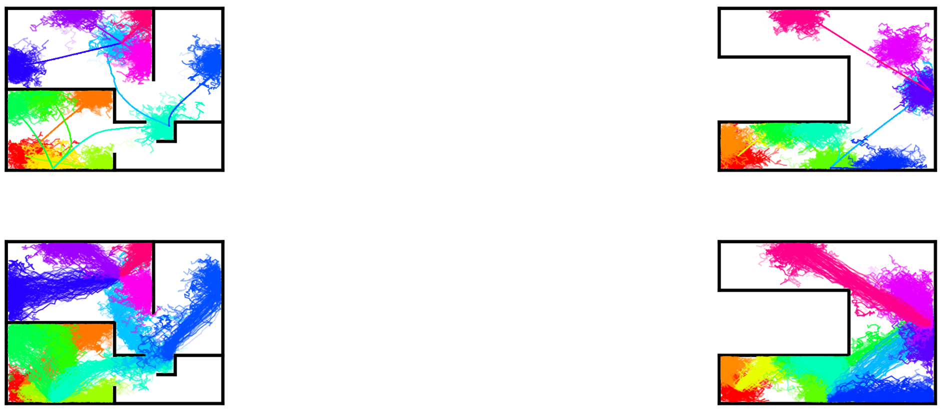

Figure 8: Difficulty of `UPSIDE` to cover the mazes if the hyperparameters T, H are set too large w.r.t. the environment size (here, $T = H = 40$, and we recall that the mazes are of size 50×50). Top (resp. bottom) row corresponds to the stochastic (resp. deterministic) executions of the policies of the directed parts of the skills.

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 κ 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 Sect. 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 a reward of 1 at state s if $\|s - g\|_2^2 < 1$.

C.5. 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 (explained in App. C.4), with the same grid search over hyperparameters, with additional seeds, which we call *downstream seeds*.

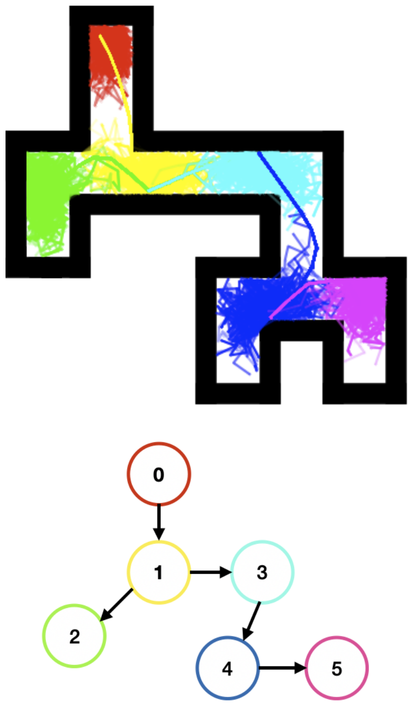

Figure 9: *(Top)* Unbalanced tree-shaped maze and *(Bottom)* the tree structure learned by UPSIDE. We see that it can successfully *map* the environment's underlying structure.

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.

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.5.

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. 7 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. 8. 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).

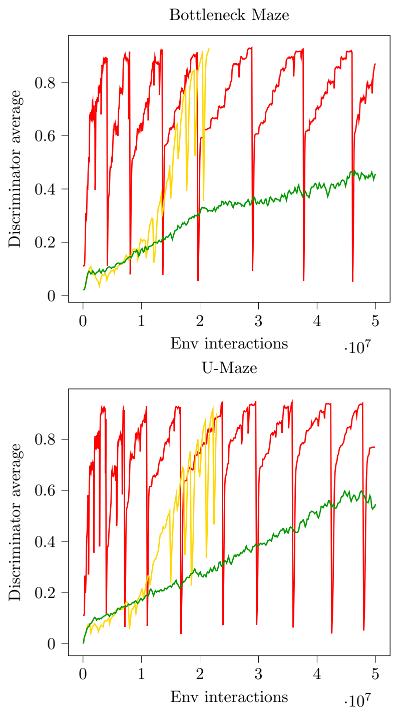

Figure 10: Average discriminability of the skills during training in Bottleneck maze and U-maze.

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 Fig. 9, 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 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. Average discriminator performance

Fig. 10 reports the average discriminability of the skills (UPSIDE, DIAYN-curriculum and DIAYN-50) during training in Bottleneck maze and U-maze. We observe that the DIAYN-50 skills (green) suffer from low discriminability, while UPSIDE (red) (as well as DIAYN-curriculum in yellow) achieves much higher discriminability by removing redundant skills.