Intrinsically Motivated Discovery of Temporally Abstract Graph-based Models of the World

Akhil Bagaria, Anita de Mello Koch, Rafael Rodriguez-Sanchez, Sam Lobel, George Konidaris

Keywords: Hierarchical RL, model-based RL, exploration.

Summary

We seek to design reinforcement learning agents that build plannable models of the world that are abstract in both state and time. We propose a new algorithm to construct a *skill graph*; nodes in the skill graph represent abstract states and edges represent skill policies. Previous works that learn a skill graph use random sampling from the state-space and nearest-neighbor search—operations that are infeasible in environments with high-dimensional observations (for example, images). Furthermore, previous algorithms attempt to increase the probability of all edges (by repeatedly executing the corresponding skills) so that the resulting graph is robust and reliable everywhere. However, exhaustive coverage is infeasible in large environments, and agents should prioritize practicing skills that are more likely to result in higher reward. We propose a method to build skill graphs that aids exploration, without assuming state-sampling, distance metrics, or demanding exhaustive coverage.

Contribution(s)

1. We provide an algorithm that learns a graph-based, plannable abstraction of the environment, even when the observations are high-dimensional; for example, images.
   Context: Prior work learned graph abstractions by assuming a distance metric over the state-space (for example, Bagaria et al. (2021b)), and hence cannot be easily applied to environments with image-based observations.
2. We use ideas from Intrinsic Motivation to design a graph construction algorithm that serves as a high-level exploration objective without needing to learn an accurate one-step model of the world—the drive to build the skill graph allows the agent to solve five challenging exploration problems, directly from pixels.
   Context: Most prior work in model-based exploration either operates in non-image based domains (for example, Sharma et al. (2020b)), or require the agent to learn one-step models (for example, Hafner et al. (2022); Mendonca et al. (2021)).
3. We provide a method for converting a goal-conditioned value function into a plannable abstract world model, which allows us to use dynamic programming to determine which option to execute at each state.
   Context: Lo et al. (2024) also study this problem, but they assume that option subgoals and initiation regions are provided to the agent; furthermore, they use the model for reward-shaping (Ng et al., 1999) rather than for option selection at decision time.

Intrinsically Motivated Discovery of Temporally Abstract Graph-based Models of the World

Akhil Bagaria[1†], **Anita de Mello Koch**[2], **Rafael Rodriguez-Sanchez**[2], **Sam Lobel**[2], **George Konidaris**[2]
akhilbg@amazon.com,
{anita_de_mello_koch,rafael_rodriguez,slobel,gdk}@brown.edu

[1]**Amazon, New York, USA**
[2]**Department of Computing Science, Brown University, Providence, Rhode Island, USA**

[†] Work done while at Brown University

Abstract

We seek to design reinforcement learning agents that build plannable models of the world that are abstract in both state and time. We propose a new algorithm to construct a *skill graph*; nodes in the skill graph represent abstract states and edges represent skill policies. Previous works that learn a skill graph use random sampling from the state-space and nearest-neighbor search: operations that are infeasible in environments with high-dimensional observations (for example, images). Furthermore, previous algorithms attempt to increase the probability of all edges (by repeatedly executing the corresponding skills) so that the resulting graph is robust and reliable everywhere. However, exhaustive coverage is infeasible in large environments, and agents should prioritize practicing skills that are more likely to result in higher reward. We show that our agent can solve 4 challenging image-based exploration problems more rapidly than vanilla model-free RL and state-of-the-art novelty-based exploration. Then, we show that the resulting abstract model solves a family of tasks in VISUALPINBALL not provided during the agent's exploration phase.

1 Introduction

Hierarchical reinforcement learning (HRL) (Klissarov et al., 2025) augments the agent's primitive actions with temporally extended actions called *options* (Sutton et al., 1999). An agent with temporally extended options can plan more effectively because the accuracy of its predictions do not degrade as rapidly over time (Sutton et al., 1999; 2011). Furthermore, well-designed options empower the agent to explore more efficiently (Machado, 2019). But, where do good options come from and how can an agent construct them autonomously solely from interaction with the environment? This problem of option discovery (Mcgovern, 2002) has been studied extensively (Precup, 2001; Mcgovern, 2002; Ravindran, 2004; Konidaris, 2011; Bacon, 2018; Machado, 2019), but most algorithms do not simultaneously tackle the complementary problem of learning state abstractions (Konidaris, 2019). An agent that simultaneously learns both can build a simplified version of the environment (Konidaris et al., 2018), which can then be exploited by off-the-shelf planners to yield solutions to complex, long-horizon tasks (Machado et al., 2023; Sutton et al., 2022; 2024).

One algorithm that simultaneously discovers options and their corresponding state abstractions is Deep Skill Graphs (DSG; Bagaria et al., 2021b). DSG learns options that form a graph, where the nodes are subgoals and the edges are the policies that navigate between them. To construct

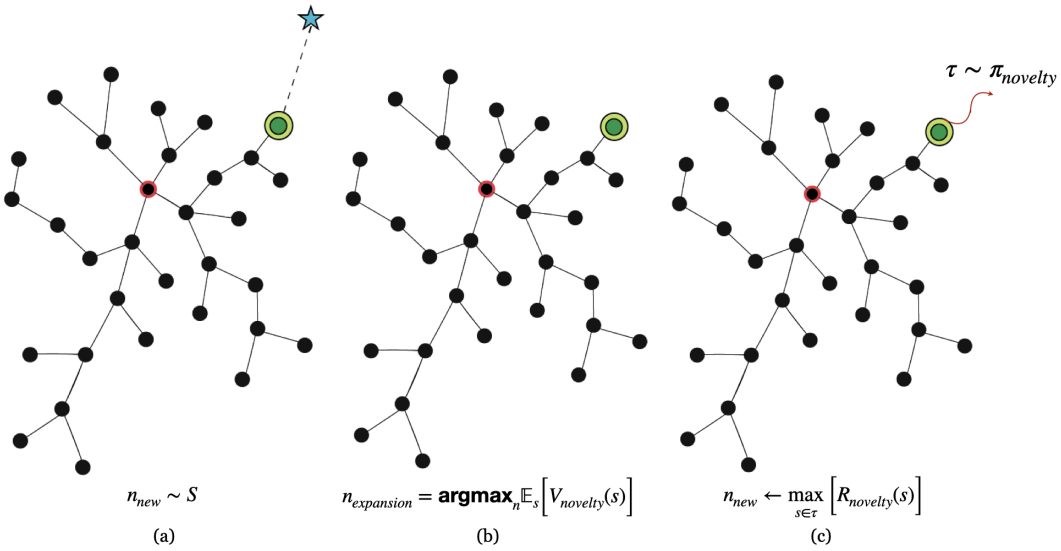

Figure 1: (a) In the original DSG algorithm, a state is sampled uniformly at random (blue star) from the state-space S and the graph is pulled towards it via its nearest neighbor node (green). (b) In the new algorithm, IM-DSG, the agent uses intrinsic motivation to identify a node to expand using an exploration value function V_{novelty}. (c) Once the agent reaches the expansion node, it executes an exploration policy π_{novelty}, and the most novel state in the resulting trajectory τ is identified as a target for a new skill.

this graph, DSG first samples a state uniformly at random from the state-space and then selects its nearest node n_e in the graph. Then, it uses planning inside the graph to reach n_e, after which it moves towards the randomly sampled state. This simple combination of random sampling and expanding the nearest neighbor node is inspired by motion planning (LaValle, 1998) and provably causes the graph to expand towards large unexplored regions of the state-space (Lindemann & LaValle, 2004). While effective, DSG makes the following assumptions that limit its applicability: (a) the ability to sample meaningful states uniformly at random from the state-space (a state-sampler), and (b) the ability to efficiently compute the nearest neighbor node in the graph (a distance metric).

How can we incrementally build the skill graph so that it retains its expansion property, but does not require a state-sampler or nearest-neighbor search to do so? Instead of generating a random state and *pulling* the graph towards it (see Figure 1(a)), we use ideas from intrinsic motivation (Barto, 2013; Schmidhuber, 2010) to identify an existing node for expansion and *push* the graph outward from it (Figure 1(b); Lindemann & LaValle 2004). The agent plans with its existing skills to reach the expansion node, and then executes a *low-level exploration policy* trained using RL to maximize a novelty-based intrinsic reward (Sutton, 1990; Taïga et al., 2019). Since the intrinsic reward is likely to be higher in states that are outside the graph, executing the exploration policy will result in a trajectory that will tend to move away from the existing graph, i.e., towards those states where the agent does not yet have a reliable abstract model (Figure 1(c)). The agent then identifies the most interesting state (Chentanez et al., 2005) in that trajectory and builds a new skill to target it (Simsek & Barto, 2004). Over time, the agent refines the skill that reaches this new node in the graph, and in doing so, converts this once novel region into an area over which it has gained mastery via its abstract model (Veeriah et al., 2018). We call this algorithm Intrinsically Motivated Deep Skill Graphs (IM-DSG).

We test IM-DSG in a series of exploration problems with image-based observation spaces, to which DSG cannot be applied. Our qualitative results show that IM-DSG discovers options that achieve intuitively meaningful subgoals in the environment and that the acquired skill graph incrementally

expands outward from the start state. Quantitatively, we show that our method outperforms pure novelty-maximization techniques (Taïga et al., 2019) as well as baseline model-free RL (Kapturowski et al., 2019) in MINIGRID (Chevalier-Boisvert et al., 2023), VISUAL TAXI (Dietterich, 2000; Diuk et al., 2008; Allen et al., 2021), and SOKOBAN (Schrader, 2018); finally we show that the learned abstract model is useful in a multi-task context to achieve held-out goals in a zero-shot fashion in VISUAL PINBALL (Konidaris & Barto, 2009; Rodriguez-Sanchez & Konidaris, 2024).

2 Background and Related Work

We model the interaction of the agent with its environment as a Markov Decision Process $M = (S, A, R, T, \gamma)$ (Sutton & Barto, 2018). To aid in reward maximization, our agent will discover options $o \in \mathcal{O} = (\mathcal{I}_o, \pi_o, \beta_o)$ (Sutton et al., 1999), where $\mathcal{I}_o(s)$ is the probability that o can be initiated in state s, π_o is a policy which outputs primitive actions and $\beta_o : s \to \{0, 1\}$ denotes the termination region of the option, which we also call the subgoal of the option. Additionally, each option has a maximum horizon of H steps and an effect set $\text{Eff}(o) \subseteq \beta_o$, which denotes the set of states in which the option actually enters its subgoal β_o (Konidaris et al., 2018).

Universal Value Function Approximators (UVFAs) (Schaul et al., 2015) provide a scalable way to learn goal-conditioned value functions $V_g : S \to \mathbb{R}$, $Q_g : S \times A \to \mathbb{R}$ and their corresponding goal-conditioned policies $\pi_g : S \to A$ (Kaelbling, 1993) using function approximation. We use UVFAs to parameterize option value functions and policies: instead of representing each option's policy separately (Sutton et al., 2011), all options can condition the same function approximator on its own subgoals (Bagaria et al., 2021a): $\pi_o(s) = \text{argmax}_a\, Q_g(s, a) := \text{argmax}_a\, Q_{\beta_o}(s, a)$.

Exploration in RL. In tabular RL, uncertainty about the transition and reward models can be captured using visitation counts: if the bonus reward (Sutton, 1990) for action a from state s decays as $1/\sqrt{N[s, a]}$, the agent can rapidly learn the optimal policy π^* (Lattimore & Szepesvári, 2020; Strehl & Littman, 2008). When the environment is too big to permit tabular counting, counts can be generalized to *pseudocounts* $N(s)$ (Bellemare et al., 2016): states that are similar to one another have similar pseudocounts. Coin Flip Network (CFN) (Lobel et al., 2023) is a simple, principled, and state-of-the-art technique which uses a neural network ϕ to predict pseudocounts in high-dimensional state-spaces. Novelty seeking agents exhibit impressive performance (Kapturowski et al., 2022), but they do not learn a model suitable for high-level planning. Our algorithm combines novelty maximization (which we refer to as "low-level exploration") with a drive to learn an abstract, plannable model of the world ("high-level exploration").

Option discovery methods. A large body of work has sought to discover options from the set of primitive actions available to the agent. At a high-level, option discovery techniques can be categorized in the following way:

- **Reward-driven methods.** Option critic (Bacon et al., 2017) and feudal methods (Dayan & Hinton, 1993; Vezhnevets et al., 2017; Nachum et al., 2018b; Levy et al., 2019) learn skills using the overall reward maximizing objective of the agent. Since all learning stems from the extrinsic reward function, they struggle in hard exploration problems with sparse rewards; there are some rare exceptions (McClinton et al., 2021), but they are plagued by the difficulty that the action-space for the high-level policy is the entire state-space of the environment (Nachum et al., 2018a).
- **Empowerment** maximizing methods learn a diverse set of options (Eysenbach et al., 2019b; Zahavy et al., 2021) that increase the agent's control over its environment (Mohamed & Jimenez Rezende, 2015; Choi et al., 2021; Gregor et al., 2016). These algorithms have shown promise in challenging control (Sharma et al., 2020a) and exploration problems (Baumli et al., 2021; Hansen et al., 2021; Campos Camúñez et al., 2020). Among these, DADS (Sharma et al., 2020a) combines empowerment with high-level planning, but it decomposes the agent's life into separate training and test phases, i.e., they do not provide a way to *interleave* exploration and

model learning. Furthermore, methods like DADS require learning one-step transition models, perhaps explaining why they have not been applied to image-based problems.

- **Spectral methods** use the graph Laplacian (Chung & Graham, 1997) to find the principal directions of the state-space (Mahadevan & Maggioni, 2007) for exploration (Machado et al., 2017). Typically, an option is associated with each eigenvector of the Laplacian; these options demonstrate strong exploratory properties (Jinnai et al., 2019; 2020; Klissarov & Machado, 2023). Both empowerment and spectral methods are promising directions for exploration research, but have not yet been combined with model-based planning, which is our core motivation.

Deep Model-based RL methods like Dreamer (Hafner et al., 2018) and MuZero (Schrittwieser et al., 2021) learn a one-step model of the environment for planning. These algorithms have demonstrated impressive results in a variety of domains (Hafner et al., 2023). However, all learning (even in LEXA (Mendonca et al., 2021) and Director (Hafner et al., 2022), which learn temporal abstractions) is predicated on accurately approximating **one-step** transition and reward models in the agent's native state-action space. One-step models suffer from the "curse of horizon" (Sutton et al., 1999): model prediction errors compound over time (Talvitie, 2017). Our work seeks to bypass the challenges of model-learning in the raw state-action space: model-free policies are learned in the low-level and temporally-extended models are learned and used in the high-level.

Go-Explore shares a similar motivation to our algorithm: exploration is more effective when initiated from the frontiers of the agent's experience (Ecoffet et al., 2021). But, they require a prespecified "cell" representation and exploit the near-determinism of Atari domains (Machado et al., 2018) to return to the frontier (Oh et al., 2018). Crucially, Go-Explore does not learn a plannable model while doing exploration, which is the focus of our work.

Existing graph-based HRL methods like deep skill graphs (DSG) (Bagaria et al., 2021b; Lee et al., 2022; Lo et al., 2024) do not generalize to image-based observation spaces. Others do (Konidaris et al., 2018; Mann et al., 2015; Sutton et al., 2024; Rodriguez-Sanchez & Konidaris, 2024), but assume that options are given. Eysenbach et al. (2019a); Huang et al. (2019); Campos Camúñez et al. (2020); Nasiriany et al. (2019); Sharma et al. (2020a) decompose the agent's training into a pre-training stage and a planning phase: skills are learned over the entire domain and *then* planning is done to improve learned skills. These methods typically perform uniform resets: the agent is randomly reset to a new location at the beginning of every episode (Eysenbach et al., 2019a; Huang et al., 2019; Zhang et al., 2021); this bypasses the exploration problem, which is a key motivation for our work.

3 Intrinsically Motivated Deep Skill Graphs

Exploration and planning are deeply intertwined. Exploration aids planning by preferentially seeking out data from regions where the model is currently too weak to support effective planning. But, planning also supports effective exploration: the agent can plan with its abstract model so that it can reach interesting areas reliably, and then conduct local exploration in a more targeted fashion. The Intrinsically Motivated Deep Skill Graphs (IM-DSG) agent interleaves exploration and planning in this way. To support planning, it builds a graph-based model of the environment. Exploration proceeds both inside the graph (to strengthen the existing model) and outside it (to increase the region over which the agent can plan). Inside the graph, the agent refines skills that are currently weak and could lead to high reward; outside the graph, the agent uses a novelty and reward maximizing policy to search for new salient regions that can be reached reliably in the future using its model.

3.1 Semantics of the Skill Graph

Before discussing our algorithm for discovering options and planning with them, we first outline the major components of the skill graph:

Nodes and Edges. The skill graph is a weighted and directed graph $\mathcal{G} = (\mathcal{N}, E, W)$. Each **node** $n \in \mathcal{N}$ represents the terminating states of an option o_n—i.e., its subgoal—determined by the termination condition $\beta_{o_n} : s \to \{0, 1\}$. Each node n maintains an empirical distribution over states where its option o_n terminated. We denote this as $n(s)$, which assigns probability $1/|\text{Eff}(o_n)|$ to each $s \in \text{Eff}(o_n)$ and 0 otherwise. Moving forward, we abuse notation by taking expectations with respect to the states of a node, $\mathbb{E}_{s\sim n}[\cdot]$ to mean expectations taken with respect to the effect distribution $\mathbb{E}_{s\sim\text{Eff}(o_n)}[\cdot]$, which is done simply by sampling from the effect set.

An **edge** $e_{n_i \to n_j} \in E$ exists between nodes n_i and n_j if executing the option policy $\pi_{o_{n_j}}$ from states in node n_i is likely to reach node n_j (Konidaris et al., 2018). The weight of an edge is the probability of successfully traversing it in a single option execution, which is equivalent to the initiation probability of option o_j from states inside node n_i: $w_{n_i \to n_j} = \mathbb{E}_{s\sim n_i}\left[\mathcal{I}_{o_{n_j}}(s)\right]$ (Bagaria et al., 2023).

Mapping from states to nodes and back. Given a state s, $\mathcal{N}(s)$ denotes the set of nodes that s maps to:

$$\mathcal{N}(s) = \{n \in \mathcal{N} \mid \beta_{o_n}(s) = 1\}. \tag{1}$$

Of course, it is possible that a state does not satisfy *any* option's termination condition; so, we augment the nodes in the graph with a null vertex φ such that $\mathcal{N}(s) = \{\varphi\}$ for such states; informally, φ represents "falling off the graph".

The descendants $\mathcal{D}(n)$ of a node n is the set of nodes that can be reached from n with nonzero probability, while staying inside the graph:

$$\mathcal{D}(n) = \{n' \in \mathcal{N} \mid \mathcal{G}.\texttt{has-path}(n, n')\}, \tag{2}$$

where $\mathcal{G}.\texttt{has-path}(n, n')$ is a sub-routine implemented using breadth-first search on the adjacency matrix representation of the graph $\mathcal{G}$. By combining Equations 1 and 2, we can enumerate the nodes reachable from the current state s_t as $\mathcal{D}(s_t) = \cup_{n\in\mathcal{N}(s_t)}\mathcal{D}(n)$.

3.2 Graph Construction Algorithm

At a high-level, the graph construction algorithm proceeds as follows: the IM-DSG agent first identifies which node in the graph to expand (Section 3.2.1), then it constructs (Section 3.2.4) and solves an abstract MDP (Section 3.2.5), which yields a policy over options that guides the agent to the expansion node. After following this abstract policy and reaching the expansion node, the agent executes a low-level exploration policy (Section 3.2.6), and finally extracts a new node from the resulting trajectory to add it to the skill graph (Section 3.2.7).

3.2.1 Identifying the Expansion Node

To select an expansion node, we need to quantify how much each node would contribute to graph expansion: nodes that cause the graph to extend further should have a higher probability of being selected for expansion. To capture this idea, we first construct an intrinsic reward function that is higher the further away a state is from the graph. But rather than relying on distance functions, which are usually unavailable in high-dimensional observation spaces, we use pseudocounts (Bellemare et al., 2016): the further away a state is from the graph, the fewer times it has likely been visited. We use Coin Flip Networks (CFN) (Lobel et al., 2023) to estimate pseudocounts and use it as our intrinsic reward:

$$r_{\text{novelty}}(s) = \frac{1}{\sqrt{N_\phi(s)}},$$

where $N_\phi(s)$ is the pseudocount of state s as approximated using CFN parameters ϕ.

We now have an intrinsic reward function r_{novelty} that is higher along the frontiers of the graph, where the agent has less experience. To turn this into an expansion probability, we ask: if the agent

executes a novelty-maximizing policy π_{novelty} starting from each graph node, how much intrinsic reward can it expect to accumulate? Fortunately, this concept is well captured by the exploration value function V_{novelty}. But, solely using novelty will result in graph expansion in all possible directions; to forego exhaustive coverage and prioritize high reward regions, we add the intrinsic and extrinsic rewards together while learning V_{novelty}. Concretely, we define the utility of expanding a node n as $U(n) = \mathbb{E}_{s\sim n}\Big[V_{\text{novelty}}(s)\Big] = \mathbb{E}_{s\sim n}\Big[\sum_{t=0}^{\infty}\gamma^t\{R(s_t,a_t) + b\cdot r_{\text{novelty}}(s_t)\} \mid s_0 = s\Big]$; we use $b = 0.01$[1].

Finally, we restrict the choice of expansion node to the set of nodes that are reachable from the current state, i.e, the descendants of the nodes that map to the current state of the agent s_t. Therefore, we define the probability of expanding a node n as follows:

$$n_{\text{expansion}} \sim \mathbb{P}_n = \frac{U(n)}{\sum_{n'\in\mathcal{D}(s_t)} U(n')} = \frac{\mathbb{E}_{s\sim n}\Big[V_{\text{novelty}}(s)\Big]}{\sum_{n'\in\mathcal{D}(s_t)} \mathbb{E}_{s'\sim n'}\Big[V_{\text{novelty}}(s')\Big]}, \forall n \in \mathcal{D}(s_t). \tag{3}$$

3.2.2 Estimating Edge Probabilities

Every edge in the skill graph has a weight $w_{n_i\to n_j}$, which represents the probability with which an option o_{n_j} initiating from states in node n_i will successfully terminate in node n_j. As pointed out earlier, this is equivalent to the expected initiation probability of option o_{n_j}, where the expectation is taken with respect to states inside the node n_i. Following Bagaria et al. (2023), we interpret the agent's goal-conditioned value function (UVFA) (Schaul et al., 2015) as an initiation probability:

$$w_{n_i\to n_j} = \mathbb{E}_{s\sim n_i}\Big[\mathcal{I}_{o_j}(s)\Big] \approx \mathbb{E}_{s\sim n_i}\Big[V_{n_j}(s)\Big] := v_{n_i\to n_j}. \tag{4}$$

It is possible to infer the edge probabilities from the UVFA because the goal-conditioned reward function that is used to train it is $\mathcal{R}_o(s) = \beta_o(s)$, i.e., it is $+1$ for states inside the option's subgoal region and 0 otherwise. However, naively using the UVFA prediction as edge probability can hinder learning because:

1. $v_{n_i\to n_j}$ for an edge can be low if traversing it is actually unlikely in a single option execution (even under an optimal policy) *or* because the agent does not yet have enough data, and,
2. when we add new nodes to the graph, the probability of reaching them will initially be low. To increase these edge probabilities, the agent must continually practice the options that target them, but this is unlikely to happen owing to their low starting probabilities.

Therefore, to address these issues, we are *optimistic* about the value of the edge probabilities:

$$w^{+}_{n_i\to n_j} := v_{n_i\to n_j} + \mathcal{U}(n_i, n_j) \approx v_{n_i\to n_j} + \frac{c}{\sqrt{N[n_i, n_j] + 1}}, \tag{5}$$

where $\mathcal{U}(n_i, n_j)$ represents the uncertainty about our value prediction $v_{n_i\to n_j}$, which we obtain using $N[n_i, n_j]$ (Brafman & Tennenholtz, 2002; Strehl & Littman, 2008), the number of times the agent has executed option o_{n_j} starting from states in node n_i ($c \in [0, 1]$ is a hyperparameter).[2] When $N[n_i, n_j]$ is low, the agent is optimistic about traversing that edge, and so it is encouraged to practice the corresponding option policy. But, as the agent attempts to traverse edge $e_{n_i\to n_j}$, the count $N[n_i, n_j]$ increases, and the optimistic bias decays; eventually, the edge probability approaches the goal-conditioned value, which we expect to become more accurate with more data. By incorporating an optimistic bias via the edge probabilities, we ensure that the policies computed by the planner achieve a form of **exploration within the graph**, which prioritizes paths that reach the expansion node while improving newly added skills.

[1] We chose $b = 0.01$ to make the corresponding intrinsic return, $0.01/(1-\gamma) = 1$, the same scale as the extrinsic reward.
[2] Since the counts are over discrete states of the AMDP, we simply maintain tabular counts in a $|\bar{S}| \times |\bar{S}|$ matrix.

3.2.3 Deciding when there is an edge between two nodes

Using Equation 5, we can determine the probability of traversing an edge in a single option execution. Naïvely, this would result in a dense, fully-connected graph, which would be problematic: suppose there is an existing node n_1 and we just added a new node n_2 that is far away (even under an optimal policy) from n_1. The agent's initial desire to connect them would be high (because $N[n_1, n_2]$ will be low). As a result, the agent will spend a lot of time realizing that n_1 and n_2 should actually have a low probability of being directly connected to one another. Not only is this sample-inefficient, it also results in graph construction *slowing down over time*: for every new node added to the graph, the agent must determine how the new node relates to every other existing node. To mitigate this issue, we use the following heuristics to determine when two nodes n_1 and n_2 should have an edge connecting them:

- **Observed multi-step transition.** When we observe a transition between two nodes $n_1 \rightarrow n_2$ within H steps (the maximum horizon of an option), we connect them with an edge.
- **High goal-conditioned value.** We connect nodes $n_1 \rightarrow n_2$ with an edge when the goal-conditioned value between them, $\mathbb{E}_{s\sim n_1}\left[V_{n_2}(s)\right]$, is high enough. To ensure that the resulting model is amenable to planning, we add edges only when we are *highly confident* that execution of one option will allow for the execution of another (Konidaris et al., 2018). We instantiate this principle via an adaptive threshold that is high to begin with, and tapers to a fixed hyperparameter $\tau \in [0, 1]$ when more experience is gained:

$$e_{n_1 \rightarrow n_2} = \mathbb{I}\left[\mathbb{E}_{s\sim n_1}\left[V_{n_2}(s)\right] - \frac{c}{\sqrt{\min\left\{N[n_1], N[n_2]\right\}}} > \tau\right], \tag{6}$$

  where $\mathbb{I}$ is the indicator function and $N[n]$ is the number of times the agent has visited node n. The inverse square root term acts as an uncertainty penalty that diminishes as more data is collected. We use a different form of the count in the denominator than in Equation 5 because the pairwise counts are likely 0 before edge creation.

3.2.4 Constructing the Abstract MDP

After the agent identifies the expansion node $n_{\text{expansion}}$, it needs to compute a plan that will guide it from its current state s_t to $n_{\text{expansion}}$. Our graph has two properties that will inform an appropriate choice of planning algorithm:

1. **Probabilistic transitions**. The skill graph has probabilistic edges because (a) the environment's transition function and the agent's option policies are stochastic, and (b) the option policies are being learned online, causing the abstract transition function to also appear stochastic and non-stationary (Nachum et al., 2018b; Levy et al., 2019; Bagaria et al., 2021a). We represent these dynamics with a transition matrix, $\bar{T}$, where each entry $\bar{T}_{ij}$ is the probability that executing the option targeting node n_j from node n_i successfully leads to node n_j. When option execution is unsuccessful, we assume the agent "falls off" the graph, modeling it by assigning the remaining probability $(1 - \bar{T}_{ij})$ to an absorbing failure abstract state φ.
2. **Incorporating extrinsic reward.** Given two paths from s_t that both reach $n_{\text{expansion}}$, we would prefer one that collects extrinsic rewards along the way. This allows us to form a reward-respecting abstract model (Sutton et al., 2024), and prioritizes skill refinement in parts of the graph that are likely to result in higher reward. To incorporate this, we define a reward matrix, $\bar{R}$, where each entry $\bar{R}_{ij}$ is the sum of extrinsic rewards collected during the execution of the option and an intrinsic reward that is high at the expansion node. Formally, we write: $\bar{R}_{ij} = \kappa \bar{R}^{\text{ext}}_{ij} + \bar{R}^{\text{int}}_{j}$, where $\kappa \in \mathbb{R}$ scales the extrinsic reward, and $\bar{R}^{\text{int}}_{j}$ equals 1 if node n_j is the expansion node, and 0 otherwise.

We also define an abstract discount factor, $\bar{\gamma}$, which reflects the probability of staying within the graph during an option execution. Putting these elements together, our AMDP is defined as: $\bar{M} =$

$(\bar{S}, \bar{A}, \bar{R}, \bar{T}, \bar{\gamma})$, where $\bar{S}$ and $\bar{A}$ represent the set of nodes in the skill graph. Planning on this abstract MDP is a Stochastic Shortest Path (SSP) problem (Bertsekas & Tsitsiklis, 1991). We solve this SSP using value iteration (Bellman, 1966), which yields a deterministic abstract policy $\bar{\pi}$. This abstract policy maps each node (abstract state) to an option, guiding the agent toward the expansion node while balancing the collection of extrinsic rewards and exploration of uncertain edges. An illustrative example of this AMDP construction is provided in Figure 7 in Appendix C.

3.2.5 Navigating to the Expansion Node

The abstract policy $\bar{\pi}$ tells the agent which options to execute to reach the expansion node. While following $\bar{\pi}$, if the agent finds itself in a state that is not in the graph ($s_t \in \varphi$), it targets the closest node in the graph: $o_t = \text{argmax}_o V_{\beta_o}(s_t), \forall o \in \mathcal{O}$. Each option execution corresponds to rolling out the corresponding option policy $\pi_o(s)$, which is trained using recurrent deep Q-learning (Kapturowski et al., 2019) to maximize the pseudo reward function $R_o(s) = \beta_o(s)$. In practice, all option policies are parameterized using the same goal-conditioned policy $\pi(s|g;\theta)$ (Schaul et al., 2015) and each option conditions the policy with goal states sampled from its own termination region (Bagaria et al., 2021a). To improve the sample-efficiency of learning effective option policies, we use hindsight experience replay (Andrychowicz et al., 2017).

3.2.6 Generating Behavior Outside the Graph

After reaching the expansion node, the IM-DSG agent attempts to extend the graph. It does so by executing the low-level exploration policy π_{CFN}, which is trained to maximize the weighted sum of intrinsic and extrinsic rewards:

$$R_{\text{CFN}}(s,a) = R(s,a) + b \cdot r_{\text{novelty}}(s),$$

where $r_{\text{novelty}}(s) = 1/\sqrt{N_\phi(s)}$, $b \in [0,1]$ controls the exploration-exploitation trade-off (we use $b = 0.01$) and $N_\phi(s)$ is the CFN visitation count prediction for state s. We use recurrent deep Q-learning (Kapturowski et al., 2019) to learn V_{novelty} and π_{novelty}. The result of this policy execution is a trajectory τ_{novelty}, which is likely to move outward, toward regions where the agent's abstract model is more uncertain; this is a way of doing exploration *outside* the graph.

3.2.7 Adding a New Node to the Graph

Having generated a low-level exploration trajectory τ_{novelty} in the frontier of the graph, the agent must now identify a new node to add to the graph. Again, we take inspiration from intrinsic motivation methods (Simsek & Barto, 2004): the agent identifies the state in τ_{novelty} with the highest predicted novelty; if the novelty of that state is higher than the running mean of the novelty-based rewards, then we flag it as a new target and add it to the skill graph:

$$s_{\text{target}} = \underset{s \in \tau_{\text{novelty}}}{\text{argmax}} \left[r_{\text{novelty}}(s) \right] \text{ if } \max_s \left[r_{\text{novelty}}(s) \right] > \mu_t + k\sigma_t, \text{ else } \varphi, \tag{7}$$

where μ_t is the running mean and σ_t is the running standard deviation of r_{novelty}, computed incrementally over the lifetime of the agent (Welford, 1962); $k \in \mathbb{R}^+$ is a hyperpameter (we use $k = 1$). Equation 7 identifies a target state, which we need to convert to a termination classifier so that it can serve as a node in the skill graph. We could use domain knowledge (for example, the "cell" representations of Ecoffet et al. (2021)), pixel-wise equality (Veeriah et al., 2018), or a temporal distance based classifier (Savinov et al., 2018); we opt for simplicity and use an off-the-shelf template matching algorithm, which is a simple call to OpenCV's `matchTemplate`(s, g) (Bradski, 2000):

$$n_{\text{new}} = \beta_{o_{\text{new}}}(s) = \mathbb{I}\Big[\texttt{matchTemplate}(s, s_{\text{target}}) > \tau_{\text{template}}\Big], \tag{8}$$

where $\mathbb{I}$ is the indicator function, `matchTemplate` computes the normalized correlation coefficient between image s and template s_{target}, and $\tau_{\text{template}} = 0.5$ is the similarity threshold.

Algorithm 1 Intrinsic Motivation Directed Skill Graph (IM-DSG)

Initialize:
Initialize CFN to learn psuedocounts N_ϕ, value function V_{novelty}, and policy π_{novelty}.
Initialize goal-conditioned R2D2 to learn value function V_g and policy π_g.
Initialize graph $\mathcal{G}$ with null node $\{\varphi\}$ and no edges.
while True **do**
    Sample expansion node using Equation 3.
    Construct AMDP using procedure described in Section 3.2.4.
    Solve AMDP using value iteration to get abstract policy $\bar{\pi}$.
    Follow abstract policy until agent reaches expansion node or episode terminates.
    if agent reached expansion node **then**
        Roll out exploration policy π_{novelty} to get trajectory τ_{novelty}.
    end if
    Potentially add a new node from τ_{novelty} to the graph using Equation 7.
    Update V_g and π_g using Hindsight Experience Replay (HER).
    Update π_{novelty} to maximize the sum of intrinsic and extrinsic rewards: $R(s,a)+b\cdot r_{\text{novelty}}(s)$.
    Update the edge traversal $N[i,j]$ and node visitation counts $N[i]$ using the full trajectory.
    Update the edge probabilities in the skill graph $\mathcal{G}$ using V_g (Sections 3.2.2 and 3.2.3).
end while

3.2.8 Summary of Graph Discovery Algorithm

Algorithm 1 summarizes IM-DSG. The agent maintains two value functions V_g, V_{novelty} and two policies $\pi_g, \pi_{\text{novelty}}$; R2D2 (Kapturowski et al., 2019) is used to learn both sets of policies and value functions. The agent first chooses an expansion node using Equation 3. Which option is executed is then determined by solving the resulting Abstract MDP using value iteration. New nodes are added to the graph by executing π_{novelty} from the expansion node. Finally, graph edges are updated based on V_g and edge traversal counts $N[i,j]$. The choice of expansion node and subsequent low-level exploration encourage the skill graph to expand along directions that jointly maximize novelty and extrinsic reward.

4 Experiments

We evaluate the IM-DSG agent on five challenging exploration problems, all of which have image-based observations and discrete actions (details in Appendix A). We chose the MiniGrid (Chevalier-Boisvert et al., 2023) environments because they were identified as being challenging for existing RL algorithms by the survey of Colas et al. (2022). We chose Sokoban (Schrader, 2018) because the survey of Hamrick et al. (2020) identified it as being a problem in which both learning and planning are important for good performance. Finally, we chose VisualTaxi and VisualPinball because non-image versions of these domains have been used extensively to evaluate state abstraction and option discovery algorithms in isolation (for example, Lo et al., 2024; Konidaris & Barto, 2009; Dietterich, 2000).

4.1 Qualitative Evaluation

To build intuition, we first qualitatively evaluate the IM-DSG agent. Specifically, we want to understand what the skill-graph looks like, what parts of it the agent uses most frequently, and whether the skill graph aids more effective exploration.

4.1.1 Visualizing the Skill Graph in MiniGrid

Figure 2 is a visualization of the skill graph at the end of training in MiniGrid-KeyCorridor. To visualize each node in the graph, we randomly sample a state in which that termination condition

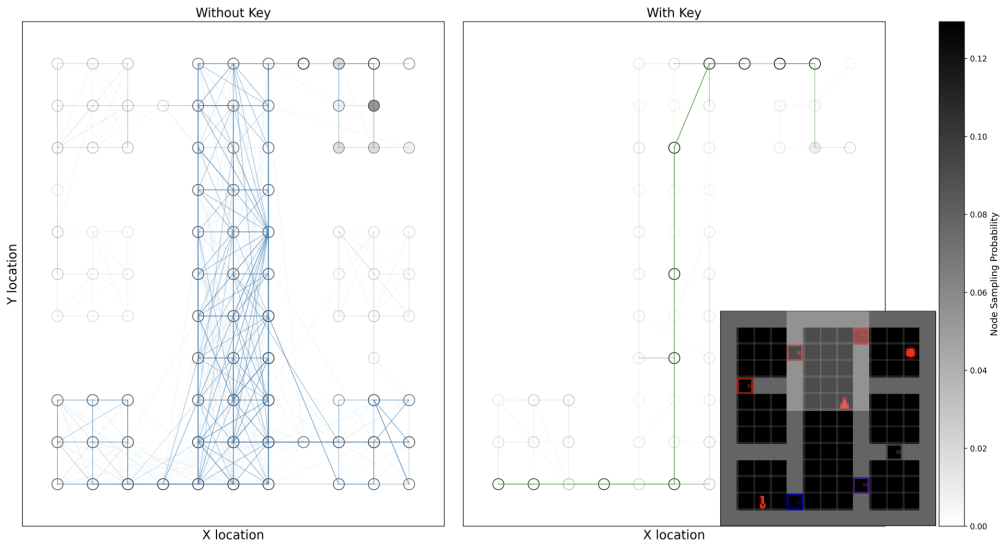

Figure 2: **Discovered skill graph in MINIGRID-KEYCORRIDOR**. The domain is shown in the bottom-right inset: it involves picking up a key in the bottom-left room, opening a locked door and then picking up a ball in the top-right room. The left subplot shows the subgraph where the agent does *not* have a key in its possession (other state variables are not shown for simplicity). The nodes and edges of the graph are shaded based on their expected abstract value $\mathbb{E}[\bar{V}(n)]$. Although we visualize the location and inventory, the agent itself only observes images. The graphs show that, when the agent lacks the key, it aims to explore down the corridor and in the rooms at the bottom of the maze, but when it does have the key, interesting options are on the path to the goal.

was triggered; only the location and inventory is visualized here for simplicity. Each node is colored based on its probability of serving as the expansion node. Furthermore, the transparency α_n of a node is its average abstract value:

$$\alpha_n = \max\left\{\mathbb{E}_{n_e}\left[\bar{V}_{n_e}(n)\right], 0.05\right\},$$

where the expectation is over the node expansion probability distribution $\mathbb{P}_n$ (Equation 3), $\bar{V}_g(n)$ is the abstract value of node n when targeting node g (obtained via Value Iteration on the AMDP; Section 3.2.4). The transparency of an edge is the average transparency of the nodes that it connects.

Understanding Figure 2. Nodes in the top-right room have a higher expansion probability. This is because (a) navigating to the top-right room is novel as it requires unlocking the red door with the red key, and (b) once the agent enters the top-right room, it can solve the remainder of the task easily. These two facts result in high V_{novelty} and hence high expansion probabilities, which is why the graph in the bottom-left and top-right rooms have high abstract value. Nodes in the central corridor also have high abstract value because the agent must navigate through them for the majority of its target expansion nodes. This demonstrates that the agent focuses on parts of the graph that are either important for expansion by themselves, or those subgraphs that serve as *hubs*, i.e., they lead to other sub-graphs that are important for expansion (Şimşek et al., 2005; Shah & Srivastava, 2024).

4.1.2 Exploration in SOKOBAN

Figure 3 shows IM-DSG's high-level policy when it encounters the task-goal for the first time: the expansion node is chosen to be a state that corresponds to placing two out of the three boxes in their correct locations. The agent plans with its learned options to reach this state. Subsequent novelty-driven exploration is more targeted as it only seeks to explore those states in which most of

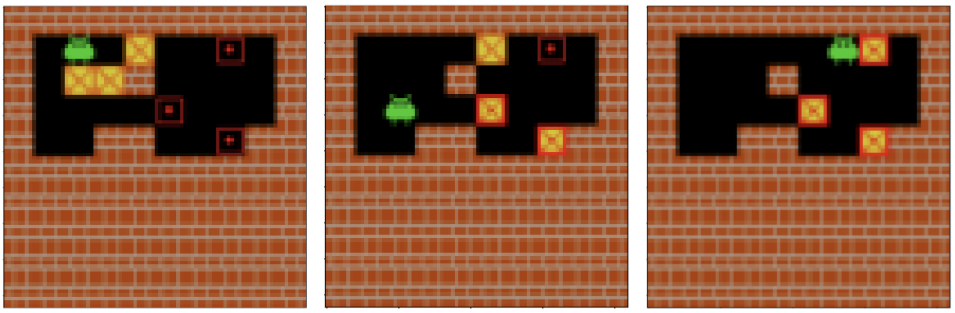

Figure 3: Example of hierarchical policy execution in SOKOBAN: *(left)* start state in which *no* box (boxes shown in yellow) is in its desired location (target locations are shown in red). *(Middle)* selected expansion node which has 2 out of 3 boxes placed in their desired locations; the agent uses planning to get to this state. *(Right)* After reaching the expansion node, the agent does local exploration using π_{novelty}, which reaches the task goal.

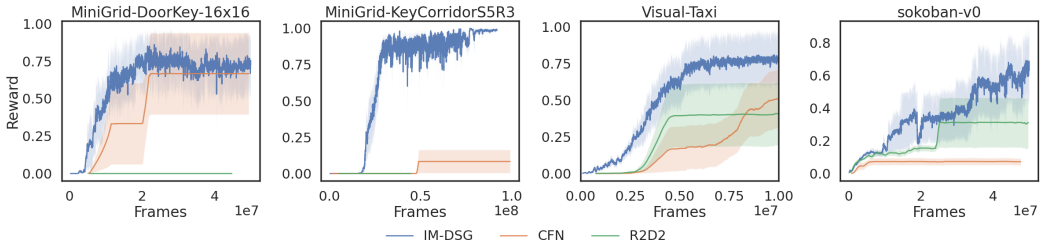

Figure 4: Comparing the performance of our proposed algorithm, IM-DSG (blue), novelty-based CFN (orange) and vanilla model-free R2D2 (green). Each curve represents the mean reward and the shaded area represents the standard deviation over 5 random seeds.

the task has already been solved, increasing the probability that random exploration would find the proverbial "needle in a haystack".

4.2 Quantitative Evaluation

First, we seek to understand whether the IM-DSG agent can solve challenging exploration problems in a more sample-efficient manner than our baselines. Second, even though the graph construction is prioritized along directions where the agent expects to gather extrinsic reward, we want to understand whether the graph can still be used in a multi-task context to solve goal-reaching problems that were not encountered during training.

4.2.1 Sample Efficient Exploration in Single Task MDPs

We compare the IM-DSG agent to the following baselines:

- **Flat model-free RL.** R2D2 is a distributed variant of DQN that uses a recurrent neural network to represent the value function (Kapturowski et al., 2019). Since the IM-DSG agent uses R2D2 to learn policies, we include it as our model-free baseline.
- **Novelty driven exploration.** We use CFN to explore outside the graph, and it is a state-of-the-art technique for novelty-driven RL; so, we include it as our exploration baseline.

Figure 4 shows our quantitative results: we find that the IM-DSG agent consistently outperforms R2D2 and CFN in our domains. This suggests that although novelty-maximizing exploration is in principle sufficient to solve these problems, the discovery and use of the model for planning leads to a more sample-efficient agent in these domains.

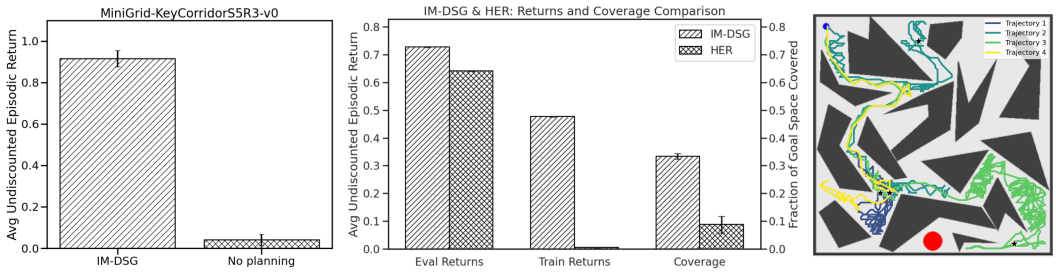

Figure 5: **Ablation and graph usefulness in a multi-task context.** *(left)* Ablating the use of planning in IM-DSG in MINIGRID-KEYCORRIDOR: bars represent the average per-episode reward during training. *(middle)* Evaluation performance in VISUALPINBALL on 10k randomly sampled goal states not used during training: comparison against HER in terms of test-time rewards, train-time rewards, and coverage, which is the fraction of the maze explored during training. All error bars represent standard deviation over 5 random seeds. *(right)* Solution trajectories found by the IM-DSG agent for 4 randomly sampled test-time goals (shown using black stars); the blue dot in the top-left of the maze is the start state, the big red ball in the bottom is the train-time goal. Note that, unlike in the original, this version of pinball uses images as observations.

Model-based planning ablation. Figure 5 *(left)* shows that when we disable planning, the IM-DSG agent is unable to solve the MINIGRID-KEYCORRIDOR problem. This indicates that high-level planning with the learned skill graph is essential—simply combining CFN with HER is not enough to solve this task in the allotted training budget. This ablation also serves a rough comparison to algorithms like Skew-Fit (Pong et al., 2019) and Omega (Pitis et al., 2020) that combine HER with novelty-based exploration, but do not do model-based planning.

4.2.2 Generalization to New Goals in VISUALPINBALL

In this section, we test whether the skill graph discovered during exploration can be used to solve other, unseen tasks at test-time. In particular, we train our agent to solve the VISUALPINBALL problem, which requires guiding the ball to the big red circle at the bottom of the maze. After training, we present the agent with 10k randomly sampled held-out goals and test its ability to reach these goals. We compare our agent against Hindsight Experience Replay (HER; Andrychowicz et al., 2017), which in principle can also generalize to unseen goals at test-time. We follow the experimental setup of Sharma et al. (2020a); more details can be found in Appendix B.

Figure 5 shows that IM-DSG outperforms HER at test-time. To understand why, we compare their train-time performances and find that HER is unable to explore the maze as extensively as IM-DSG, as a result of which, it is unable to reach goals that are far away from the start-state at test-time.

5 Discussion and Conclusion

We introduced an algorithm that builds a graph abstraction of large MDPs with image-based observations. Nodes of this graph correspond to option termination regions, edges represent option policies, and edge probabilities are determined using option initiation functions. The skill graph expands towards the frontier of the agent's experience without assuming access to state-sampling or a distance metric; the acquired graph enables planning in increasingly large portions of the state-space.

Our method suffers from at least the following limitations:

- Each option drives the value of all state variables to a certain range of values. In more complex environments, it may be unnecessary, or even impossible, to control all state variables at once.

- After reaching the expansion node, the IM-DSG agent executes the low-level exploration policy (CFN) until the end of the episode; more fine-grained control over when to enter and exit low-level exploration (Pislar et al., 2022) would generalize our algorithm to continuing problems.

Despite these shortcomings, IM-DSG takes a small step towards the eventual goal of building RL agents that continually discover their own state and action abstractions (Gershman, 2017; Konidaris, 2019), and use them for effective exploration (Gopnik, 2020) and planning (Sutton et al., 2024).

Acknowledgements

We would like to thank the members of the Brown Intelligent Robots Lab (IRL) for their support and suggestions. This research was support by NSF grant 1955361 and NSF CAREER 1844960.

References

Cameron Allen, Neev Parikh, Omer Gottesman, and George Konidaris. Learning markov state abstractions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 34:8229–8241, 2021.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Pierre-Luc Bacon. *Temporal Representation Learning*. PhD thesis, McGill University Libraries, 2018.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Akhil Bagaria and Tom Schaul. Scaling goal-based exploration via pruning proto-goals. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pp. 3451–3460. ijcai.org, 2023.

Akhil Bagaria, Jason Senthil, Matthew Slivinski, and George Konidaris. Robustly learning composable options in deep reinforcement learning. In *30th International Joint Conference on Artificial Intelligence*, 2021a.

Akhil Bagaria, Jason K. Senthil, and George Konidaris. Skill discovery for exploration and planning using deep skill graphs. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 521–531. PMLR, 2021b.

Akhil Bagaria, Ben M Abbatematteo, Omer Gottesman, Matt Corsaro, Sreehari Rammohan, and George Konidaris. Effectively learning initiation sets in hierarchical reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Andrew G. Barto. Intrinsic motivation and reinforcement learning. In Gianluca Baldassarre and Marco Mirolli (eds.), *Intrinsically Motivated Learning in Natural and Artificial Systems*, pp. 17–47. Springer, 2013.

Kate Baumli, David Warde-Farley, Steven Hansen, and Volodymyr Mnih. Relative variational intrinsic control. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pp. 6732–6740, 2021.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

Dimitri P Bertsekas and John N Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

Ronen I Brafman and Moshe Tennenholtz. R-Max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

Víctor Campos Camúñez, Alex Trott, Caiming Xiong, Richard Socher, Xavier Giró Nieto, and Jordi Torres Viñals. Explore, discover and learn: unsupervised discovery of state-covering skills. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119, pp. 1317–1327, 2020.

Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 1281–1288, 2005.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *CoRR*, abs/2306.13831, 2023.

Jongwook Choi, Archit Sharma, Honglak Lee, Sergey Levine, and Shixiang Shane Gu. Variational empowerment as representation learning for goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, pp. 1953–1963. PMLR, 2021.

Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research*, 74:1159–1199, 2022.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pp. 271–278, 1993.

Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

Carlos Diuk, Andre Cohen, and Michael L. Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 240–247, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems 32*, pp. 15246–15257. 2019a.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019b.

Samuel J Gershman. On the blessing of abstraction. *SAGE Publications Sage UK: London, England*, 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Alison Gopnik. Childhood as a solution to explore–exploit tensions. *Philosophical Transactions of the Royal Society B*, 375(1803):20190502, 2020.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse domains through world models. *CoRR*, abs/2301.04104, 2023.

Jessica B Hamrick, Abram L Friesen, Feryal Behbahani, Arthur Guez, Fabio Viola, Sims Witherspoon, Thomas Anthony, Lars Buesing, Petar Veličković, and Théophane Weber. On the role of planning in model-based deep reinforcement learning. *arXiv preprint arXiv:2011.04021*, 2020.

Steven Hansen, Guillaume Desjardins, Kate Baumli, David Warde-Farley, Nicolas Heess, Simon Osindero, and Volodymyr Mnih. Entropic desired dynamics for intrinsic control. *Advances in Neural Information Processing Systems*, 34:11436–11448, 2021.

Zhiao Huang, Fangchen Liu, and Hao Su. Mapping state space using landmarks for universal goal reaching. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 1940–1950, 2019.

Yuu Jinnai, Jee Won Park, David Abel, and George Konidaris. Discovering options for exploration by minimizing cover time. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019.

Yuu Jinnai, Jee Won Park, Marlos C. Machado, and George Konidaris. Exploration in reinforcement learning with deep covering options. In *International Conference on Learning Representations*, 2020.

Leslie Pack Kaelbling. Learning to achieve goals. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1094–1099. Citeseer, 1993.

Steven Kapturowski, Georg Ostrovski, Will Dabney, John Quan, and Remi Munos. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019.

Steven Kapturowski, Víctor Campos, Ray Jiang, Nemanja Rakićević, Hado van Hasselt, Charles Blundell, and Adrià Puigdomènech Badia. Human-level atari 200x faster. *arXiv preprint arXiv:2209.07550*, 2022.

Martin Klissarov and Marlos C Machado. Deep laplacian-based options for temporally-extended exploration. *arXiv preprint arXiv:2301.11181*, 2023.

Martin Klissarov, Akhil Bagaria, Ziyan Luo, George Dimitri Konidaris, Doina Precup, and Marlos C. Machado. Discovering temporal structure: An overview of hierarchical reinforcement learning. *ArXiv*, abs/2506.14045, 2025.

G. Konidaris. *Autonomous Robot Skill Acquisition*. PhD thesis, University of Massachusetts at Amherst, 2011.

George Konidaris. On the necessity of abstraction. *Current Opinion in Behavioral Sciences*, 29: 1–7, 2019.

George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems*, pp. 1015–1023, 2009.

George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61:215–289, 2018.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

Seungjae Lee, Jigang Kim, Inkyu Jang, and H. Jin Kim. DHRL: A graph-based approach for long-horizon and sparse hierarchical reinforcement learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations*, 2019.

Stephen R Lindemann and Steven M LaValle. Incrementally reducing dispersion by increasing voronoi bias in RRTs. In *IEEE International Conference on Robotics and Automation*, volume 4, pp. 3251–3257, 2004.

Chunlok Lo, Kevin Roice, Parham Mohammad Panahi, Scott M Jordan, Adam White, Gabor Mihucz, Farzane Aminmansour, and Martha White. Goal-space planning with subgoal models. *Journal of Machine Learning Research*, 25(330):1–57, 2024.

Sam Lobel, Akhil Bagaria, and George Konidaris. Flipping coins to estimate pseudocounts for exploration in reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22594–22613. PMLR, 23–29 Jul 2023.

Marlos C. Machado. *Efficient Exploration in Reinforcement Learning through Time-Based Representations*. PhD thesis, PhD thesis, University of Alberta, Canada, 2019.

Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. A laplacian framework for option discovery in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pp. 2295–2304, 2017.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Marlos C. Machado, Andre Barreto, and Doina Precup. Temporal abstraction in reinforcement learning with the successor representation. *Journal of Machine Learning Research (JMLR)*, 24 (80):1–69, 2023.

Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(Oct):2169–2231, 2007.

Timothy A Mann, Shie Mannor, and Doina Precup. Approximate value iteration with temporally extended actions. *Journal of Artificial Intelligence Research*, 53:375–438, 2015.

Willie McClinton, Andrew Levy, and George Konidaris. HAC explore: Accelerating exploration with hierarchical reinforcement learning. *CoRR*, abs/2108.05872, 2021.

Elizabeth Amy Mcgovern. *Autonomous discovery of temporal abstractions from interaction with an environment*. PhD thesis, University of Massachusetts at Amherst, 2002.

Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 24379–24391, 2021.

Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 28, 2015.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018a.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 3303–3313, 2018b.

S. Nasiriany, V. Pong, S. Lin, and S. Levine. Planning with goal-conditioned policies. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, volume 99, pp. 278–287, 1999.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International conference on machine learning*, pp. 3878–3887. PMLR, 2018.

Miruna Pislar, David Szepesvari, Georg Ostrovski, Diana L. Borsa, and Tom Schaul. When should agents explore? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

Silviu Pitis, Harris Chan, Stephen Zhao, Bradly C. Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7750–7761. PMLR, 2020.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *Proceedings of the 37th International Conference on Machine Learning, ICML*, 2019.

Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts at Amherst, 2001.

Balaraman Ravindran. *An algebraic approach to abstraction in reinforcement learning*. PhD thesis, University of Massachusetts at Amherst, 2004.

Rafael Rodriguez-Sanchez and George Konidaris. Learning abstract world models for value-preserving planning with options. *RLJ*, 4:1733–1758, 2024.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320, 2015.

Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990-2010). *IEEE Trans. Auton. Ment. Dev.*, 2(3):230–247, 2010.

Max-Philipp B. Schrader. gym-sokoban. https://github.com/mpSchrader/gym-sokoban, 2018.

Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model. *Advances in Neural Information Processing Systems*, 34:27580–27591, 2021.

Naman Shah and Siddharth Srivastava. Hierarchical planning and learning for robots in stochastic settings using zero-shot option invention. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 10358–10367. AAAI Press, 2024.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations (ICLR)*, 2020a.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. (arXiv:1907.01657), February 2020b. arXiv:1907.01657 [cs, stat].

Özgür Simsek and Andrew G. Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In Carla E. Brodley (ed.), *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

Özgür Şimşek, Alicia P Wolfe, and Andrew G Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd international conference on Machine learning*, pp. 816–823. ACM, 2005.

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Bruce W. Porter and Raymond J. Mooney (eds.), *Machine Learning, Proceedings of the Seventh International Conference on Machine Learning, Austin, Texas, USA, June 21-23, 1990*, pp. 216–224. Morgan Kaufmann, 1990.

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

Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

Richard S Sutton, Michael Bowling, and Patrick M Pilarski. The Alberta plan for AI research. *arXiv preprint arXiv:2208.11173*, 2022.

Richard S. Sutton, Marlos C. Machado, G. Zacharias Holland, David Szepesvari, Finbarr Timbers, Brian Tanner, and Adam White. Reward-respecting subtasks for model-based reinforcement learning (abstract reprint). In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan

(eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 22713. AAAI Press, 2024.

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

Adrien Ali Taïga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. Benchmarking bonus-based exploration methods on the arcade learning environment. *arXiv preprint arXiv:1908.02388*, 2019.

Erik Talvitie. Self-correcting models for model-based reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Vivek Veeriah, Junhyuk Oh, and Satinder Singh. Many-goals reinforcement learning. *arXiv preprint arXiv:1806.09605*, 2018.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3540–3549, 2017.

B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4:419–420, 1962.

Tom Zahavy, Brendan O'Donoghue, Andre Barreto, Sebastian Flennerhag, Volodymyr Mnih, and Satinder Singh. Discovering diverse nearly optimal policies with successor features. In *ICML Workshop on Unsupervised Reinforcement Learning*, 2021.

Lunjun Zhang, Ge Yang, and Bradly C. Stadie. World model as a graph: Learning latent landmarks for planning. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12611–12620. PMLR, 2021.

A Domain Descriptions

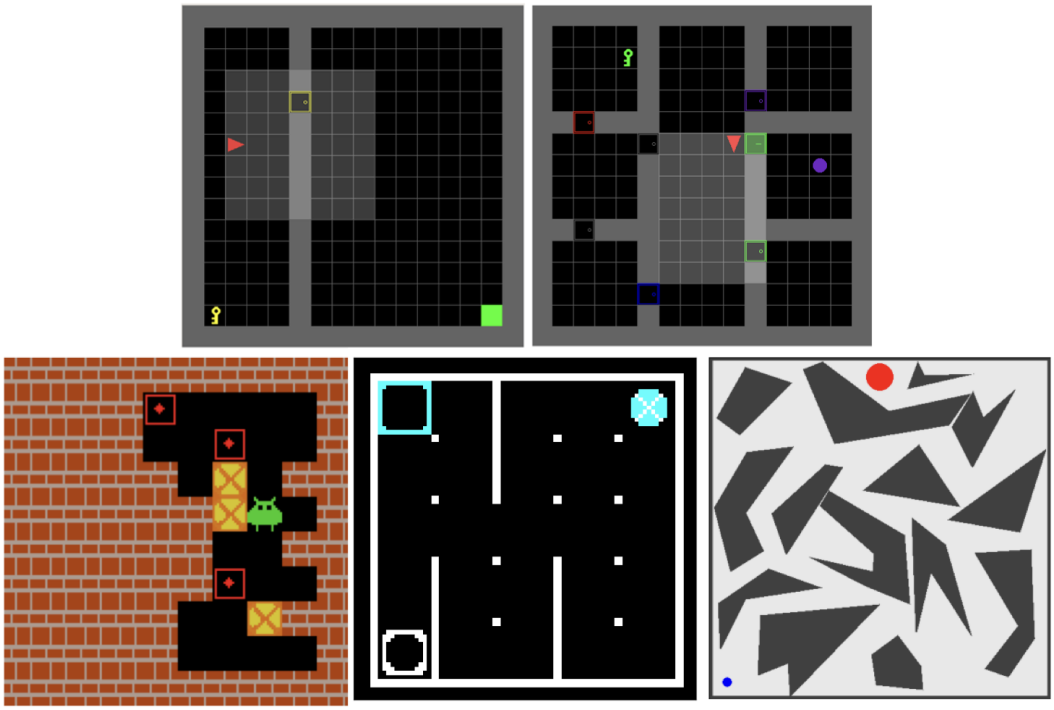

Figure 6: Domains used to test the IM-DSG agent. From top-left in left to right order: MINIGRID-DOORKEY, MINIGRID-KEYCORRIDOR, SOKOBAN, VISUALTAXI, and VISUALPINBALL.

1. MINIGRID-DOORKEY: To reach the goal, the agent must first pick up a key and then use that key to unlock a door. There is no intermediate reward for picking up the key, only a sparse terminating reward for reaching the goal. Each episode lasts a maximum of 200 steps.
2. MINIGRID-KEYCORRIDOR: This domain also has a key and a locked door, but additionally has other doors that can be open and closed. The goal is to pick up the purple ball that is placed in locked room (center-right room in Figure 6(c)). Each episode lasts a maximum of 1000 steps.
3. SOKOBAN: In this classic puzzle, the agent must place 3 boxes into their target locations (red border squares in Figure 6). The environment provides intermediate rewards for putting each box into its place and a terminating reward for correctly placing the final box. Each episode lasts a maximum of 200 steps.
4. VISUALTAXI: this is an image-based version of the classic Taxi problem (Dietterich, 2000). A passenger awaits in one of four depots and must be dropped off at a destination depot. Successful completion of the full task yeilds a sparse terminating reward of $+1$. Each episode lasts a maximum of 50 steps.
5. VISUALPINBALL: image-based version of the under-actuated Pinball domain (Konidaris & Barto, 2009; Bacon et al., 2017) in which the agent provides acceleration to the ball in one of 4 directions (or no-op), causing the velocity of the ball to change over time. We pick the hardest configuration from Konidaris & Barto (2009) during training, and sample goal locations at random during testing. Test-time goals are communicated to the agent as images of the pinball in the desired location. Each episode lasts a maximum of 1000 steps.

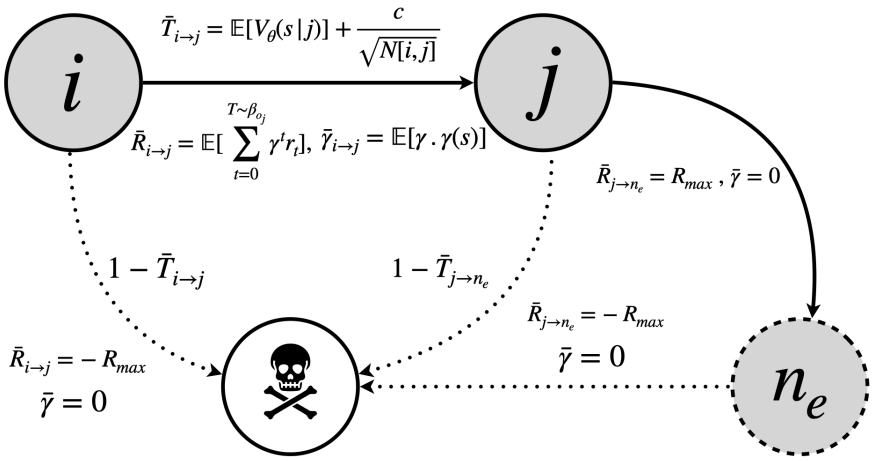

Figure 7: Summary illustration of a toy abstract MDP (AMDP) with 3 nodes. i and j are two ordinary nodes, n_e is the expansion node and the skull node shows a fictitious node that represents "falling off the graph". The transition probabilities $\bar{T}_{i\to j}$ are obtained using the UVFA V_θ and edge-traversal counts $N[i,j]$; the reward $\bar{R}_{i\to j}$ and discount models $\bar{\gamma}$ are obtained via monte carlo estimation from option executions.

B VISUALPINBALL Experiment Details

Both the IM-DSG agent and the Hindsight Experience Replay (HER) agent get the same training budget of 50M frames. During evaluation, both algorithms are tested on the same set of 10k goals and are given a budget of 10M frames to achieve them. Each episode lasts for at most for 1k steps. Error bars are computed over 5 training runs of the respective agents.

To compute the coverage metric shown in Figure 5 *(right)*, we first need to compute the maximum possible coverage in the maze. This is done by first discretizing the (x, y) plane into 100×100 squares, and then subtracting the approximate area occupied by the obstacles in the maze. To compute the coverage achieved by the two agents, we consider what fraction of the free space is occupied by all the states visited by the two agents.

Test-time modifications for VISUALPINBALL. During training, IM-DSG picks the expansion node based on V_{novelty}. At test-time, this would not be a good strategy because there is no reason that V_{novelty} would guide the agent to the test-time goal. So instead, we pick the expansion node to be the one that is closest to the test-time goal g_{test}:

$$n^{\text{test}}_{\text{expansion}} = \operatorname*{argmax}_n \mathbb{E}_{s\sim n}\left[V_{g_{\text{test}}}(s)\right] \tag{9}$$

Once the agent reaches $n^{\text{test}}_{\text{expansion}}$, it then executes its goal-conditioned policy $\pi(s, g_{\text{test}})$ to reach the test-time goal g_{test}.

C AMDP Construction

Figure 7 illustrates a 3 node toy AMDP: n_e is the expansion node, i and j are intermediate nodes and the skull represents the null node φ. Nodes of the graph are abstract states and outgoing edges from a node are available options. The abstract model, which is a combination of $\bar{T}, \bar{R}, \bar{\gamma}$, is estimated using the agent's goal-conditioned value function V_g and Monte Carlo estimation respectively.

Hyperparameter	Value	Tuned
Learning rate	**0.0001**, 0.00001	Yes
Trace length	40	No
Sequence period	20	No
Batch size	32	No
Burn-in length	0	No
Max replay size	500000	No
Target update period	**600**, 1200	Yes
Discount factor	0.997	No
Number of actors	32	No

Table 1: R2D2 Hyperparameters.

Hyperparameter	Value	Tuned
τ from Eq 6	0.7, **0.8**, 0.9	Yes
c from Eq 5	0.1, **0.2**	Yes
k in Eq 7	**1**, 2	Yes
τ_{template} in Eq 8	0.5	No

Table 2: IM-DSG Specific Hyperparameters.

D UVFA Training

We represent the agent's goal-conditioned value function using the following neural network architecture: a convolutional neural network (CNN; Goodfellow et al., 2016) torso encodes the image representing the current state s into $\Phi_1(s)$; another CNN encodes the goal g into $\Phi_2(g)$, where g is sampled from an option's effect set. $\Phi_1(s)$ is input into an LSTM (Goodfellow et al., 2016); the output of the LSTM, y_t is concatenated with $\Phi_2(g)$ and then passed through a multi-layered perceptron (MLP), which outputs the Q-value for each action in the MDP's action space. The goal encoder is identical to the state encoder, whose exact architecture, along with that of the LSTM and MLP, are taken without modification from Kapturowski et al. (2019). We follow Bagaria & Schaul (2023) and use novelty to pick which 5 achieved goals in a trajectory should be replayed in hindsight.

E Hyperparameters

Table 1 lists the hyperparameters for R2D2, Table 2 lists those hyperparameter settings that are specific to IM-DSG, and finally, Table 3 lists those IM-DSG hyperparameters that were environment specific. For non-IM-DSG hyperparameters, we first tune them based on the performance of the CFN agent on the MINIGRID-KEYCORRIDOR problem. Then, the same CFN and R2D2 hyperparameters are used in the IM-DSG algorithm, and other domains, without modification.

The descendant threshold for VISUALTAXI is higher than 0 because the passenger's destination is part of the state description. The threshold filters the descendant set $\mathcal{D}(s_t)$ to only include nodes that match the current passenger destination, ensuring expansion candidates are contextually relevant to the current task.

Hyperparameter	**DoorKey**	**KeyCorridor**	**VisualTaxi**	**VisualPinball**	**Sokoban**
Option Horizon H	200	400	50	200	50
Descendant threshold	0	0	0.3	0	0

Table 3: IM-DSG Hyperparameter settings for different domains