Discovering Temporal Structure: An Overview of Hierarchical Reinforcement Learning

Martin Klissarov[*] MARTIN.KLISSAROV@MAIL.MCGILL.CA
Mila, McGill University

Akhil Bagaria[*†] AKHILBG@AMAZON.COM
Amazon

Ziyan "Ray" Luo ZIYAN.LUO@MAIL.MCGILL.CA
Mila, McGill University

George Konidaris GDK@CS.BROWN.EDU
Brown University

Doina Precup[‡] DPRECUP@CS.MCGILL.CA
Mila, McGill University
Canada CIFAR AI Chair

Marlos C. Machado[‡] MACHADO@UALBERTA.CA
Amii, University of Alberta
Canada CIFAR AI Chair

Abstract

Developing agents capable of exploring, planning and learning in complex open-ended environments is a grand challenge in artificial intelligence (AI). Hierarchical reinforcement learning (HRL) offers a promising solution to this challenge by discovering and exploiting the temporal structure within a stream of experience. The strong appeal of the HRL framework has led to a rich and diverse body of literature attempting to discover a useful structure. However, it is still not clear how one might define what constitutes good structure in the first place, or the kind of problems in which identifying it may be helpful. This work aims to identify the benefits of HRL from the perspective of the fundamental challenges in decision-making, as well as highlight its impact on the performance trade-offs of AI agents. Through these benefits, we then cover the families of methods that discover temporal structure in HRL, ranging from learning directly from online experience to offline datasets, to leveraging large language models (LLMs). Finally, we highlight the challenges of temporal structure discovery and the domains that are particularly well-suited for such endeavours.

1. Introduction

Reinforcement learning (RL) is a general computational framework for building agents that learn to maximize a scalar reward from their experience. RL agents sense their environment and produce actions at every single timestep, yet effective reward maximization in complex environments requires reasoning and learning over many timescales, spanning vast horizons. Consider how we typically go about our day: as we actuate muscles every few milliseconds,

∗. Equal contribution.
†. Work done while at Brown University.
‡. Equal supervision.

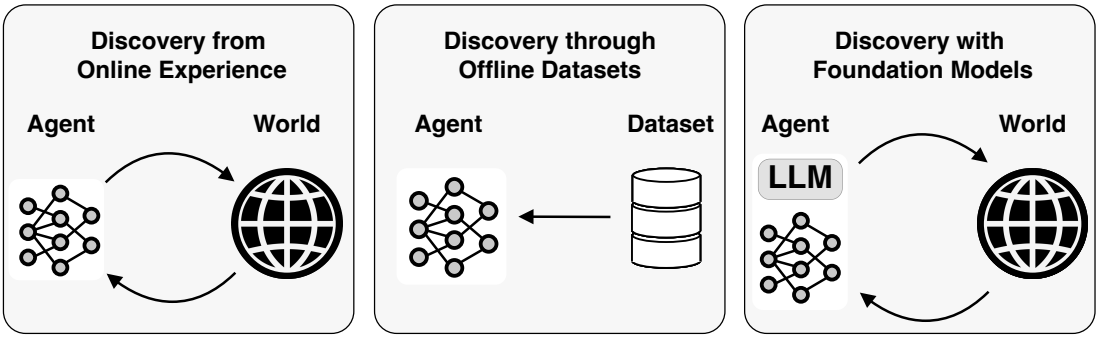


Figure 1: **Overview of the methods for temporal structure discovery.** We focus on the problem of discovering temporal structure autonomously from data. We put the discovery problem in perspective of the overall agent, covering the major benefits of Hierarchical Reinforcement Learning as well as the associated challenges and trade-offs.

we simultaneously perform high-level decisions such as choosing presents for a loved one, deciding what to eat for lunch, figuring out meaningful scientific questions, and so on. Such abstract decision-making allows us to make decisions in a complex world, without being overwhelmed with unnecessary detail.

Hierarchical reinforcement learning (HRL) formalizes the idea of flexibly reasoning over different timescales by developing agents that learn, predict, and act in the world at multiple levels of abstraction. At its core, HRL builds on the temporal structure revealed through interaction with an environment. These can be leveraged either within a learning algorithm, for example, as a curriculum over goals, or by defining a set of useful and reusable skills. When the temporal structure is defined by human specialists, HRL can dramatically ease the decision-making burden of the agent by improving exploration (Bellemare et al., 2020), learning (Vinyals et al., 2019), and generalization (Ahn et al., 2022). On the other hand, when the temporal structure underpinning HRL is poorly defined, it can hamper learning—for example, resulting in pathologically bad exploration (Jong et al., 2008). These appeals and drawbacks naturally lead to the question: how can agents autonomously discover useful temporal structures in HRL?

Before designing algorithms that successfully address the discovery problem, we are faced with the question of what constitutes a "good" temporal structure in the first place. Is there one type of "good" structure that yields higher rewards in all possible environments? Are there specific types of problem settings, like multi-task learning (Plappert et al., 2018) and continual learning (Khetarpal et al., 2020c), where we expect HRL to outperform non-hierarchical RL, and others where we do not? How can prior knowledge, for instance, through the integration of large language models (LLMs), alleviate the difficulties of discovery? This work presents various perspectives on what constitutes "good" temporal structures through the lens of the fundamental problems of RL—specifically, how HRL can aid **exploration**, **credit assignment**, **transfer**, and **interpretability**.

Key Contributions. The discovery of useful temporal structures has been a prolific, albeit challenging, topic of research. Before we present the various algorithms that tackle this problem, we take a step back and discuss the potential **benefits** of HRL methods as

well as their **trade-offs** in the context of sequential decision-making. It is through the lens of these benefits and trade-offs that we introduce the diverse approaches that have been developed to tackle the fundamental question of discovery. While recent surveys in HRL (Pateria et al., 2021; Hutsebaut-Buysse et al., 2022) present papers based on their technical differences and domains of application, we present the literature based on how each method contributes to these core benefits. We then discuss the **challenges** associated with discovering structure in HRL and the **domains** that are particularly well-suited for such methods.

Scope. Almost all of the algorithms we cover are compatible with deep neural networks. We categorize approaches in terms of the amount of prior knowledge, presenting works that (1) learn directly from the agent's **online experience**, (2) leverage **offline datasets** through offline RL, and (3) build on **foundation models** such as LLMs to define policies and rewards.

Overview. Section 2 discusses the benefits of the HRL framework and the different trade-offs faced when discovering temporal structure. Section 3 introduces the notation and fundamental concepts used throughout the paper. In Section 4, 5, and 6, we present methods that try to answer the central question in HRL: how can agents effectively discover temporal structure in a stream of data? These sections are divided into methods that learn directly from interaction, methods that leverage offline datasets, and more recent methods that build on foundation models. In Section 7, we present approaches that investigate how an agent might deliberate over the skills it has mastered to achieve different goals. In Section 8, we discuss the challenges of discovering temporal structure through HRL. In Section 9, we explore additional related fields to HRL and how they are interconnected, such as research on state and action abstractions, continual RL, and programmatic RL. Finally, in Section 10, we highlight environments and domains that are particularly promising for HRL research, with a particular focus on open-ended systems.

2. What is Hierarchical Reinforcement Learning for?

Hierarchical reinforcement learning (HRL) aims to exploit the **temporal structure** of sequential decision-making problems. Solutions to complex problems can often be *approximated* by deconstructing the problem into simpler sub-problems that are modular and composable. **Modularity** refers to the property that a solution to a subproblem can be reused without concern for exactly how it was solved. **Compositionality** means that sub-problems can subsequently be recombined to create solutions to a wide range of more complex problems.

To better understand what such a structure might represent in practice, consider a programmer with an abundance of time who cares about solving only a single task. In such a scenario, Assembly language might be the optimal choice because its precise control over hardware resources potentially maximizes memory efficiency and minimizes execution time. However, in practice, programmers often opt for higher-level programming languages and use external software libraries because they offer compositional modules that solve common programming subtasks, and therefore make the writing of most new programs more efficient, at the cost of increasing execution time. In fact, such programming languages allow us to quickly solve complex problems; without them, most large software projects would simply be

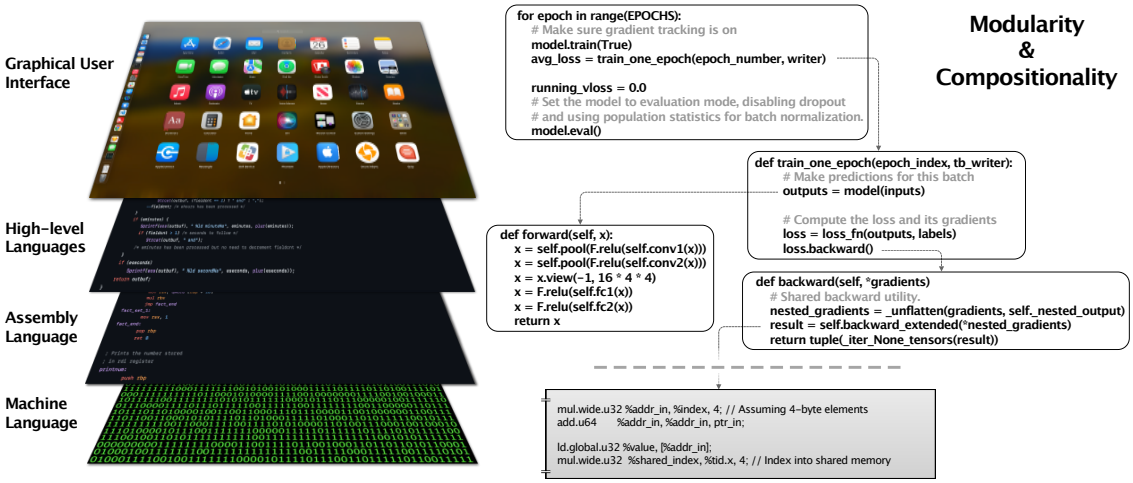


Figure 2: **(Left)** The hierarchy of software and languages which make computers especially useful, allowing humans to directly interact with a user interface or a high-level language to achieve a diversity of goals. **(Right)** a code snippet, revealing the usefulness of modular and compositional software structures that researchers use every day. The PyTorch language, which abstracts over the PTX language used within CUDA kernels, allows exploring research ideas efficiently.

infeasible. This idea is visually represented in Figure 2 on the left, where abstract interfaces allow us to manipulate machine language efficiently. Modularity and compositionality are also particularly appealing properties for software expected to undergo changes throughout its life cycle. In a software library, each function typically handles a specific subtask and can be composed within a sequence of function calls to achieve a larger objective. In some more complex tasks, functions might call other functions. Developing such a library requires careful consideration of the right code organization to adopt and which guiding principles to follow, balancing execution time, readability, and performance. However, once a library is written, the user can focus on the overall program's behaviour without needing to understand the implementation details of each function (Wilkes et al., 1958), greatly empowering the user's ability to achieve their goals. This is represented in Figure 2 on the right, where the modular and compositional nature of PyTorch allows researchers to efficiently explore research ideas.

The temporal structure at the core of HRL is analogous to functions and subroutines in programming languages.[1] Just as a human programmer writing a complex program is faced with the difficulty of breaking their task into subtasks, so must RL agents autonomously identify hierarchical structure in a stream of data. The modularity and compositionality properties are therefore good indicators as to the kind of problems in which we might find HRL particularly useful (see Section 10). We now discuss how, by discovering and leveraging such temporal structure, HRL methods can help address fundamental challenges in decision-making.

1. We provide more details about this analogy in Section 9.3.

2.1 The Benefits of Hierarchical Reinforcement Learning

Just as modular and compositional codebases can facilitate effective software development of complex systems, HRL can leverage an environment's structure to improve decision-making. This is particularly powerful when an agent is faced with tasks spanning vast horizons. By breaking down such long horizons into manageable subgoals, HRL effectively affords learnability. How can we understand this more precisely? In this section we attempt to provide a comprehensive perspective on the benefits of HRL through the lens of three fundamental challenges agents face when learning from interaction: how to select the right data to collect (*exploration*), how to efficiently learn from this data (*credit assignment*), and how to transfer knowledge and behaviour to new situations (*transferability*). Additionally, as agents become increasingly more capable, a new challenge emerges: understanding their decision-making processes (*interpretability*). When covering the different families of HRL methods (Sections 4, 5, and 6), we will explicitly consider how these benefits are instantiated in practice.

Exploration. Broadly speaking, RL agents must solve two problems: (a) how to use existing data to learn useful behaviours, and (b) what data to collect in the first place. The latter problem, known as the *exploration* problem, is both unique and central to RL; the agent must learn how to collect data that improves its understanding of the world even if doing so does not immediately maximize reward in the short term (Amin et al., 2021). By exploiting the temporal structure of an environment, an agent can improve exploration in at least three ways. First, it can seek subgoals that are closer and more achievable than the overarching task's goal, potentially creating a progressive curriculum of subgoals that allows the agent to explore more effectively. Second, it can explore in a diversity of directions, each defined by a skill in the agent's skill set. Finally, agents can explore at a higher level of abstraction than individual actions, enabling them to search the solution space more efficiently. Consider a researcher tackling an important scientific question. By learning a high-level programming language, such as PyTorch, and writing modular code, the researcher can iterate faster to investigate many high-level ideas. When iterating over ideas, the researcher may seek to achieve some important milestones, such as a proof of concept, that can reveal new perspectives and provide insights into possible future courses of action.

Credit Assignment. To improve its decisions over time, an agent must identify the key moments in a sequence of decisions that best explain the observed result. RL algorithms typically leverage multi-step error propagation (Sutton, 1988) to learn about temporally distant, or *delayed*, outcomes. An agent leveraging the environment's temporal structure could more efficiently identify the origins of an outcome by propagating errors at the abstraction level defined by this structure. Consider our previous example of a researcher performing a scientific experiment. Completing such an experiment consists of a sequence of high-level decisions, such as the choices of data preprocessing or evaluation metrics. Each of these high-level decisions is instantiated through a series of keystrokes that make up the final working code. By reflecting on the validity of the sequence of high-level decisions, rather than of each individual keystroke, the researcher could better identify which ones were critical for the observed outcome and how this sequence could be improved. By breaking down a task into such segments, it is also easier to identify if a particular segment is completed,

narrowing down the search for lower-level mistakes, such as where an errant keystroke might have introduced a bug.

Transfer. HRL offers a particularly promising way of exploiting structure shared between a family of problems: skills acquired in one task can seamlessly be *transferred* to another. Agents could achieve this by breaking a complex task into simpler subtasks that have the potential to recur in many contexts and then learn skills that achieve such subtasks. Faced with a new challenge, such an agent can re-compose the skills, either by sequencing them or acting according to a mixture of them efficiently. Consider our previous example of an AI researcher conducting experiments and writing a paper for a particular conference. The collection of research code subroutines and writing skills learned while writing this initial paper could substantially reduce the complexity of writing a follow-up article, further improving their ability to conduct impactful research. A set of skills can also serve as a foundation for learning increasingly more complex ones, as a form of auto-curriculum.

By addressing the three aforementioned fundamental challenges, HRL aims to achieve *faster learning and planning*, ultimately improving the agent's problem-solving capabilities. Beyond these, HRL also has the potential to tackle the additional challenge of interpretability.

Interpretability. While not all HRL algorithms produce interpretable behaviour, those that do offer the unique advantage of allowing human observers to better understand an agent's decision-making process. As such, agents become increasingly powerful; they will eventually be deployed in real-world situations where the consequences of their actions carry considerable stakes. A crucial requirement would then be our capacity to ensure their alignment, and interpreting their decisions is a key aspect of this challenge (Amodei et al., 2016). HRL could provide an interpretable interface for AI alignment via a more abstract decision formulation than the one defined in the environment. For instance, in a robot navigation task, low-level actions such as the forces applied at the joints are particularly hard to interpret compared to a sequence of semantically meaningful skills such as "reach the stairs" and "descend to the first floor". Humans may be able to follow the agent's reasoning at that level of abstraction, and provide feedback as to the type of goals that are to be preferred.

2.2 Trade-offs

HRL builds on the inductive bias that tasks can be naturally decomposed into simpler, modular, and compositional subtasks, making it especially effective when such a hierarchical structure is apparent. However, while it offers several potential advantages, consistent with the No Free Lunch Theorem (Wolpert and Macready, 1997), HRL is not guaranteed to outperform non-hierarchical RL methods across all tasks. Its bias may lead to suboptimal outcomes when the assumed structure does not match the problem's underlying one. In other words, a poorly chosen task decomposition can sometimes make a problem harder, not easier. Taking the analogy of programming, adding abstractions in a codebase can simultaneously help in seeing a broader picture, but also obscure the important details (Victor, 2011). As a result, for any task or environment, HRL agents face a trade-off between *performance*, *sample efficiency*, and *computation efficiency*, as illustrated in Figure 3.

We first consider the trade-off between performance and sample efficiency. This trade-off can be appreciated from a theoretical point of view: under standard assumptions, the optimal

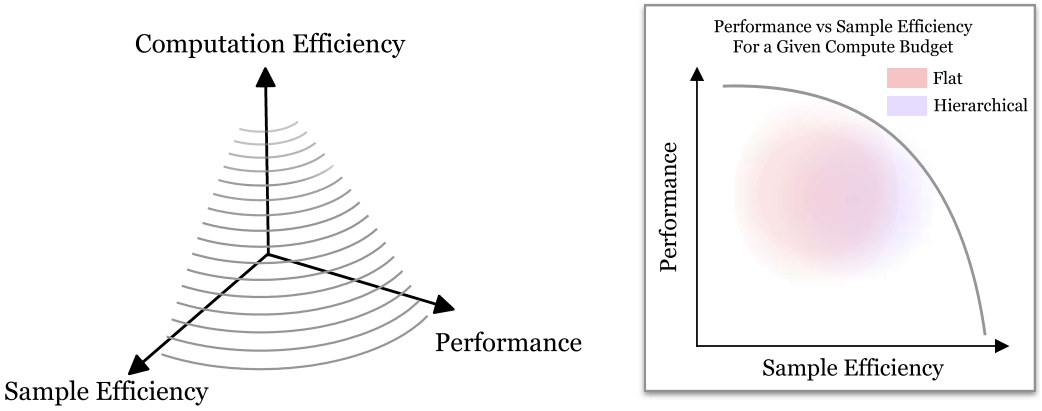


Figure 3: **(Left)** Agents trade-off between three different objectives: performance (in terms of reward), sample efficiency (the amount of data required to reach a certain performance), and computational efficiency (amount of computation needed to do so). The Pareto frontier between performance and sample efficiency shifts with different compute budgets. For instance, with unlimited compute (a low computational efficiency), high performance can be achieved at moderate sample efficiency. **(Right)** A qualitative illustration of a "flat" agent and a hierarchical agent under the Pareto frontier of the performance vs. sample efficiency trade-off, given a fixed compute budget. A "flat" agent, as opposed to a hierarchical one, does not use temporally extended actions. While it is not always the case, hierarchical agents tend to trade some amount of performance for improved sample efficiency.

policy can always be represented using primitive actions alone (Bertsekas, 1995). However, learning the optimal policy for large and realistic environments is often simply intractable. Rather than pursuing perfect solutions, an agent should embrace efficient learning algorithms to develop reasonable but often suboptimal policies. This is one of the main appeals of HRL: by re-composing existing solutions (i.e., behaviours achieving particular subtasks), an agent may be able to quickly find approximate solutions for a variety of tasks, offering a promising way to trade off optimality with sample efficiency. For example, a pre-trained skill that opens doors allows an agent to bypass learning the complex motor controls for that specific action. However, this very abstraction can be limiting; if a door is stuck and requires an unusual push-and-jiggle motion, the rigid pre-defined skill might fail, preventing the agent from solving an edge case that a more flexible, low-level policy could have.

Another important challenge faced by agents interacting with complex and realistic environments is computational efficiency: the amount of computation spent selecting the right action at each timestep. Such computation can correspond to the neural network size, the maximum depth allowed for an agent using Monte Carlo Tree Search (Coulom, 2006), or the length of the reasoning trace of an LLM. Suppose the computation time is unrestricted, e.g., for agents acting in a simulator with the liberty of performing thousands of imagined rollouts for each timestep. In such a case, large amounts of computation can be spent on planning the next action. However, in real-world scenarios, the compute time per timestep is constrained. As HRL agents make decisions at a high level of abstraction, computation time can be managed more flexibly. For example, a robotic agent equipped with an LLM

may plan over a set of semantically meaningful skills, which is significantly smaller than the underlying continuous action space. Since each high-level decision made by such an agent typically takes place over multiple timesteps, the cost of deliberating is naturally amortized over such timescales.

On the Importance of Knowledge Reuse. A common pitfall in HRL applications is that the number of interactions required to discover the hierarchical structure of a problem can be greater than the number of interactions needed to solve the problem itself, highlighting the importance of carefully considering the types of problem for which HRL is used (see Section 10). This cost can be amortized in different ways. For example, it may be offset if the agent is expected to complete *many different tasks within its lifetime*, allowing the learned subtasks to be potentially reused. Alternatively, reusing existing knowledge—such as **offline datasets** (Section 5) and **foundation models** (Section 6)—can also help mitigate this cost. As we will see, HRL's formalism offers a natural and particularly promising way to incorporate such prior knowledge.

3. Formalizing Hierarchical Reinforcement Learning

We use the notation introduced by Sutton and Barto (2018): capital letters refer to random variables, whereas lowercase letters refer to their instantiation. Table 2 summarizes the notation used in this section.

3.1 Reinforcement Learning

We consider an agent interacting with an environment where the agent is in state $S_t \in \mathcal{S}$ at timestep t, selects an action $A_t \in \mathcal{A}$, and in response the environment emits a scalar reward $R_{t+1} \in \mathbb{R}$ and transitions to a new state, $S_{t+1} \in \mathcal{S}$. This transition happens according to a transition probability distribution,

$$p(s'|s,a) = p(S_{t+1} = s'|S_t = s, A_t = a). \tag{1}$$

The agent's goal is to find a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ is the distribution over $\mathcal{A}$, that maximizes the expected discounted sum of rewards (return):

$$G_t = \mathbb{E}_\pi \left[\sum_{i=t}^{\infty} \gamma^{i-t} R_{i+1} \right], \tag{2}$$

where $\gamma \in [0, 1)$ is the discount factor. This 5-tuple, $\langle \mathcal{S}, \mathcal{A}, R, p, \gamma \rangle$ defines a Markov Decision Process (MDP) (Puterman, 1994), the most commonly accepted formalism in RL.

When following a particular policy π, the value of each state can be represented by the state value function,

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t | S_t = s \right]. \tag{3}$$

Similarly, we may consider the value of being in state s and taking action a, following policy π afterward, represented by the action value function, or q-function,

$$q_\pi(s,a) = \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a \right]. \tag{4}$$

This function can be written recursively,

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \mid S_t = s, a_t = a, \pi] \\ &= r(s,a) + \gamma \sum_{s'} p(s'|s,a) v_\pi(s') \\ &= r(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a'} \pi(s',a') q_\pi(s',a'). \end{aligned} \tag{5}$$

A similar derivation is possible for the value function, and this set of equations is referred to as the Bellman equations for evaluation (Bellman, 1957).

The goal of an RL agent is to maximize the rewards it gets from interacting with the environment. In an MDP, there exists at least one optimal policy, defined as,

$$\pi^* = \arg\max_\pi q_\pi(s,a). \tag{6}$$

In most settings, this quantity is impractical to compute exactly, and we must resort to approximation. Such approximations stem from two families of algorithms for learning reward-maximizing policies. The first family of methods, called value-based methods, greedily maximizes an estimated action-value function. Q-Learning (Watkins and Dayan, 1992) is likely the most used algorithm to estimate the optimal policy, π^*, whose update rule takes the following form,

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t) \right]. \tag{7}$$

This update has been the basis of the Deep Q-Networks algorithm (Mnih et al., 2015).

The second family of methods directly maximizes the quantity of interest, that is, the discounted sum of returns. The policy gradient theorem (Sutton et al., 1999a) provides the gradient of the expected discounted return from an initial state distribution, $d(s_0)$, with respect to a stochastic policy, $\pi_\zeta(\cdot|s)$, parameterized by ζ,

$$\frac{\partial J(\zeta)}{\partial \zeta} = \sum_s d_\pi^\gamma(s) \sum_a \frac{\partial \pi_\zeta(a|s)}{\partial \zeta} q_\pi(s,a), \tag{8}$$

where $d_\pi^\gamma(s) = \sum_{s_0} d(s_0) \sum_{t=0}^\infty \gamma^t \sum_a P_\pi(S_t = s | S_0 = s_0)$ is the discounted state occupancy measure, and $P_\pi(S_t = s|S_0 = s_0)$ the probability of reaching state s from s_0 in t steps when following policy π. This update has been the basis of many modern algorithms, including the well-known proximal policy optimization (Schulman et al., 2017).

3.2 Hierarchical Reinforcement Learning

The temporal structure an agent learns using HRL has been formalized in a variety of names, such as **skills**, **options**, **temporal abstractions**, or **goal-conditioned policies**, amongst others. These frameworks carry their own notations and focus on particular methodologies or research questions. We adopt the options formalism (Sutton et al., 1999b; Precup and Sutton, 2000) as it provides a useful and comprehensive framework for expressing temporal structure. In Section 3.2.2, we expand on how alternative formalisms are fundamentally connected by focusing on what constitutes HRL at its core.

3.2.1 Options: A Mathematical Formalism

An HRL agent makes use of a set of options, $\mathcal{O}$, which are defined by three components: a policy, an initiation function, and a termination function. These components can be implemented through parameterized functions, such as neural networks, or symbolically through code (e.g., see Section 6). In Figure 4, we illustrate the temporal structure exhibited by options while interacting with an environment. More formally,

- $\pi : \mathcal{S} \times \mathcal{O} \to \Delta(\mathcal{A})$ is an option policy, which selects an action according to the current state and the current option.[2] This quantity can also be referred to as the skill's policy, the intra-option policy, or the goal-conditioned policy (see Section 3.2.2). When this function is parameterized by a set of weights θ, we will write $\pi_\theta(a|s, o)$.

- $\beta : \mathcal{S} \times \mathcal{O} \to [0, 1]$ is the option termination function, giving the probability with which option o should stop executing if it reaches state s. When this function is parameterized, we will use the notation $\beta_\psi(s, o)$, where ψ represents the termination function parameters.

- $\mathcal{I} : \mathcal{S} \times \mathcal{O} \to [0, 1]$ is the option initiation function, which determines to what degree an option o can start its execution from a certain state. Traditionally, this component is referred to as the initiation set, which determines the set of states in which an option can initiate. When this component is parameterized, we will use the notation $\mathcal{I}_\chi(s, o)$, where χ represents the initiation function parameters.

Additionally, to select among the set of options, an HRL agent uses:

- $\mu : \mathcal{S} \to \Delta(\mathcal{O} \cup \mathcal{A})$, the high-level policy, which outputs a probability distribution over the set of options $\mathcal{O}$ and actions $\mathcal{A}$ given a state s. When this probability is parameterized, we will use the notation $\mu_\kappa(o|s)$, where κ represents the high-level policy parameters. Similarly to the previous components, in practice, this policy can also be instantiated by other means, e.g., a programmatic policy that directly encodes domain knowledge or follows predefined rules (see Section 9.3).

When planning with options, an HRL agent will do so through:

- $P_\mathcal{O} : \mathcal{S} \times \mathcal{O} \times \mathcal{S} \to \mathbb{R}$, the option model function. This function takes as input a state s, an option o, a future state s', and the discount factor γ, and outputs a measure of how likely the option o will terminate at state s', at any point in the future.

Not all of the papers covered in this work will explicitly define each of these components. It is common for research in HRL to only highlight the components for which a significant contribution is made and to make assumptions about the other components. For instance, the termination function is often assumed to output termination after a fixed number of timesteps. Similarly, the option initiation function is often assumed to allow option initiation across the whole state space. We will highlight the relevant aspects within the presentation of each work.

2. We slightly abuse the notation here with respect to the symbol representing the policy of a non-hierarchical RL agent. The distinction between the two will be clear through context.

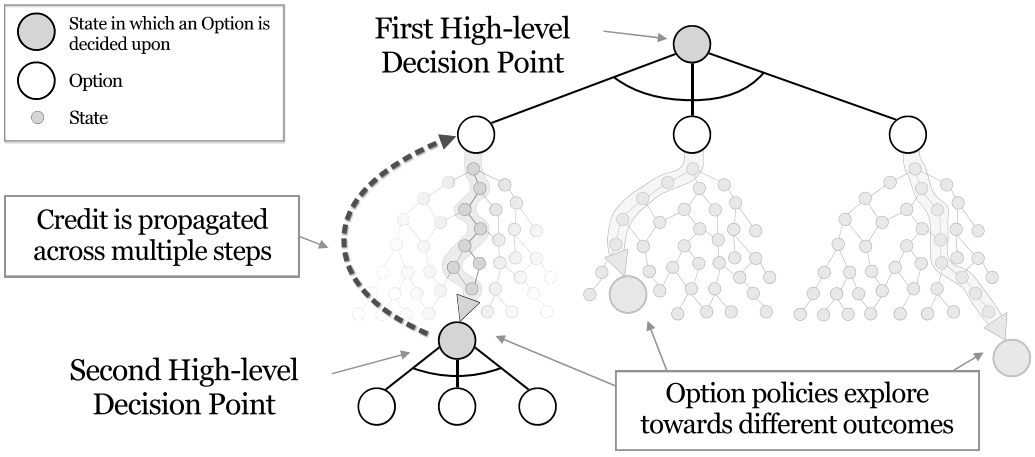


Figure 4: A simplified diagram illustrating the decision process of a hierarchical agent: large white nodes signify high-level decisions made over options, and large grey nodes represent the state observed by the agent at that moment. The high-level decisions can be made over a potentially infinite set of options, such as when option policies are represented as goal-conditioned policies. Grey trails represent state transitions during option execution. This diagram illustrates how different options last for different timescales and traverse the environment in a diversity of directions. After an option finishes execution, the agent must make its next high-level decision.

Using the presented terms, we can now define the option value function,

$$q_\pi(s, o) = \sum_a \pi(a \mid s, o) q_u(s, o, a), \tag{9}$$

where $q_u : \mathcal{S} \times \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ is the value of executing action a in the context of a state-option pair:

$$q_u(s, o, a) = r(s, a) + \gamma \sum_{s'} p(s' \mid s, a) u_\beta(o, s'). \tag{10}$$

The function $u_\beta : \mathcal{O} \times \mathcal{S} \to \mathbb{R}$ is called the option-value function upon arrival, that is, the value of executing option o upon entering a state s' is given by:

$$u_\beta(o, s') = (1 - \beta(s', o)) q_\pi(s', o) + \beta(s', o) v_\mu(s'). \tag{11}$$

Finally, the function $v_\mu : \mathcal{S} \to \mathbb{R}$ is defined as the value function over a set of options

$$v_\mu(s) = \sum_o \mu(o|s) q_\pi(s, o). \tag{12}$$

Subgoal Options. Technically, an option is simply described by a way of initiating, a way of acting, and a way of terminating—its behaviour need not maximize any objective at all. As an example, consider an option that initiates everywhere, terminates nowhere, and whose policy arbitrarily maps each state to an action; this is a well-defined option but does

not optimize any useful objective. However, for option discovery, rather than searching for these three quantities in their raw form, it is often more convenient to think of options as achieving *subgoals*. In fact, the vast majority of the literature on option discovery can be seen as achieving subgoals (e.g., McGovern and Barto, 2001; Precup, 2001; Colas et al., 2022; Sutton et al., 2023); we refer to these options as *subgoal options* (Bagaria, 2025). One way to learn options that achieve subgoals is through the following:

- $r^o : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, the option reward function is a function conditioned on an option o. We also refer to this quantity as the goal reward function. It takes as input a state s, an action a, and a next state s', and outputs a scalar reward. When maximized, it would produce the corresponding option policy. When parametrized, this function will use parameter notation ν.

It is important to note that not all subgoal options need to maximize option reward functions. Indeed, some approaches learn a set of useful behaviours through imitation learning (e.g., Le et al., 2018; Team et al., 2024). Alternatively, subgoal options can be defined by mapping states to actions through symbolic functions such as code (see Sections 6 and 9.3). It is also possible these quantities take a slightly different set of inputs, for example, the option reward function may only receive as input the current state s, written as $r^o(s)$.

3.2.2 Related Terminologies and Formalisms

The previous section uses the language of options to formalize the learned temporal structure. As the field of HRL is rich and diverse, some researchers may feel misrepresented by such a formalism. Therefore, throughout the paper, we may interchangeably refer to options (with the notation o for each option) as skills (with the notation z for each skill), goal-conditioned policies (with the notation g for each goal), or simply refer to the general term of temporal abstractions. We believe such differences in language are mostly superficial and may hinder the integration of the best practices from each of these fields. We now highlight the differences among various related formalisms and illustrate their connections.

Skills. The skill terminology has largely been used informally in the HRL literature to refer to temporally extended behaviours. Skills can most commonly be formalized using the options framework, but they can also be formalized using other formalisms such as macro-actions (Fikes and Nilsson, 1971), feudal hierarchies (Dayan and Hinton, 1993), MAXQ (Dietterich et al., 1998), and HAMs (Parr and Russell, 1997).

Goal-conditioned RL. A goal can be formally defined using a triple: (g, r^g, γ_g), where $g : \mathcal{S} \to \mathbb{R}^d$ is a goal vector that can, for example, be used to condition a policy, $\pi(a|s, g)$, or a value function, $v(s|g)$, $r^g : \mathcal{S} \to \mathbb{R}$, is a goal reward function that maps each state to real-valued number, and, finally, $\gamma_g : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ describes the goal's continuation function, and hence the timescale for achieving that goal (Kaelbling, 1993a; Schaul et al., 2015). Most work in goal-conditioned RL (GCRL) considers the state space to be the set of goals an agent should reach (Andrychowicz et al., 2017). To obtain useful measures of the distance between the current state and the goal state, a key emerging research question is defining representations that afford meaningful distance measures. Although some of the works from the GCRL literature are present in this work, we refer the reader to Liu et al. (2022) for an in-depth review.

Feudal RL. In Feudal RL (Dayan and Hinton, 1993), decision-making is divided across multiple levels of the hierarchy, where higher-level "managers" set subgoals for lower-level "workers" who are rewarded by their managers for achieving these subgoals. The space from which the manager draws subgoals is usually continuous, whereas options are usually instantiated as a discrete set of policies. In the previous section, we intentionally refrained from specifying the nature of the option set, accommodating both discrete and continuous sets. The concept of a continuous option set can be interpreted through the lens of parameterized skills (da Silva et al., 2012).

3.2.3 Beyond Architectural Choices

In the previous section, we mentioned that skills, options, and goal-conditioned policies were slightly different instantiations of the same fundamental principle. We now attempt to clarify this statement. In Figure 5, we depict a set of common instantiations of hierarchical architectures. One such architecture is a modular architecture of hierarchical components: a high-level policy is explicitly defined, together with a collection of options, each potentially implemented through neural networks. Our previous statement makes it obvious that HRL is not restricted to such a hierarchical architecture, despite the fact that it is quite common in the literature.

An alternative instantiation is the goal-conditioned neural network, which can be instantiated by an LLM (see Section 6). However, HRL is also not restricted to such an architecture. In fact, we argue that **HRL is fundamentally defined through the algorithm, not the agent architecture.** In the most general case, HRL can produce agents that are simply instantiated by a single, large neural network where the options and goals are implicitly learned and defined within the neurons themselves.

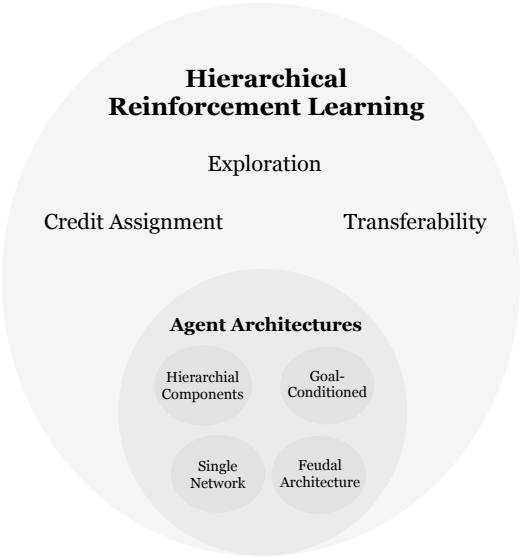


Figure 5: The agent architecture is a subproblem to the main question posed in HRL: how to discover temporal structure?

An HRL algorithm empowers the agent's exploration by selecting goals across time, and rewarding the agent for achieving them. It also facilitates more effective credit assignment by decomposing a long, continuous stream of experience into meaningful subtasks. Additionally, HRL can better prepare an agent for future challenges by promoting the learning of reusable behaviours, which can be explicitly or implicitly defined. These essentially represent the core benefits of HRL, as outlined earlier in this work in Section 2.1.

4. Discovery from Online Experience

In this section, we present work in option discovery that takes place in the *online* setting: the agent seeks to construct useful options by simply interacting with the environment. This setting has received significant attention because it holds the promise of scalability (Sutton,

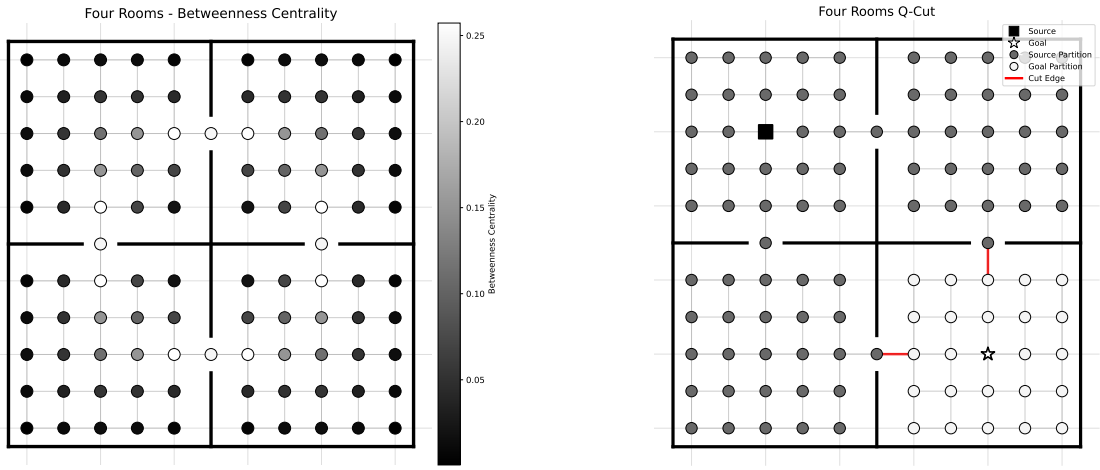
Table 1: A summary of HRL methods shown in Section 4, 5, and 6 that discover temporally abstract behaviors, **highlighting the main benefits** elaborated in Section 2.1. Each method links to the corresponding section. A single black dot (•) indicates that a class of methods generally contributes to addressing a specific challenge, while a double black dot (••) signifies that the class of methods is explicitly designed to tackle that challenge.

Categories	**Methods**	**Reference**	**Focus of Methodologies on Different Benefits**			
			Credit Assignment	Exploration	Transferability	Interpretability
Discovery from Online Experience	Bottleneck Principle	4.1	•	•	•	•
	Spectral Methods	4.2	•	••	•	
	Skill Chaining	4.3	••	••	•	•
	Empowerment Maximization	4.4	•	••	•	
	Via Environmental Reward	4.5	••		•	•
	Directly Optimizing the Benefits of HRL	4.6	•	•		
	Meta-Learning	4.7		•	••	
	Curriculum Learning	4.8		••	•	
	Intrinsic Motivation	4.9		••	•	
Discovery through Offline Datasets	Variational Inference	5.1	••	•	•	
	Hindsight Sub-goal Relabelling	5.2	••			•
Discovery with Foundation Models	Embedding Similarity	6.1		••	•	•
	Providing Feedback	6.2	•	••	•	•
	Reward as Code	6.3			••	•
	Directly Modeling the Policy	6.4		•	••	•

2019)—a long-lived agent that can learn new, useful options simply via interaction and can potentially keep increasing its competence in the world, bootstrapping new skills with previously discovered ones (Ring, 1995; Schmidhuber, 2010).

We broadly categorize this literature based on the proxy objectives used for option discovery. For each family of methods, we first describe the core intuition and key methodological patterns. Then, we discuss how each category contributes to the core benefits of HRL (as outlined in Section 2.1). Finally, we discuss some limitations of each category and highlight opportunities for research.

Before presenting the methods in detail, we direct the reader's attention to Table 1, which provides an overview of all the discovery methods discussed in this work. For each method, we highlight which benefits have been mostly studied by researchers from the field, where a single black dot (•) indicates that a class of methods generally contributes



(a) Betweenness Centrality

(b) Q-Cut

Figure 6: **Bottleneck Discovery in Four Rooms.** Skill discovery using (a) betweenness centrality, a measure of the likelihood that a state lies on the shortest path between any two other states, and (b) Q-cuts, which finds the edge that solves the Min-Cut problem on the transition graph. Both classes of methods attempt to identify bottleneck states and use them as option subgoals.

to addressing a specific challenge, while a double black dot (●●) signifies that the class of methods is explicitly designed to tackle that challenge.

4.1 Bottleneck Discovery

Many challenging problems in RL have *bottlenecks*, which are small regions of states that an agent must pass through to reach a larger, potentially more interesting region of the state space. For example, in the *Two Rooms* task (Sutton et al., 1999b; Solway et al., 2014), the agent must go through a doorway state to access the goal in the other room. Another example is a player in a video game who must pick up a key to unlock a door that leads to the other levels. In these examples, the doorway and the key act as bottlenecks—reaching those states grants the agent access to an entirely new region of interesting states. When an agent identifies such bottleneck states during learning, it defines a subgoal option (as in Section 3.2.1) to reach it. Specifically, the option terminates with a positive subgoal reward when it reaches the bottleneck state, and continues without termination or reward otherwise. When bottleneck states are successfully identified and targeted with subgoal options, the agent often improves exploration, credit assignment, and transfer, as we will soon discuss. Given this intuitive appeal, several papers have proposed algorithms for identifying bottlenecks.

Most algorithms for finding bottlenecks begin with a graph-based view of the MDP: states are treated as nodes and an edge exists between two states, (s, s'), when the agent can reach s' from s in a single timestep:

$$G = (\mathcal{S}, E), e_{s,s'} \in E = \mathbb{1}_{\sum_{a \in \mathcal{A}} p(s'|s,a) > 0}. \tag{13}$$

In this graph, bottlenecks have been described, and identified, using the following approaches:

Diverse Density

An early approach to option discovery used the concept of diverse density, which measures how much more likely a state is to lie on a successful trajectory than an unsuccessful one. McGovern and Barto (2001) formulate bottlenecks as states with highly diverse density and propose a simple algorithm to identify them. Concretely, consider that the agent has a set of successful trajectories, $\mathcal{T}^+$, (each is a sequence of states leading to a goal state) and a set of unsuccessful trajectories, $\mathcal{T}^-$, (sequences that did not reach the goal state). The diverse density score, $\text{DD}(s)$, captures the probability that a given state, s, occurs in successful trajectories and does not occur in unsuccessful trajectories:

$$\text{DD}(s) = \prod_{\tau \in \mathcal{T}^+} P(s \in \tau) \prod_{\tau \in \mathcal{T}^-} \Big(1 - P(s \in \tau)\Big), \tag{14}$$

where the probability that a state occurs in a trajectory can be computed in tabular domains using visitation counts:

$$P(s \in \tau) = \frac{\text{Number of times } s \text{ appears in } \tau}{|\tau|}. \tag{15}$$

States with a diverse density greater than a threshold are chosen as bottlenecks, and subgoal options are created to reach them. A drawback is that trajectories must be classified as positive or negative depending on whether they were on the path to a goal state. Stolle and Precup (2002) address this shortcoming by defining diverse density over a *family of tasks*: bottleneck states are those that are repeatedly visited while solving many goal-reaching tasks.

Graph partitioning

Under the graph view of MDPs, bottlenecks can be interpreted as "accumulation" nodes—states in which many paths or trajectories coincide. These accumulation nodes, or bottleneck states, tend to separate loosely connected sub-graphs, which are otherwise densely connected among themselves. To see why bottlenecks separate loosely connected sub-graphs, Menache et al. (2002) describe the problem of going from a start state, s, to a goal state, g, as a Max-Flow problem (Ahuja et al., 1993): the agent should maximize the accumulation (or flow) of probability along paths that originate in s and terminate in g. But, the problem of maximizing the flow in a graph is equivalent to the Min-Cut problem (Ford and Fulkerson, 1962), which requires identifying the lowest probability edges that can be removed to completely separate the source state, s, from the goal state, g. Off-the-shelf algorithms can be used to identify these min-cuts, which are interpreted as bottleneck states, and used as a target for new subgoal options (Kazemitabar and Beigy, 2009).

Specifically, the Q-Cut algorithm (Menache et al., 2002) finds such bottlenecks by solving the Min-Cut problem. In Min-Cut, the nodes of the graph are divided into disjoint sets, U and V ($U \cup V = \mathbb{S}$ and $U \cap V = \varnothing$), such that the source state belongs to the first set, $s \in U$, and the goal state belongs to the second set, $g \in V \setminus U$. The cut-value between them

is defined as the sum of probabilities along the edges that connect the two subsets:

$$\text{Cut}(U, V \setminus U) = \sum_{(i,j)\in E: i\in U, j\in V\setminus U} p(j|i,a). \tag{16}$$

Additionally, the *min-cut* is the solution to the following optimization problem, which searches for the edges that separate source s and goal g while minimizing the sum of probabilities along the cut edges:

$$\text{MinCut}(G) = \{(i,j) \in E : i \in U^*, j \in V \setminus U^*\}, \tag{17}$$

$$\text{where } U^* = \arg\min_{U\subset\mathcal{S}} \text{Cut}(U, V \setminus U). \tag{18}$$

Although there are exponentially many valid cuts, the Min-Cut problem can be solved in polynomial time (Ford and Fulkerson, 1962). Finally Menache et al. (2002) define the bottlenecks as the destination nodes of the min-cut edges: $B = \{j \mid (i,j) \in \text{Min-Cut}(G)\}$.

A drawback of Q-cut is that the entire MDP must be described with a global graph, which is not scalable. To address this shortcoming, L-cut (Şimşek et al., 2005) constructs "local graphs" using states visited in an episode. Instead of searching for individual states, Mannor et al. (2004) suggest identifying clusters of states and then connecting them using options; this approach has recently been extended using more sophisticated clustering techniques (Metzen, 2012; Srinivas et al., 2016; Campos et al., 2020; Bacon, 2013). Notably, Evans and Şimşek (2023)'s use of graph modularity (Newman and Girvan, 2004) as the metric for clustering allows them to efficiently learn multi-level hierarchies, where each level operates at a different timescale.

Graph Centrality

In graph theory, centrality measures the importance of each node within a graph. The search for useful subgoals in an MDP can be viewed as being analogous to identifying central nodes in a graph. Centrality measures are theoretically well-understood, and several efficient algorithms exist for computing them in large graphs, so it is attractive to use these methods for option discovery. Although many different graph centrality measures exist, Şimşek and Barto (2008) advocate for *betweenness* centrality because of its ability to find bottlenecks in large graphs. Betweenness quantifies how important a node is in a network by counting how many times it appears on the shortest path between other nodes (Şimşek and Barto, 2008). Specifically, the betweenness score $b(v)$ for a vertex (or equivalently, a state) is given as:

$$b(v) = \sum_{s\neq t\neq v} \frac{\sigma_{st}(v)}{\sigma_{st}} w_{st}, \tag{19}$$

where σ_{st} is the number of shortest paths from state s to state t, $\sigma_{st}(v)$ is the number of those paths that pass through state v, and w_{st} is the weight assigned to paths from vertex s to vertex t. The ratio in Equation 19 is the fraction of all-pairs shortest paths in the state transition graph that go through vertex v. When w_{st} is the same for all pairs of nodes, then Equation 19 is the betweenness centrality measure on graphs. To tailor this centrality measure to MDPs, w_{st} is set to the expected reward while going from state $s \to t$.

4.1.1 Benefits and Opportunities

Having introduced the major approaches for identifying bottlenecks, we now discuss how the resulting algorithms contribute to the aforementioned benefits of HRL.

Exploration. If an agent can easily reach the bottleneck states in an environment, it can perform more effective exploration. This is because states that were once hard to reach become more accessible, even under a random policy (these states are often referred to as *access states*). For example, picking up a key makes it easy for the player of a video game to visit previously unseen rooms. When this bottleneck discovery is done in an incremental fashion, as in L-Cuts (Şimşek et al., 2005), the agent expands the frontier of its experiences in the environment.

Credit Assignment. Methods like that of McGovern and Barto (2001), and Şimşek and Barto (2008) require the agent to solve the problem several times before option discovery can even begin; in such cases, exploration is clearly not the main benefit of discovering options. However, once the agent identifies bottlenecks, it can perform rapid credit assignment. This is primarily because of three reasons: (a) rather than progressing step-by-step, value can propagate in large, multi-step "jumps" from the states in which option execution terminates to the states from where it initiates (Sutton et al., 1999b), (b) value from rewarding events only needs to propagate along trajectories that pass through the bottleneck, greatly reducing the state-action pairs whose values need to be updated, and (c) in long-horizon problems, the difference in value between different actions—the *action-gap* (Bellemare et al., 2016a)—tends to approach zero (Lehnert et al., 2018), making it impossible to learn an accurate action-value function; in such cases Lehnert et al. (2018) suggest partitioning the state-space along bottlenecks, so that each partition can be treated as a short-horizon problem, inducing a larger action-gap, and hence, easier credit assignment.

Transfer. Bottlenecks are useful for transfer because they are largely task agnostic—they focus on capturing structure in the transition function, and so the same bottlenecks are often useful for a family of tasks or reward functions. For example, in the *Two Rooms* task, the ability to quickly and reliably reach the doorway enables the agent to reach the goal, no matter where it is placed in the second room (McGovern and Barto, 2001).

Opportunities for Research.

- **Scalability.** Most methods for finding bottlenecks apply to discrete graphs; as a result, these techniques often struggle to scale to large, continuous MDPs. Notable exceptions include spectral methods (discussed in Section 4.2), which compute continuous properties of the underlying graph, without explicitly representing the graph in the first place.

- **Performance guarantees.** It is generally not well understood how the proxy objective of targeting bottlenecks contributes to high-level objectives of the agent, such as reward maximization or faster planning. Methods outlined in Section 4.6 attempt to answer this question in general for all option discovery methods, but given the number of option discovery algorithms related to bottlenecks, it would be useful to find if there is a high-level objective of the agent that is maximized (at least to some degree) while optimizing for this proxy objective.

4.2 Spectral Methods

Many option discovery methods are based on the idea of leveraging the state space's topology, be it to discover options that identify key states that connect different partitions of the environment (Şimşek et al., 2005), that connect states that are far from each other when looking at the diffusion properties of the environment (e.g., Machado and Bowling, 2016; Machado et al., 2017), or that easily allow the agent to traverse the environment in a reusable manner (Liu et al., 2017; Klissarov and Machado, 2023). They are termed spectral methods because, through the eigenvectors of a matrix representation of the environment, they extract information from the state space, such as connectivity or diffusion.

The different algorithms in this group leverage the different ways of representing the environment as a matrix and the different types of information one can extract from such matrices. Originally, heavily inspired by results from the graph theory literature, these methods were based on the graph Laplacian and its eigenfunctions,[3] which can approximate any function on the graph (Chung, 1997). The normalized graph Laplacian, $\mathcal{L}$, for example, is defined as

$$\mathcal{L} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-1/2}, \tag{20}$$

where $\mathbf{A}$ is the graph's adjacency matrix obtained by modelling each state in the environment as a node. The adjacency matrix reflects the degree of connectivity between two states. The matrix $\mathbf{D}$ is a diagonal matrix whose entries are the row sums of $\mathbf{A}$. In the reinforcement learning literature, the eigenvectors of the graph Laplacian are also known as proto-value functions (PVFs; Mahadevan, 2005; Mahadevan and Maggioni, 2007).

Importantly, when considering the eigendecomposition, $\mathcal{L}\mathbf{e} = \lambda\mathbf{e}$, the eigenvector of the graph Laplacian associated to the second smallest eigenvalue captures the number of connected components in a graph (Shi and Malik, 2000), allowing one to easily identify bottleneck states (Şimşek et al., 2005), as discussed in Section 4.1. The eigenvectors of the graph Laplacian, in general, capture different time scales of diffusion, which can be used to discover options that promote exploration, such as eigenoptions (Machado et al., 2017, 2018; Machado, 2019), covering options (Jinnai et al., 2019b, 2020), and covering eigenoptions (Machado et al., 2023).

Eigenoptions, for example, are defined such that each option, o_i, is associated with the corresponding eigenvector, $\mathbf{e}_i$, of the graph Laplacian. Their policy is defined as the policy that maximizes the intrinsic reward that incentivizes the agent to navigate alongside the direction pointed by $\mathbf{e}_i$, which, in the linear function approximation (and tabular) case, is formalized as

$$r^{\mathbf{e}_i}(s, s') = \mathbf{e}_i^\top \big(\boldsymbol{\phi}(s') - \boldsymbol{\phi}(s)\big), \tag{21}$$

where $\boldsymbol{\phi}(s)$ denotes the feature representation of state s. Originally, an option o_i was defined to terminate in state s if $q_\pi^{\mathbf{e}_i}(s, a) \leq 0$ for all $a \in \mathcal{A}$, where $q_\pi^{\mathbf{e}_i}$ is defined w.r.t. $r^{\mathbf{e}_i}(\cdot, \cdot)$. All other states in the environment were defined to be in the initiation set.

Naturally, explicitly representing an environment through its underlying graph is not scalable. Existing methods leverage approximations of the eigenfunctions of the graph Laplacian that can be obtained through neural networks trained with stochastic gradient descent (Wu et al., 2019; Wang et al., 2021; Gomez et al., 2023). The underlying idea

3. Eigenfunctions can be seen as a generalization of eigenvectors to continuous state spaces.

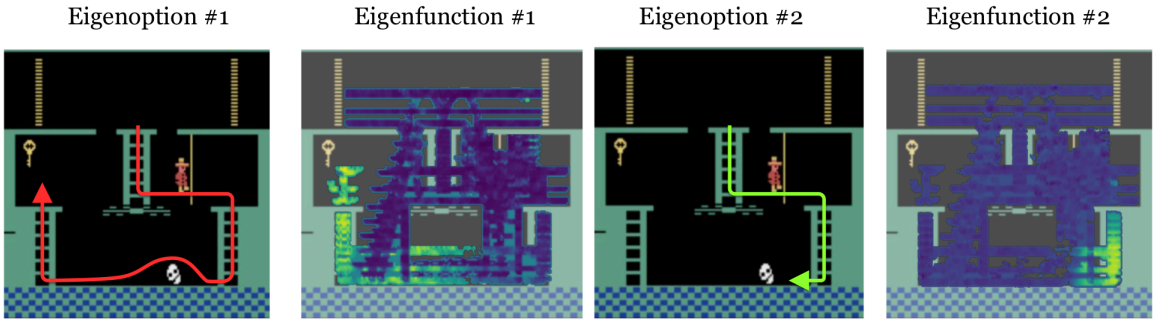


Figure 7: Visualization of the first and second eigenfunctions on Montezuma's Revenge, an Atari 2600 game, discovered by the algorithm proposed by Klissarov and Machado (2023). The arrows depict what the eigenoption induced by these eigenfunctions could end up being.

behind these methods is to learn a representation that captures the properties of the approximated eigenvectors such that observations that happen "close in time" are close in representation space and that different eigenfunctions are indeed orthogonal to each other. The current state-of-the-art method for doing so is called the *augmented Lagrangian Laplacian objective* (ALLO; Gomez et al., 2023). It consists of the following max-min objective for approximating d eigenfunctions:

$$\max_{\boldsymbol{\beta}} \min_{\mathbf{u}\in\mathbb{R}^{d|\mathcal{S}|}} \sum_{i=1}^{d} \langle \mathbf{u}_i, \mathcal{L}\mathbf{u}_i \rangle + \sum_{j=1}^{d}\sum_{k=1}^{j} \omega_{jk}\big(\langle \mathbf{u}_j, [\![\mathbf{u}_k]\!] \rangle - \delta_{jk}\big) + b\sum_{j=1}^{d}\sum_{k=1}^{j} \big(\langle \mathbf{u}_j, [\![\mathbf{u}_k]\!] \rangle - \delta_{jk}\big)^2, \tag{22}$$

where $\mathcal{L}$ denotes the graph Laplacian again, $[\![\cdot]\!]$ the stop gradient operator, δ_{jk} the Kronecker delta, b is a scalar hyperparameter, and $\boldsymbol{\omega} = [\omega_{1,1}, \omega_{2,1}, \omega_{2,2}, \cdots, \omega_{d,1}, \cdots, \omega_{d,d}] \in \mathbb{R}^{d(d+1)/2}$ is a vector containing all of the dual variables of the objective. Note that the optimal dual variables, $\boldsymbol{\omega}^*$, are proportional to the smallest eigenvalues of $\mathcal{L}$. These approximations have now been used to learn options that are effective in various domains, including continuous control tasks (Jinnai et al., 2020), 3D navigation tasks, and Atari 2600 games (Klissarov and Machado, 2023). An issue these methods had to circumvent was that most of these approximation objectives assume the ability to sample uniformly the entire state space. This is currently addressed by iteratively increasing the region covered by the agent (e.g., Machado et al., 2023); some methods even do so explicitly in the objective they minimize (Erraqabi et al., 2022).

The process to compute the intrinsic reward maximized by the option is slightly different when using neural network estimates of the eigenfunctions of the graph Laplacian. Instead of first computing the eigenvectors, one usually directly estimates the components of the eigenfunction associated with a particular state, s. Formally,

$$r^{f_{e_i}}(s, s') = f_{e_i}(s') - f_{e_i}(s), \tag{23}$$

where we used $f_{e_i}(s)$ to denote the value of i-th eigenfunction of the graph Laplacian associated with state s. In this setting, stochastic option terminations are more common in practice due to the difficulties generalization introduces to accurately estimating action-value functions without interference (e.g., Klissarov and Machado, 2023).

Many other mathematical objects are somewhat equivalent to the eigenvectors of the graph Laplacian and have also been used for option discovery. Slow Feature Analysis (SFA; Wiskott and Sejnowski, 2002; Sprekeler, 2011), for example, are a key component of Continual Curiosity-driven Skill Acquisition (CCSA; Kompella et al., 2017). The eigenvectors of the successor representation (SR; Dayan, 1993) have also been shown to be equivalent to the eigenvectors of the graph Laplacian (Machado et al., 2018).

The equivalence between the eigenvectors of the SR and of the graph Laplacian is particularly important due to the predictive nature of the SR and the ease with which one can learn it incrementally. In fact, the SR now has a quite prominent role in the option literature, being used in the discovery of options for both faster credit assignment (Ramesh et al., 2019) and exploration (Machado et al., 2018; Machado, 2019).

The successor representation is defined as

$$\boldsymbol{\Psi}_\pi(s, s') = \mathbb{E}_{\pi,p}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{\{S_t=s'\}} \mid S_0 = s\right], \tag{24}$$

where $\mathbb{1}$ denotes the indicator function. The SR was originally introduced through an intuition that is very similar to the one outlined above: one should capture the environment's dynamics by assigning similar values to temporally close states, thus creating a representation of the underlying structure. It can also be estimated with temporal-difference learning methods (Sutton, 1988), which, as we mentioned above, allows us to learn it incrementally:

$$\Psi(S_t, j) \leftarrow \Psi(S_t, j) + \eta\Big(\mathbb{1}_{\{S_t=j\}} + \gamma\Psi(S_{t+1}, j) - \Psi(S_t, j)\Big), \tag{25}$$

where we used $\Psi(\cdot, \cdot)$ to denote a sample-based approximation of $\boldsymbol{\Psi}_\pi$.

Importantly, beyond the discovery methods mentioned above; as a representation, which was its original purpose, the SR can also be used to combine options without additional learning (Barreto et al., 2019a), and recent results in neuroscience and cognitive sciences suggest the SR can model activations in the hippocampus (Stachenfeld et al., 2017) and explain some human behaviour (Momennejad et al., 2017). These results have led Machado et al. (2023) to propose that the successor representation should be seen as the "natural substrate for the discovery and use of temporal abstractions" in reinforcement learning.

In terms of scalability, again, there have been many proposals on how to scale the SR to function approximation settings ranging from specific neural network architectures (Kulkarni et al., 2016; Machado et al., 2018; Chua et al., 2024) to ideas such as successor features (Barreto et al., 2017) and successor measures (Touati and Ollivier, 2021; Farebrother et al., 2023). Successor features, for example, can be seen as a projection of the SR onto the space realizable by the representation, $\boldsymbol{\phi}$. In matrix form, if we use $\boldsymbol{\Phi} \in \mathbb{R}^{|\mathcal{S}|\times d}$ to denote the matrix encoding the d-dimensional feature representation of each state, successor features are defined as $\boldsymbol{\Psi}_\pi = \sum_{t=0}^{\infty}(\gamma\mathbf{P}_\pi)^t\boldsymbol{\Phi} = (I - \gamma\mathbf{P}_\pi)^{-1}\boldsymbol{\Phi}$.

4.2.1 Benefits and Opportunities

Exploration. The eigenoptions line of work (Machado et al., 2017) has popularized the idea of leveraging temporal abstraction for exploration. Eigenoptions can significantly decrease

the diffusion time[4] in an environment, and this afforded exploration can lead to faster learning. Machado et al. (2018) further extends previous work to the function approximation case by estimating the successor representation and then performing a singular value decomposition on it. Jinnai et al. (2019b) introduce covering options, arguing that rather than constructing an option for every eigenvector of the graph Laplacian, a single option constructed based on the second eigenvector is sufficient. This is because that single option minimizes the cover time of the underlying MDP, which loosely refers to how long it takes for a random high-level policy to visit all states. Leveraging direct approximations of the eigenfunctions of the graph Laplacian, Jinnai et al. (2020) extended covering options to the function approximation case, and Klissarov and Machado (2023) did the same for covering eigenoptions (Machado et al., 2023), demonstrating strong exploration properties in a variety of reinforcement learning problems.

Transferability. Options are often thought to be important in lifelong/continual learning settings where skills can be reused in an ever-changing world. The benefit of Laplacian-based options in such settings has been demonstrated both in simpler tabular problems in which the goal location changes regularly (Liu et al., 2017) and in more complex, high-dimensional settings in which not only the goal location would change but also the topology of the environment (Klissarov and Machado, 2023).

Opportunities for Research.

- **Improving Representations.** Machado et al. (2023) have proposed the perspective that spectral methods consist of a phase in which a representation is first learned (e.g., PVFs, SR), followed by a phase in which options are then derived from such a representation. This process can even be done in a cycle, which Machado et al. (2023) called Representation-driven Option Discovery (ROD) cycle. Thus, better representation learning methods are an exciting research frontier for this line of work in which the learned representation informs the option discovery process. This can be investigated from the SR perspective (e.g., Touati and Ollivier, 2021; Carvalho et al., 2023; Farebrother et al., 2023), or from the perspective of directly estimating the spectral decomposition of the SR (e.g., Pfau et al., 2018; Wang et al., 2021, 2022; Gomez et al., 2023), including non-symmetric settings (Wang et al., 2023b).

- **Planning.** Another promising line of work involves further exploring the recent success of Laplacian-based methods in planning and credit assignment in general, as these options are often used in a reward-agnostic way (e.g., Sutton et al., 2023). Validating these results beyond the tabular case and extending existing results to partially-observable settings are also intriguing lines of work.

- **Reward-Aware Representations.** The representations discussed in this section rely on the topology of the environment without considering the underlying reward function. There is an interesting question of whether one should define proximity not only in terms of when observations take place but also in terms of the reward associated with them. Interestingly, the linear MDP formalism (Todorov, 2006, 2009b) gives rise to representations akin to the SR but that are reward-aware. In this context, Tse et al.

4. The diffusion time encodes the expected number of "decisions" required to navigate between two states randomly chosen in an environment (Machado et al., 2017).

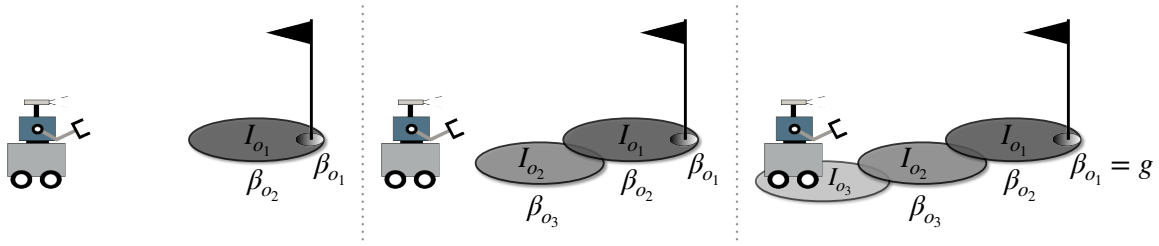


Figure 8: **Sequentially Composable Options.** The skill chaining algorithm incrementally learns options backwards from the goal, such that the subgoal of each option is the initiation region of another option. First the agent finds the states from which it can reliably reach the goal *(left)*, then it finds the states from where it can reach the first region *(middle)*, and so on, until there is a high probability of success from the environment's start state *(right)*.

(2025) has shown that options derived from the eigenvectors of such a reward-aware representation, termed the default representation (Piray and Daw, 2021), exhibit qualitatively different exploratory behaviour when faced with regions of negative reward in the state space.

4.3 Sequentially Composable Options

Options are said to be *sequentially executable* when each option terminates in a region where another option can successfully achieve its own subgoal. Sequentially composable options are more useful for high-level planning (Konidaris et al., 2018) and even result in highly robust solutions (Tedrake et al., 2010). While most methods attempt to sequentially compose discovered options *post hoc*, some methods explicitly incorporate sequential composition into the option discovery objective. A prominent family of such methods is that of Skill Chaining (Konidaris and Barto, 2009; Bagaria and Konidaris, 2020).

Figure 8 illustrates the skill chaining algorithm. Given a target region of states $g \subset \mathcal{S}$ (for example, the task goal) (shown as a flag in Figure 8), skill chaining discovers subgoal options that can be sequenced together so that each option execution roughly brings the agent closer to g. This is done by learning options backward from the goal: first, the agent learns option o_1 such that $\beta(o_1) = g$; this entails learning two functions: (a) the option policy $\pi(a|s, o_1)$, which aims to maximize the subgoal reward $r^{o_1}(s) = \beta(o_1)$, and (b) the initiation function $\mathcal{I}(o_1)$, which is defined to be the states from which $\pi(\cdot|s, o_1)$ can reliably reach g. Shortly after, the agent creates another option o_2 so that its subgoal is the initiation of the previous option—this is because the agent can reach the goal with high probability from states inside the first option's initiation region. This process continues until the start state, $s_0 \sim \rho_0$, of the MDP is inside the initiation region of some option. This is because when the initiation probability is high at the start state, the agent can simply execute its learned options to achieve its goal g. Skill composability is explicitly enforced by setting the termination region of each option $\beta(o_i)$ to be the states in which another option has a high initiation probability, i.e., $\mathcal{I}(o_{i-1})$ is greater than some pre-specified threshold $c \in [0, 1]$.

In skill chaining, the initiation set of an option has special meaning: it represents the states from which option execution is likely to achieve its subgoal. Learning the initiation

function is usually framed as a binary classification problem: states along successful option trajectories (those that achieve the option's subgoal) are considered as positive examples $\mathbf{s}^+ = \{s_1^+, \cdots, s_n^+\}$ and states along unsuccessful trajectories are considered as negative examples $\mathbf{s}^- = \{s_1^-, \cdots, s_m^-\}$. Then, a probabilistic classifier (with parameters χ) is fit on these training examples using the binary cross-entropy loss. Now, when a new state s is encountered during learning, $\mathcal{I}(s, o)$ represents the probability that the agent can reach option o's subgoal $\beta(o)$ in a single execution of $\pi(\cdot|s, o)$. While this classification approach is simple to implement, some of its drawbacks include: (a) the classifier struggles to adapt to changing option policies, and (b) the agent must wait until the end of option execution to update its initiation function. To address this issue, Bagaria et al. (2023) frame the initiation function as a general value function (Sutton et al., 2011): the agent uses each experience tuple $(s, a, \beta(o), s')$ to update its prediction of whether an option execution will achieve its subgoal; this is done using the following temporal difference (TD) error and stochastic gradient descent update rule:

$$\delta_{\mathcal{I}}(s, o) = \beta(s', o) + \mathcal{I}(s', o) - \mathcal{I}(s, o), \tag{26}$$

$$\chi = \chi - \alpha \delta_{\mathcal{I}} \nabla_{\chi} \mathcal{I}(s, o), \tag{27}$$

where $\alpha \in \mathbb{R}^+$ is a step size parameter and χ are the initiation function parameters. However, at a given state s, an option's initiation probability $\mathcal{I}_{\chi}(s, o)$ can be low either because the option policy is unlikely to successfully reach its subgoal from state s or because the agent does not have enough data to confidently estimate $\mathcal{I}_{\chi}(s, o)$. As a result, the skill chaining agent additionally estimates its uncertainty $\mathcal{U}(s, o)$ about its initiation function's predictions: when deciding whether an option is executable from a state s, it is optimistic with respect to that uncertainty, but when targeting another option's initiation region, it is pessimistic with respect to it (Bagaria et al., 2021a).

Algorithm 1 summarizes the skill chaining algorithm. First, the high-level policy picks an option with the aim of maximizing extrinsic reward, while attending to the initiation probability of each option. Actions are selected using the chosen option's policy, which is rewarded for achieving its own subgoal. Transitions encountered during option execution are used to update the low-level option policy, the high-level policy, and the option's initiation function. When the agent is confident that there is no option that could reach its subgoal from the start states of the environment, it creates a new option and adds it to the skill chain. This new option's subgoal region is the states where the previous option in the skill chain has a high initiation probability, thereby enforcing sequential composability.

4.3.1 Benefits and Opportunities

Planning. Each option execution drives the agent to a small, predictable region of the state-space. Since those states are constructed to be inside the initiation region of another option, they can be sequentially composed. In practice, each option's initiation and termination region is parameterized using probabilistic classifiers, so there is a *probability* that two options can be executed in sequence, which eventually permits computation of the probabilistic feasibility of entire plans. Bagaria et al. (2023) used graph-search to find recursively optimal solutions and Bagaria et al. (2021a) provided a dynamic programming algorithm to approximate hierarchically optimal ways of planning with subgoal options.

Algorithm 1 Skill Chaining Algorithm

Initialize:
Initialize first option o_1's subgoal as task goal: $\beta(o_1) = g$.
Initialize o_1's initiation function $\mathcal{I}(s, o_1)$, uncertainty $\mathcal{U}(s, o_1)$, and policy $\pi_\theta(\cdot|s, o_1)$.
Initialize the agent's option set using the first option: $\mathcal{O} = \{o_1\}$.
Hyperparameters:
Option horizon H_o and initiation function thresholds $c_1, c_2 \in [0, 1]$ for each option.
while True **do**
    Sample an option o from the following distribution:

$$\frac{\mu(o|s)\mathcal{I}^+(s, o)}{\sum_{o' \in \mathcal{O}} \mu(o'|s)\mathcal{I}^+(s, o')}, \forall o \in \mathcal{O},$$

    where $\mathcal{I}^+(s, o') = \texttt{clip}(\mathcal{I}(s, o') + \mathcal{U}(s, o), 0, 1)$.
    while option o does not terminate **do**
        Sample an action $a \sim \pi(\cdot \mid s, o)$.
        Execute the action to get reward r and next state s'.
        Update the option policy $\pi(\cdot|s, o)$ using reward $r^o(s, a, s') = \beta(s', o)$.
        Update the high-level policy using extrinsic reward r.
        Update the option's initiation function using generalized TD-Error:

$$\delta_\mathcal{I}(s, o) = \beta(s', o) + \mathcal{I}(s', o) - \mathcal{I}(s, o).$$

    end while
    if $\mathbb{E}_{s_0 \sim \rho_0}[\mathcal{I}(s_0, o)] < c_1$ & $\mathbb{E}_{s_0 \sim \rho_0}[\mathcal{U}(s_0, o)] < c_2, \forall o \in \mathcal{O}$ **then**
        Extract the last option in the chain, ω.
        Create new option o' such that $\beta(s, o') = \mathbb{1}\Big(\mathcal{I}(s, \omega) > c\Big)$.
        Add the new option o' to the agent's option set $\mathcal{O}$.
    end if
end while

Credit Assignment. Skill chaining has demonstrated more sample-efficient credit assignment in goal-reaching tasks than non-hierarchical RL, which can be attributed to the following reasons. (1) *Jumpy transitions*: Skill chaining methods usually use the entire T-step option transition $(s_t, o, \sum r_{t:t+T}, s_{t+T})$ to update the high-level policy $\mu(o|s)$. Much like n-step returns and TD(λ) in non-hierarchical RL, this has the effect of rapidly propagating credit among state-action pairs. (2) *Focused next-state distribution*: not only does each option execute for multiple timesteps, but it also guides the agent to states that are closer to the goal. In other words, options in the skill chain modify the agent's state distribution to make states closer to the goal more likely. Since these states are usually the ones with non-zero values, bootstrapping-based value learning (e.g., TD) progresses more rapidly.

Exploration. Since skill discovery proceeds backward from the goal, the algorithm requires either an exploration policy or a set of demonstration trajectories (Konidaris et al., 2010; Kang and Oh, 2022) that achieve the task goal. This advocates for a view of skill chaining

as producing options that are good for exploitation, which can be combined with options that are good for exploration. Deep skill graphs (DSG) (Bagaria et al., 2021b) overcome this limitation: the agent finds intrinsically motivating states and learns skill chains that connect them to each other; the resulting chains form a graph abstraction of the environment, which is useful for planning. Furthermore, the graph building process has a *Voronoi bias* (LaValle, 1998; Lindemann and LaValle, 2004), meaning that it tends to grow towards parts of the state-space where the agent has the least experience.

Opportunities for Research.

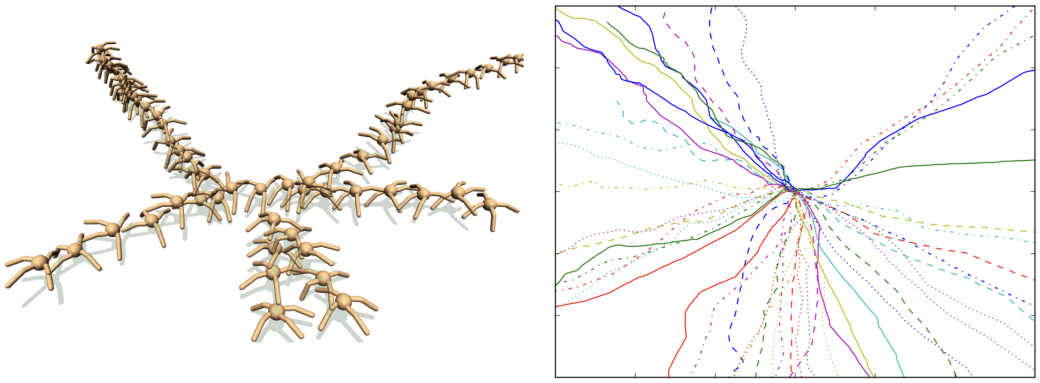
- **Goal-reaching options.** To learn the initiation set of each option in the chain, its subgoal must be described using a binary function: either the subgoal is achieved in the current state, or it is not. Such a subgoal description is not universal, as it cannot be used to describe continuing tasks like maintaining a constant velocity or repeating periodic motions. If the initiation cumulant (Bagaria et al., 2023) can be formulated for general reward functions, then skill chaining can be applied to non-goal-reaching tasks as well.

- **Controlling all state variables at the same time.** If we think of states being composed of different state variables (a property known as *factoredness* Boutilier et al. 2000), then skill chaining drives the value of all variables to a certain range of values. In more complex environments, it may be unnecessary, or even impossible, to control all state variables at the same time. Future work could create a version of skill chaining that leverages the factoredness of the state-space and only controls a subset of all the factors at any given time.

Additional connections to control theory and motion planning. Lozano-Perez et al. (1984), Mason (1985), and Burridge et al. (1999) popularized the view of policies as *funnels*: these policies drive a large set of ordinary states to a small set of desired states. Policies can be sequentially composed to reach some target set of states by placing the end (narrow part) of each funnel inside the beginning (broad part) of some other funnel. Tedrake et al. (2010); Ames and Konidaris (2019) provided a way to compute these initiation regions for complex, dynamical systems using convex optimization and built robust controllers for fixed-wing UAVs (Tedrake et al., 2010). Later, Konidaris and Barto (2009) extended this idea to model-free RL. Bagaria and Konidaris (2020) then upgraded the skill-chaining algorithm with deep learning so that it could be applied to higher-dimensional systems. Variants of deep skill chaining have been used in robotic surgery (Huang et al., 2023), manipulation (Lee et al., 2021; Vats et al., 2023), multi-agent RL (Xie et al., 2022), and task and motion planning (Mishra et al., 2023).

4.4 Empowerment Maximization

Empowerment-based methods discover diverse skills by maximizing an agent's control over its environment. At its core, empowerment quantifies how much influence an agent has over its future observations—an agent is more empowered when it can reliably cause a wider variety of outcomes (Klyubin et al., 2005; Salge et al., 2014). For example, having access to a car empowers you to reach many different locations; learning to swim empowers you to



(*a*) (*b*)

Figure 9: Empowerment-based skill-discovery methods learn skills that generate trajectories that are maximally different from one another, with the constraint that, having observed a trajectory, it should be clear which skill generated it. (*a*) Trajectories generated by 6 distinct skills in MuJoCo Ant; (*b*) (x, y) location of the center of mass of the Ant plotted after executing skills learned by the DADS algorithm. Figure from Sharma et al. (2020b), used with permission.

survive in water. Empowerment can also be seen as a way to maximize social influence in multi-agent settings (Jaques et al., 2019), or to seek agreement between future states and the agent's internal representations (Hafner et al., 2020).

Formally, empowerment is defined as the mutual information between an agent's actions and its future states. Mutual information $I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$ measures how much information one random variable provides about another, equaling zero when the variables are independent and increasing as they become more statistically dependent.

Now, consider an agent that executes a sequence of n actions $\mathbf{a} = (a_t, a_{t+1}, \ldots, a_{t+n-1})$ starting from state s_t, resulting in state s_{t+n}. The n-step empowerment at state s_t is:

$$\mathcal{E}_n(s_t) = \max_{p(\mathbf{a})} I(\mathbf{a}; s_{t+n}|s_t), \tag{28}$$

where $p(\mathbf{a})$ is the probability distribution over action sequences that the agent can choose. This captures the maximum amount of information that action sequences can provide about future states, optimized over all possible action distributions $p(\mathbf{a})$. Expanding the mutual information reveals its intuitive meaning:

$$I(\mathbf{a}; s_{t+n}|s_t) = \mathcal{H}(s_{t+n}|s_t) - \mathcal{H}(s_{t+n}|s_t, \mathbf{a}), \tag{29}$$

where the first term represents uncertainty about future states given only the current state, while the second represents remaining uncertainty after choosing actions. Empowerment measures how much this uncertainty can be reduced through deliberate action choice.

As the mutual information is intractable, Mohamed and Rezende (2015) propose to estimate it through variational inference. Specifically, the authors estimate $p(\mathbf{a}|s_{t+n})$ using the variational approximation $q_\phi(\mathbf{a}|s_{t+n})$ and leverage the non-negativity of the KL divergence

to obtain[5]:

$$I(\mathbf{a}; s_{t+n}|s_t) = \mathcal{H}(\mathbf{a}|s_t) - \mathcal{H}(\mathbf{a}|s_t, s_{t+n}) \tag{30}$$
$$= \mathcal{H}(\mathbf{a}|s_t) + \mathbb{E}[\log p(\mathbf{a}|s_t, s_{t+n})] \tag{31}$$
$$\geq \mathcal{H}(\mathbf{a}|s_t) + \mathbb{E}[\log q_\phi(\mathbf{a}|s_t, s_{t+n})] \quad \text{(Variational Bound)} \tag{32}$$

This variational bound (Equation 32) provides a practical way to compute empowerment in high-dimensional continuous spaces using neural networks (with parameters ϕ). However, this objective finds open-loop action sequences $\mathbf{a}$, and we want to discover *skills* (closed-loop policies). Gregor et al. (2017) addressed this concern by introducing Variational Intrinsic Control (VIC), which replaces fixed action sequences with parameterized skills $\pi_\theta(a|s, z)$ conditioned on skill variables z. VIC maximizes the mutual information between skills and final states reached from skill execution:

$$J_{\text{VIC}} = I(z; s_{t+n}|s_t), \tag{33}$$

where s_t is the initial state and s_{t+n} is the final state after executing skill z for n timesteps. We can expand this as:

$$I(z; s_{t+n}|s_t) = \mathcal{H}(z|s_t) - \mathcal{H}(z|s_{t+n}, s_t). \tag{34}$$

Similar to Mohamed and Rezende (2015), VIC uses the variational lower bound:

$$I(z; s_{t+n}|s_t) \geq \mathcal{H}(z|s_t) + \mathbb{E}[\log q_\phi(z|s_{t+n}, s_t)]. \tag{35}$$

This variational lower bound can be optimized by training two neural networks: a policy $\pi_\theta(a|s, z)$ that executes skills, and a discriminator $q_\phi(z|s_{t+n}, s_t)$ that predicts which skill was used based on the final state.

Eysenbach et al. (2019) simplified VIC's approach in their method *Diversity is All You Need* (DIAYN). While VIC maximizes mutual information between skills and final states, DIAYN instead focuses on making skills distinguishable from the states they visit throughout execution. DIAYN builds on maximum entropy reinforcement learning, which augments the standard RL objective with an entropy bonus $\mathcal{H}(A|S)$ to encourage exploration. DIAYN learns skills by maximizing:

$$J_{\text{DIAYN}} = I(s; z) + \mathcal{H}(a|s) - I(a; z|s), \tag{36}$$

which has an intuitive interpretation: skills should be distinguishable from the states they visit ($I(s; z)$), actions should be diverse ($\mathcal{H}(a|s)$), but skills should be consistent in their behaviour ($-I(a; z|s)$). This objective can further be simplified by expanding the mutual information in terms of the conditional entropies, and then applying the variational approximation similar to Mohamed and Rezende (2015):

$$J_{\text{DIAYN}} = \Big(\mathcal{H}(z) - \mathcal{H}(z|s)\Big) + \mathcal{H}(a|s) - \Big(\mathcal{H}(a|s) - \mathcal{H}(a|s, z)\Big) \tag{37}$$
$$= \mathcal{H}(z) - \mathcal{H}(z|s) + \mathcal{H}(a|s, z) \tag{38}$$
$$= \mathcal{H}(a|s, z) + \mathbb{E}[\log p(z|s)] - \mathbb{E}[\log p(z)] \tag{39}$$
$$\geq \mathcal{H}(a|s, z) + \mathbb{E}[\log q_\phi(z|s)] - \mathbb{E}[\log p(z)]. \tag{40}$$

5. Note that we are departing from our usual notation to denote p and q using the conventional notation in variational inference.

The final step implies the use of a discriminator $q_\phi(z|s)$ to variationally approximate $p(z|s)$; the result is an intra-skill pseudo reward function for each skill z:

$$r^z(s) = \log q_\phi(z|s) - \log p(z), \tag{41}$$

assuming that the $\mathcal{H}(a|s,z)$ term is maximized using an maximimum entropy RL formulation (Ziebart et al., 2008). This leads to a practical algorithm: sample skill $z \sim p(z)$, execute policy $\pi_\theta(a|s,z)$, train discriminator $q_\phi(z|s)$ to predict skills from states, and update the policy using pseudo-reward in Equation 41.

While DIAYN successfully learns diverse behaviours, Sharma et al. (2020b) observed that it can discover skills with unpredictable effects, making them difficult to sequentially compose downstream. Their method, *Dynamics-Aware Discovery of Skills* (DADS), addresses this by explicitly encouraging predictable skill dynamics while maintaining diversity. Their formulation captures two desirable properties simultaneously: different skills should lead to different future states (diversity), and given the current state and skill, the future state should be predictable. Expanding the mutual information:

$$I(z; s_{t+n}|s_t) = \mathcal{H}(s_{t+n}|s_t) - \mathcal{H}(s_{t+n}|s_t, z) = \mathbb{E}\left[\log \frac{p(s_{t+n}|s_t, z)}{p(s_{t+n}|s_t)}\right]. \tag{42}$$

DADS uses variational approximation with two learned models: a skill dynamics model $q_\psi(s_{t+n}|s_t, z)$ and a marginal dynamics model $q_\xi(s_{t+n}|s_t)$. The skill reward function becomes

$$r^z(s_t, s_{t+n}) = \log q_\psi(s_{t+n}|s_t, z) - \log q_\xi(s_{t+n}|s_t), \tag{43}$$

encouraging skills that make state transitions more predictable than the marginal dynamics. Resulting skills have focused effects, and examples of learned skills are visualized in Figure 9. These methods share the key insight that useful skills should be distinguishable from their effects on the environment—whether through final states (VIC), visited states (DIAYN), or state transitions (DADS), each approach learns discriminators that identify which skill was executed from observations. This creates an intrinsic reward signal driving the discovery of diverse, meaningful behaviours without external supervision.

Let $\tau = (s_t, a_t, s_{t+1}, a_{t+1}, ..., s_{t+n})$ denote a trajectory of states and actions. DIAYN's update rule relies on the approximation that $\log(q_\phi(z|\tau)) = \sum_{i=0}^{n} \log(q_\phi(z|s_{t+i}))$, i.e., as a sum of per-timestep log probabilities along the trajectory generated by the skill z. Instead of the sum-based decomposition of the trajectory, which treats each transition as being independent from the others, VALOR approximates $q_\phi(z|\tau)$ using an LSTM architecture (Achiam et al., 2018). Strouse et al. (2022) noticed that the discriminator, q_ϕ is pessimistic in new states; to address this pessimism, their algorithm DISDAIN, augments skill learning with a novelty bonus. CIC (Laskin et al., 2022) improves the optimization of the mutual information objective using contrastive learning. Relative VIC (Baumli et al., 2021) departs from this view slightly by introducing a new term to the optimization: rather than requiring that trajectories be distinguishable from the observed trajectory, they additionally require that the trajectory *not* be distinguishable from the final state alone, thereby encouraging skills to cause characteristic *changes* in state, rather than taking the agent to different parts of the state-space alone.

4.4.1 Benefits and Opportunities

Exploration. Empowerment serves as an intrinsic motivation that encourages agents to seek states where they have the most control over future outcomes (Klyubin et al., 2005). Several recent empowerment methods seek to address the exploration problem in RL by explicitly optimizing for skill diversity (Eysenbach et al., 2019) and state-space coverage (Campos et al., 2020); by learning a diverse set of skills, agents not only explore effectively in a single-task setting (Massari et al., 2021), but also adapt quickly to new tasks, reducing sample complexity (Sharma et al., 2020b; Baumli et al., 2021; Hansen et al., 2020). One difficulty in obtaining better exploration with empowerment-based methods is that the learned skills tend to be localized, i.e., only cover a small area. Lipschitz-constrained Skill Discovery (LSD) (Park et al., 2022) replaces mutual information estimation with a Lipschitz-constrained objective, ensuring that learned skills correspond to large, meaningful state transitions rather than minor variations. Exploration is further enhanced by paying attention to the the parts of the environment that are within the agent's control (Park et al., 2023)—this is done by using a controllability-aware distance function that assigns higher values to harder-to-control state transitions, leading to more complex skill acquisition, such as object manipulation, without direct external supervision.

Credit Assignment. While the primary motivation of empowerment-driven techniques is that of exploration, recent methods seek to improve the process of learning policies over discovered skills. For example, Leibfried et al. (2019) derive Bellman operators that combine empowerment-, and reward-maximization; Sharma et al. (2020b,a) advocate for learning option models during the skill-discovery phase so that the learned options can be composed via a planner at test-time. In fact, the simplification of empowerment-based objectives to state-reaching (Pitis et al., 2020) can be viewed as a compromise between learning more expressive skills and creating stationary objectives that ease online policy learning for utilizing discovered skills.

Transfer. Most empowerment-based skill-discovery algorithms learn skills that are distinguishable via specific states encountered in sampled trajectories (for example, the last state of the trajectory). This approach leads to skills that are tied to specific states encountered during skill-learning; in other words, learned skills do not transfer to unseen, related parts of the state-space. Relative VIC is a promising approach for learning transferrable skills because it rewards skill policies for causing characteristic *changes* in state, rather than targeting specific states themselves. Some algorithms use successor features to enable transfer (Zahavy et al., 2021), but more research is needed on learning skills that simultaneously maximize empowerment and enable reuse across different portions of the state-space.

Opportunities for Research.

- **Optimization challenges.** Despite significant progress in online mutual information estimation, empowerment remains challenging to estimate and optimize (Achiam et al., 2018). To address this, Park et al. (2024c) introduce a Wasserstein variant of the mutual information objective, where the KL divergence in MI is replaced with the Wasserstein distance. Finding such ways to ease the estimation and optimization of the empowerment objective is a key area of current research.

- **Connections to causal learning.** Gopnik (2024) hypothesizes that if an agent learns an accurate causal model of the world, it will necessarily increase its empowerment, and, conversely, increasing empowerment will lead to a more accurate (albeit implicit) causal model of the world (Salge et al., 2014). This could enable model-based planning for complex, long-horizon problems (Kahneman, 2011), fully unleashing the power of HRL.
- **Possible signatures in human learning.** There is mounting evidence in developmental cognitive science that the drive to learn causal models of the world is behind many of the exploratory capabilities of children (Gopnik and Wellman, 2012). For example, Rovee-Collier and Gekoski (1979) show that infants as young as 3 months old vary their actions to observe their causal effects on their environment; Du et al. (2023b) show that children playing some video games can be thought of as maximizing their empowerment. Gopnik (2024) hypothesizes that empowerment maximization in RL could become the new dominant paradigm (after Bayesian approaches that struggle to scale to large hypothesis spaces) for explaining exploration in humans and other animals.

Additional Connections to goal-based exploration. When the variational distribution in Equation 35 is Gaussian and fixed, empowerment objectives reduce to goal-based exploration in RL (Choi et al., 2021), by which we mean methods that propose random target states and use a goal-conditioned policy (Schaul et al., 2015) to reach them, for example, in hindsight experience replay (Andrychowicz et al., 2017) and Go-Explore (Ecoffet et al., 2020). In fact, it is possible to think of goal-based exploration and variational empowerment as lying on a spectrum: the more expressive the variational distribution, the more powerful, albeit non-stationary, the associated representation learning problem (Choi et al., 2021). Furthermore, Warde-Farley et al. (2019) advocate for taking a mutual information maximization approach to goal-conditioned reward functions, Pitis et al. (2020) argue that empowerment maximization is roughly equivalent to maximizing the size of the set of goals that can be achieved by the agent's policy, and Levy et al. (2023) find that goal-conditioning can make the empowerment objective significantly easier to compute and optimize. These findings further blur the lines between goal-based exploration in RL and empowerment maximization.

4.5 Via Environment Rewards

Most of the work on learning skills has focused on discovering intrinsic reward functions, which are then used to learn option policies. There are, however, two important lines of work that instead aim to learn behaviour directly through the rewards given by the environment.

Feudal Methods

The first set of approaches builds on *feudal reinforcement learning* (Dayan and Hinton, 1993). In this framework, the agent is decomposed hierarchically into managers and workers: managers set subgoals for workers to achieve, and workers use non-hierarchical RL to achieve those subgoals. In this way, goal-setting is decoupled from goal-achievement; each level in the hierarchy communicates to the level below it what must be achieved, but does not specify how to do so. The manager maximizes the reward coming from the environment to define the goals that the worker should achieve. Feudal RL was extended to deep RL through

Feudal Networks (FuN) (Vezhnevets et al., 2017). FuN learns a two-level hierarchy in which the higher-level manager outputs a goal vector g_t at time t that specifies the direction in which the lower-level worker should modify the agent's current state. Specifically, a linear transformation, ϕ, then maps the last c goals outputted by the manager into an embedding vector w_t,

$$w_t = \phi\big(\sum_{i=t-c}^{t} g_i\big). \tag{44}$$

The worker's policy is then defined through this embedding vector and a matrix of learnable parameters U_t, that is $\pi^{\text{worker}} = \text{SoftMax}(U_t w_t)$.

The worker policy is trained through the standard policy gradient update rule, where the rewards are the goal vectors,

$$r^{\text{worker}}(s_t, g_{t-i}, s_{t-i}) = 1/c \sum_{i=1}^{c} d_{cos}(s_t - s_{t-i}, g_{t-i}), \tag{45}$$

where $d_{cos}(x, y)$ is the cosine similarity measure between vector x and y. The manager policy is learned with the task reward function; however, the authors propose the following update rule, which they term directional policy gradient,

$$\nabla\mu(g_t|s_t) = \nabla d_{cos}(s_{t+c} - s_t, g_t)\big(Q^{\text{manager}}(s_t, g_t) - V^{\text{manager}}(s_t)\big). \tag{46}$$

This update closely puts an emphasis on the direction in which a goal vector points to, and whether that direction was achieved by transitioning from s_t to s_{t+c}.

As with most skill discovery methods, the high-level policy is trained at the same time as the low-level policy; as the low-level policy changes during learning, data from a high-level action taken in the past may not yield the same low-level behaviour in the future (Nachum et al., 2018). This non-stationarity is addressed using relabeling tricks and off-policy learning in the HIRO algorithm (Nachum et al., 2018). In their work, as well as following literature, the worker's intrinsic reward is defined as,

$$r^{\text{worker}}(s_t, g_t, s_{t+1}) = -||s_t + g_t - s_{t+1}||. \tag{47}$$

This definition forgoes the explicit use of the cosine similarity; however, it maintains the idea that a goal vector would represent a delta between state transitions. Later, Levy et al. (2019) present the algorithm Hierarchical Actor Critic algorithm, which improves upon HIRO by removing the need for dense reward functions, by instead using hindsight experience replay (Andrychowicz et al., 2017). In a separate direction, Hafner et al. (2022) instantiate the feudal architecture within a model-based algorithm called Director, which shows strong performance across a wide range of environments. Their approach additionally provides interpretability as the world model can decode goals into images.

Option-Critic

The second set of approaches is based on the *option-critic* (Bacon et al., 2017). In this work, the authors derive both the intra-option policy gradient theorem as well as the termination gradient theorem, which provide the update rules for learning option policies and termination

functions, respectively. The intra-option policy gradient theorem leads to the following update,

$$\frac{\partial q_\pi(s,o)}{\partial \theta} = \sum_{s,o} d^\gamma_{\pi,\mu,\beta}(s,o) \sum_a \frac{\partial \pi_\theta(a|s,o)}{\partial \theta} q_u(s,o,a), \tag{48}$$

where, $d^\gamma_{\pi,\mu,\beta}(s,o) = \sum_t \gamma^t P_{\pi,\mu,\beta}(S_t = s, O_t = o)$, is the γ-discounted occupancy measure over state-option pairs. In the policy gradient theorem (Sutton et al., 1999a), the flat policy is multiplied by the state-action value function, leading to an increased probability for actions whose future discounted return is higher. In the case of the intra-option policy gradient, the quantity modulating the action probabilities is the state-action-option value function; therefore, a strict generalization of the policy gradient theorem. The termination gradient theorem used to learn the termination function is derived from the option value of option o upon arrival in state s (see Equation 11). The update rule takes the following form,

$$\frac{\partial u_\beta(s',o)}{\partial \psi} = \sum_{s,o} d^\gamma_{\pi,\mu,\beta}(s,o) \frac{\partial \beta_\psi(s',o)}{\partial \psi} A(s',o), \tag{49}$$

where $A(s',o) = q_\pi(s',o) - v_\mu(s')$ is the advantage function over options, representing how advantageous it is to be in state s' with option o with respect to the value of state s' averaged over all options. Usually, the advantage function is implemented as a heuristic for reducing the variance of the estimator, but in this case, it comes naturally from the derivation of the theorem. Later, Bacon (2018) unified these different objectives and derivation through the following objective,

$$J_\alpha(\omega) = \sum_{s,o} \alpha(s,o) Q_\omega(s,o) = \mathbb{E}_{\alpha,\omega}\left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t)\right], \tag{50}$$

where $\alpha : Dist(\mathcal{S} \times \mathcal{O})$ is a distribution of an initial state-option pair, and where ω defines all the parameters within the options framework, including the termination, option policies, and high-level policy. The authors show how, by assuming independence between the parameters of these components, the previous update rules can be recovered.

The line of work surrounding the option-critic has received significantly more attention than we can present in detail in this section. Some of the contributions include learning safe policies (Jain et al., 2018), using multiple discount factors (Harutyunyan et al., 2019b), learning option termination in an off-policy manner (Harutyunyan et al., 2019a), extending the theorems to multiple levels of hierarchy (Riemer et al., 2018), and theoretical derivations that take parameter sharing between options into consideration (Riemer et al., 2019).

4.5.1 Benefits and Opportunities

Credit assignment. Vezhnevets et al. (2017) show strong performance of their feudal method on a set of Atari 2600 games from the Arcade Learning Environment (Bellemare et al., 2012) and 3D navigation challenges (Beattie et al., 2016). These domains are long-horizon and require the agent to propagate credit across multiple steps. The authors report significantly better results than a baseline not leveraging such a hierarchy.

Transfer. Bacon et al. (2017) present experiments where the learned options improve the ability to generalize across changes in the Four rooms environment (Sutton et al., 1999b) compared to non-hierarchical RL algorithms. Such changes included modifying the goal location and the agent's starting location. This benefit is later reinforced by multiple works (Zhang and Whiteson, 2019; Khetarpal et al., 2020b; Kamat and Precup, 2020; Klissarov and Precup, 2021) showcasing the transferability of options learned through the option-critic method in more complex environments such as locomotion control (Todorov et al., 2012) and 3D navigation (Chevalier-Boisvert et al., 2023). In these transfer experiments, the agent usually first learns to perform a task before some component of the task is changed.

Interpretability. A particular highlight of the option-critic line of work is that interpretability naturally emerges by learning options directly from environmental rewards. For example, Bacon et al. (2017) report experiments where the termination function would highlight bottleneck states, which are often seen as key in learning temporal abstraction (Stolle and Precup, 2002). Findings on interpretability are similarly reported across different domains (Harb et al., 2018; Klissarov et al., 2017; Zhang and Whiteson, 2019).

Opportunities for Research.

- **Avoiding option degeneracy.** An important practical obstacle when learning options through the update rules proposed by the option-critic is that it may lead to degenerate solutions (Luo et al., 2023). Options tend to reduce to actions where each of the options' duration is only one timestep long. Another observed phenomenon is that only one option ends up being executed throughout all episodes. In both cases, the essence of temporal abstraction is lost. To avoid such undesirable behaviour, the authors add a penalty term c_{delib} to the termination gradient's advantage function: $A(s', o) + c_{\text{delib}}$. This term essentially discourages the termination to prefer switching, unless the advantage in doing so is greater than the value of c_{delib}. A thorough theoretical derivation was later done to justify the use of such a term, which was coined as the deliberation cost (Harb et al., 2018). This cost is introduced as a hyperparameter, which raises the question of what value we should choose for a specific environment. Discovering more general solutions to option degeneration remains an open area of research.

- **Reliance on the environment reward.** The strength of the methods we presented in this section is that they do not require a human-defined objective for learning the hierarchy. As such, such methods heavily rely on an informative environment reward. For example, in feudal methods, if the high-level policy is poorly trained due a sparse environmental rewards, it might output goals that fail to drive the learning progress of the lower-level policy. To address the exploration challenge, recent methods like HAC-Explore incorporate a novelty-based intrinsic rewards (McClinton et al., 2021) or demonstrations (Gupta et al., 2019) to solve longer-horizon tasks.

4.6 Directly Optimizing for the Benefits of Hierarchical Reinforcement Learning

Many of the option discovery methods that we have discussed so far rely on *proxy* objectives; these objectives include finding bottleneck states, empowerment maximization, more reliable

composability, and so on. The intuition is that if the agent had options that maximized these proxy objectives, it would unlock agent-level capabilities such as effective exploration, credit assignment, or transfer. Indeed, these methods often show empirical success in some scenarios, but the formal connection between these proxy objectives and the overall objectives of the agent is unclear (Solway et al., 2014). For example, options that target bottleneck states are empirically useful in some tasks, but what kind of performance can we expect from the same technique in an entirely different problem? In fact, several papers have shown that not all skills are created equal—that is, options that are perfectly suited for a particular task, might severely hurt agent-level objectives in other tasks (Jong et al., 2008; Solway et al., 2014). To address this gap, a class of methods—initiated by Solway et al. (2014)—has sought to discover options with precise guarantees on agent-level objectives. These methods explicitly state the performance criterion of the agent and then derive an algorithm that discovers options with bounded loss on that criterion.

4.6.1 Benefits and Opportunities

Planning. In the planning context, option discovery can be framed as the search for a set of options that minimizes the *planning time*—defined as the number of iterations a planning algorithm (e.g., value iteration) takes to approximate the optimal value function v^* within some accuracy ϵ (Silver and Ciosek, 2012; Jinnai et al., 2019a). Formally, given a maximum allowable value error $\max_{s \in \mathcal{S}} |v^*(s) - \hat{v}(s)| \leq \epsilon$, the goal is to find a set of at most k options $\mathcal{O}$ that minimizes L_ϵ, the number of iterations needed to reach this accuracy:

$$\min_{\mathcal{O}} \; L_\epsilon \quad \text{s.t. } |\mathcal{O}| \leq k. \tag{51}$$

Jinnai et al. (2019a) prove that this problem is NP-hard, even in deterministic tabular MDPs. They introduce approximation algorithms with provable guarantees, but their results are limited to *point options*—options that initiate and terminate in a single state.

While their method minimizes worst-case planning time, Average Options (Ivanov et al., 2024) focuses instead on minimizing the expected planning time across a distribution of tasks. These tasks share the same transition dynamics, but differ in their start and goal states. The idea is to discover options that reduce the expected cost of reaching any state from any other:

$$\arg\min_{\mathcal{O}} \quad d_{\mathcal{O}}(G) = \arg\min_{\mathcal{O}} \sum_{s \in \mathcal{S}} \sum_{s' \in \mathcal{S}} d_{\mathcal{O}}(s, s'), \tag{52}$$

where $d_{\mathcal{O}}(s, s')$ is a non-symmetric distance metric (e.g., shortest path length) in the MDP graph augmented with options $\mathcal{O}$; such an augmentation adds edges to the graph, while leaving nodes unchanged. Like the worst-case version, this problem is also NP-hard. However, by reducing it to the well-studied k-medians with penalties problem in graph theory (Meyerson and Tagiku, 2009), Ivanov et al. (2024) derive efficient approximation algorithms with bounded suboptimality. Planning can also be sped up using options in the single-task setting: Wan and Sutton (2022) present an option discovery algorithm that seeks options that maximize reward—similar to option-critic (Harb et al., 2018)—but reduces the number of options available at different states to reduce planning time.

Exploration. In the context of exploration, Jinnai et al. (2019b) formalize the performance criterion of the agent as minimizing the number of steps needed for a policy to visit every state (as a proxy for discovering some unknown reward). They show that this performance criterion is related to the graph-theoretic property of *cover-time*, which measures the number of steps needed by a random walk to visit every edge in a graph. To define the cover time C, we first need the hitting time H_{ij} between two states $i \to j$: the hitting time in a Markov chain is the greatest lower-bound on the number of steps needed to get from source state i to destination state j: $H_{ij} = \inf\{t : \mathcal{S}_t = j \mid \mathcal{S}_0 = i\}$. Then, the cover time C_i starting in state i is the maximum hitting time over all possible destination states: $C_i = \max_{j \in \mathcal{S}} H_{ij}$. Jinnai et al. (2019b) show that the expected cover time $\mathbb{E}[C_i]$—where the expectation is with respect to the dynamics induced by a random walk—can be most effectively reduced by creating an option that connects the two states that are furthest apart according to the second eigenvector of the graph Laplacian (see Equation 20). Jinnai et al. (2019b) also show that finding options that minimize cover-time in a graph is NP-Hard; but, they provide an approximation algorithm that minimizes an upper-bound on the expected cover-time. This method was later extended to continuous environments using deep learning-based approximations of the graph Laplacian (Jinnai et al., 2020; Wu et al., 2019), further suggesting strong connections to the eigenoptions literature (Machado et al., 2017, 2023; Klissarov and Machado, 2023) (c.f. Section 4.2).

Credit assignment. As discussed earlier, options can accelerate policy evaluation by enabling value updates that span multiple steps, rather than progressing one step at a time. Bacon and Precup (2016) formalize this intuition using the lens of *matrix splitting*, a technique from numerical linear algebra that speeds up the solution of linear systems. In their view, each set of options defines a modified Bellman operator that can be interpreted as a *preconditioned* version of the original policy evaluation problem. Recall that the Bellman expectation equation for a fixed policy π is:

$$v = r_\pi + \gamma P_\pi v, \tag{53}$$

where $v \in \mathbb{R}^{|\mathcal{S}|}$ is the value function, r_π is the expected reward vector, and P_π is the transition matrix under policy π. This is a linear system of the form $Av = b$, with $A = I - \gamma P_\pi$, and $b = r_\pi$. Planning with options induces a *matrix splitting* $A = M - N$ (Varga, 2000), leading to an iterative update of the form:

$$v_{k+1} = M^{-1} N v_k + M^{-1} b. \tag{54}$$

In this formulation, the matrix M reflects the dynamics induced by the options, and is chosen to be easy to apply and invert; the remaining part N captures what is not directly handled by the options. The matrix $M^{-1}N$ is known as the *iteration matrix*, as it governs how the current value estimate v_k influences the next one v_{k+1}. This kind of transformation is known as *preconditioning*: a way of rewriting the problem so that the resulting iterative updates converge more quickly. The speed of convergence is governed by the *spectral radius* $\rho_r(M^{-1}N)$: the largest absolute eigenvalue of the iteration matrix. A smaller spectral radius means that errors shrink faster with each iteration. From this perspective, a good set of options is one that minimizes $\rho_r(M^{-1}N)$, enabling value information to propagate more efficiently. While Bacon and Precup (2016) do not introduce a concrete option discovery

algorithm, they offer a powerful design principle: discover options that act as preconditioners for value propagation. This opens the door to leveraging ideas from numerical linear algebra in option discovery.

Transfer. In the context of transfer, Solway et al. (2014) define the optimal set of options as those that maximize the efficiency with which an agent can learn the optimal policy for other, possibly unseen, sets of tasks. They show that in this setting, optimal options are those that maximize Bayesian model evidence under the distribution of tasks that the agent is expected to solve. Specifically, a hierarchy that maximizes model evidence, also provably minimizes the geometric mean of the number of samples needed to find the optimal policy for any task in the given task distribution. Brunskill and Li (2014) consider a similar formulation of the option transfer problem: given interaction data from a set of tasks, how can an agent learn options that minimize the sample complexity of learning in a future stream of tasks? They find that this problem is at least as hard as the set cover problem in Operations Research, and is thus also NP-hard. They use a greedy approximation algorithm for option discovery and evaluate it empirically in a tabular MDP.

Opportunities for Research.

- **Guarantees in more general settings.** The papers discussed in this section emphasize the importance of formally stating the objective of option discovery and relating that to the overall objectives of the agent. However, this research is still nascent, and more papers exploring this subject are needed. For instance, can we develop formal algorithms that bound planning time without needing the assumption of "point options"? Can we bound planning time or cover time when using function approximation? Although Brunskill and Li (2014) derive an algorithm to minimize sample complexity during transfer, the greedy approximation algorithm they present does not bound sample complexity; future work could extend their theoretical results to bound the performance of the greedy approximation algorithm. Finally, can we write down the problems of option-driven exploration, planning, and policy evaluation in different ways that result in HRL algorithms with even stronger guarantees or better scaling properties?

4.7 Meta Learning

RL algorithms, such as Q-learning, learn policies; meta-RL algorithms, in contrast, aim to learn the RL algorithm itself, or parts of it, to subsequently learn a policy. This creates a bilevel optimization: the algorithm for learning the RL algorithm itself is called the *outer-loop*, while the learned algorithm (which learns a policy) is called the *inner-loop* (Schmidhuber, 1987; Thrun and Pratt, 1998; Beck et al., 2023). The appeal of meta-RL approaches is that if the environment demands certain properties from the RL agent (for example, transferability), then such properties will automatically be learned from data, without the explicit need for careful human ingenuity and design in every part of the training process (Silver et al., 2021).

Typically, a meta-RL algorithm consists of an inner and an outer loop. Within each of these loops, a set of parameters is being maximized. Concretely, let ω_{out} represent the parameters learned by the outer loop, and ω_{in} the parameters learned by the inner loop.

These parameters in practice represent a particular subset of the option parameters presented in Section 3. For example, in the work by Veeriah et al. (2021), the inner loop optimizes the parameters option policies and the high-level policy, whereas in the work by Frans et al. (2018) the parameters of the high-level policy are part of the outer loop.

Meta-Gradients

A common instantiation of meta-RL algorithms is through the use of meta-gradients. In the inner loop, the agent updates the inner parameters,

$$\omega'_{\text{in}} \leftarrow \omega_{\text{in}} + \alpha \nabla_{\omega_{\text{in}}} J_{\text{in}}(\omega_{\text{in}}), \tag{55}$$

where J_{in} is an arbitrary objective that depends on ω_{in}. To obtain the meta-gradients, we assume that the outer parameters depend on the inner parameters. Data is then collected with the updated inner parameters such to proceed to the following update,

$$\omega'_{\text{out}} \leftarrow \omega'_{\text{out}} + \alpha \nabla_{\omega_{\text{out}}} J_{\text{out}}(\omega'_{\text{in}}(\omega_{\text{out}})), \tag{56}$$
$$\omega'_{\text{out}} \leftarrow \omega'_{\text{out}} + \alpha \nabla_{\omega'_{\text{in}}} J_{\text{out}}(\omega'_{\text{in}}(\omega_{\text{out}})) \nabla_{\omega_{\text{out}}} \omega'_{\text{in}}(\omega_{\text{out}}), \tag{57}$$

where $\nabla_{\omega_{\text{out}}} \omega'_{\text{in}}(\omega_{\text{out}})$ encodes how the outer loop parameters affected the updated inner loop parameters. The objectives J_{in} and J_{out} may differ in various ways, such as defining different distributions over tasks.

Veeriah et al. (2021) leverage meta-gradients to learn options in high-dimensional navigation environments. In the inner loop, they update the option policies parameters, θ, and the high-level policy parameters κ,

$$\theta' \leftarrow \theta + \alpha_\theta (G_t - q_\pi(s_t, o_t)) \cdot \nabla_\theta [\log \pi_\theta(a_t | s_t, o_t) - q_\pi(s_t, o_t)], \tag{58}$$
$$\kappa' \leftarrow \kappa + \alpha_\kappa (G^\mu_t - v_\mu(s_t)) \cdot \nabla_\kappa [\log \mu_\kappa(o_t | s_t) - v_\mu(s_t)], \tag{59}$$

where G_t is the option policy return (see Section 3.2.1) and G^μ_t is a n-step return for the high-level policy defined as, $G^\mu_t = \sum_{j=1}^{n} \gamma^j r_{t+j} - \gamma^n c + \gamma^{n+1} V_\mu(s_{t+n})$ where c is a switching cost added on option terminations, similar to Harb et al. (2018). The outer loop is instantiated through these updates to the parameters ν of the option reward function, and the parameters ψ of the termination function,

$$\psi \leftarrow \psi + \alpha_\psi (G^\mu_t - v_\mu(s_t)) \nabla_\psi \log \pi_{\theta'(\psi,\nu)}(a_t | s_t, o_t), \tag{60}$$
$$\nu \leftarrow \nu + \alpha_\nu (G^\mu_t - v_\mu(s_t)) \nabla_\nu \log \pi_{\theta'(\psi,\nu)}(a_t | s_t, o_t). \tag{61}$$

The outer loop updates the option-reward and termination meta-parameters using a new trajectory generated by interacting with the environment using the most recent inner-loop parameters, $\theta'(\psi, \nu)$ and $\kappa'(\psi, \nu)$, which depend on the outer loop parameters. The update in the outer loop assesses the impact of updates to the high-level policy, μ_κ, and option policies, π_θ, and it may involve a different distribution of tasks than the one used in the inner loop, as is common in meta learning (Finn et al., 2017).

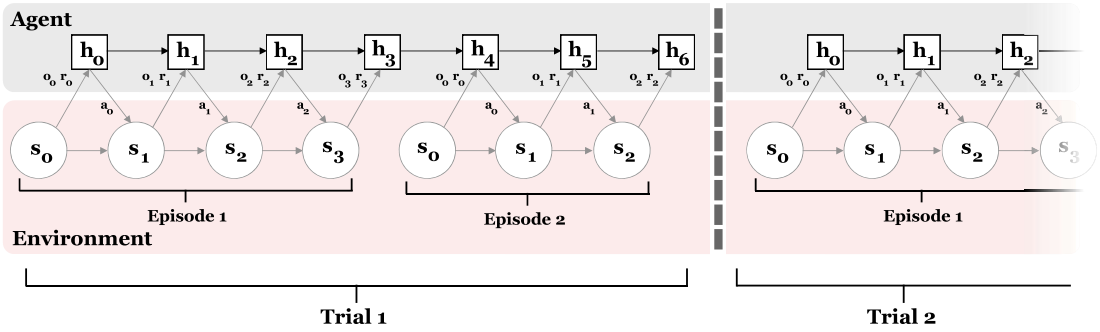


Figure 10: Black-box meta reinforcement learning. Trials consist of multiple episodes during which the hidden state, h_i, of the agent is unrolled. The hidden state is only reset between trials. Figure reproduced from (Duan et al., 2016).

Black-box Meta Reinforcement Learning

In black-box meta RL (Wang et al., 2016; Duan et al., 2016), an agent interacts with a sequence of different tasks drawn from an arbitrary distribution, $p^{\xi} : \xi \to \mathbb{R}_+$. Each interaction with a task, or distribution of tasks, is considered a trial, which itself consists of N episodes, represented in Figure 10. During a trial, the agent receives observations, rewards, and termination signals from the environment, where episode termination signals represent the episode boundaries. These variables are used to update the agent's internal memory h, which is typically represented by the hidden state of an RNN (Hochreiter and Schmidhuber, 1997) or the context of a transformer network (Vaswani et al., 2017). Importantly, the agent continuously updates h across episodes within the same trial; the memory is only reset at the end of each trial. The overall goal is to maximize the total reward accumulated over an entire trial,

$$\max_{\pi} \mathbb{E}_{\xi \sim p^{\xi}} \left[\sum_{\text{episode}=1}^{N} \mathbb{E}_{\pi} \left[\sum_{t} r^{\xi}(s_t, a_t) \right] \right], \tag{62}$$

where r^{ξ} is the reward associated with task ξ. This objective incentivizes the agent to learn how to adapt its policy based on the experience gathered so far during a trial, effectively forcing it to implicitly learn, through the updates to its policy's memory h, a reinforcement learning rule capable of efficient adaptation to new tasks. When further conditioning the policy π on the task's goal g, as done by Bauer et al. (2023), this approach can lead to human-timescale adaptation.

4.7.1 Benefits and Opportunities

Transfer. As discussed earlier, a major benefit of learning options is that of reuse: options learned in one part of the state-space could speed up learning in another (Taylor and Stone, 2009; Konidaris and Barto, 2007). Some methods have tried to discover transferable options using meta-learning. For example, MLSH (Frans et al., 2018) discovers a set of policies and trains a high-level policy to select among them. The meta-objective trains these components such that the high-level policy can quickly learn to solve new tasks from a distribution by

reusing the learned skills, making them reusable across a pre-specified task distribution (Nam et al., 2022; Gupta et al., 2018; Fu et al., 2023). MODAC (Veeriah et al., 2021) uses meta-gradients (Xu et al., 2018; Oh et al., 2020) to do the same: an outer loop learns option reward functions and termination functions that an inner loop maximizes using policy gradients. The outer loop of the optimization learns from the reward coming from the environment.

Exploration. Meta-learning approaches have also sought to address the exploration question in non-stationary and multi-task settings. When the agent finds itself in a new environment, how can it leverage its past experiences to targetedly explore this new environment? This problem is called *meta-exploration* by Beck et al. (2023). For example, when someone is in a new house and they have to look for utensils, they begin their search from the kitchen; similarly, we would like to create RL agents that can direct their exploration for quick adaptation in new environments (Gupta et al., 2018). This is one of the motivations for the Adaptive Agent (AdA) (Bauer et al., 2023) that uses meta-learning to train a policy capable of human-timescale adaptation in a massive, combinatorial task space (OEL Team et al., 2021). Specifically, they use black-box meta-RL: the policy is implemented as a Transformer-XL model (Dai et al., 2019). This model π_θ takes the history h of interactions within the current episode (past states, actions, rewards) and goal description, g, as input to determine the next action. The adaptation happens implicitly within the recurrent state of the model. The combinatorial complexity of the environments allows for careful selection of tasks that are at the appropriate difficulty given the current agent capabilities, generating an effective meta-learning curriculum. As such, AdA mixes ideas from meta-learning as well as curriculum learning, which we cover in the next section.

Opportunities for Research.

- **Relaxing the multi-task formulation.** Meta-learning approaches have demonstrated abilities of transfer, adaptation, and meta-exploration—abilities that have been challenging to scalably acquire using other techniques. Furthermore, meta-learning via in-context learning (Dong et al., 2022; Bauer et al., 2023; Raparthy et al., 2023) provides a scalable, and potentially simpler way, to acquire these crucial capabilities. Existing meta-learning approaches rely on a specialized multi-task formulation, with clear task boundaries and episodes. Methods that lift these assumptions will be able to bring these capabilities to a wider variety of settings. For an in-depth review of meta learning approach, please refer to Beck et al. (2023).

4.8 Curriculum Learning

Within a complex environment, there exists a diversity of goals that are interesting for an agent. Some of these goals might be easily achievable, whereas others would simply be impossible to complete for an agent's current capabilities. How could, then, such an agent learn to achieve difficult goals? An effective strategy would be to try to achieve a curriculum of goals, where the complexity of each attempted goal increases continuously with the agent's capabilities. The idea of curriculum learning has a long history in AI that goes beyond the RL setting (Kaplan and Oudeyer, 2003; Schmidhuber, 2004; Bengio et al., 2009; Schmidhuber, 2011). A central question then becomes, how should one prioritize

which goal or task should be attempted at any given time? Taking the HRL perspective, we can rephrase this question as: how should the high-level policy select the next goal? This question can be formalized through the following objective function:

$$\max_{\mu} \sum_{g \in \mathcal{G}} \mathbb{E}_{\pi'}[r^g], \tag{63}$$

where the goal-conditioned policy, π', used in the expectation, $\mathbb{E}_{\pi'}[\cdot]$, depends on the choice of the goal selection distribution μ. Specifically, π' is obtained by starting with an initial policy π_0 and applying N iterative updates. For each iteration $k = 1, \ldots, N$, a goal $g_k \sim \mu$ is sampled, and the policy is updated via $\pi_k = U_{g_k}(\pi_{k-1})$, where the update rule U_{g_k} aims to maximize the reward r^{g_k} associated with goal g_k, i.e. through policy gradient updates. The final policy used in Equation 63 is $\pi' = \pi_N$. The optimization will thus find the distribution μ which, when used to update π, would lead to the best performance as measured across all goals $\mathcal{G}$.

The objective of Equation 63 is also referred to as the global learning progress (LP), and is, as such, intractable. Researchers have thus approximated this objective through local measures of LP_{local} (Baranes and Oudeyer, 2013; Stout and Barto, 2010; Forestier et al., 2017; Colas et al., 2018) which can be defined as,

$$\text{LP}_{\text{local},g} = V_{\pi_t,g} - V_{\pi_{t-i},g}, \tag{64}$$

where $V_{\pi_t,g}$ is the estimate of the performance of the updated policy after t iterations on goal g and $V_{\pi_{t-i}}$ is that of the policy at iteration $t - i$. These values are usually obtained through Monte Carlo estimates by rolling out policies over multiple episodes, thus possibly covering a subset of the possible goals within the goal space.

In such methods, the high-level policy μ is often optimized through multi-arm bandit algorithms rather than through RL. In other words, μ maximizes the following one-step reward: $\max_\mu = \mathbb{E}[r^\mu] = \mathbb{E}[\text{LP}_{\text{local},g}]$, which can then be defined as

$$\mu_t(g) = \frac{\exp(|E_t(g)|/e)}{\sum_{g \in \mathcal{G}} \exp(|E_t(g)|/e)}, \tag{65}$$

where e is the temperature and E_t is an exponential moving average of the rate of change in performance on goal g,

$$E_{t+1}(g) = (1 - \alpha)E_t(g) + \alpha \text{LP}_{\text{local},g}. \tag{66}$$

The global learning process objective can be approximated through other means, which we discuss in the following sub-section on the benefits and opportunities.

In addition to covering methods that produce curricula by explicitly generating goals according to a certain distribution, we also include a discussion around *implicit curricula*. In these methods, certain properties of the learning algorithm itself create a curriculum-like effect. A prominent example of an implicit curriculum is hindsight experience replay (HER) (Andrychowicz et al., 2017), which stores experience generated by seeking a certain goal g in a buffer called an experience replay, and relabels such experience with a variety of other goals g'. We present HER in Algorithm 2, where we highlight the operations that differ

from the standard use of an experience replay. We use the symbol of the high-level policy, μ, as the operator that relabels experience. In its most common form, HER relabels stored trajectories that do not reach their intended goals with whatever final state was reached. The relabeled goals then tend to naturally progress from those easily achievable by a random agent to increasingly challenging ones.

Algorithm 2 Hindsight Experience Replay (HER)

Require: Goal sampling strategy $\mathcal{S}$
Initialize replay buffer $\mathcal{D}$
for episode = 1 to M **do**
    Sample an initial state s_0 and a goal g
    Generate an episode trajectory $(s_0, a_0, s_1, \ldots, s_T)$
    for $t = 0$ to $T - 1$ **do**
        Calculate reward $r_t = r^g(s_t, a_t)$
        Store transition $(s_t, a_t, r_t, s_{t+1}, g)$ in $\mathcal{D}$
        Sample a set of additional goals $\mathcal{G}' = \mathcal{S}$ (current episode)
        for all $g' \in \mathcal{G}'$ **do**
            Calculate hindsight reward $r'_t = r^{g'}(s_t, a_t)$
            Store transition $(s_t, a_t, r'_t, s_{t+1}, g')$ in $\mathcal{D}$
        end for
    end for
end for

Research on learning from a curriculum of goals has received much more attention than we can cover here, producing a diversity of approximations to the global learning progress (Forestier and Oudeyer, 2016; Matiisen et al., 2017; Kovač et al., 2020; Akakzia et al., 2021). For an in-depth review please see the surveys by Colas et al. (2020c) and Portelas et al. (2020).

4.8.1 Benefits and Opportunities

Exploration. One of the main benefits of leveraging curriculum learning to achieve goals is that the agent will be continuously pushed to the limits of its capacity. By doing so, it might discover new locations in an environment or learn completely new behaviour from the combination of previously achieved goals. A family of methods approximates the global learning progress, specifically with the intent of seeking intermediate difficulty. Florensa et al. (2018) propose using a Goal Generative Adversarial Network (Goal GAN) to automatically generate a curriculum of tasks for reinforcement learning agents. The method focuses on generating Goals of Intermediate Difficulty (GOID), defined as goals where the agent's current policy, π, achieves an expected performance v_π within a specific range:

$$\text{GOID}_i := \{g : v_{\min} \leq v_\pi \leq v_{\max}\}. \tag{67}$$

Here, $v_{\min}$ and $v_{\max}$ represent the minimum and maximum desired performance, ensuring goals are neither too easy nor too hard for the current policy π_i. The generator in Goal GAN is trained to output goals within the GOID set, whereas the discriminator is trained to distinguish between goals that are within the set from those that are not. Racanière et al.

(2019) introduce a setter-solver paradigm with three criteria represented through values in $[0, 1]$: validity, feasibility, and coverage. Goals are sampled according to the distribution defined by these criteria, allowing for a balanced selection. Their findings highlight that these criteria, along with conditioning on the current version of the environment, are crucial for an effective learning curriculum. Sukhbaatar et al. (2018) instead rely on asymmetric self-play to generate a curriculum of explorative goals in reversible or resettable environments, leading to improved performance on a diverse set of tasks. Campero et al. (2021) train a goal-generating teacher to guide a goal-conditioned student policy by proposing goals that are neither too hard nor too easy, as measured by the number of timesteps to reach the goal,

$$r^{\mu} = \begin{cases} +a & \text{if } t^{+} \geq t^{*} \\ -b & \text{if } t^{+} < t^{*}, \end{cases} \tag{68}$$

where a, b are hyperparameters that quantify the bonus and penalty, t^{π} represents the time it took the policy to reach the goal, and t^{*} is a hyperparameter.

Additionally, HER-based approaches (Andrychowicz et al., 2017; Fang et al., 2018; Yang et al., 2021a) have demonstrated promising results in improving exploration compared to curiosity-driven methods. Curriculum-guided HER (Fang et al., 2019) introduces an explicit curriculum that transitions from curiosity-driven selection early on to goal-proximity focus in later stages, mimicking human-like exploration. Complementing HER, CER (Liu et al., 2019) enhance exploration by introducing a competitive dynamic between two agents learning the same task, where one agent is penalized for revisiting states explored by the other. Many more works show how curriculum learning can help in hard-to-explore environments (Colas et al., 2018; Zhang et al., 2020; Pitis et al., 2020; Colas et al., 2020a).

Transfer. Learning successfully through curricula produces a whole set of behaviours that were previously not seen. OEL Team et al. (2021) leverage population-based training to quantify progress on goal completion within large, open-ended, and procedurally-generated environments and tasks. Through a continuum of task difficulty, the authors show that the resulting goal-conditioned agent can generalize zero-shot to new situations.

Opportunities for Research.

- **Refining the measure of progress.** One of the main challenges in deriving a curriculum of goals is accurately measuring how difficult a chosen goal is for a learning algorithm at a certain point in time. Different heuristics can work well for certain environments, for example, the number of timesteps required for reaching a goal (Campero et al., 2021), or might involve a combination of heuristics (OEL Team et al., 2021). However, such formalizations might not be generally applicable. Ideas from unsupervised environment design, where the environment evolves as well as the agent's parameters, could be particularly promising (Dennis et al., 2020; Jiang et al., 2021; Parker-Holder et al., 2022; Samvelyan et al., 2023). Another important desideratum is that a curriculum should continuously increase the difficulty of the goals, but should also generate interesting goals. Finding a formalization that would encode a general measure of interestingness and difficulty is still an open question. However, for many tasks of interest, such as tasks where human prior knowledge would be relevant, leveraging foundational models offers a particularly promising way to

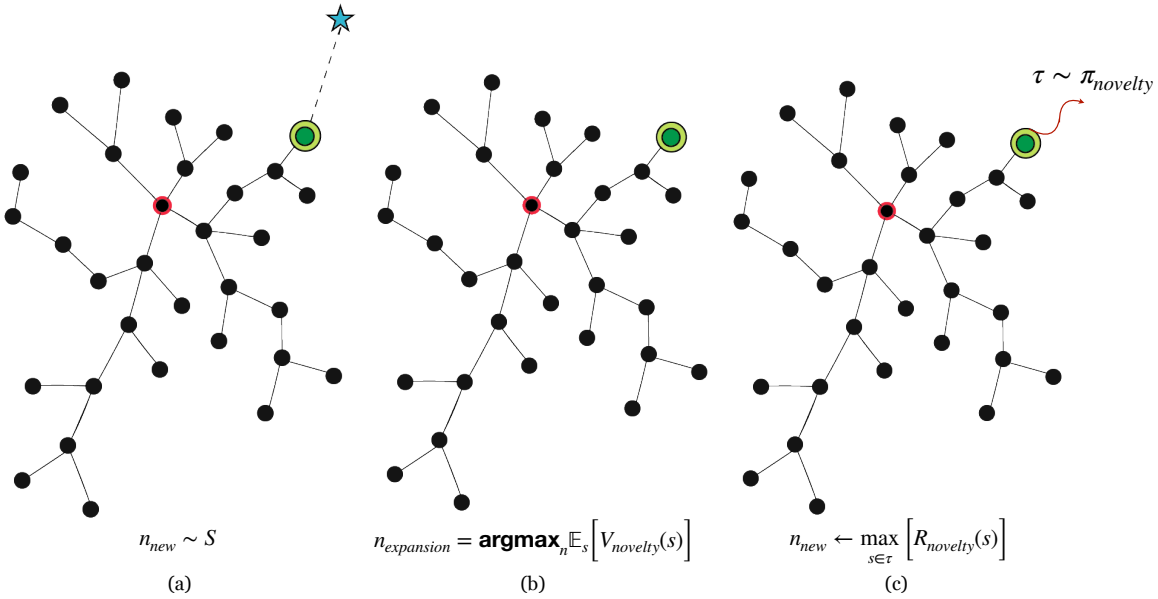


Figure 11: (a) In the original DSG algorithm, a state is sampled uniformly at random (blue star) from the state-space $\mathcal{S}$ and the graph is pulled towards it via its nearest neighbor node (green). (b) The IM-DSG agent uses intrinsic motivation to identify a node to expand using an exploration 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 is identified as a target for a new skill.

define such metrics for curriculum learning, as covered in Section 6. One notable work is that of Zhang et al. (2024), which investigates whether an LLM's common sense can be a good measure of interestingness in open-ended environments.

4.9 Intrinsic Motivation

Intrinsic Motivations (IM) drive actions for their own sake, meaning that they are not in service of achieving an obvious, externally specified goal; instead, they are in service of augmenting knowledge and learning skills whose utility only becomes apparent later on (Oudeyer and Kaplan, 2007; Barto and Şimşek, 2005; Berlyne, 1965; Harlow, 1950). Computationally, this can be formalized using notions of information gain (Bellemare et al., 2016b)—an agent may take actions that result in new information about its environment, even if it requires forsaking extrinsic reward in the short term. IM underpins a developmental approach where an agent learns reusable skills autonomously, preparing it for various future challenges. For example, children, as intrinsically motivated biological agents, develop skills by engaging in activities that yield interesting, memorable outcomes (Gopnik et al., 2009); these skills improve in efficiency with repetition and can be strategically reproduced for specific goals (Barto et al., 2004). Such behaviours are well represented by options, with the intended outcomes encapsulated in the options' subgoals (Singh et al., 2004). While

we have discussed IM-based option discovery approaches like empowerment maximization, spectral methods, and bottleneck discovery, this section explores additional methods not covered by these categories.

One common intrinsic motivation signal is novelty, which decreases with repeated state visitations. For example, upon visiting state s, an agent might receive

$$r^{\text{int}}(s) = N(s)^{-1/2}, \tag{69}$$

where $N(s)$ is the visit count. This count-based bonus encourages exploration of infrequently seen states by making familiar states less rewarding (Auer et al., 2008; Strehl and Littman, 2008; Bellemare et al., 2016b; Lobel et al., 2023). An example of using this for option discovery is provided by the Relative Novelty algorithm (Simsek and Barto, 2004). Here, a state s is deemed a good subgoal if it leads to experience that is significantly more novel than the experience preceding it. Let $n(s)$ be a novelty score (e.g., inverse visitation count from Equation 69), and let w^+ and w^- be fixed-size windows of future and past states, respectively. The relative novelty at time t is then computed as

$$\text{RN}(s_t) = \frac{1}{|w^+|} \sum_{s \in w^+} n(s) \Big/ \left(\frac{1}{|w^-|} \sum_{s \in w^-} n(s) \right). \tag{70}$$

States with high relative novelty are likely to be gateways to unexplored regions (e.g., doorways in Figure 6), and can be automatically selected as subgoals. An option is then created to reach each such subgoal from a broader initiation set, often by learning a dedicated policy using intrinsic reward. In this way, the agent transforms spikes in novelty into reusable skills without supervision or knowledge of external task rewards.

A related example is First Return then Explore (FRTE) (Ecoffet et al., 2020), which formalizes intrinsic motivation using count-based novelty over discretized "cells", which are pre-specified state abstractions that serve as option subgoals (an example state abstraction is spatially downsampling image-based observations). Every time a new cell is encountered, the agent logs it in an archive. The policy is then conditioned to return to these cells to deepen exploration. FRTE selects target cells based on their inverse visitation count, returning to underexplored frontiers before taking random, exploratory actions. This loop results in a growing archive of reachable states, each effectively defining an option subgoal. A model-based approach has recently been proposed by Bagaria et al. (2025b), who extend the Deep Skill Graphs algorithm discussed in Section 4.3 to image-based observation spaces where a meaningful distance metric is not readily apparent. Their algorithm, *Intrinsically Motivated Deep Skill Graphs* (IM-DSG), learns a graph-based model of the world—nodes of the graph represent option subgoal regions (abstract states) and edges represent option policies (abstract actions). Figure 11 iilustrates the main steps of the algorithm: first, the agent picks an existing node based on how much that node is expected to contribute to exploration (Figure 11(b)), then the agent plans with its abstract model using dynamic programming to determine the options it needs to executes to reach the sampled node, from where it executs a novelty seeking exploration policy (Figure 11(c)). States visited by the novelty-seeking policy are candidates for creating a new node in the graph, similar to the Relative Novelty algorithm discussed earlier.

A different line of intrinsically motivated option discovery leverages structural state-space features to define internal subgoals. In particular, methods for factored MDPs (Boutilier et al., 2000) use the causal dependencies between state features to propose options. A factored MDP assumes the state can be described by a vector of state variables and that the transition dynamics factorize according to a dynamic Bayesian network (DBN). HEXQ (Hengst et al., 2002) was an early algorithm in this category. HEXQ automatically decomposes a factored MDP into subtasks by detecting *exits*—states where a change in one state variable causes a change in another variable (or termination). More formally, if X and Y are state variables, an *exit* can be identified where the conditional entropy $\mathcal{H}(Y' \mid X, a)$ increases sharply, indicating that a transition cannot be explained without accounting for additional variables. Each such transition is marked as an exit and treated as a subgoal. HEXQ then learns a hierarchy of options, each option driving the agent toward one of these subgoals—lower-level options correspond to frequently-changing variables, whereas higher-level options handle more slowly-changing aspects. In this way, HEXQ yields an option hierarchy spanning multiple levels of temporal abstraction.

A similar approach is proposed by Jonsson and Barto (2006), who analyze the structure of a learned DBN to extract a causal abstraction hierarchy. For each action, they examine the DBN's parent-child dependencies: if variable X_i influences the next-state value of X_j, i.e., $X_i \in \text{Parent}(X'_j \mid a)$ for some action a, then X_i is said to causally affect X_j. These dependencies define a directed graph over state variables, which is decomposed into strongly connected components (SCCs). The SCCs are then topologically ordered to yield levels of abstraction—variables in earlier components are controlled first, while later components are conditioned on them. This is because earlier variables in the topological ordering tend to be those that causally influence, but are not influenced by, variables in later components. While their algorithm assumes access to the transition model, Vigorito and Barto (2008) propose a model-free algorithm that incrementally builds DBNs online through exploration. When new dependencies are detected—e.g., when variable X_i begins to influence X_j—a new option is instantiated to induce that dependency reliably. This learning process can be guided by structure learning techniques (e.g., maximizing Bayesian Information Criterion), or by identifying transitions that lead to salient changes in state abstractions. More recently, Nayyar and Srivastava (2024) cluster states based on the temporal-difference (TD) error:

$$\delta_t = r_t + \gamma v(s_{t+1}) - v(s_t), \tag{71}$$

which serves as a proxy for learning progress. Regions with high variance in δ_t are recursively split, producing a symbolic abstraction over state variables and spawning new options targeted to regions where prediction error remains high. This method resembles HEXQ, but instead of focusing on the frequency of variable changes, it uses the TD-error incurred by candidate state abstractions.

One downside of these factored approaches is that they assume the agent observes factored state variables, which requires significant domain knowledge. Bagaria et al. (2025a) address this limitation by developing an agent that learns to identify relevant features directly from image observations. When their agent encounters a particularly novel state, it uses counterfactual analysis to isolate which visual features are responsible for the novelty of that state. Then, the agent learns a classifier that focuses only on these salient features (Singh et al., 2004), ignoring other aspects of the image. This feature-specific classifier serves

as an abstract subgoal (Bagaria and Schaul, 2023) for option learning, enabling factored skill discovery without requiring pre-specified state decompositions.

4.9.1 Benefits and Opportunities

Exploration. The primary role of intrinsic motivation in RL is to facilitate exploration, especially when extrinsic rewards are sparse, delayed, or misleading. By rewarding novelty, surprise, or learning progress, IM helps the agent identify and prioritize skills that could result in mastery of the environment (Veeriah et al., 2018). The resulting behaviours are often more structured and directed than undirected exploration strategies like epsilon-greedy or softmax sampling.

Transfer. Options discovered through intrinsically motivated factor-based methods are transferable because they target changes in individual state variables or abstract subspaces (Sutton et al., 2023). For example, in HEXQ, an option that changes a frequently occurring variable—like an agent's position—can be reused across many contexts where that variable matters, regardless of the values of other variables. Similarly, in methods based on causal abstraction (Jonsson and Barto, 2006; Vigorito and Barto, 2008), options are constructed to affect only a specific part of the environment while assuming other parts are stable or independently controllable. This modularity reduces interference between options because each option specializes in a different part of the environment, which additionally encourages compositionality in behaviour space. As a result, once such an option is learned, it can be reused across multiple tasks or contexts without retraining, greatly improving sample efficiency.

Opportunities for Research.

- **Generalize factored approaches to large observation spaces.** Factored approaches like HEXQ assume access to explicit state variables, limiting their applicability to high-dimensional observation spaces such as images or sensor data. Recent works (Bagaria et al., 2025a; Higgins et al., 2016; Kim and Mnih, 2018) demonstrate some promising initial results, but significant challenges remain in automatically discovering meaningful factors without domain knowledge. Future research should focus on developing methods that can identify relevant features and their causal dependencies in continuous, high-dimensional spaces while maintaining the transferability and combosability benefits of factored approaches.

- **Improved estimates of novelty.** While count-based novelty measures work well in discrete spaces, they struggle in continuous environments where exact state revisitation is unlikely. Recent advances (Bellemare et al., 2016b; Burda et al., 2019; Lobel et al., 2023; Guo et al., 2022) provide neural network-based alternatives, but fundamental challenges persist in distinguishing meaningful novelty from environmental stochasticity (sometimes colloquially called the "noisy TV" problem). Future work should develop more robust novelty estimation methods that can maintain exploration incentives across different timescales, handle function approximation errors gracefully, and integrate structural priors about environment dynamics to focus on causally relevant state changes.

- **Connections to other discovery techniques.** Intrinsic motivation approaches have developed largely independently from other option discovery methods like graph clustering, empowerment, and spectral decomposition techniques. However, recent theoretical work suggests that some of these approaches may be unified under information-theoretic frameworks (Achiam et al., 2018). Establishing formal connections between intrinsic motivation and other discovery paradigms (for example, (Machado et al., 2020)) could enable hybrid approaches that leverage the complementary strengths of different methods.

5. Discovery through Offline Datasets

Offline RL (or batch RL) aims to learn policies from pre-collected datasets, avoiding active data collection. This enables scalable deployment in domains such as robotics, autonomous driving, education, and healthcare (Levine et al., 2020), where interaction data can be difficult to obtain. Similarly, *offline skill discovery* leverages these datasets to extract temporally abstract behaviours that can later serve as high-level primitives (in either offline or online RL) to accelerate learning.

In this section, to facilitate the discovery of skills, we assume access of an offline dataset $\mathcal{D} = \{\tau_i\}_{i=1}^N$ where $\tau_i = (s_t^i, a_t^i, r_t^i)_{t=1}^T$ represents a trajectory interacting with the environment. The dataset can be populated with expert demonstrations or acquired through arbitrary policies. It is also possible that not all components are present in the dataset; indeed, numerous works do not assume access to rewards (most methods in Section 5.1) or actions (Kim et al., 2019).

A closely related line of work, which we do not cover in detail, focuses on learning useful representations from offline datasets (Ma et al., 2020; Touati et al., 2023; Farebrother et al., 2023; Chen et al., 2023; Park et al., 2024b; Tirinzoni et al., 2025). A key idea in these methods involves learning representations that decouple environment dynamics from specific task rewards. This is often done by modeling discounted future state occupancies or their features (Dayan and Hinton, 1993; Barreto et al., 2017), which then allows for rapid adaptation to new reward functions or goal specifications, typically by linear combination of the learned representations based on the new reward.

5.1 Variational Inference of Skill Latents

A prominent class of methods in offline skill discovery focuses on the **reconstruction** loss of pre-collected trajectories τ, typically optimized through likelihood maximization of the observed data. In these approaches, skills will be defined as the latent variables within the reconstruction loss. The methods rely on *unlabeled experiences*, $\tau = (s_t, a_t)_{t=1}^T$—that is, data collected without explicit reward feedback—and in some cases, even excludes actions (Kim et al., 2019). This is often referred to as "unsupervised skill discovery" (Eysenbach et al., 2019). We model each trajectory τ with a latent skill sequence:

$$\zeta = (z_t, b_t)_{t=1}^T, \qquad z_t \in \mathbb{R}^d,\ b_t \in \{0, 1\}, \tag{72}$$

where z_t encodes the skill active at time t and b_t is a boundary signal that indicates when a skill starts or ends, i.e. the analogue of an option-termination signal.[6]

6. Both z_t and b_t can be in different parameterization (e.g., $b_t \in \mathbb{Z}$ as boundary indicator variable in Kipf et al. (2019)); some implementations even dispense with b_t (e.g., OPAL (Ajay et al., 2021)).

Equation below states the maximum-likelihood objective: the parameters ϕ are adjusted to maximize the average log likelihood that the model assigns to the trajectories observed in the dataset:

$$J(\phi) = \mathbb{E}_{\tau\sim\mathcal{D}}\big[\log p_\phi(\tau)\big]. \tag{73}$$

Here the term $p_\phi(\tau) = \int p_\phi(\tau,\zeta)\,d\zeta$ has already *marginalized* the latent skill sequence ζ, so maximizing $J(\phi)$ encourages the model to explain the observed trajectories without fixing any particular skills in advance. Because the integral over ζ is usually intractable, we replace $\log p_\phi(\tau)$ by its *evidence lower bound* (ELBO), obtained by introducing an approximate posterior $q_\phi(\zeta \mid \tau)$ and applying Jensen's inequality:

$$\begin{aligned}\log p_\phi(\tau) &= \log \int q_\phi(\zeta \mid \tau)\frac{p_\phi(\tau,\zeta)}{q_\phi(\zeta \mid \tau)}d\zeta \\ &\geq \mathbb{E}_{q_\phi(\zeta|\tau)}\big[\log p_\phi(\tau,\zeta) - \log q_\phi(\zeta \mid \tau)\big] \\ &= \mathbb{E}_{q_\phi}\big[\log p_\phi(\tau \mid \zeta)\big] - D_{\mathrm{KL}}\big(q_\phi(\zeta \mid \tau)\,\|\,p_\phi(\zeta)\big).\end{aligned} \tag{74}$$

Averaging over $\tau\sim\mathcal{D}$, and introducing the β-weight[7] (Higgins et al., 2016) which balances reconstruction and regularization terms yields the training objective:

$$J_{\mathrm{ELBO}}(\phi) = \underbrace{\mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q_\phi(\zeta|\tau)}\big[\log p_\phi(\tau \mid \zeta)\big]}_{\text{reconstruction}} - \beta\,\mathbb{E}_{\tau\sim\mathcal{D}}\,\underbrace{D_{\mathrm{KL}}\big(q_\phi(\zeta \mid \tau)\,\|\,p_\phi(\zeta)\big)}_{\text{regularization}}, \tag{75}$$

which is *maximized* with respect to ϕ. The first term obliges the model to *reconstruct* each trajectory by segmenting it at boundaries b_t and encoding each segment with a latent skill vector z_t; the second term *regularizes* those encodings toward the prior, encouraging the emergence of a compact skill space. In practice, three distinct models are employed whose parameters are jointly denoted by ϕ:

- **Prior** $p_\phi(\zeta)$: defines a prior distribution over latent skill sequences. A common choice is a fixed, factorized prior (e.g., unit Gaussian for z_t and Bernoulli for b_t), but the prior can instead be endowed with learnable parameters and conditioned on the current state, yielding $p_\phi(\zeta \mid s)$ (Ajay et al., 2021; Nam et al., 2022). This can also serve as a prior on the policy over skills, $\mu_\kappa(z \mid s)$;
- **Decoder** $p_\phi(\tau \mid \zeta)$: models the probability of a trajectory conditioned on a given skill sequence. Importantly, the decoder can also be formulated as a skill-conditioned policy, $\pi_\theta(a \mid s,\zeta)$, that reconstructs only the actions in the trajectory, as done in several works (Kipf et al., 2019; Ajay et al., 2021; Pertsch et al., 2021; Nam et al., 2022). To do so, we need to adjust Equation 75 accordingly:

$$J_{\mathrm{ELBO}}(\phi,\theta) = \underbrace{\mathbb{E}_{\tau\sim\mathcal{D},\,\zeta\sim q_\phi(\zeta|\tau)}\Big[\sum_{t=1}^{T}\log\pi_\theta\big(a_t \mid s_t,\zeta\big)\Big]}_{\text{action-reconstruction}} - \beta\,\mathbb{E}_{\tau\sim\mathcal{D}}\,\underbrace{D_{\mathrm{KL}}\big(q_\phi(\zeta \mid \tau)\,\|\,p_\phi(\zeta)\big)}_{\text{regularisation}}. \tag{76}$$

7. We slighlty abuse the notation here, where usually in our work β refers to the termination function.

- **Encoder** $q_\phi(\zeta \mid \tau)$: given an observed trajectory, returns a distribution over the skill sequence that likely produced it.

Skill discovery algorithms differ in both the optimization procedure adopted for Equation 75 and the specific parameterization of latent variables. The largest group, variational-autoencoder (VAE)-based methods (Kingma and Welling, 2014), directly maximize the ELBO in Equation 75. Alternative strategies include the Expectation-Gradient framework (Fox et al., 2017; Krishnan et al., 2017) and adversarial approaches inspired by generative adversarial networks (Sharma et al., 2018), each offering distinct bias-variance trade-offs and inductive biases for learning reusable skill spaces.

The learned skills naturally support a hierarchy. In such works, a low-level controller, $\pi_\theta(a_t \mid s_t, z_t)$, is typically trained offline via behavioural cloning to execute any given skill, while a high-level policy, $\mu_\kappa(z_t \mid s_t)$, is subsequently optimized, either online (Pertsch et al., 2021; Nam et al., 2022; Salter et al., 2022), or with offline RL (Ajay et al., 2021), thereby accelerating learning efficient policies. The skills can also augment the primitive action space, expanding the agent's control repertoire (Fox et al., 2017; Kipf et al., 2019; Jiang et al., 2022), or be transformed into intrinsic reward signals to enhance long-term credit assignment (Liu et al., 2023b).

Beyond pure likelihood maximization, it is also common to add a **compression** regularizer grounded in the *minimum description length* (MDL) principle (Rissanen, 1978). MDL prefers the model that can be transmitted with the fewest total bits of (i) the model parameters and (ii) the data encoded through that model. Viewing the latent variables as skills and boundaries (Equation 72), the model parameters are the decoder (and prior if parameterized), and the data are the offline trajectories; hence, a concise skill set shortens the overall description length.

The *bits-back* coding argument (Hinton and Zemel, 1993; Honkela and Valpola, 2004; Zhang et al., 2021b) shows that maximizing the ELBO (Equation 75) approximately minimizes the description length, but with an ill-chosen prior $p(\zeta)$, the optimum can collapse to a degenerate representation (e.g., a single skill encoding that simply mirrors the observations). To avoid this, LOVE (Jiang et al., 2022) augments the ELBO with an MDL-inspired information-cost term that explicitly penalizes skills increasing the expected code length of transmitting trajectories, yielding a representation that is both *informative* (high ELBO) and *economical* (low description length):

$$\min_{\phi} \; L_{\mathrm{DL}}(\phi) \quad \text{s.t.} \quad J_{\mathrm{ELBO}}(\phi) \;\geq\; C, \tag{77}$$

$$L_{\mathrm{DL}}(\phi) \;= \mathbb{E}_{\tau \sim D,\, \{b_t, z_t\} \sim q_\phi(\cdot \mid \tau)} \Big[- \sum_{t=1}^{T} b_t \, \log \, p_z^*\big(z_t; \phi\big) \Big], \tag{78}$$

where $b_t = 1$ indicates the start of a new skill at time t, and C is a constant. The optimal prior on z, p_z^*, that minimizes the expected description length is defined by:

$$p_z^*(z; \phi) \;= \frac{\mathbb{E}_{\tau \sim D,\, \{b_t, z_t\} \sim q_\phi(\cdot \mid \tau)} \Big[\sum_{t=1}^{T} b_t \, \delta\big(z_t = z\big) \Big]}{\mathbb{E}_{\tau \sim D,\, \{b_t, z_t\} \sim q_\phi(\cdot \mid \tau)} \Big[\sum_{t=1}^{T} b_t \Big]}, \tag{79}$$

with $\delta(\cdot)$ denoting the indicator function. Intuitively, $\mathcal{L}_{\text{DL}}$ penalizes having too many skill boundaries and distinct skill choices, driving the method toward a concise skill decomposition, and longer skills encompassing common structures are generally favored to avoid the degenerate solution where each skill represents a single action. Salter et al. (2022) instead leverages the concept of *bottleneck* option by introducing a *predictability* objective, encouraging option-level transitions to be predictable. The authors show that maximizing this predictability reduces the conditional entropy and thus the optimal code length, and this objective is equivalent to applying the MDL principle.

Vlastelica et al. (2023) cast offline skill discovery as **empowerment maximization**, presented in Section 4.4, under an imitation constraint. To ensure that each skill remains faithful to the offline demonstrations, they constrain the divergence between the induced state occupancy and the state occupancy d_E from a skill-independent expert dataset, resulting in a constrained optimization problem:

$$\max_{\{\pi_z\},\, q_\phi} \; \mathbb{E}_{z\sim p(z), s\sim d_\pi}[\log q_\phi(z \mid s)] \quad \text{s.t. } D_{\text{KL}}\big(d_\pi \,\|\, d_E\big) \le \varepsilon,\ \forall z. \tag{80}$$

Here, d_π denotes the state-occupancy measure of the skill-conditioned policy, estimated in the offline setting via density-based learning. The term q_ϕ represents a skill discriminator that tightens a variational lower bound on the mutual information between skills and states. It simultaneously *diversifies* behaviours by maximizing the lower bound on the mutual information between states and skills, and *regularizes* them by penalizing departures from the expert state distribution d_E.

5.1.1 Benefits and Opportunities

Credit Assignment. By extracting hierarchical structure from offline datasets, agents can break complex trajectories into more manageable subgoals. This segmentation makes it easier to understand why certain results occur, as it allows each outcome—such as achieving a subgoal—to be more directly linked to the specific actions and conditions that produced it. In other words, by focusing on a sequence of subgoals rather than the full sequence of primitive actions, the learning algorithm can more easily attribute success or failure to specific decisions or events. For example, Kipf et al. (2019) tackle credit assignment in maze-like environments with delayed feedback. Their method infers segment boundaries $q(b_i \mid s, a)$, with $b_i \in [1, T+1]$ functioning similarly to option termination functions, and encodes each segment using $q(z_i \mid s, a)$ as latent skill (subgoal) descriptors. This segmentation captures subgoal structure, facilitating effective credit assignment across subgoals when applying the skills in sparse-reward settings. Similarly, Kim et al. (2019) improve credit assignment in goal-oriented navigation tasks by decomposing action-free trajectories into subsequences by inferring skill descriptors z_t and binary termination signals $b_t \in \{0, 1\}$.

Transfer. By discovering skills from offline datasets, agents develop a foundational set of versatile competencies. These competencies can then be transferred to new tasks with minimal adjustment. Pertsch et al. (2021) show promising results on transferring skills obtained in the offline dataset to more complex simulated robotic tasks unseen in the dataset (e.g., maze navigation with larger maps in evaluation). Similarly, Jiang et al. (2022) and Salter et al. (2022) show that by optimizing a compression objective, in addition to the

reconstruction one, the discovered skills help transfer across multiple tasks. Nam et al. (2022) demonstrate that by meta-training a high-level policy, $\pi_\theta(z_t \mid s_t, e)$, where e is a task encoding, and executing a low-level policy, $\pi_\theta(a_t \mid s_t, z_t)$, which is learned via behavioural cloning, the agent can solve a wide range of new tasks in a meta-RL setting.

Exploration. When reusing the offline discovered skills for online interaction, this can reduce the difficulty of exploration since the agent can quickly apply well-tested, pre-learned behaviours rather than learning them through trial-and-error in real-time. Fox et al. (2017) show promising exploration results in a simple four-room domain by augmenting the action space with discovered parameterized options. Salter et al. (2022) show that the learned temporally compressed bottleneck options are beneficial for exploration in maze-like environments with delayed rewards.

Avoiding Distributional Shift. In offline RL, distributional shift describes the discrepancy between the action distribution present in the training dataset and the actions chosen by the policy during evaluation or deployment. This occurs when the policy selects actions that are rarely or never observed in the dataset, leading to unreliable value estimates. Ajay et al. (2021) leverage offline skill discovery to mitigate this issue. Their approach encodes short trajectories (e.g., every K steps) from the dataset into a skill descriptor z. By maximizing the log-likelihood of actions in trajectories τ, conditioned on states s_t and skill descriptors z, the method captures recurring behaviours present in the data. The offline dataset can then be enhanced using z, and a high-level policy, $\pi_\theta(z \mid s)$, can be derived by off-the-shelf offline RL algorithms. The authors show that such a temporal structure reduces compounding errors for extrapolating out-of-distribution actions in offline RL.

Opportunities for Research.

- **Optimization challenges.** Evident in some studies (Jiang et al., 2022), optimization challenges can lead to degraded skill quality if the learning dynamics are not carefully managed. Additionally, the under-utilization of reward signals in existing datasets creates an opportunity to further refine learned skills, and incorporating offline RL methods—rather than relying solely on reconstruction-based approaches—into HRL may unlock greater performance gains, as Hu and Leung (2023) provide provably positive results on sample efficiency.

- **Broader scope of test environments.** Additionally, many methods in this field primarily validate their concepts on simulated robotic navigation tasks, which typically involve deterministic transitions and rewards (Gao et al., 2024), and often favor specific inductive biases. A natural extension would be scaling this paradigm to real-world, image-based tasks, or other practical applications with different properties.

5.2 Hindsight Subgoal Relabeling

In Section 5.1, we discussed the methods that automatically infer the skill descriptors, usually characterized by latent variables. In this section, we explore methods that identify and relabel subgoals within an offline dataset, effectively leveraging existing transitions to learn how to achieve subgoals (Kaelbling, 1993b). Offline experiences offer valuable insights into identifying subgoals, which can be viewed as milestones or waypoints for accomplishing

a task (Gupta et al., 2019; Park et al., 2024a), or abstracting bottleneck states to make a good partition of the state space (Paul et al., 2019). This is conceptually similar to hindsight experience replay (Andrychowicz et al., 2017) we discussed in Section 4.8, which relabels the final or intermediate states reached in a trajectory as if they were the intended goals.

In an examplar work by Paul et al. (2019), a reward-free trajectory dataset, $\mathcal{D} = \{(s_1^{(i)}, a_1^{(i)}, \ldots, s_{n_i}^{(i)})\}_{i=1}^{n_d}$, is segmented into an *ordered* list of n_g disjoint partitions, $G = \{1, \ldots, n_g\}$, that serve as subgoals.

Initial labeling. Each trajectory is equipartitioned by assigning consecutive subgoal indices:

$$g_t^{(i)} = j \quad \text{iff} \quad \left\lfloor \tfrac{(j-1)n_i}{n_g} \right\rfloor < t \leq \left\lfloor \tfrac{j\,n_i}{n_g} \right\rfloor,\ j \in G. \tag{81}$$

Iterative refinement. Repeat the following steps until the label change falls below a threshold. Alternating these steps until convergence yields a classifier μ_κ that both partitions the state space and respects the required ordering. Such a $\mu_\kappa(g \mid s)$ can provide information on whether a state is a milestone or bottleneck.

- *Learning step*: fit a classifier $\mu_\kappa(g \mid s)$ by cross-entropy on the current labels:

$$L(\kappa) \;=\; \mathbb{E}_{(s,g)\sim D}\big[-\log \mu_\kappa(g \mid s)\big]. \tag{82}$$

- *Inference step*: enforce the trajectory order $1 \prec 2 \prec \ldots \prec n_g$ with Dynamic Time Warping[8] (Müller, 2007) over the posterior sequence $\langle \mu_\kappa(\,\cdot \mid s_t)\rangle$.

Potential function and intrinsic reward. The most probable class defines a potential $\Phi_\kappa(s) = \arg\max_g \mu_\kappa(g \mid s)$, and an intrinsic reward r' can be defined as:

$$r'(s, a, s') \;=\; \gamma\,\Phi_\kappa(s') - \Phi_\kappa(s), \tag{83}$$

where γ is the discount factor of the MDP.

Learning schedule. The policy π_θ is first initialized through behaviour cloning, after which reinforcement learning proceeds with the augmented reward $r + r'$. This phase exploits subgoal guidance to supply dense progress signals without additional expert interaction and still preserves the original optimum.

As another example, Relay Policy Learning (RPL) (Gupta et al., 2019) relabels demonstration trajectories $\tau = (s_0, a_0, \ldots, s_T)$ with *relay* subgoals, producing two goal-augmented datasets:

$$\mathcal{D}_\ell = \big\{\, (s_t, a_t, g_\ell) \;\big|\; g_\ell = s_{t+w},\ 0 \leq t < T,\ 1 \leq w \leq W_\ell,\ t + w \leq T \,\big\}, \tag{84}$$

$$\mathcal{D}_h = \big\{\, (s_t, g_\ell, g_h) \;\big|\; g_\ell = s_{t+\min(W_\ell, w)},\ g_h = s_{t+w},\ 0 \leq t < T,\ 1 \leq w \leq W_h,\ t + w \leq T \,\big\}. \tag{85}$$

$\mathcal{D}_\ell$ offers *short-horizon* examples (subgoal horizon $\leq W_\ell$) for training a low-level controller $\pi_\theta(a \mid s, g_\ell)$ to reach nearby states, whereas $\mathcal{D}_h$ pairs each *long-horizon* target g_h (up to

8. An algorithm that aligns two sequences by allowing non-uniform time shifts, so that similar patterns are matched even if they occur at different timesteps or speeds.

W_h steps away) with a feasible intermediate subgoal g_ℓ, enabling hierarchical planning by a high-level goal-setter $\mu_\kappa(g_\ell \mid s, g_h)$. The imitation objective is:

$$\max_{\kappa,\theta} \mathbb{E}_{(s,a,g_\ell)\sim\mathcal{D}_\ell}\big[\log \pi_\theta(a \mid s, g_\ell)\big] \; + \; \mathbb{E}_{(s,g'_\ell,g_h)\sim\mathcal{D}_h}\big[\log \mu_\kappa(g'_\ell \mid s, g_h)\big],$$

with $W_\ell = 30$ and $W_h = 260$ in all experiments of RPL. During execution, every W_ℓ steps the high-level policy samples a new subgoal $g_\ell \sim \mu_\kappa(\cdot \mid s, g_h)$ and the low-level controller tracks it step-by-step until the next subgoal is issued.

5.2.1 Benefits and Opportunities

Credit Assignment. Subgoal relabeling decomposes complex tasks into manageable intermediate objectives by highlighting which actions contribute to reaching the final goal.

Paul et al. (2019) address credit assignment by constructing an intrinsic reward based on a subgoal policy $\mu_\kappa(g \mid s)$, which identifies a state's progress toward a final goal. This classifier is trained via an EM-style procedure that enforces an ordering constraint over subgoal indices, ensuring that states later in a demonstration receive higher labels. The resulting potential function, $\Phi_\kappa(s) = \arg\max_g \mu_\kappa(g \mid s)$, induces a shaped reward (Equation 83) which provides dense feedback aligned with behavioural progress. This intrinsic signal facilitates temporal credit assignment by rewarding transitions that advance the agent through the learned subgoal structure, even when extrinsic rewards are sparse or delayed.

Alternatively, RPL (Gupta et al., 2019) addresses the credit assignment challenge from sparse, delayed rewards by relabeling demonstrations with overlapping sliding-window subgoals. It trains a goal-conditioned low-level policy to reach short-horizon targets, while a high-level policy selects subgoals. This hierarchical structure ensures that each low-level episode ends with an intrinsic success signal. As a result, external rewards propagate after at most one window. This transforms the long-horizon problem into a sequence of locally supervised updates, enabling faster and more stable credit assignment than flat or single-level baselines.

Similarly, Park et al. (2024a) relabel subgoals as the state s_{t+k} that lies exactly k steps ahead of the current state s_t. A high-level policy, $\mu_\kappa\big(s_{t+k} \mid s_t, g\big)$, proposes such waypoints conditioned on the ultimate goal g, while a low-level policy, $\pi_\theta\big(a_t \mid s_t, s_{t+k}\big)$, outputs primitive actions that move the agent toward the subgoal. Both policies are optimized via a shared goal-conditioned value function V_ψ: μ_κ maximizes $V_\psi(s_{t+k}, g)$, whereas π_θ maximizes $V_\psi(s_{t+1}, s_{t+k})$. Specifically, μ_κ is trained to choose subgoals s_{t+k} that maximize $V_\psi(s_{t+k}, g)$, while π_θ is trained to select actions that make the next state s_{t+1} have high value $V_\psi(s_{t+1}, s_{t+k})$ relative to the current subgoal. Because different subgoals induce much larger variations in V_ψ than individual actions, the high-level receives a more reliable learning signal, and since π_θ queries V_ψ only for nearby states where estimates are more accurate, the entire hierarchy is less susceptible to noise and approximation errors in the value function, resulting in a more robust policy and credit assignment.

Interpretability. Identifying subgoals within the offline dataset can provide insights into understanding the decision-making process. For example, Paul et al. (2019) present visualizations of the state space in robotic navigation tasks, such as AntMaze and AntTarget, demonstrating that the state space can be structurally partitioned using discovered subgoals. The structural decomposition is intuitively meaningful to humans, facilitating better understanding and verification.

Opportunities for Research.

- **Enhancing interpretability of decision making with interpretable subgoals.** Although positive empirical result is shown in (Paul et al., 2019), current offline relabeling schemes select subgoals with limited transparency into why particular states are chosen or how they steer the learned policy. Embedding interpretability or alignment objectives, such as attributing subgoal selection to human-understandable criteria, would not only clarify the decision rationale, but also foster trust and diagnosability.
- **Scaling to complex observations by identifying latent subgoals.** Researchers can extend offline subgoal relabeling to environments with high-dimensional inputs, such as images, language, or tactile data, with Park et al. (2024a) as an example. To do so, methods could identify subgoal representations in some latent space that pinpoint meaningful milestones within these spaces. By focusing on compact embeddings, relabeling can remain effective even when raw observations are noisy, partial, or multimodal.

6. Discovery with Foundation Models

Agents that learn skills from scratch through environment interactions are directly exposed to the inherent complexities of the domains in which they operate. Such agents must learn from their stream of experience how to organize the collected data into meaningful chunks in order to derive a useful set of skills. To mitigate these challenges, we can instead build on prior knowledge contained in large pretrained models to guide the discovery of useful skills in complex environments. The starting assumption for such methods is that pretrained models contain knowledge about the environment of interest. Although this assumption may not always hold, it is likely applicable to many domains of interest and will grow as the training paradigm of LLMs expands in scope.

Simultaneously, an interesting feature of LLM-based methods is that, as these large models are based on human priors and are instantiated through natural language, the set of behaviours will generally be more interpretable. In fact, leveraging language, without using LLMs, has produced a prolific line of work (Shu et al., 2018; Bahdanau et al., 2018; Fu et al., 2019; Jiang et al., 2019; Colas et al., 2020a,b; Akakzia et al., 2021).[9] These works underscore, amongst other features, the compositional nature of language. This quality makes it a particularly useful space to represent a variety of goals.

It is therefore natural to consider LLMs for HRL as they provide both useful inductive biases from pre-training on human data and a meaningful abstraction space through natural language. This connection is reinforced by the fact that **LLMs, by their very nature, can represent goal-conditioned policies, where goals are specified linguistically**.

As such, many recent works leverage LLMs to decompose tasks into subtasks (Pignatelli et al., 2024; Yang et al., 2024; Wang et al., 2024c), an operation done according to their pre-existing understanding of the task's underlying structure. Another perspective is to use LLMs as a measure of interestingness to propose a curriculum of goals and tasks in open-ended domains (Colas et al., 2023; Zhang et al., 2024; Faldor et al., 2025; Wang et al., 2024b; Zala et al., 2024). The key to converting an LLM's latent knowledge into a functional agent lies in efficiently learning the options required to execute the decomposed goals.

9. For an in-depth review of the use of language in RL, please refer to the work by Luketina et al. (2019).

In this section, we investigate four families of methods that propose solutions to this problem. The first family consists of methods using embeddings from large pretrained models as representations from which option rewards are defined. Next, we present methods that use large pretrained models to provide feedback, in the form of rewards or preferences, for learning different skills. Building on the code generation capabilities of large models, we present two families of methods that write code to either craft reward functions to learn specific behaviours. Finally, as LLMs can be seen as goal-conditioned policies, we cover methods that use them directly to specify goals and achieve them.

Not all the presented methods in this section are hierarchical by nature. For example, some papers focus on defining rewards or policies for a set of tasks, rather than a set of options. However, given the promising potential for such methods to drive progress for HRL, we include them as well.

6.1 Embedding Similarity

As foundation models are trained on Internet-scale datasets, their embeddings contain useful structure for a variety of tasks. Such embeddings can be the result of contrastive pretraining on image and text pairs, for instance, the Contrastive Language-Image Pretraining (CLIP) encoder (Radford et al., 2021). Let $\mathbf{w}_i \in \mathbb{R}^d$ represent the normalized feature vector (embedding) generated by the image encoder for the i-th image, I_i. Similarly, let $\mathbf{u}_j \in \mathbb{R}^d$ be the normalized feature vector generated by the text encoder for the j-th text, T_j. The similarity between image i and text j is computed using the cosine similarity (which simplifies to the dot product for normalized vectors):

$$C_{ij} = \mathbf{w}_i^\top \mathbf{u}_j. \tag{86}$$

These embeddings can then be used to represent image or language goals and define reward functions by taking the cosine similarity between the embeddings of the goal and the observation in which the agent is currently situated,

$$r^g(s) = \mathbf{w}(s)^\top \mathbf{u}(g). \tag{87}$$

This reward function is then maximized by a goal-conditioned policy interacting with an environment to learn behaviours that achieve the specified goals.

To obtain these vectors, the objective is formulated as minimizing a cross-entropy loss, applied symmetrically for both image-to-text and text-to-image prediction tasks. The loss for predicting the correct text caption for a given image i (considering all N text captions in the batch) is defined as:

$$L_{\text{image}_i} = -\log \frac{\exp(c_{ii}/e)}{\sum_{j=1}^{N} \exp(c_{ij}/e)}, \tag{88}$$

where e is the temperature hyperparameter. The loss for predicting the correct image for a given text caption i (considering all N images in the batch) is:

$$L_{\text{text}_i} = -\log \frac{\exp(c_{ii}/e)}{\sum_{j=1}^{N} \exp(c_{ji}/e)}. \tag{89}$$

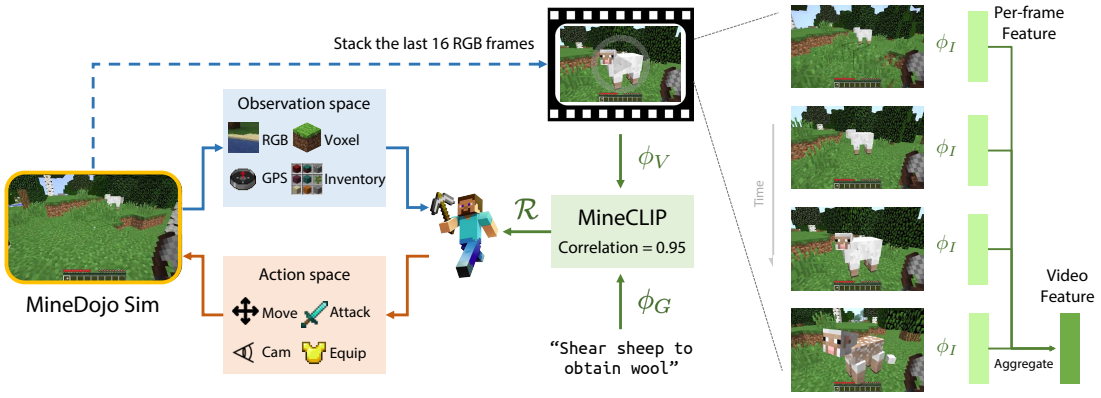


Figure 12: Illustration of the method of embedding similarity for defining option reward functions. Visual observations and language goal descriptions are converted into embeddings, and their similarity (e.g., via MineCLIP) is used to define reward functions for goal-conditioned policies. In this example, the agent is rewarded for successfully performing sheep shearing. Figure taken from Fan et al. (2022).

Fan et al. (2022) instantiate this idea in the open-ended Minecraft game (Johnson et al., 2016; Kanervisto et al., 2021). To do so, they introduce the MineDojo framework. The authors collect a large dataset of Minecraft gameplay for training a reward function that would map textual goals and a sequence of observations to a scalar value indicating their similarity. The language goal is encoded through the pretrained CLIP encoder (Radford et al., 2021) whereas the video encoder is composed of an image encoder and a temporal aggregator that accumulates 16 consecutive frames from the video. This leads to the following non-Markovian reward,

$$r^g(s_{t-16:t}) = \mathbf{w}(s_{t-16:t})^\top \mathbf{u}(g). \tag{90}$$

The authors train their reward model, called MineCLIP, on the aforementioned dataset using the same losses as in Equation 88 and 89. This resulting reward function excels at capturing correct behaviour on a wide collection of tasks, such as "Combat zombie". Lifshitz et al. (2023) build on this work to obtain an instruction-following agent in Minecraft, where language instructions represent goals.

CLIP-based methods have also been applied to robotics. Xiao et al. (2022) fine-tune the CLIP model on a small dataset of robotic tasks and then utilize the model to label, using a set of predefined annotations, a much larger dataset of unlabeled observations. Using this larger dataset, the authors then train language-conditioned policies to achieve goals through imitation learning. Further improving the sample efficiency of embedding-based methods, Palo et al. (2023) show the possibility of efficiently fine-tuning the same CLIP model on as little as 1000 data points. Avoiding the costly operation of fine-tuning large pretrained models, Cui et al. (2022) investigate the prospect of using the CLIP model in a zero-shot fashion for defining goal-conditioned policies, obtaining good results on robotics tasks. Similarly, Rocamonde et al. (2024) leverage a fixed pretrained CLIP model and study the scaling effect of such models on the resulting RL performance.

6.1.1 Benefits and Opportunities

Exploration. Methods building on embedding-based rewards empirically show improved exploration in complex tasks. In particular, in open-ended environments such as Minecraft, the dense nature of the reward functions obtained from embedding similarity significantly helps with exploration, leading to sophisticated behaviour (Fan et al., 2022; Lifshitz et al., 2023). The dense nature of such reward functions is also particularly useful for approaches studying the challenge of robotics (Xiao et al., 2022; Fu et al., 2024) and web navigation (Baumli et al., 2023). Du et al. (2023c) investigate how guiding exploration with an LLM during a pretraining phase can help an agent's downstream performance. To do so, the authors introduce the idea of restricting the reward function through a similar threshold,

$$r^{g,T}(s_t, a_t, s_{t+1}) = \begin{cases} r^g(s_t, a_t, s_{t+1}) & \text{if } > T, \\ 0 & \text{otherwise }, \end{cases} \tag{91}$$

where T, the threshold, is a hyperparameter. This reduces the noise of possibly imperfect embeddings used to define the reward function, further improving exploration.

Transfer. Another important benefit from LLM-based approaches to skill discovery stems from the compositional nature of language, which easily allows for specifying a variety of goals. For example, Du et al. (2023a) study how pretraining the agent on self-generated goals, where good behaviour is rewarded by the embeddings of an LLM, can lead to improved downstream performance on a set of complex goals. To encourage reaching a diversity of goals that will transfer well, the authors additionally prompt the LLM to generate k goals and reward the agent on the goal with the greatest reward,

$$r^{g_{max}} = \max_{i=1...k} r^{g_i,T}(s_t, a_t, s_{t+1}). \tag{92}$$

Similar generalization to different language-conditioned goals is reported by Lifshitz et al. (2023). Instead of directly training with a goal-conditioned model, Mahmoudieh et al. (2022) efficiently train a discrete set of smaller policies, used as a basis of behaviour. This is then distilled into a single language-conditioned neural network, which can better generalize on a larger spectrum of behaviours than the basis.

Opportunities for Research.

- **Understanding the trade-offs of different embeddings.** An important question when working with embedding similarity measures is with respect to the origin of the embeddings themselves. Most of the presented papers rely on CLIP, but other embeddings have been used, such as the Bidirectional Encoder Representations (BERT) embeddings (Devlin et al., 2019) and the Reusable Representations for Robot Manipulation (R3M) Adeniji et al. (2023), which is pretrained on the Ego4D dataset (Grauman et al., 2021) through a combination of contrastive and video-language alignment losses. When considering a wide range of tasks, it is not clear which model shows greater performance, or is more amenable to fine-tuning.
- **Expanding beyond text-image similarity.** Most works compute the similarity between a language goal instruction and the current observation. Sontakke et al. (2023)

instead compute the similarity between an agent attempting to reach a goal and a demonstration of such a successful behaviour. Moreover, contrary to most works, the authors compute the reward at the trajectory level, that is, the reward is only given at the end of an interaction. The authors show that their approach can be applied even in the case where the demonstration is done by a human physically completing the task, rather than teleoperating a robot, which presents greater opportunities for generalization.

6.2 Providing Feedback

Leveraging the embeddings of foundation models to measure the similarity between a desired goal and the current state places significant emphasis on the quality of the embeddings themselves. One way to avoid this is by considering the auto-regressive nature of LLMs, which allows for chain-of-thought (Wei et al., 2022) and in-context learning (Brown et al., 2020). Such capabilities can be particularly useful to define option reward functions. This can be done by taking as input a state, or a trajectory, as well as a goal description, and using LLMs to output a scalar feedback, representing the degree of success with respect to the goal. Alternatively, preferences can be elicited from an LLM over pairs of states that are then converted into a reward model through preference-based learning (Wirth et al., 2017).

Direct Reward

To obtain a success measure, Du et al. (2023a) combine a sequence of observations from the environment together with a question such as "Did the agent successfully place the cactus left of the sofa?" to query a multimodal model (Alayrac et al., 2022) for a binary answer. Formaly, $\texttt{LLM} : \mathcal{S} \times \mathcal{G} \to \mathcal{Y}$ where the goal g is represented by the question and $y \in \{0, 1\}$. The goal reward is then defined through this binary output,

$$r^g(s) = y = \texttt{LLM}(s, g). \tag{93}$$

Such reward functions are evaluated on a diversity of domains: embodied simulations (Abramson et al., 2021), robotic manipulation with a 6DoF device, and human interactions in the Ego4D dataset (Grauman et al., 2021). To obtain accurate success measures, the authors have to initially fine-tune the model on a large dataset of expert interactions. Instead of costly model updates, Kwon et al. (2023a) propose to replace weight updates with few-shot in-context examples, building on improved learning capabilities in the employed LLM. Pan et al. (2024) show that measures of success obtained from a multimodal LLM have high agreement (up to 92.9%) with oracle evaluators. Such results are reported on WebArena (Zhou et al., 2024) and Android-in-the-Wild (Rawles et al., 2023) benchmarks. Leveraging the strong performance of LLMs as direct reward modelers, Bai et al. (2024) successfully train robust RL policies on a variety of goals derived from changing web interfaces.

Eliciting Preferences

When an LLM's output directly functions as the reward signal, it often lacks the granularity to effectively measure the relative merit of a specific state against the full spectrum of alternatives. Instead, we can leverage the idea of reinforcement learning from AI feedback

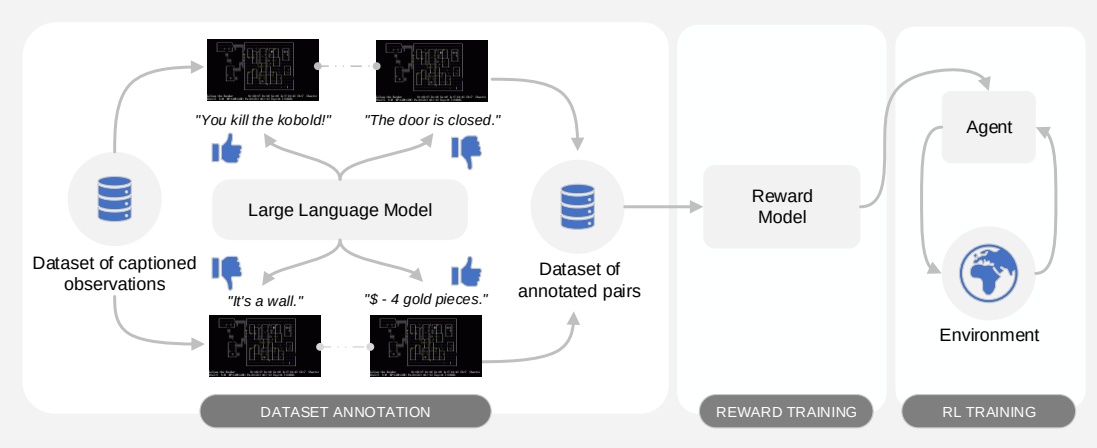


Figure 13: Learning option rewards from AI feedback proceeds in three phases. In the first phase, an LLM is conditioned on a behaviour description and queried for preferences over pairs of observations, which are stored with their preference labels within a dataset. In the second phase, the preferences are distilled into an observation-based scalar reward function. Finally, an agent is trained interactively with RL, receiving a scalar signal at every step through the reward function extracted from the preferences.

(Bai et al., 2022), introduced in the context of fine-tuning large models and relying on preference-based learning (Wirth et al., 2017; Thomaz et al., 2006). Building on this idea, Klissarov et al. (2024) introduce the Motif algorithm, which leverages an LLM's feedback to guide an agent acting in the open-ended NetHack environment (Küttler et al., 2020). Observations from the environment are presented to an LLM before querying, using chain-of-thought prompting, to provide a preference over which observation is more desirable for a certain goal.

Formally, the annotation function is given by $\texttt{LLM} : \mathcal{S} \times \mathcal{S} \times \mathcal{G} \to \mathcal{F}$, where $\mathcal{S}$ is the space of states, where $\mathcal{G}$ is the space of goals defined through natural language, and $\mathcal{F} = \{1, 2, \varnothing\}$ is a space of preferences for either the first, the second, or none of the captions. These preferences are then distilled into a reward function through the Bradley-Terry model (Bradley and Terry, 1952) and given to an RL agent interacting with the environment,

$$\mathcal{L}(\nu) = -\mathbb{E}_{(s_1,s_2,g,y)\sim\mathcal{D}_{\text{pref}}}\Big[\mathbb{1}[y=1]\log P_\nu[s_1 \succ s_2|g] + \mathbb{1}[y=2]\log P_\nu[s_2 \succ s_1|g] \\ + \mathbb{1}[y=\varnothing]\log\Big(\sqrt{P_\nu[s_1 \succ s_2|g]\cdot P_\nu[s_2 \succ s_1|g]}\Big)\Big], \tag{94}$$

where $P_\nu[s_a \succ s_b|g] = \frac{e^{r^g_\nu(s_a)}}{e^{r^g_\nu(s_a)}+e^{r^g_\nu(s_b)}}$ is the probability of preferring a state s_a to another s_b given a goal g; r^g_ν is the reward defined with respect to the goal specified in the LLM's prompt.

Through the process of comparing states to alternatives, eliciting LLM preferences, or receiving AI feedback, nuanced and fine-grained reward functions can be provided. Such

reward functions can also be understood as process-based rewards (Uesato et al., 2023; Lightman et al., 2023). Klissarov et al. (2024) leverage this characteristic to learn a set of policies that exhibit a certain behaviour across time, such as preferring generally more cautious strategies when exploring. This is in contrast to the work on LLM as direct reward modelers, which typically define rewards for reaching goal states as binary success detectors (Du et al., 2023a). As illustrated in the MaestroMotif algorithm (Klissarov et al., 2025a), the flexibility offered by AI feedback is key in designing HRL agents capable of subtle behaviours and fast adaptation. Adding to the generality of AI feedback, Wang et al. (2024a) investigate the resulting policies across a range of continuous control domains using pixel observations and a multimodal LLM. Their findings show that reward functions generated through AI feedback yield more performant policies compared to embedding similarity approaches or methods that directly query the LLM for scalar rewards.

6.2.1 Benefits and Opportunities

Exploration. Klissarov et al. (2024) illustrate the potential of AI feedback-based rewards to significantly improve exploration on the complex open-ended world of NetHack. The obtained reward function is shown to be naturally dense and encodes a variety of important milestones, such as unlocking doors or picking up items. The authors hypothesize that, by querying the model on thousands of pairs of observations from the environment, the LLM's common sense reasoning and domain knowledge are distilled into a useful reward function.

Credit Assignment. Wang et al. (2024a) report that the reward obtained from preferences monotonically increases as the agent advances towards the goal, naturally assigning credit to states in between the starting state and the goal. Klissarov et al. (2025b) further study the dense nature of such reward functions and reveal a strong correlation with value functions obtained at the end of training. As such, value functions have been trained to propagate information through temporal difference learning (Sutton, 1988), the authors argue that this high correlation is another indication that the reward functions based on LLM feedback are useful for credit assignment. An equivalent perspective is that the resulting dense reward can be seen as a form of reward redistribution (Arjona-Medina et al., 2018; Hung et al., 2019; Klissarov and Precup, 2020; Ni et al., 2023), which is an established method for improving credit assignment.

Transfer. In MaestroMotif, Klissarov et al. (2025a) show how a set of semantically meaningful skills can be easily re-composed zero-shot to adapt to complex new tasks. Leveraging the code generation abilities of LLMs, they propose a neuro-symbolic approach where skill policies are neural networks trained by reinforcement learning, and the high-level policy is defined through code. The authors then use the in-context learning abilities of LLMs to re-compose the skills, significantly outperforming baselines that are trained specifically on each of the tasks. Their approach highlights how the compositional nature of language can be particularly helpful when combined with a set of linguistically-defined skills, leading to an easily promptable agent.

Opportunities for Research.

- **Simplifying the reward learning process.** Despite the strength of preference-based methods for crafting rewards through LLMs, they are more complex than directly

querying for a reward signal. Is there a way to leverage the improved exploration and credit assignment without the additional complexity? Is an existing dataset of observations needed for eliciting useful preferences? Zheng et al. (2024) provide an initial answer to these questions by comparing different ways in which the LLM feedback is leveraged, for example, by using it as a label for a classification loss. Their results show surprisingly strong performance of some of these simpler baselines, even when querying the LLM with online interactions.

6.3 Reward as Code

Instead of relying on LLMs to evaluate good and bad behaviour from observations, it is possible to rely on their code generation abilities to craft helpful rewards. In this line of work, a goal description is given to the LLM as input, as well as additional information from the environment,

$$\texttt{code}^g \sim \texttt{LLM}(g, \text{info}), \tag{95}$$

$$r^g(s) = \texttt{code}^g(s). \tag{96}$$

This additional information often constitutes important symbolic information, such as low-level features, that is used to define the code. This code is then executed alongside the environment simulation to generate a reward for every state s. Xie et al. (2024) explore the possibility of leveraging an LLM's capacity to code reward functions for robotics tasks. The authors provide the LLM with additional information in the form of a symbolic representation of the environment (e.g., Python classes describing each object and methods to access specific information about it). Furthermore, the authors provide the LLM with helpful functions from different packages (such as quaternion computation in NumPy) to be used for reward generation. Finally, their algorithms also allow for integrating human feedback. Yu et al. (2023) similarly investigate how LLMs can generate reward functions for learning robotics skills. In their approach, an LLM takes as input a detailed language description of a goal and instantiates a set of reward functions.

Another notable work is that of Ma et al. (2024), which presents Evolution-driven Universal REward Kit for Agent (EUREKA). They provide a task description to the LLM, such as "make the pen spin to a target orientation", and proceed to do an evolutionary search on the reward function space. This process is supported through additional context given to the LLM in the form of selected parts of the source code of the environment. For each candidate reward function that the LLM generates, a complete learning run through massively distributed RL experiments using IsaacGym (Makoviychuk et al., 2021). The most promising reward function candidates are then retained and given to the LLM together with the learning statistics, such that the model performs in-context learning and suggests a new batch of candidates.

6.3.1 Benefits and Opportunities

Transfer. The ability to efficiently generate reward functions, without human supervision, is particularly important for transfer. For example, Ma et al. (2024) achieve super-human level reward design for complex robotics skills across a variety of embodiments. In the

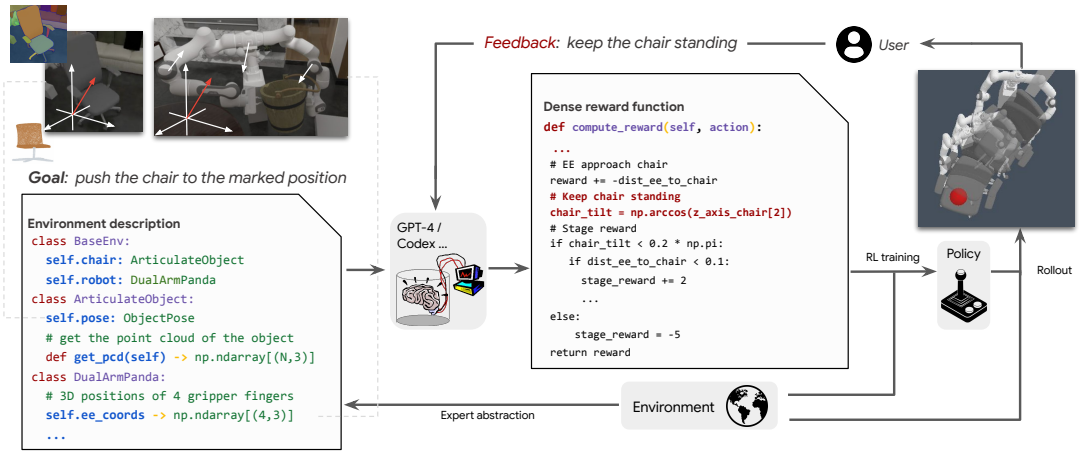


Figure 14: Defining reward functions as code requires access to a symbolic representation of the environment. This is done through an *expert abstraction* function that represents the environment as a hierarchy of Pythonic classes. The *user instruction* describes, in natural language, the goal to be achieved. The agent then interacts with the environment to maximize this symbolic reward function. It is also possible to include *user feedback* that summarizes the failure modes of the current reward code. Figure taken from Xie et al. (2024).

domain of Minecraft, Li et al. (2023) show how reward functions as code can be used to solve a variety of long-horizon goals given access to the symbolic features from the environment.

Opportunities for Research.

- **Going beyond symbolic representations.** Generating a reward function as code is a powerful paradigm: it avoids the need to query the LLM during the RL phase and does not require learning a parametric reward model. However, by definition, such an approach requires access to symbolic features from the domain of interest, which can be limiting if this involves real-world interactions with humans. Venuto et al. (2024) propose to query the LLM to craft its own symbolic representation from high-dimensional observations, similar to the work by Palo and Johns (2024). These representations are then used to define reward functions in code. However, their approach requires access to expert demonstrations, which future work could alleviate.

6.4 Directly Modeling the Policy

So far, we have covered methods that leverage foundation models to define goal reward functions through a variety of strategies, such that goal-conditioned policies can be obtained by maximizing the reward functions. Alternatively, there exists a line of work that uses LLMs to directly model the policy itself, where goals are defined through prompts and conditioning the LLM on them, effectively serving as goal-conditioned policies. In this setting, the LLM is oftentimes updated through in-context learning (Wei et al., 2022) to obtain policy improvements, bypassing the need for performing parameter updates, which can be costly and time-consuming. Building on the code generation capabilities of LLMs,

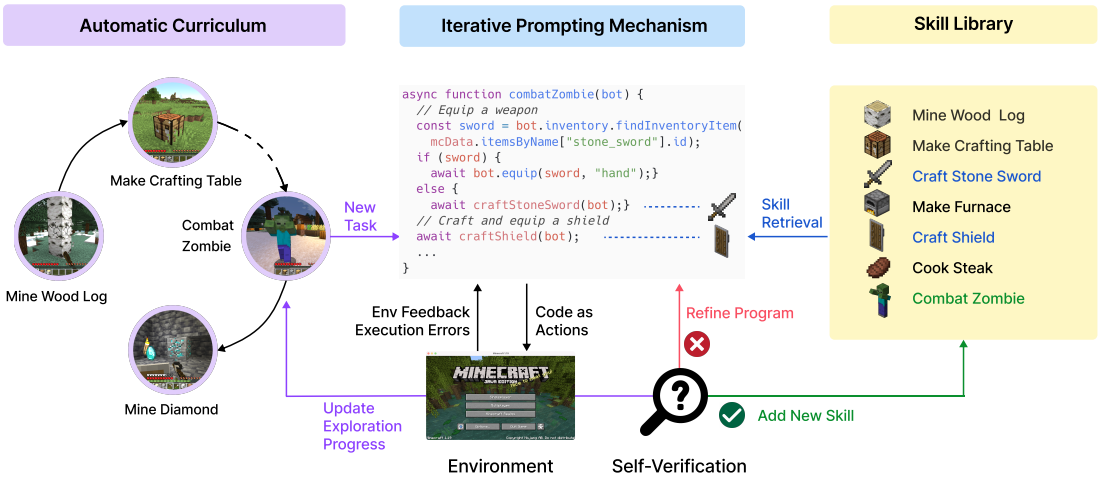


Figure 15: An LLM is conditioned on a goal description and generates snippets of code which instantiate skill policies. When interacting in multimodal environments, such as Minecraft, a bridge between this symbolic skill policy representation and the high-dimensional nature of the environment has to be present. Under such a setting, an LLM can act as an HRL agent, efficiently achieving complex goals. Figure taken from Wang et al. (2023a).

Liang et al. (2022) propose to define robotic skills as policies in the form of Python code,

$$\texttt{code}^g \sim \texttt{LLM}(g), \tag{97}$$

$$a \sim \texttt{code}^g(s), \tag{98}$$

where the goal-conditioned $\texttt{code}^g$ acts as a the goal-conditioned policy $\pi(a|s,g)$. They show that the LLM can re-compose calls to an API such that a new code policy achieves a specific goal. In particular, they propose hierarchical code-generation that recursively defines undefined functions from existing functions, leading to strong performance on robotics tasks. Kwon et al. (2023b) extend this work, removing assumptions such as providing in-context examples or requiring the LLM to predict end-effector poses. In the complex open-ended environment of Minecraft, Wang et al. (2023a) propose Voyager, a method leveraging an LLM to continually expand a library of skills. Such skills are crafted by prompting the LLM to define specific behaviours in code, building on an existing JavaScript API (PrismarineJS, 2013) that allows for grounding the generated code in the multimodality of Minecraft. Voyager further uses ideas of auto-curriculum and self-reflection to update the set of skills, learn new ones, or define their composition for a given task.

Another line of work directly queries the LLM for actions by giving as context the natural language description of a goal and the current state,

$$a \sim \texttt{LLM}(s, g). \tag{99}$$

In these settings, LLMs output the low-level actions in an environment, effectively as a goal-conditioned policy. This particular instantiation highlights that LLMs are already particularly effective HRL agents that can be conditioned on goal descriptions. The current

focus of such models is on computer-based tasks (Anthropic, 2024; OpenAI, 2025). Despite the appeal of generalizing zero-shot to new language instructions, current LLMs are still quite limited in successfully performing long-horizon tasks by directly selecting low-level actions (OpenAI, 2024; Zhou et al., 2024).

6.4.1 Benefits and Opportunities

Exploration. By relying on an LLM's common sense, prior knowledge, and possible API libraries, researchers have shown that agents explore their environment significantly better. By directly modeling the policy, it is possible to condition the LLM on a wider variety of goals and find well-performing policies for a subset of easier goals. This allows for making progress on very hard exploration problems by breaking the task into achievable milestones. Examples include collecting diamonds in Minecraft (Wang et al., 2023a) or intricate web navigation tasks (Zhou et al., 2024). By conditioning an LLM on language and directly outputting a sequence of actions to achieve tasks, agents achieve tasks that would be extremely difficult, or even impossible, to learn by maximizing a reward function.

Transfer. Directly acting with an LLM greatly simplifies how users can leverage the compositional nature of language. For example, the same LLM can be directly conditioned on a variety of computer interaction tasks and achieve them zero-shot (Anthropic, 2024). Alternatively, a library of skills can be re-composed through in-context learning to craft new skills (Wang et al., 2023a).

Opportunities for Research.

- **Lifting restrictions on the action space and action frequency.** The prospect of directly generating a wide spectrum of behaviours simply by querying a large pretrained model is particularly appealing. It essentially encompasses the fundamental promise of HRL for fast adaptation thanks to the compositional nature of language. However, it also poses interesting challenges. For example, would such a model be restricted to a certain action space, or is there a way to efficiently adapt to a variety of embodiments? Are there limitations in terms of action frequencies? The domain of computer navigation is especially promising as grounding an LLM in the action space of computers would allow a model to achieve many economically useful tasks. However, the same model could not be used to control an embodied robot, unless fine-tuning is performed, which for large models is costly. A varying action space also raises the necessity to co-fine-tune the model to avoid catastrophic forgetting (Brohan et al., 2023).

7. Using Temporally Abstract Behaviour

In the previous sections, we presented a variety of approaches addressing the option discovery problem. This naturally leads to the question: how might an agent effectively use this set of behaviours to inform decision-making? In this section, we outline a spectrum of possible ways of integrating options and discuss different learning strategies.

7.1 Different Ways of Deliberating over Options

Let us consider the most common way of integrating options within an agent: the call-and-return model. In this model, a single option is chosen at every high-level decision point, and this option selects actions in the environment until its termination or interruption. This process repeats, and the high-level policy selects again amongst the available options. This model is by far the most predominant one across all HRL approaches we have covered in this work and was also used to give a simplified presentation of HRL itself in Figure 4. The call-and-return model presents a straightforward way to think about HRL: a computational cost is paid at every high-level decision point for the high-level policy to deliberate and decide on an option. This cost can come in the form of a forward pass in a large neural network, chain-of-thought deliberation in LLMs, or a planning budget using option models. Once this cost is paid, the computational burden is reduced to the amount of computation required for the option to pick primitive actions.

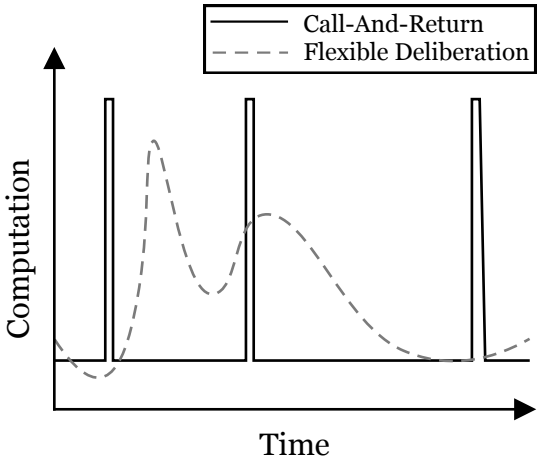


Figure 16: Depiction of the distribution of computation over time for the standard call-and-return model of execution. We assume that high-level decisions incur a greater computation cost compared to the low-level ones. This is illustrated in the spikes that characterize the call-and-return model. We also present a hypothetical model that would distribute computation over time in a more flexible way.

The call-and-return model proposes to spend computation as a binary choice: either the model deliberates over options or executes them. However, one could allocate computation according to a different distribution by allowing various degrees of deliberation to happen across timesteps. We illustrate this through Figure 16. Some states could require extensive deliberation, for example, in the form of long chains of thought during the reasoning process. Some other states could require shorter deliberations to decide on the correct action. The line of work on the generalized policy iteration and the generalized policy evaluation (Barreto et al., 2017, 2019b, 2020) is a concrete example of how one might redistribute computation across all timesteps. In this work, additional computation is spent at every timestep to select an action that is at least as good as the actions that would be chosen by any of the individual option policies in isolation.[10]

7.2 Learning High-level Policies

The agent's high-level policy, $\mu(o|s)$, is responsible for selecting an option. We present different approaches to learning this quantity by separating methods into three categories: model-free approaches, model-based approaches, and approaches that rely on in-context learning using LLMs.

10. Combining options at execution time is also explored by looking at their value functions (Todorov, 2009a; Hunt et al., 2018; Haarnoja et al., 2018).

7.2.1 Model-free approaches

Usual model-free RL methods (like Q-learning) can, with slight modifications, be used to learn a policy that selects options $\mu(o|s)$. These modifications simply involve discounting rewards obtained during option execution appropriately and using the state at the end of option exection as the next environment state, i.e., the experience tuple used to update the agent is $(s_t, o_t, \sum_{k=0}^{\tau} \gamma^k r_{t+k}, s_{t+\tau})$, where τ is the duration of execution of option o_t (Bradtke and Duff, 1994). This approach, while simple, treats option execution as a black-box. When the chosen option is Markov, meaning that its duration τ can be written purely as a function of state (and not time), then intra-option learning can be used for improved sample-efficiency. As long as states observed during option execution are inside the option's initiation set, then the corresponding transitions can be used to update $\mu(o|s)$ (Sutton et al., 1998). Specifically, an SMDP transition $(s_t, o_t, \sum_{k=0}^{\tau} \gamma^k r_{t+k}, s_{t+\tau})$ can be decomposed into up to τ transitions of the kind $(s_i, o_t, \sum_{k=0}^{\tau} \gamma^k r_{i+k}, s_{i+\tau})$ for all i such that $s_i \in \mathcal{I}_o$. Bacon (2018) later generalize these insights to policy gradient methods by proposing the option gradient theorem.

Bandits that maximize learning progress. A popular model-free approach is to treat the high-level policy as a contextual bandit (which can be thought of as an MDP with $\gamma = 0$). The reward function for the bandit is designed to carefully trade off various objectives. For example, when the extrinsic reward is dense and informative, the bandit simply chooses the option expected to maximize the reward (Schaul et al., 2019). When the reward function is sparse or deceptive, then a measure of *learning progress* (LP) is often added to the extrinsic reward; the idea is that the agent should pick options that (in addition to greedily maximizing reward) would also improve its knowledge of the environment and its own competence in the environment (Colas et al., 2022). Although measuring LP itself is intractable, proxies are used in practice. Competence progress (Oudeyer and Kaplan, 2007; Stout and Barto, 2010) prioritizes skills whose capabilities change the most with time—these skills represent subgoals of intermediate difficulty (Florensa et al., 2018). Count-based bonuses prioritize options that lead to high novelty (Bagaria and Schaul, 2023; Badia et al., 2020b,a), and density-based approaches (Pong et al., 2020) attempt to maintain a high entropy distribution for option selection from different states (Pitis et al., 2020).

7.2.2 Model-based approaches

Typically, in model-based RL, the agent first learns transition and reward models of the world, and then uses those models to look ahead in the future, before finally making a decision at the current timestep. When the agent learns single-timestep models of the world, it must roll out these models over a long horizon. This is problematic because model-prediction errors compound over time (Talvitie, 2017; Janner et al., 2019) and small errors in model prediction can lead to massive errors in value approximation (Kearns and Singh, 2002). Options allow the agent to learn temporally extended models of the world, which afford longer-horizon planning.

Learning option models. The agent's stream of interaction data can be used to learn option models in two ways: (a) on-policy: where the agent updates the models for an option after it is executed (Sutton et al., 1999b), or (b) off-policy: where the agent uses intra-option learning (Sutton et al., 1998) to simultaneously learn about many options from

the data collected at every timestep. Some methods learn the option model in the agent's observation space, while others operate in an abstract state space. Models trained in the raw observation space must contend with the challenges of high-dimensional inputs and outputs (Nair and Finn, 2020). When state abstraction is learned alongside options, the agent must also manage *drift*, where option models must rapidly adapt to changes in the evolving abstract state representation.

Abstract planning. Options enable procedural abstraction, but the agent still has to plan in its original observation space, which is challenging when that observation space is high-dimensional. More effective planning can be achieved by combining options with a suitable state abstraction. This combination of state and action abstraction can result in abstract decision processes that are simpler to plan in, but this often comes at a cost—the coarser the abstraction, the greater the potential for suboptimality of the resulting plans, mirroring the trade-offs discussed in the context of options in Section 2.2. We now briefly discuss some approaches that combine options with state abstraction for model-based planning.

- *Expectation models.* There are at least three choices for representing an option models: (a) distribution model: predict the distribution over possible next states, (b) sample model: generate a sample from the next state (and reward) distribution, and use sample-based planning techniques such as Monte-Carlo Tree Search (Coulom, 2006), and (c) expectation model, where the agent predicts the expected next state and reward. When the value function is linear in the agent's state representation, then expectation models are sufficient for planning (Wan et al., 2019). Due to its simplicity, expectation models can be learned efficiently by solving a system of linear equations (Sutton et al., 2023), making it an attractive choice for HRL agents that simultaneously learn state representations that evolve over time. There have also been proposals of using temporal abstractions as a mechanism for focusing on local, subgoal-conditioned models that are possibly easier to learn than a complete model of the environment (Lo et al., 2024).

- *Skills to symbols* (Konidaris et al., 2018). When options have the property that their policy drives all state variables to a small range of values, then the abstract state representation needed for planning is that of a *graph*. Nodes of this graph correspond to abstract states and edges correspond to options; an edge exists between two nodes when one option terminates in a state from which another option has a high probability of being successful in its own subtask. The discovery of options with this property of sequential composability was discussed in Section 4.3. The Deep Skill Graphs algorithm (Bagaria et al., 2021b, 2025b) simultaneously learns options and such a graph representation for planning in continuous environments. However, skills cannot always control all state variables—they often set some state variables, while leaving others unchanged. When options have this property, then the representation needed for planning is that of a type of factored MDP (Boutilier et al., 2000), which can be succinctly described using Planning Domain Definition Language (PDDL) (McDermott et al., 1998). The advantage of generating a PDDL description of the problem is that it can then be efficiently solved by off-the-shelf classical planners, even when the planning problem is long-horizon and combinatorial in the number of state variables.

The algorithms of Konidaris et al. (2018) provide a way to learn such abstract state representations, enabling an agent to compute the probability with which a given plan will be successful. Recently, Rodriguez-Sanchez and Konidaris (2024) proposed a way to learn continuous state representations that lead to provably bounded *value loss* (Li et al., 2006; Abel et al., 2016, 2020)—meaning that when the agent plans solely with its learned abstract state representations, it foregoes no more than a bounded amount of reward compared to an agent that plans in the MDP's native state-space. An additional challenge when learning option compatible state abstractions for planning is that of transfer—learned representations should be resusable in future tasks encountered by the agent. To learn transferrable representations, James et al. (2019) leverage a simple insight: when the same agent is used to solve a family of related problems, then state representations that are expressed from the point-of-view of the agent are more amenable to transfer than state representations that uniquely describe each individual task (Konidaris and Barto, 2007). For example, a home robot that solves many tasks in many homes, does so with the same set of sensors and actuators; so representations expressed from the perspective of that robot are reusable across many different contexts. By applying this insight to learned symbolic representations, James et al. (2019) reduce the number of samples required to solve each additional task in a given sequence of tasks.

In summary, the combination of options with appropriate state abstractions offers a powerful framework for efficient model-based planning in complex environments. These approaches address fundamental challenges in reinforcement learning by enabling longer planning horizons, reducing the dimensionality of the planning space, and mitigating error propagation in learned models (Bagaria et al., 2025b). The trade-off between abstraction granularity and solution optimality remains a central consideration, with different methods offering various compromises between planning efficiency and performance guarantees. As hierarchical reinforcement learning continues to evolve, integrating these state and action abstraction techniques with advances in representation learning and approximate planning promises to further enhance the scalability and applicability of RL to increasingly complex real-world problems. Future research directions include developing more robust methods for discovering suitable abstractions automatically, improving the theoretical understanding of abstraction hierarchies, and bridging the gap between symbolic planning and continuous control.

7.2.3 Large Language Models

If options are represented using LLMs, in-context learning can be used to learn the high-level policy, $\mu(o|s)$. This can be done by having the LLM output Python code that implements skill-selection logic (Wang et al., 2024c; Klissarov et al., 2024), or to output formal plans described using PDDL (Silver et al., 2024). Such policies can then be updated by providing execution traces as context to the LLMs and asking for code refinements. It is also possible to directly deploy the LLM in the environment to select skills at every high-level decision point (Ahn et al., 2022). Since such approaches do not require gradient updates, they potentially offer faster adaptation. However, the nature of in-context is currently not well understood, for example, in terms of generalization and robustness, and is an active area of research.

8. Challenges of Discovery

Arguably, one of the biggest challenges in discovering temporal abstractions comes from the fact that there is a **lack of agreed-upon objective** that would yield meaningful options across a variety of domains. This can be observed by the wide diversity of methods presented in Section 4, 5, and 6. Additionally, the **complexity overhead** that HRL sometimes introduces can make it less appealing from a practical perspective. The time invested by a practitioner in setting up an HRL algorithm is valuable. If this time investment does not lead to significantly improved performance on a particular task, or is not generally applicable across tasks, the practitioner will likely choose a simpler approach.

The two aforementioned points indicate that there is a lot of potential for research in HRL in order to find **reliable and general** solutions as well as understanding *where to apply them* (see Section 10). In what follows, we highlight prominent technical challenges that arise when attempting to discover temporal abstractions.

8.1 Non-stationarity

One of the main difficulties in learning a hierarchy of behaviours stems from its modular nature. A hierarchical agent has to learn, potentially simultaneously, about the option policies, option reward functions, termination functions, initiation functions, and high-level policy. As each of these modules is being learned, it creates non-stationary targets for the other modules.

A straightforward approach to deal with this non-stationarity is by learning the different components separately. For instance, this can be done by leveraging offline datasets (methods in Section 5) to first learn a set of option rewards or a set of option policies, before fixing them. These components can then be provided to a high-level policy that will learn to achieve a certain task. Similarly, we can leverage the LLM's prior knowledge to define, beforehand, option reward functions or to directly model the option policies (methods in Section 6). These would create stationary targets for the remainder of the components. LLMs can also be used to model the high-level policy itself, either by directly querying them or by leveraging their coding abilities to define the skill execution logic. The in-context learning abilities of LLMs could further allow for fast, gradient-free adaptation with respect to a changing option set.

When learning tabula rasa, the non-stationarity can be particularly challenging. It is common for methods to first define an option learning phase, where the high-level policy acts according to a more exploratory behaviour, for example by uniformly choosing over the options (Machado et al., 2017; Eysenbach et al., 2019). Such a phase is meant to provide experience in learning the option reward functions and option policies. Nachum et al. (2018) emphasize the difficulty of non-stationarity in HRL when learning from past experiences that are stored in a dataset, called an experience replay buffer (Lin, 1991). An option that was previously sampled and stored within a replay buffer, together with the experience it generated, would not produce the same data distribution if we were to sample it now. To alleviate this, Nachum et al. (2018) relabel which option was used for a stored datapoint with the option that is currently most likely to generate the actions seen in this datapoint.

Bagaria et al. (2023) illustrate how the non-stationary challenge affects the initiation set. They argue that learning the initiation function using binary classification (or, equivalently,

Monte Carlo value estimation) is only a sound approach when the option policy is fixed. In their approach, the initiation function captures the capability of the current option policy to achieve its goal. As the option policy evolves, so must the initiation function. As a consequence, when an option is unsuccessful from a state, its initiation probability at that state goes down, and so does the probability that the option policy improves in and around that state. While this is unproblematic when the option policy is fixed, it eventually leads to overly conservative initiation functions: options tend to initiate in smaller and smaller parts of the state-space during the agent's lifetime. To address these issues, they incorporate tools from off-policy evaluation and use exploration bonuses to increase the initiation probability of states from which policy improvement is most likely.

8.2 Learning About Multiple Behaviours

One of the appeals of the HRL is that if an agent has access to a large collection of options, it may efficiently achieve good performance on a variety of tasks by re-composing them. However, such a large library of behaviours also comes at the cost of first learning the options themselves, highlighting some of the fundamental trade-offs presented in Section 2.2.

To approach this problem, it is convenient to turn to off-policy algorithms (Precup et al., 2000). Such algorithms allow for learning from data that was not generated by the current policy. Klissarov and Precup (2021) propose update rules to improve all options simultaneously by relying on a decomposition of the state-option distribution, introducing a minimal amount of off-policy corrections, and remaining compatible with any policy optimization algorithm. Their method can also be seen as an all-options policy optimization, similar to all-action updates in RL (Sutton et al., 2001). Daniel et al. (2016) instead leverage the perspective in which options are seen as latent variables. The authors adopt an expectation-maximization approach, which assumes a linear structure of the option policies. Smith et al. (2018) alleviate this assumption and derive a policy gradient objective that improves the data efficiency and interpretability of the learned options. A conceptually related work is proposed by Wulfmeier et al. (2020), which leverages dynamic programming to infer option and action probabilities in hindsight.

We have previously introduced the methods of hindsight relabeling (Andrychowicz et al., 2017) as part of the skill discovery methods. We can reframe their approach through this question: if you have a multitude of options, or even a continuous spectrum, which other option should you update for a given trajectory? The authors answer this question by relabeling the trajectory stored in the replay buffer with the final state that was reached. This essentially leverages off-policy as the experience generated by one policy is used to update another policy. The importance of learning off-policy through re-labeling is emphasized by Nachum et al. (2018), which shows significantly faster learning, and by Levy et al. (2019), which extends the ideas of re-labeling experience through hindsight goal transitions.

Is it possible to sample-efficiently learn about multiple options from a single stream of experience? Barreto et al. (2020) propose the Generalized Policy Improvement update rules to answer this question. The authors extend the concept of improvement from a single policy to multiple policies simultaneously. Specifically, their theorem states that, for a given set of

policies, $\pi_1, \pi_2, ..., \pi_n$, and their associated approximate Q values, $Q^{\pi_1}, Q^{\pi_2}, ..., Q^{\pi_n}$,

$$\pi(s) \in \operatorname*{argmax}_a \max_i Q^{\pi_i}(s, a), \tag{100}$$

then $Q^{\pi}(s, a) \geq \max_i Q^{\pi_i}(s, a)$. This update rule is used by Barreto et al. (2019a) to efficiently learn how to execute a combination of options. Thakoor et al. (2022) further generalize the results beyond Markov policies, in particular, to options whose execution duration follows a geometric distribution. The idea of learning efficiently about multiple policies is closely related to concepts such as the successor representations (Dayan, 1993) and successor features (Barreto et al., 2017), as well as other decompositions of the transition function (Touati et al., 2023).

8.3 Combining Rewards

When learning option policies through their option reward functions, we are faced with another important question: how should we balance between the option reward and the environmental reward? Dayan and Hinton (1993) argue that the option policies should be agnostic to the environmental reward and learned only through the intrinsic one, leading to specialised options. Vezhnevets et al. (2017) take a softer approach and provide both rewards, possibly as the environmental reward contains rich information in the environments that were considered. In other cases, there is no intrinsic reward at all (Bacon et al., 2017). Sutton et al. (2023) investigate these questions from the perspective of planning and learning with options that either respect or do not respect the environmental reward. The authors show that reward-respecting options (that is, options that take the environment reward into consideration) are much more effective when used for planning. Zahavy et al. (2022) propose a point of view of constrained optimization to balance these objectives and leverage Lagrange multipliers in practice. A thorough examination concerning the trade-offs of how hierarchical agents combine environmental reward and intrinsic reward is yet to be made.

9. Related Fields

We now discuss the fields related to HRL, covering different types of abstractions, continual RL, programmatic RL, and cooperative multi-agent RL.

9.1 State and Action Abstractions in Reinforcement Learning

Scaling RL for real-world applications faces challenges in handling high-dimensional or noisy observations and large action spaces. Accordingly, the RL community has long explored *abstraction*, which in computer science practice suppresses irrelevant low-level details so that reasoning can proceed at a higher conceptual level (Colburn and Shute, 2007), as a means to mitigate the curse of dimensionality and improve sample efficiency (Konidaris, 2019; Ho et al., 2019; Abel, 2022). Abstraction can be accomplished either through explicit *aggregation* of states and actions (Li et al., 2006), or by using neural networks as a mapping from the raw state or action space to an abstract space—a process often referred to as *representation learning* (Abel, 2022). Various forms of abstraction have been proposed in the RL literature, each targeting distinct equivalence relations to capture different aspects of the learning problem.

State abstraction offers a principled approach to scaling RL to control tasks involving high-dimensional observations, such as images, which often contain substantial task-irrelevant details. Li et al. (2006) survey a spectrum of state-abstraction schemes, each defined by its own equivalence criterion. For example, some merge states that yield identical immediate reward and transition dynamics under every action, while others require the same optimal action-value functions. In contrast, bisimulation metrics (Ferns et al., 2004, 2011; Castro, 2020; Zhang et al., 2021a; Luo et al., 2025) dispense with such rigid equivalence by quantifying how much two states differ in their reward distributions and transition kernels, which enables grouping those whose combined divergence falls below a chosen threshold. To make state abstraction more deep-learning-friendly, recent approaches introduce differentiable objectives, specifically reward prediction and self-prediction losses defined with respect to a learned representation, to train compact, informative embeddings (Gelada et al., 2019; Ni et al., 2024).

Another line of work focuses on *state-action abstraction*, notably MDP homomorphism, which maps state–action pairs to abstract equivalents while preserving transition and reward structure (Ravindran, 2004; Ravindran and Barto, 2001, 2004; Narayanamurthy and Ravindran, 2008; Rezaei-Shoshtari et al., 2022). This aggregation of the state-action space, termed *model minimization*, forms an abstract MDP capable of capturing symmetrical aspects of the environment.[11]

As for *action abstraction*, it can be classified into two categories: per-timestep and multiple-timestep. Per-timestep action abstraction is commonly applied to mitigate the computational complexity associated with large action spaces, involving action elimination (Even-Dar et al., 2006; Zahavy et al., 2018), action embedding or transformation (Van Hasselt and Wiering, 2009; Dulac-Arnold et al., 2015; Jiang et al., 2023), and affordances (Abel et al., 2014; Fulda et al., 2017; Khetarpal et al., 2020a), which reduces the effective action space to only those that satisfy a given intent or task-relevant criterion under the current state. Per-timestep action abstraction can also be extended to *policy abstraction* (Barreto et al., 2019a; Zhang et al., 2023), which provides a framework for generalizing and compressing policy behaviours by mapping detailed decision-making strategies into a succinct abstract space. Multiple-timestep action abstraction, often referred to as temporal abstraction, is a fundamental aspect of HRL. It can be either closed-loop as described in the option framework (Sutton et al., 1999b), or open-loop as a compression of an action sequence (Pertsch et al., 2021).

These abstraction types naturally interface with HRL, which provides a framework for integrating them effectively. In addition to temporal abstraction, HRL facilitates the integration of various types of state and action abstractions. In classical HRL, two common forms of state abstraction are employed: first, state abstraction within the high-level controller, enabling learning or planning in a more tractable space. Feudal RL (Dayan and Hinton, 1993), as a prominent example, employs information hiding to abstract low-level details from the state observed by the manager. Second, state abstraction within the low-level controller, which abstracts states irrelevant to a particular option. State abstractions within MAXQ (Dietterich et al., 1998) and option models are natural examples, as options can be defined exclusively for states where the option is applicable. Classical HRL also incorporates per-timestep action abstractions. In the option framework (Sutton et al.,

11. Symmetrical aspects denote invariances under transformations of states and actions that leave both the transition dynamics and reward function unchanged.

1999b), the initiation set serves as a high-level per-timestep action abstraction, indicating the affordance of a specific option in different states.

Several HRL methods leverage state and action abstractions in addition to temporal abstraction. Relativized options (Ravindran and Barto, 2002; Ravindran, 2003; Ravindran and Barto, 2003) integrate state-action abstraction (MDP homomorphism) techniques within an HRL framework to generate concise representations of a related task family. These options are defined without an absolute frame of reference, and their policies adapt according to the circumstances of their invocation, enabling effective multi-task knowledge transfer. Portable options (Konidaris and Barto, 2007) extend this concept, ensuring that the option depends solely on abstract states characterized by task-invariant descriptors. Castro and Precup (2010) apply a bisimulation metric for two different MDPs to facilitate knowledge transfer and propose an option-bisimulation metric to quantify the behavioural discrepancy between states under an option. Abel et al. (2020) propose a value-preserving abstraction, combining state abstractions and options to ensure the representation of near-optimal policies is maintained. In their approach, the state abstraction ϕ, which maps the state to an abstract state, defines the initiation and termination functions for a set of *ϕ-relative options*. Khetarpal et al. (2021) extend their definition of affordances (Khetarpal et al., 2020a), introducing temporally extended intents and option affordances that benefit planning in temporally abstract partial models. Hansen-Estruch et al. (2022) connect GCRL and bisimulation metrics. The authors propose a state-goal bisimulation metric to learn a shared state-goal representation, improving representation learning across tasks defined by different goals.

9.2 Continual Reinforcement Learning

Continual RL defines the problem setting in which any component of the environment, such as the transition function, the reward function, the state space, or the action space, may change over time (Khetarpal et al., 2020c). Continual RL emphasizes the stability-plasticity dilemma (Carpenter and Grossberg, 1988) which arises when training neural networks under non-stationarity: should we prioritize recent experiences or remember previous experiences? A common example is when an agent is faced with a series of tasks within a complex environment, without being told when tasks are changing. Such an example illustrates the importance of fast adaptation as a desirable quality in a continually learning agent. A related and well-known difficulty is in avoiding catastrophic forgetting, where an agent adapts adequately to the latest experiences, but completely forgets what it learned in past experiences.

To face the challenges posed by the continual RL problem setting, there exists a variety of methods, such as explicit knowledge retention mechanisms or leveraging the structure shared across tasks. Agents empowered by a set of reusable skills are a part of the latter category: they have the potential to efficiently adapt to new tasks by recombining or fine-tuning their library of skills, minimizing the need to relearn from scratch (e.g., Klissarov and Machado, 2023). Additionally, HRL agents could mitigate catastrophic forgetting by expanding and filtering their set of skills over time. One of the fundamental reasons for the synergy between HRL and continual RL is that both fields rarely focus on optimally solving any of the tasks that are being given. Instead, they are concerned about fast adaptation and transferability.

While promising, integrating HRL and continual RL presents open research challenges. As mentioned in Section 8, it is necessary to develop scalable skill discovery methods that

can function in non-stationary settings, devise frameworks that jointly optimize for continual learning and HRL objectives, and design benchmarks and metrics for evaluating agents.

9.3 Programmatic Reinforcement Learning

As stated in Section 2, HRL conceptually makes an analogy to programming languages and formal systems. An example of this connection is *Hoare Logic* (Hoare, 1969), a formal system for assessing the correctness of imperative programs, which shares a similar structure with the option framework (see Section 3.2) including initiation sets (pre-conditions), policies (commands), and termination conditions (post-conditions). Both frameworks facilitate reasoning about action sequences, thereby enhancing the structuring of complex decision-making processes. Research efforts have been undertaken to bridge the gap between HRL, programming languages, logic, and formal methods.

Programs as high-level policy. Early approaches, HAM (Parr and Russell, 1997) and PHAM (Andre and Russell, 2000) utilized hierarchies of partially specified finite-state machines (FSM) to structure policies. There are four types of states in HAMs: *Action* states execute actions, *Call* states execute subroutines, *Choice* states select subsequent states non-deterministically, and *Stop* states halt execution and return control to prior call states. This provides a prototype for early HRL methods, allowing for better compositionality, transferability (Andre and Russell, 2000), and state abstraction (Andre and Russell, 2002). More recent approaches utilize programs, specifically in domain-specific languages (DSLs), as high-level policies to guide lower-level RL agents. These are often called programmatic policies. Such an approach allows the system designer to inject biases that could, for example, improve sample efficiency over neural representations (Moraes et al., 2025).

Programs convey structured, interpretable, and unambiguous information, and their incorporation into the policy space can reduce the search space for the overall solution and offer a natural method for integrating prior knowledge symbolically. The structured representation of these programs allows one to decompose policies into options that can also be used to induce spaces that are more conducive to search (Moraes and Lelis, 2024; Moraes et al., 2025). In general, the programs can be either hand-crafted (Andreas et al., 2017; Sun et al., 2020), synthesized automatically by construction or synthesis on a predetermined syntax (Carvalho et al., 2024) or semantic (latent) space (Yang et al., 2021b; Hasanbeig et al., 2021; Moraes et al., 2023; Moraes and Lelis, 2024), by parameterizing the program space, also known as *neuro-symbolic* (Sheth and Roy, 2023) approaches (Denil et al., 2017; Sohn et al., 2018; Trivedi et al., 2021; Zhao et al., 2021; Qiu and Zhu, 2022; Liu et al., 2023a; Lin et al., 2024) or by leveraging foundation models (Wang et al., 2023a; Klissarov et al., 2025a; Moraes et al., 2025). Learning search guidance for these spaces is an active area of research (Medeiros et al., 2022; Aleixo and Lelis, 2023). The idea of decomposing policies into subprograms has also been explored even when the underlying policy is a neural network (Alikhasi and Lelis, 2024).

Programs to intrinsic rewards. Akin to the intrinsic reward described in Section 4, recent studies (Jothimurugan et al., 2019; Icarte et al., 2022; Furelos-Blanco et al., 2023; Venuto et al., 2024) demonstrate the feasibility of "translating" the formal languages (e.g., programs or FSMs) into the reward signal to enhance the RL agent.

Distilling the neural policies to interpretable programs. A series of studies focuses on condensing an agent's policy into more hierarchical, interpretable, and verifiable formats such as programs (Verma et al., 2018, 2019) or Decision Trees (Bastani et al., 2018), enhancing both lightness and clarity.

9.4 Cooperative Multi-Agent Reinforcement Learning

Cooperative multi-agent RL (Cooperative MARL) and HRL can be seen as conceptually connected: managing problem complexity using the structure of the problem. By breaking down large-scale problems into more manageable sub-problems, both approaches improve tractability and facilitate learning. In cooperative MARL, decomposition is achieved by distributing the decision-making process among multiple agents, whereas in HRL, it is accomplished through temporal abstraction. As an example, Feudal RL (Dayan and Hinton, 1993) can be viewed as a multi-agent system comprising managers and workers. This framework naturally extends to cooperative MARL settings (Ahilan and Dayan, 2019). Extensive research has explored the integration of HRL with cooperative MARL; interested readers are referred to Section 3.5 of the work by Pateria et al. (2021) for further details.

10. Promising Domains for Hierarchical Reinforcement Learning

In this work, we have examined a wide diversity of HRL approaches, each time highlighting the important ways in which they help decision-making through the benefits we laid out in Section 2.1. The vast body of research in HRL encompasses a wide spectrum of methods spanning multiple environments and domains. Under this diversity of approaches, a key question emerges: **in what domains should we expect HRL to be most effective?** One obvious criterion is for the domain to contain tasks that are temporally extended tasks, as short-horizon tasks offer limited opportunities for leveraging temporal abstractions. For example, decomposing short-horizon tasks into subtasks is likely to be less fruitful than long-horizon ones. However, can we go beyond this simple criterion to predict the suitability of HRL methods?

As illustrated in Section 2.2, one of the motivations for the HRL formalism is that it is a way to efficiently obtain good solutions within a certain sample and computation budget. This is particularly relevant in complex environments, where optimality is impractical. Should HRL then be considered as a fallback option when non-hierarchical RL fails in complex environments? This perspective positions HRL as a last resort when the task is too hard, but importantly, does not rely on any concrete intuition as to why HRL should even work in such situations. To provide a more informative answer, we go back to the fundamental idea that was used to introduce the methods in this work. This idea is that **HRL methods exploit structure**. A complex environment lacking exploitable structure might not benefit from HRL. Similarly, a complex environment where we only care about a single task might limit HRL's advantages, given the inherent overhead of learning a hierarchy. Therefore, task complexity alone is not a sufficient condition for the effectiveness of HRL methods.

HRL appears best suited for long-horizon environments that allow for a diversity of goals that share a structural overlap (whether these goals are defined by the environment or the agent itself). From this perspective, **open-ended systems are particularly promising domains for HRL methods**. Hughes et al. (2024) define an open-ended system as

one that presents a constant flow of novel and possibly learnable goals. It is common in such systems that these goals share, to a degree, a common underlying structure, which makes HRL particularly appealing. Below, we showcase specific domains exemplifying these characteristics. Importantly, this list is not exhaustive but rather serves to illustrate settings where HRL might excel.

10.1 Example Environments and Applications

Web Agents. The World Wide Web, a dynamic and ever-changing environment, presents a unique challenge for AI agents. The recent surge in interest has led to a variety of implementations of challenging domains, such as Android-in-the-Wild (Rawles et al., 2023) or WebArena (Zhou et al., 2024). Its near-infinite tasks and constantly evolving goals demand adaptability and the ability to decompose complex objectives into manageable subgoals. As mentioned Section 3, even if the resulting agent is not hierarchical (i.e., does not explicitly carry a set of skills), learning to navigate the web through HRL methods, such as curriculum-based ones, is particularly important to address the sheer complexity of the web. Indeed, Web Agents must learn to navigate a constantly shifting landscape of information and services, adapting to new data, evolving user preferences, and the emergence of novel websites and services. Another important characteristic is that many tasks of interest share a lot of underlying structure, a key point of HRL. Overall, this complex and open-ended domain requires agents capable of learning, adapting, and generalizing across multiple timescales, ultimately revolutionizing how we interact with the online world.

Robotics. Robotics, with its emphasis on embodied intelligence and real-world interaction, presents a compelling domain for exploring the potential of HRL methods. The tasks robots face, from navigating complex environments to manipulating objects with dexterity, involve long horizons where, at each step, a low-level action is sampled from a continuous action space. HRL offers a natural framework for decomposing these complex tasks into manageable sub-policies, allowing robots to learn and refine abstract skills while also developing higher-level strategies for sequencing and coordinating them. Practical implementations of interest include AI2-THOR (Kolve et al., 2017), Habitat (Szot et al., 2021; Puig et al., 2024), CALVIN (Mees et al., 2022) and OGBench (Park et al., 2025a).

The ability to recompose learned skills into novel combinations is crucial for robots operating in unstructured and dynamic environments, where adaptability and generalization are key. For instance, a robot learning to grasp objects might develop sub-policies for reaching, orienting its gripper, and applying the appropriate force. Ideally, these individual skills could then be recombined and adapted to grasp a wide variety of objects in different contexts, without requiring retraining from scratch. The long horizons inherent in many robotic tasks, coupled with the need for flexible and adaptable skill acquisition, make HRL a promising approach for developing robots capable of performing complex, real-world tasks with increasing autonomy and efficiency.

Open-ended games. Training AI agents on games has a long history of striking successes in domains like Go (Silver et al., 2017) or Atari 2600 games (Mnih et al., 2015).[12] However,

12. Note that this applies less to the case of hard exploration games such Montezuma's Revenge, Private Eye, or Pitfall!, amongst others.

for HRL to be particularly effective, the domain should be complex, long-horizon, and open-ended. We have seen in Section 4.8 such an example, where a goal-conditioned policy trained on a large diversity of tasks led to human-timescale adaptation. Key to this success was the fact that data was readily available through fast simulation, allowing for quicker research iteration. This makes it particularly interesting to study open-ended games in order to better understand HRL methods. We now provide two such examples. **NetHack** is a complex roguelike game, and it is an ideal environment for exploring the benefits of HRL. It has been brought to the RL community through the NetHack Learning Environment (Küttler et al., 2020). Its open-ended nature, procedurally generated dungeons, and long-horizon gameplay require exploration, planning, and adaptation across multiple timescales. Success requires not just immediate tactical decisions, but also strategizing towards long-term goals, demanding credit assignment across extended temporal spans. The vast diversity of situations encountered also requires generalization, making HRL's ability to learn reusable sub-policies and higher-level strategies particularly valuable. **Minecraft.** Minecraft (Johnson et al., 2016; Kanervisto et al., 2021), with its expansive, procedurally generated world and open-ended gameplay, presents a compelling testbed for HRL algorithms. The game requires navigating across diverse biomes, gathering resources, crafting tools, and structures, and ultimately, surviving and thriving. This requires planning and execution across multiple timescales. For instance, while the immediate goal might be chopping down a tree for wood, this action serves the higher-level objective of building a shelter for protection against nocturnal mobs. Furthermore, Minecraft's crafting system inherently embodies a hierarchical structure. Creating complex items like diamond tools requires a chain of prerequisite crafting steps, each with its own subgoals and resource requirements. HRL agents could learn to decompose these complex tasks into manageable sub-policies, mirroring the hierarchical nature of crafting itself.

11. Conclusion

In this paper, we have attempted to cover the rich, complex, and ever-expanding field of hierarchical reinforcement learning. We have started by highlighting the importance of modularity and compositionality as environment characteristics for hierarchical reinforcement learning to discover useful structure. We have then defined how hierarchical reinforcement learning can benefit an agent by looking through the lens of the fundamental problems in decision-making.

It is through these benefits, namely exploration, credit assignment, transferability, and interpretability, that we subsequently presented all the methods covered in this work. These benefits also more clearly characterize what hierarchical reinforcement learning is: it is not about an agent architecture, but rather about discovering structure and using it to achieve these fundamental benefits. When presenting the existing techniques, we have grouped them into three categories: (1) discovery from online experience, (2) discovery through offline datasets, and (3) discovery with foundational models. Within each of these categories, we have further decomposed the large corpus of methods into families that share fundamental insights about the kind of structure an agent should attempt to discover. We have also discussed how one might leverage temporally abstract behaviour from an agent's perspective. Finally, we have emphasized the important challenges that exist when

discovering structure through hierarchical reinforcement learning and the environments in which it is most promising to do so.

Throughout this work, we have constantly strived to connect technical knowledge of the methods to fundamental principles, for instance by explicitly referring to the benefits of hierarchical reinforcement learning. We have also made connections to highly relevant related fields, for example, programmatic RL or other types of abstractions in RL (see Section 9). There exist many other research areas that would be particularly interesting to understand for hierarchical reinforcement learning research. One such field is that of search algorithms (Telikani et al., 2021) such as quality diversity algorithms (Lehman and Stanley, 2011; Cully et al., 2015; Ding et al., 2024). The potential for hierarchical reinforcement learning research is continually expanding as we enter an age where AI models are required to be generalists. In fact, as we tackle tasks with increasingly long horizons, finding decompositions that afford learnability may simply be unavoidable in reinforcement learning (Park et al., 2025b). Consequently, simple and scalable methods for discovering and leveraging structure are more pressing than ever. We hope this work provides a useful foundation for realizing this potential and inspires future innovation.

Notation

Table 2: Glossary of notations used in RL and HRL (see Section 3).

Reinforcement Learning (RL)	
$S_t \in \mathcal{S}$	State at time step t
$A_t \in \mathcal{A}$	Action at time step t
$R_{t+1} \in \mathbb{R}$	Reward at time step $t+1$
$p(s' \mid s, a)$	Transition probability
$\gamma \in [0, 1)$	Discount factor
$\pi(a \mid s)$	Policy over actions
$d_\pi^\gamma(s)$	Discounted state occupancy under π
$q_\pi(s, a)$	Action-value function under policy π
$v_\pi(s)$	State-value function under policy π
$Q(s, a)$	Estimated Q-function
Hierarchical Reinforcement Learning (HRL)	
$o \in \mathcal{O}$	Option (temporally extended action)
$z \in \mathcal{Z}$	Skill (alternative term for option)
$g \in \mathcal{G}$	Goal
$\mu(o \mid s)$	High-level policy
$\pi(a \mid s, o)$	Option policy
$\beta(s, o)$	Option termination function
$\mathcal{I}(s, o)$	Option initiation function
$P_\mathcal{O}(s, o, s')$	Option transition model
$q_\pi(s, o)$	Option-value function
$q_u(s, o, a)$	Action-value within option context
$u_\beta(o, s')$	Option value upon arrival
$v_\mu(s)$	Value function under high-level policy μ
$r^o(s, a, s')$	Intra-option reward function for option o

Acknowledgements

We would like to thank Xujie Si, Khimya Khetarpal, Seohong Park, Bartłomiej Cupiał, Isabeau Prémont-Schwarz, Levi Lelis, Andrew Levy, Alex Ivanov, Nishanth Anand, and Jonathan Colaço Carr for their valuable feedback. This research is supported in part by NSF 1955361 and NSF CAREER 1844960. The research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Canada CIFAR AI Chair Program.

References

Abel, D. (2022). *A Theory of Abstraction in Reinforcement Learning*. PhD thesis, University of Texas.

Abel, D., Barth-Maron, G., MacGlashan, J., and Tellex, S. (2014). Toward affordance-aware planning. In *First Workshop on Affordances: Affordances in Vision for Cognitive Robotics*.

Abel, D., Hershkowitz, D. E., and Littman, M. L. (2016). Near Optimal Behavior via Approximate State Abstraction. In *International Conference on Machine Learning*.

Abel, D., Umbanhowar, N., Khetarpal, K., Arumugam, D., Precup, D., and Littman, M. L. (2020). Value Preserving State-Action Abstractions. In *International Conference on Artificial Intelligence and Statistics*.

Abramson, D. I. A. T. J., Ahuja, A., Brussee, A., Carnevale, F., Cassin, M., Fischer, F., Georgiev, P., Goldin, A., Harley, T., Hill, F., Humphreys, P. C., Hung, A., Landon, J., Lillicrap, T. P., Merzic, H., Muldal, A., Santoro, A., Scully, G., von Glehn, T., Wayne, G., Wong, N., Yan, C., and Zhu, R. (2021). Creating Multimodal Interactive Agents with Imitation and Self-Supervised Learning. *arXiv*.

Achiam, J., Edwards, H., Amodei, D., and Abbeel, P. (2018). Variational Option Discovery Algorithms. *arXiv*.

Adeniji, A., Xie, A., Sferrazza, C., Seo, Y., James, S., and Abbeel, P. (2023). Language Reward Modulation for Pretraining Reinforcement Learning. *arXiv*.

Ahilan, S. and Dayan, P. (2019). Feudal multi-agent hierarchies for cooperative reinforcement learning. *arXiv*.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., Yan, M., and Zeng, A. (2022). Do As I Can and Not As I Say: Grounding Language in Robotic Affordances. In *Conference on Robot Learning*.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows - Theory, Algorithms and Applications*. Prentice Hall.

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. (2021). Opal: Offline primitive discovery for accelerating offline reinforcement learning. *International Conference on Learning Representations*.

Akakzia, A., Colas, C., Oudeyer, P.-Y., Chetouani, M., and Sigaud, O. (2021). Grounding Language to Autonomously-Acquired Skills via Goal Generation. In *International Conference on Learning Representations*.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J., Borgeaud, S., Brock, A., Nematzadeh, A., Sharifzadeh, S., Binkowski, M., Barreira, R., Vinyals, O., Zisserman, A., and Simonyan, K. (2022). Flamingo: a Visual Language Model for Few-Shot Learning. *Neural Information Processing Systems*.

Aleixo, D. S. and Lelis, L. H. S. (2023). Show Me the Way! Bilevel Search for Synthesizing Programmatic Strategies. In *The Association for the Advancement of Artificial Intelligence*.

Alikhasi, M. and Lelis, L. H. S. (2024). Unveiling Options with Neural Network Decomposition. In *International Conference on Learning Representations*.

Ames, B. and Konidaris, G. D. (2019). Bounded-Error LQR-Trees. In *International Conference on Intelligent Robots and Systems*.

Amin, S., Gomrokchi, M., Satija, H., van Hoof, H., and Precup, D. (2021). A Survey of Exploration Methods in Reinforcement Learning. *arXiv*.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., and Mané, D. (2016). Concrete Problems in AI Safety. *arXiv*.

Andre, D. and Russell, S. (2000). Programmable reinforcement learning agents. *Neural Information Processing Systems*.

Andre, D. and Russell, S. J. (2002). State abstraction for programmable reinforcement learning agents. In *The Association for the Advancement of Artificial Intelligence*.

Andreas, J., Klein, D., and Levine, S. (2017). Modular multitask reinforcement learning with policy sketches. In *International conference on machine learning*, pages 166–175. PMLR.

Andrychowicz, M., Crow, D., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2017). Hindsight Experience Replay. In *Neural Information Processing Systems*.

Anthropic (2024). Developing a Computer Use Model. https://www.anthropic.com/news/developing-computer-use.

Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., and Hochreiter, S. (2018). RUDDER: Return Decomposition for Delayed Rewards. In *Neural Information Processing Systems*.

Auer, P., Jaksch, T., and Ortner, R. (2008). Near-optimal Regret Bounds for Reinforcement Learning. In *Neural Information Processing Systems*.

Bacon, P. and Precup, D. (2016). A Matrix Splitting Perspective on Planning with Options. *arXiv*.

Bacon, P.-L. (2013). On the Bottleneck Concept for Options Discovery: Theoretical Underpinnings and Extension in Continuous State Spaces. Master's thesis, McGill University.

Bacon, P.-L. (2018). *Temporal representation learning*. PhD thesis, McGill University.

Bacon, P.-L., Harb, J., and Precup, D. (2017). The Option-Critic Architecture. *The Association for the Advancement of Artificial Intelligence*.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., and Blundell, C. (2020a). Agent57: Outperforming the Atari Human Benchmark. In *International Conference on Machine Learning*.

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., and Blundell, C. (2020b). Never Give Up: Learning Directed Exploration Strategies. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Bagaria, A. (2025). *Skill Discovery for Exploration and Planning*. Doctor of philosophy, Brown University.

Bagaria, A., Abbatematteo, B. M., Gottesman, O., Corsaro, M., Rammohan, S., and Konidaris, G. (2023). Effectively Learning Initiation Sets in Hierarchical Reinforcement Learning. In *Neural Information Processing Systems*.

Bagaria, A., Koch, A. D. M., and Konidaris, G. (2025a). Going Beyond State-Reaching: Learning Abstractions for Intrinsically Motivated Skill Discovery. In *Multi-Disciplinary Conference on Reinforcement Learning and Decision Making*.

Bagaria, A., Koch, A. D. M., Rodriguez-Sanchez, R., Lobel, S., and Konidaris, G. (2025b). Intrinsically Motivated Discovery of Temporally Abstract Graph-based Models of the World. *Reinforcement Learning Journal*.

Bagaria, A. and Konidaris, G. (2020). Option Discovery using Deep Skill Chaining. In *International Conference on Learning Representations*.

Bagaria, A. and Schaul, T. (2023). Scaling Goal-based Exploration via Pruning Proto-goals. In *International Joint Conference on Artificial Intelligence*.

Bagaria, A., Senthil, J., Slivinski, M., and Konidaris, G. (2021a). Robustly learning composable options in deep reinforcement learning. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*.

Bagaria, A., Senthil, J. K., and Konidaris, G. (2021b). Skill discovery for exploration and planning using deep skill graphs. In *International Conference on Machine Learning*.

Bahdanau, D., Hill, F., Leike, J., Hughes, E., Hosseini, S., Kohli, P., and Grefenstette, E. (2018). Learning to Understand Goal Specifications by Modelling Reward. In *International Conference on Learning Representations*.

Bai, H., Zhou, Y., Cemri, M., Pan, J., Suhr, A., Levine, S., and Kumar, A. (2024). DigiRL: Training In-The-Wild Device-Control Agents with Autonomous Reinforcement Learning. *Neural Information Processing Systems*.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., Chen, C., Olsson, C., Olah, C., Hernandez, D., Drain, D., Ganguli, D., Li, D., Tran-Johnson, E., Perez, E., Kerr, J., Mueller, J., Ladish, J., Landau, J., Ndousse, K., Lukošiūtė, K., Lovitt, L., Sellitto, M., Elhage, N., Schiefer, N., Mercado, N., DasSarma, N., Lasenby, R., Larson, R., Ringer, S., Johnston, S., Kravec, S., Showk, S. E., Fort, S., Lanham, T., Telleen-Lawton, T., Conerly, T., Henighan, T. J., Hume, T., Bowman, S., Hatfield-Dodds, Z., Mann, B., Amodei, D., Joseph, N., McCandlish, S., Brown, T. B., and Kaplan, J. (2022). Constitutional AI: Harmlessness from AI Feedback. *arXiv*.

Baranes, A. and Oudeyer, P.-Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61:49–73.

Barreto, A., Borsa, D., Hou, S., Comanici, G., Aygün, E., Hamel, P., Toyama, D., Hunt, J. J., Mourad, S., Silver, D., and Precup, D. (2019a). The Option Keyboard: Combining Skills in Reinforcement Learning. *Neural Information Processing Systems*.

Barreto, A., Borsa, D., Quan, J., Schaul, T., Silver, D., Hessel, M., Mankowitz, D. J., Zídek, A., and Munos, R. (2019b). Transfer in Deep Reinforcement Learning Using Successor Features and Generalised Policy Improvement. *International Conference on Machine Learning*.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. (2017). Successor features for transfer in reinforcement learning. *Neural Information Processing Systems*.

Barreto, A., Hou, S., Borsa, D., Silver, D., and Precup, D. (2020). Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117:30079–30087.

Barto, A., Singh, S., and Chentanez, N. (2004). Intrinsically motivated learning of hierarchical collections of skills. *Proceedings of the 3rd International Conference on Development and Learning*.

Barto, A. G. and Şimşek, Ö. (2005). Intrinsic motivation for reinforcement learning systems. In *Proceedings of the thirteenth yale workshop on adaptive and learning systems*, pages 113–118. Yale University Press New Haven, CO, USA.

Bastani, O., Pu, Y., and Solar-Lezama, A. (2018). Verifiable reinforcement learning via policy extraction. *Neural Information Processing Systems*.

Bauer, J., Baumli, K., Behbahani, F. M. P., Bhoopchand, A., Bradley-Schmieg, N., Chang, M., Clay, N., Collister, A., Dasagi, V., Gonzalez, L., Gregor, K., Hughes, E., Kashem, S., Loks-Thompson, M., Openshaw, H., Parker-Holder, J., Pathak, S., Nieves, N. P., Rakicevic, N., Rocktäschel, T., Schroecker, Y., Singh, S., Sygnowski, J., Tuyls, K., York, S., Zacherl, A., and Zhang, L. M. (2023). Human-Timescale Adaptation in an Open-Ended Task Space. In *International Conference on Machine Learning*.

Baumli, K., Baveja, S., Behbahani, F. M. P., Chan, H., Comanici, G., Flennerhag, S., Gazeau, M., Holsheimer, K., Horgan, D., Laskin, M., Lyle, C., Masoom, H., McKinney, K., Mnih, V., Neitz, A., Pardo, F., Parker-Holder, J., Quan, J., Rocktaschel, T., Sahni, H., Schaul, T., Schroecker, Y., Spencer, S., Steigerwald, R., Wang, L., and Zhang, L. (2023). Vision-Language Models as a Source of Rewards. *arXiv*.

Baumli, K., Warde-Farley, D., Hansen, S., and Mnih, V. (2021). Relative Variational Intrinsic Control. In *The Association for the Advancement of Artificial Intelligence*.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. (2016). DeepMind Lab. *arXiv*.

Beck, J., Vuorio, R., Liu, E. Z., Xiong, Z., Zintgraf, L. M., Finn, C., and Whiteson, S. (2023). A Survey of Meta-Reinforcement Learning. *Foundations and Trends in Machine Learning*.

Bellemare, M., Naddaf, Y., Veness, J., and Bowling, M. (2012). The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. (2020). Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588:77–82.

Bellemare, M. G., Ostrovski, G., Guez, A., Thomas, P. S., and Munos, R. (2016a). Increasing the Action Gap: New Operators for Reinforcement Learning. In *The Association for the Advancement of Artificial Intelligence*.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016b). Unifying Count-Based Exploration and Intrinsic Motivation. In *Neural Information Processing Systems*.

Bellman, R. (1957). Dynamic Programming. *Science*, 153:34–37.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *International Conference on Machine Learning*.

Berlyne, D. (1965). *Conflict, Arousal and Curiosity*. McGraw Hill.

Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control, Two Volume Set*. Athena Scientific.

Boutilier, C., Dearden, R., and Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107.

Bradley, R. A. and Terry, M. E. (1952). Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*, 39:324.

Bradtke, S. J. and Duff, M. O. (1994). Reinforcement Learning Methods for Continuous-Time Markov Decision Problems. In *Neural Information Processing Systems*.

Brohan, A., Brown, N., and et al, J. C. (2023). RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Conference on Robot Learning*.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T. J., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. *Neural Information Processing Systems*.

Brunskill, E. and Li, L. (2014). PAC-inspired Option Discovery in Lifelong Reinforcement Learning. In *International Conference on Machine Learning*.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2019). Exploration by Random Network Distillation. In *International Conference on Learning Representations*.

Burridge, R. R., Rizzi, A. A., and Koditschek, D. E. (1999). Sequential Composition of Dynamically Dexterous Robot Behaviors. *International Journal Robotics Research*, 18(6):534–555.

Campero, A., Raileanu, R., Kuttler, H., Tenenbaum, J. B., Rocktäschel, T., and Grefenstette, E. (2021). Learning with {AMIG}o: Adversarially Motivated Intrinsic Goals. In *International Conference on Learning Representations*.

Campos, V., Trott, A., Xiong, C., Socher, R., Giro-I-Nieto, X., and Torres, J. (2020). Explore, Discover and Learn: Unsupervised Discovery of State-Covering Skills. In *International Conference on Machine Learning*.

Carpenter, G. A. and Grossberg, S. (1988). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision Graphics Image Processing*, 37:54–115.

Carvalho, T. H., Tjhia, K., and Lelis, L. H. S. (2024). Reclaiming the Source of Programmatic Policies: Programmatic versus Latent Spaces. In *International Conference on Learning Representations*.

Carvalho, W. T., Filos, A., Lewis, R., Lee, H., and Singh, S. (2023). Composing Task Knowledge with Modular Successor Feature Approximators. In *International Conference on Learning Representations*.

Castro, P. and Precup, D. (2010). Using bisimulation for policy transfer in MDPs. In *The Association for the Advancement of Artificial Intelligence*.

Castro, P. S. (2020). Scalable methods for computing state similarity in deterministic markov decision processes. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 10069–10076.

Chen, B., Zhu, C., Agrawal, P., Zhang, K., and Gupta, A. (2023). Self-Supervised Reinforcement Learning that Transfers using Random Features. In *Neural Information Processing Systems*.

Chevalier-Boisvert, M., Dai, B., Towers, M., Perez-Vicente, R. D. L., Willems, L., Lahlou, S., Pal, S., Castro, P. S., and Terry, J. K. (2023). Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks. In *Neural Information Processing Systems*.

Choi, J., Sharma, A., Lee, H., Levine, S., and Gu, S. S. (2021). Variational Empowerment as Representation Learning for Goal-Based Reinforcement Learning. In *International Conference on Machine Learning*.

Chua, R., Ghosh, A., Kaplanis, C., Richards, B. A., and Precup, D. (2024). Learning Successor Features the Simple Way. In *Neural Information Processing Systems*.

Chung, F. R. (1997). *Spectral graph theory*. American Mathematical Society.

Colas, C., Akakzia, A., Oudeyer, P.-Y., Chetouani, M., and Sigaud, O. (2020a). Language-Conditioned Goal Generation: a New Approach to Language Grounding for RL. *arXiv*.

Colas, C., Karch, T., Lair, N., Dussoux, J.-M., Moulin-Frier, C., Dominey, P. F., and Oudeyer, P.-Y. (2020b). Language as a Cognitive Tool to Imagine Goals in Curiosity-Driven Exploration. *Neural Information Processing Systems*.

Colas, C., Karch, T., Sigaud, O., and Oudeyer, P.-Y. (2020c). Autotelic Agents with Intrinsically Motivated Goal-Conditioned Reinforcement Learning: A Short Survey. *Journal of Artificial Intelligence Research*, 74:1159–1199.

Colas, C., Karch, T., Sigaud, O., and Oudeyer, P.-Y. (2022). Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research*, 74:1159–1199.

Colas, C., Oudeyer, P.-Y., Sigaud, O., Fournier, P., and Chetouani, M. (2018). CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning. In *International Conference on Machine Learning*.

Colas, C., Teodorescu, L., Oudeyer, P.-Y., Yuan, X., and Côté, M.-A. (2023). Augmenting Autotelic Agents with Large Language Models. In *Conference on Lifelong Learning Agents*.

Colburn, T. R. and Shute, G. M. (2007). Abstraction in Computer Science. *Minds and Machines*, 17(2):169–184.

Coulom, R. (2006). Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *Computers and Games*.

Cui, Y., Niekum, S., Gupta, A., Kumar, V., and Rajeswaran, A. (2022). Can Foundation Models Perform Zero-Shot Task Specification For Robot Manipulation? In *Conference on Learning for Dynamics & Control*.

Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. (2015). Robots that can adapt like animals. *Nature*, 521(7553):503–507.

da Silva, B. C., Konidaris, G. D., and Barto, A. G. (2012). Learning Parameterized Skills. In *International Conference on Machine Learning*.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Association for Computational Linguistics*.

Daniel, C., van Hoof, H., Peters, J., and Neumann, G. (2016). Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104:337–357.

Dayan, P. (1993). Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation*, 5(4):613–624.

Dayan, P. and Hinton, G. E. (1993). Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271–278.

Denil, M., Colmenarejo, S. G., Cabi, S., Saxton, D., and De Freitas, N. (2017). Programmable agents. *Neural Information Processing Systems*.

Dennis, M., Jaques, N., Vinitsky, E., Bayen, A. M., Russell, S. J., Critch, A., and Levine, S. (2020). Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design. *Neural Information Processing Systems*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the Association for Computational Linguistics*.

Dietterich, T. G. et al. (1998). The MAXQ Method for Hierarchical Reinforcement Learning. In *International Conference on Machine Learning*.

Ding, L., Zhang, J., Clune, J., Spector, L., and Lehman, J. (2024). Quality Diversity through Human Feedback. *International Conference on Machine Learning*.

Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., and Sui, Z. (2022). A survey on in-context learning. *Empirical Methods in Natural Language Processing*.

Du, Y., Konyushkova, K., Denil, M., Raju, A. S., Landon, J., Hill, F., de Freitas, N., and Cabi, S. (2023a). Vision-Language Models as Success Detectors. *Conference on Lifelong Learning Agents*.

Du, Y., Kosoy, E., Dayan, A., Rufova, M., Abbeel, P., and Gopnik, A. (2023b). What can AI Learn from Human Exploration? Intrinsically-Motivated Humans and Agents in Open-World Exploration. In *Intrinsically-Motivated and Open-Ended Learning Workshop, Neural Information Processing Systems*.

Du, Y., Watkins, O., Wang, Z., Colas, C., Darrell, T., Abbeel, P., Gupta, A., and Andreas, J. (2023c). Guiding Pretraining in Reinforcement Learning with Large Language Models. In *International Conference on Machine Learning*.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. (2016). Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv*.

Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., Mann, T., Weber, T., Degris, T., and Coppin, B. (2015). Deep reinforcement learning in large discrete action spaces. *arXiv*.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. (2020). First return, then explore. *Nature*, 590:580–586.

Erraqabi, A., Machado, M. C., Zhao, M., Sukhbaatar, S., Lazaric, A., Denoyer, L., and Bengio, Y. (2022). Temporal abstractions-augmented temporally contrastive learning: An alternative to the Laplacian in RL. In *Uncertainty in Artificial Intelligence*.

Evans, J. B. and Şimşek, Ö. (2023). Creating Multi-Level Skill Hierarchies in Reinforcement Learning. In *Neural Information Processing Systems*.

Even-Dar, E., Mannor, S., Mansour, Y., and Mahadevan, S. (2006). Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(6).

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. (2019). Diversity is All You Need: Learning Skills without a Reward Function. In *International Conference on Learning Representations*.

Faldor, M., Zhang, J., Cully, A., and Clune, J. (2025). OMNI-EPIC: Open-endedness via Models of human Notions of Interestingness with Environments Programmed in Code. In *International Conference on Learning Representations*.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D., Zhu, Y., and Anandkumar, A. (2022). MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge. In *Neural Information Processing Systems*.

Fang, M., Zhou, C., Shi, B., Gong, B., Xu, J., and Zhang, T. (2018). DHER: Hindsight experience replay for dynamic goals. In *International Conference on Learning Representations*.

Fang, M., Zhou, T., Du, Y., Han, L., and Zhang, Z. (2019). Curriculum-guided hindsight experience replay. *Neural Information Processing Systems*.

Farebrother, J., Greaves, J., Agarwal, R., Lan, C. L., Goroshin, R., Castro, P. S., and Bellemare, M. G. (2023). Proto-Value Networks: Scaling Representation Learning with Auxiliary Tasks. In *International Conference on Learning Representations*.

Ferns, N., Panangaden, P., and Precup, D. (2004). Metrics for finite Markov decision processes. In *Uncertainty in Artificial Intelligence*.

Ferns, N., Panangaden, P., and Precup, D. (2011). Bisimulation metrics for continuous Markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714.

Fikes, R. and Nilsson, N. J. (1971). STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3/4):189–208.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *International Conference on Machine Learning*.

Florensa, C., Held, D., Geng, X., and Abbeel, P. (2018). Automatic Goal Generation for Reinforcement Learning Agents. In *International Conference on Machine Learning*.

Ford, L. R. and Fulkerson, D. R. (1962). *Flows in Networks*. Princeton University Press.

Forestier, S., Mollard, Y., and Oudeyer, P.-Y. (2017). Intrinsically Motivated Goal Exploration Processes with Automatic Curriculum Learning. *J. Mach. Learn. Res.*, 23:152:1–152:41.

Forestier, S. and Oudeyer, P.-Y. (2016). Modular active curiosity-driven discovery of tool use. *International Conference on Intelligent Robots and Systems*.

Fox, R., Krishnan, S., Stoica, I., and Goldberg, K. (2017). Multi-level discovery of deep options. *arXiv*.

Frans, K., Ho, J., Chen, X., Abbeel, P., and Schulman, J. (2018). Meta Learning Shared Hierarchies. In *International Conference on Learning Representations*.

Fu, H., Yu, S., Tiwari, S., Littman, M., and Konidaris, G. (2023). Meta-learning Parameterized Skills. In *International Conference on Machine Learning*.

Fu, J., Korattikara, A., Levine, S., and Guadarrama, S. (2019). From Language to Goals: Inverse Reinforcement Learning for Vision-Based Instruction Following. In *International Conference on Learning Representations*.

Fu, Y., Zhang, H., Wu, D., Xu, W., and Boulet, B. (2024). FuRL: Visual-Language Models as Fuzzy Rewards for Reinforcement Learning. In *International Conference on Machine Learning*.

Fulda, N., Ricks, D., Murdoch, B., and Wingate, D. (2017). What can you do with a rock? affordance extraction via word embeddings. *International Joint Conference on Artificial Intelligence*.

Furelos-Blanco, D., Law, M., Jonsson, A., Broda, K., and Russo, A. (2023). Hierarchies of reward machines. In *International Conference on Machine Learning*, pages 10494–10541. PMLR.

Gao, C.-X., Wu, C., Cao, M., Kong, R., Zhang, Z., and Yu, Y. (2024). Act: empowering decision transformer with dynamic programming via advantage conditioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12127–12135.

Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. (2019). Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*.

Gomez, D., Bowling, M., and Machado, M. C. (2023). Proper Laplacian Representation Learning. *International Conference on Learning Representations*.

Gopnik, A. (2024). Empowerment as Causal Learning, Causal Learning as Empowerment: A bridge between Bayesian causal hypothesis testing and reinforcement learning. *Intrinsically Motivated Open-ended Learning Workshop, Neural Information Processing Systems*.

Gopnik, A., Meltzoff, A., and Kuhl, P. (2009). *The Scientist in the Crib: What Early Learning Tells Us About the Mind*. HarperCollins.

Gopnik, A. and Wellman, H. M. (2012). Reconstructing constructivism: causal models, Bayesian learning mechanisms, and the theory theory. *Psychological bulletin*, 138(6):1085.

Grauman, K., Westbury, A., Byrne, E., Chavis, Z., Furnari, A., Girdhar, R., and et al., J. H. (2021). Ego4D: Around the World in 3,000 Hours of Egocentric Video. *Conference on Computer Vision and Pattern Recognition*.

Gregor, K., Rezende, D. J., and Wierstra, D. (2017). Variational Intrinsic Control. In *International Conference on Learning Representations*.

Guo, Z. D., Thakoor, S., Pislar, M., Pires, B. Á., Altch'e, F., Tallec, C., Saade, A., Calandriello, D., Grill, J.-B., Tang, Y., Valḱo, M., Munos, R., Azar, M. G., and Piot, B. (2022). Byol-explore: Exploration by bootstrapped prediction. *ArXiv*.

Gupta, A., Kumar, V., Lynch, C., Levine, S., and Hausman, K. (2019). Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *Conference on Robot Learning*.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. (2018). Meta-Reinforcement Learning of Structured Exploration Strategies. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Haarnoja, T., Pong, V. H., Zhou, A., Dalal, M., Abbeel, P., and Levine, S. (2018). Composable Deep Reinforcement Learning for Robotic Manipulation. *International Conference on Robotics and Automation*.

Hafner, D., Lee, K.-H., Fischer, I., and Abbeel, P. (2022). Deep Hierarchical Planning from Pixels. In *Neural Information Processing Systems*.

Hafner, D., Ortega, P. A., Ba, J., Parr, T., Friston, K. J., and Heess, N. (2020). Action and Perception as Divergence Minimization. In *International Conference on Learning Representations*.

Hansen, S., Dabney, W., Barreto, A., Warde-Farley, D., de Wiele, T. V., and Mnih, V. (2020). Fast Task Inference with Variational Intrinsic Successor Features. In *International Conference on Learning Representations*.

Hansen-Estruch, P., Zhang, A., Nair, A., Yin, P., and Levine, S. (2022). Bisimulation makes analogies in goal-conditioned reinforcement learning. In *International Conference on Machine Learning*.

Harb, J., Bacon, P.-L., Klissarov, M., and Precup, D. (2018). When Waiting is not an Option : Learning Options with a Deliberation Cost. *The Association for the Advancement of Artificial Intelligence*.

Harlow, H. (1950). Learning and satiation of response in intrinsically motivated complex puzzle performance by monkeys. In *Journal of comparative and physiological psychology*.

Harutyunyan, A., Dabney, W., Borsa, D., Heess, N., Munos, R., and Precup, D. (2019a). The termination critic. *International Conference on Artificial Intelligence and Statistics*.

Harutyunyan, A., Vrancx, P., Hamel, P., Nowé, A., and Precup, D. (2019b). Per-Decision Option Discounting. In *International Conference on Machine Learning*.

Hasanbeig, M., Jeppu, N. Y., Abate, A., Melham, T., and Kroening, D. (2021). DeepSynth: Automata synthesis for automatic task segmentation in deep reinforcement learning. In *The Association for the Advancement of Artificial Intelligence*.

Hengst, B. et al. (2002). Discovering hierarchy in reinforcement learning with HEXQ. In *International Conference on Machine Learning*.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2016). beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*.

Hinton, G. E. and Zemel, R. (1993). Autoencoders, Minimum Description Length and Helmholtz Free Energy. In *Neural Information Processing Systems*.

Ho, M. K., Abel, D., Griffiths, T. L., and Littman, M. L. (2019). The value of abstraction. *Current opinion in behavioral sciences*, 29:111–116.

Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Association for Computing Machinery*, 12(10):576–580.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9:1735–1780.

Honkela, A. and Valpola, H. (2004). Variational learning and bits-back coding: an information-theoretic view to Bayesian learning. *IEEE Transactions on Neural Networks*, 15(4):800–810.

Hu, X. and Leung, H.-f. (2023). Provably (More) Sample-Efficient Offline RL with Options. In *Neural Information Processing Systems*.

Huang, T., Chen, K., Wei, W., Li, J., Long, Y., and Dou, Q. (2023). Value-Informed Skill Chaining for Policy Learning of Long-Horizon Tasks with Surgical Robot. In *International Conference on Intelligent Robots and Systems*.

Hughes, E., Dennis, M., Parker-Holder, J., Behbahani, F. M. P., Mavalankar, A., Shi, Y., Schaul, T., and Rocktaschel, T. (2024). Open-Endedness is Essential for Artificial Superhuman Intelligence. *International Conference on Machine Learning*.

Hung, C.-C., Lillicrap, T., Abramson, J., Wu, Y., Mirza, M., Carnevale, F., Ahuja, A., and Wayne, G. (2019). Optimizing agent behavior over long time scales by transporting value. *Nature Communications*, 10:5223.

Hunt, J. J., Barreto, A., Lillicrap, T. P., and Heess, N. M. O. (2018). Entropic Policy Composition with Generalized Policy Improvement and Divergence Correction. *International Conference on Machine Learning*.

Hutsebaut-Buysse, M., Mets, K., and Latré, S. (2022). Hierarchical Reinforcement Learning: A Survey and Open Research Challenges. *Machine Learning and Knowledge Extraction*, 4(1):172–221.

Icarte, R. T., Klassen, T. Q., Valenzano, R., and McIlraith, S. A. (2022). Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208.

Ivanov, A., Bagaria, A., and Konidaris, G. (2024). Discovering Options that Minimize Average Planning Time. In *The Association for the Advancement of Artificial Intelligence*.

Jain, A., Khetarpal, K., and Precup, D. (2018). Safe option-critic: learning safety in the option-critic architecture. *The Knowledge Engineering Review*, 36.

James, S. D., Rosman, B., and Konidaris, G. D. (2019). Learning portable representations for high-level planning. In *International Conference on Machine Learning*.

Janner, M., Fu, J., Zhang, M., and Levine, S. (2019). When to Trust Your Model: Model-Based Policy Optimization. In *Neural Information Processing Systems*.

Jaques, N., Lazaridou, A., Hughes, E., Gülçehre, Ç., Ortega, P. A., Strouse, D., Leibo, J. Z., and de Freitas, N. (2019). Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning. In *International Conference on Machine Learning*.

Jiang, M., Dennis, M., Parker-Holder, J., Foerster, J. N., Grefenstette, E., and Rocktaschel, T. (2021). Replay-Guided Adversarial Environment Design. In *Neural Information Processing Systems*.

Jiang, Y., Gu, S. S., Murphy, K. P., and Finn, C. (2019). Language as an Abstraction for Hierarchical Deep Reinforcement Learning. *Neural Information Processing Systems*.

Jiang, Y., Liu, E., Eysenbach, B., Kolter, J. Z., and Finn, C. (2022). Learning Options via Compression. *Neural Information Processing Systems*.

Jiang, Z., Zhang, T., Janner, M., Li, Y., Rocktäschel, T., Grefenstette, E., and Tian, Y. (2023). Efficient planning in a compact latent action space. *International Conference on Learning Representations*.

Jinnai, Y., Abel, D., Hershkowitz, D., Littman, M., and Konidaris, G. (2019a). Finding Options that Minimize Planning Time. In *International Conference on Machine Learning*.

Jinnai, Y., Park, J. W., Abel, D., and Konidaris, G. (2019b). Discovering Options for Exploration by Minimizing Cover Time. In *International Conference on Machine Learning*.

Jinnai, Y., Park, J. W., Machado, M. C., and Konidaris, G. (2020). Exploration in Reinforcement Learning with Deep Covering Options. In *International Conference on Learning Representations*.

Johnson, M., Hofmann, K., Hutton, T., and Bignell, D. (2016). The Malmö Platform for Artificial Intelligence Experimentation. In *International Joint Conference on Artificial Intelligence*.

Jong, N. K., Hester, T., and Stone, P. (2008). The Utility of Temporal Abstraction in Reinforcement Learning. In *International Joint Conference on Autonomous Agents and Multiagent Systems*.

Jonsson, A. and Barto, A. G. (2006). Causal Graph Based Decomposition of Factored MDPs. *Journal of Machine Learning Research*, 7:2259–2301.

Jothimurugan, K., Alur, R., and Bastani, O. (2019). A composable specification language for reinforcement learning tasks. *Neural Information Processing Systems*.

Kaelbling, L. P. (1993a). Learning to Achieve Goals. In *International Joint Conference on Artificial Intelligence*.

Kaelbling, L. P. (1993b). Learning to achieve goals. In *International Joint Conference on Artificial Intelligence*.

Kahneman, D. (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

Kamat, A. and Precup, D. (2020). Diversity-Enriched Option-Critic. *arXiv*.

Kanervisto, A., Milani, S., Ramanauskas, K., Topin, N., Lin, Z., Li, J., yong Shi, J., Ye, D., Fu, Q., Yang, W., Hong, W., Huang, Z.-H., Chen, H., Zeng, G., Lin, Y., Micheli, V., Alonso, E., Fleuret, F., Nikulin, A., Belousov, Y., Svidchenko, O., and Shpilman, A. (2021). MineRL Diamond 2021 Competition: Overview, Results, and Lessons Learned. *Neural Information Processing Systems*.

Kang, M. and Oh, S. (2022). Deep latent-space sequential skill chaining from incomplete demonstrations. *Intelligent Serice Robotics*, 15(2):203–213.

Kaplan, F. and Oudeyer, P.-Y. (2003). Maximizing Learning Progress: An Internal Reward System for Development. In *Embodied Artificial Intelligence*.

Kazemitabar, S. J. and Beigy, H. (2009). Using Strongly Connected Components as a Basis for Autonomous Skill Acquisition in Reinforcement Learning. In *Advances in Neural Networks*.

Kearns, M. and Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232.

Khetarpal, K., Ahmed, Z., Comanici, G., Abel, D., and Precup, D. (2020a). What can i do here? A theory of affordances in reinforcement learning. In *International Conference on Machine Learning*.

Khetarpal, K., Ahmed, Z., Comanici, G., and Precup, D. (2021). Temporally abstract partial models. *Advances in Neural Information Processing Systems*.

Khetarpal, K., Klissarov, M., Chevalier-Boisvert, M., Bacon, P.-L., and Precup, D. (2020b). Options of interest: Temporal abstraction with interest functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4444–4451.

Khetarpal, K., Riemer, M., Rish, I., and Precup, D. (2020c). Towards Continual Reinforcement Learning: A Review and Perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476.

Kim, H. and Mnih, A. (2018). Disentangling by Factorising. In *International Conference on Machine Learning*.

Kim, T., Ahn, S., and Bengio, Y. (2019). Variational temporal abstraction. *Neural Information Processing Systems*.

Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*.

Kipf, T., Li, Y., Dai, H., Zambaldi, V., Sanchez-Gonzalez, A., Grefenstette, E., Kohli, P., and Battaglia, P. (2019). Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*.

Klissarov, M., Bacon, P.-L., Harb, J., and Precup, D. (2017). Learning options end-to-end for continuous action tasks. *Hierarchical Reinforcement Learning Workshop, Neural Information Processing Systems*.

Klissarov, M., D'Oro, P., Sodhani, S., Raileanu, R., Bacon, P.-L., Vincent, P., Zhang, A., and Henaff, M. (2024). Motif: Intrinsic Motivation from Artificial Intelligence Feedback. *International Conference on Learning Representations*.

Klissarov, M., D'Oro, P., Sodhani, S., Raileanu, R., Bacon, P.-L., Vincent, P., Zhang, A., and Henaff, M. (2025a). MaestroMotif: Skill Design from Artificial Intelligence Feedback. In *International Conference on Learning Representations*.

Klissarov, M., Hjelm, D., Toshev, A., and Mazoure, B. (2025b). On the Modeling Capabilities of Large Language Models for Sequential Decision Making. In *International Conference on Learning Representations*.

Klissarov, M. and Machado, M. C. (2023). Deep Laplacian-based Options for Temporally-Extended Exploration. In *International Conference on Machine Learning*.

Klissarov, M. and Precup, D. (2020). Reward Propagation Using Graph Convolutional Networks. *Neural Information Processing Systems*.

Klissarov, M. and Precup, D. (2021). Flexible Option Learning. In *Neural Information Processing Systems*.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2005). All Else Being Equal Be Empowered. In *Advances in Artificial Life*.

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Deitke, M., Ehsani, K., Gordon, D., Zhu, Y., Kembhavi, A., Gupta, A. K., and Farhadi, A. (2017). AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*.

Kompella, V. R., Stollenga, M. F., Luciw, M. D., and Schmidhuber, J. (2017). Continual curiosity-driven skill acquisition from high-dimensional video inputs for humanoid robots. *Artificial Intelligence*, 247:313–335.

Konidaris, G. (2019). On the necessity of abstraction. *Current opinion in behavioral sciences*, 29:1–7.

Konidaris, G., Kuindersma, S., Grupen, R., and Barto, A. (2010). Constructing skill trees for reinforcement learning agents from demonstration trajectories. *Neural Information Processing Systems*.

Konidaris, G. D. and Barto, A. G. (2007). Building Portable Options: Skill Transfer in Reinforcement Learning. In *International Joint Conference on Artificial Intelligence*.

Konidaris, G. D. and Barto, A. G. (2009). Skill Discovery in Continuous Reinforcement Learning Domains using Skill Chaining. In *Neural Information Processing Systems*.

Konidaris, G. D., Kaelbling, L. P., and Lozano-Perez, T. (2018). From Skills to Symbols: Learning Symbolic Representations for Abstract High-Level Planning. *Journal of Artificial Intelligence Research*, 61:215–289.

Kovač, G., Laversanne-Finot, A., and Oudeyer, P.-Y. (2020). GRIMGEP: Learning Progress for Robust Goal Sampling in Visual Deep Reinforcement Learning. *IEEE Transactions on Cognitive and Developmental Systems*, 15:1396–1407.

Krishnan, S., Fox, R., Stoica, I., and Goldberg, K. (2017). Ddco: Discovery of deep continuous options for robot learning from demonstrations. In *Conference on Robot Learning*.

Kulkarni, T. D., Saeedi, A., Gautam, S., and Gershman, S. J. (2016). Deep Successor Reinforcement Learning. *arXiv*.

Küttler, H., Nardelli, N., Miller, A. H., Raileanu, R., Selvatici, M., Grefenstette, E., and Rocktäschel, T. (2020). The NetHack Learning Environment. In *Neural Information Processing Systems*.

Kwon, M., Xie, S. M., Bullard, K., and Sadigh, D. (2023a). Reward Design with Language Models. In *The Eleventh International Conference on Learning Representations*.

Kwon, T., Palo, N. D., and Johns, E. (2023b). Language Models as Zero-Shot Trajectory Generators. In *IEEE Robotics and Automation Letters*.

Laskin, M., Liu, H., Peng, X. B., Yarats, D., Rajeswaran, A., and Abbeel, P. (2022). CIC: Contrastive Intrinsic Control for Unsupervised Skill Discovery. *arXiv*.

LaValle, S. M. (1998). Rapidly-exploring random trees : a new tool for path planning. *The annual research report*.

Le, H. M., Jiang, N., Agarwal, A., Dudík, M., Yue, Y., and Daumé, H. (2018). Hierarchical Imitation and Reinforcement Learning. *International Conference on Machine Learning*.

Lee, Y., Lim, J. J., Anandkumar, A., and Zhu, Y. (2021). Adversarial Skill Chaining for Long-Horizon Robot Manipulation via Terminal State Regularization. In *Conference on Robot Learning*.

Lehman, J. and Stanley, K. O. (2011). Evolving a diversity of virtual creatures through novelty search and local competition. Genetic and Evolutionary Computation Conference.

Lehnert, L., Laroche, R., and van Seijen, H. (2018). On Value Function Representation of Long Horizon Problems. In *The Association for the Advancement of Artificial Intelligence*.

Leibfried, F., Pascual-Díaz, S., and Grau-Moya, J. (2019). A Unified Bellman Optimality Principle Combining Reward Maximization and Empowerment. In *Neural Information Processing Systems*.

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Levy, A., Konidaris, G., Platt, R., and Saenko, K. (2019). Hierarchical Reinforcement Learning with Hindsight. In *International Conference on Learning Representations*.

Levy, A., Rammohan, S., Allievi, A., Niekum, S., and Konidaris, G. (2023). Hierarchical Empowerment: Towards Tractable Empowerment-Based Skill Learning.

Li, H., Yang, X., Wang, Z., Zhu, X., Zhou, J., Qiao, Y., Wang, X., Li, H., Lu, L., and Dai, J. (2023). Auto MC-Reward: Automated Dense Reward Design with Large Language Models for Minecraft. *Computer Vision and Pattern Recognition*.

Li, L., Walsh, T. J., and Littman, M. L. (2006). Towards a unified theory of state abstraction for MDPs. *International Symposium on Artificial Intelligence and Mathematics*.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P. R., and Zeng, A. (2022). Code as Policies: Language Model Programs for Embodied Control. *2023 IEEE International Conference on Robotics and Automation*.

Lifshitz, S., Paster, K., Chan, H., Ba, J., and McIlraith, S. A. (2023). STEVE-1: A Generative Model for Text-to-Behavior in Minecraft. In *Neural Information Processing Systems*.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. (2023). Let's Verify Step by Step. *arXiv*.

Lin, L.-J. (1991). Self-improvement Based On Reinforcement Learning, Planning and Teaching. In Birnbaum, L. A. and Collins, G. C., editors, *Machine Learning Proceedings 1991*, pages 323–327. Morgan Kaufmann.

Lin, Y.-A., Lee, C.-T., Yang, C.-H., Liu, G.-T., and Sun, S.-H. (2024). Hierarchical Programmatic Option Framework. In *Neural Information Processing Systems*.

Lindemann, S. R. and LaValle, S. M. (2004). Incrementally Reducing Dispersion by Increasing Voronoi Bias in RRTs. In *International Conference on Robotics and Automation*, volume 4, pages 3251–3257.

Liu, G.-T., Hu, E.-P., Cheng, P.-J., Lee, H.-Y., and Sun, S.-H. (2023a). Hierarchical programmatic reinforcement learning via learning to compose programs. In *International Conference on Machine Learning*, pages 21672–21697. PMLR.

Liu, H., Trott, A., Socher, R., and Xiong, C. (2019). Competitive experience replay. *International Conference on Learning Representations*.

Liu, J., Zu, L., He, L., and Wang, D. (2023b). Clue: Calibrated latent guidance for offline reinforcement learning. In *Conference on Robot Learning*.

Liu, M., Machado, M. C., Tesauro, G., and Campbell, M. (2017). The Eigenoption-Critic Framework. *Hierarchical Reinforcement Learning Workshop, Neural Information Processing Systems*.

Liu, M., Zhu, M., and Zhang, W. (2022). Goal-Conditioned Reinforcement Learning: Problems and Solutions. *International Joint Conference on Artificial Intelligence*.

Lo, C., Roice, K., Panahi, P. M., Jordan, S. M., White, A., Mihucz, G., Aminmansour, F., and White, M. (2024). Goal-Space Planning with Subgoal Models. *Journal of Machine Learning Research*, 25(330):1–57.

Lobel, S., Bagaria, A., and Konidaris, G. (2023). Flipping Coins to Estimate Pseudocounts for Exploration in Reinforcement Learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 22594–22613. PMLR.

Lozano-Perez, T., Mason, M. T., and Taylor, R. H. (1984). Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1):3–24.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J. N., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T. (2019). A Survey of Reinforcement Learning Informed by Natural Language. *International Joint Conference on Artificial Intelligence*.

Luo, Z., Ni, T., Bacon, P.-L., Precup, D., and Si, X. (2025). Understanding behavioral metric learning: A large-scale study on distracting reinforcement learning environments. *arXiv preprint arXiv:2506.00563*.

Luo, Z., Zhang, Y., and Wang, Z. (2023). Does Hierarchical Reinforcement Learning Outperform Standard Reinforcement Learning in Goal-Oriented Environments? In *Goal-Conditioned Reinforcement Learning Workshop, Neural Information Processing Systems*.

Ma, C., Ashley, D. R., Wen, J., and Bengio, Y. (2020). Universal Successor Features for Transfer Reinforcement Learning. *arXiv*.

Ma, Y. J., Liang, W., Wang, G., Huang, D.-A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L., and Anandkumar, A. (2024). Eureka: Human-Level Reward Design via Coding Large Language Models. *International Conference on Learning Representations*.

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

Machado, M. C., Barreto, A., and Precup, D. (2023). Temporal Abstraction in Reinforcement Learning with the Successor Representation. *Journal of Machine Learning Research*, 24(80):1–69.

Machado, M. C., Bellemare, M. G., and Bowling, M. (2017). A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*.

Machado, M. C., Bellemare, M. G., and Bowling, M. (2020). Count-based exploration with the successor representation. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Machado, M. C. and Bowling, M. H. (2016). Learning Purposeful Behaviour in the Absence of Rewards. *Abstraction in Reinforcement Learning Workshop, International Conference on Machine Learning*.

Machado, M. C., Rosenbaum, C., Guo, X., Liu, M., Tesauro, G., and Campbell, M. (2018). Eigenoption Discovery through the Deep Successor Representation. In *International Conference on Learning Representations*.

Mahadevan, S. (2005). Proto-value functions: Developmental reinforcement learning. In *International Conference on Machine Learning*.

Mahadevan, S. and Maggioni, M. (2007). Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, pages 2169–2231.

Mahmoudieh, P., Pathak, D., and Darrell, T. (2022). Zero-Shot Reward Specification via Grounded Natural Language. In *International Conference on Machine Learning*.

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., and State, G. (2021). Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. *Neural Information Processing Systems*.

Mannor, S., Menache, I., Hoze, A., and Klein, U. (2004). Dynamic abstraction in reinforcement learning via clustering. In *International Conference on Machine Learning*.

Mason, M. (1985). The Mechanics of Manipulation. In *IEEE International Conference on Robotics and Automation*.

Massari, F. S., Biehl, M., Meeden, L., and Kanai, R. (2021). Experimental Evidence that Empowerment May Drive Exploration in Sparse-Reward Environments. *IEEE International Conference on Development and Learning*.

Matiisen, T., Oliver, A., Cohen, T., and Schulman, J. (2017). Teacher–Student Curriculum Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 31:3732–3740.

McClinton, W., Levy, A., and Konidaris, G. D. (2021). HAC Explore: Accelerating Exploration with Hierarchical Reinforcement Learning. *arXiv*.

McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL—The Planning Domain Definition Language. Technical report, Yale Center for Computational Vision and Control.

McGovern, A. and Barto, A. G. (2001). Automatic discovery of subgoals in reinforcement learning using diverse density. In *International Conference on Machine Learning*.

Medeiros, L. C., Aleixo, D. S., and Lelis, L. H. S. (2022). What Can We Learn Even from the Weakest? Learning Sketches for Programmatic Strategies. In *The Association for the Advancement of Artificial Intelligence*.

Mees, O., Hermann, L., Rosete-Beas, E., and Burgard, W. (2022). CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks. *IEEE Robotics and Automation Letters*.

Menache, I., Mannor, S., and Shimkin, N. (2002). Q-cut—dynamic discovery of sub-goals in reinforcement learning. In *European Conference on Machine Learning*.

Metzen, J. H. (2012). Online Skill Discovery using Graph-based Clustering. In *European Workshop on Reinforcement Learning*.

Meyerson, A. and Tagiku, B. (2009). Minimizing average shortest path distances via shortcut edge addition. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 272–285. Springer.

Mishra, U. A., Xue, S., Chen, Y., and Xu, D. (2023). Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518:529–533.

Mohamed, S. and Rezende, D. J. (2015). Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning. In *Neural Information Processing Systems*.

Momennejad, I., Russek, E., Cheong, J. H., Botvinick, M., Daw, N., and Gershman, S. (2017). The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1:680—692.

Moraes, R. O., Aleixo, D. S., Ferreira, L. N., and Lelis, L. H. (2023). Choosing well your opponents: How to guide the synthesis of programmatic strategies. *International Joint Conference on Artificial Intelligence*.

Moraes, R. O. and Lelis, L. H. S. (2024). Searching for programmatic policies in semantic spaces. In *International Joint Conference on Artificial Intelligence*.

Moraes, R. O., Sadmine, Q. A., Baier, H., and Lelis, L. H. (2025). InnateCoder: Learning Programmatic Options with Foundation Models. *arXiv*.

Müller, M. (2007). *Dynamic Time Warping*, pages 69–84. Springer Berlin Heidelberg.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. (2018). Data-efficient hierarchical reinforcement learning. *Neural Information Processing Systems*.

Nair, S. and Finn, C. (2020). Hierarchical Foresight: Self-Supervised Learning of Long-Horizon Tasks via Visual Subgoal Generation. In *International Conference on Learning Representations*.

Nam, T., Sun, S., Pertsch, K., Hwang, S. J., and Lim, J. J. (2022). Skill-based Meta-Reinforcement Learning. In *International Conference on Learning Representations*.

Narayanamurthy, S. M. and Ravindran, B. (2008). On the hardness of finding symmetries in Markov decision processes. In *International Conference on Machine Learning*.

Nayyar, R. K. and Srivastava, S. (2024). Autonomous Option Invention for Continual Hierarchical Reinforcement Learning and Planning. *The Association for the Advancement of Artificial Intelligence*.

Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *American Physical Society*, 69:026113.

Ni, T., Eysenbach, B., Seyedsalehi, E., Ma, M., Gehring, C., Mahajan, A., and Bacon, P.-L. (2024). Bridging State and History Representations: Understanding Self-Predictive RL. *International Conference on Learning Representations*.

Ni, T., Ma, M., Eysenbach, B., and Bacon, P.-L. (2023). When Do Transformers Shine in RL? Decoupling Memory from Credit Assignment. In *Neural Information Processing Systems*.

OEL Team, O. E. L. T., Stooke, A., Mahajan, A., Barros, C., Deck, C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M., Mathieu, M., McAleese, N., Bradley-Schmieg, N., Wong, N., Porcel, N., Raileanu, R., Hughes-Fitt, S., Dalibard, V., and Czarnecki, W. M. (2021). Open-Ended Learning Leads to Generally Capable Agents. *arXiv*.

Oh, J., Hessel, M., Czarnecki, W. M., Xu, Z., van Hasselt, H. P., Singh, S., and Silver, D. (2020). Discovering reinforcement learning algorithms. *Neural Information Processing Systems*.

OpenAI (2024). O1 System Card. https://cdn.openai.com/o1-system-card-20240917.pdf.

OpenAI (2025). Computer-Using Agent. https://openai.com/index/introducing-operator/.

Oudeyer, P. and Kaplan, F. (2007). What is intrinsic motivation? A typology of computational approaches. *Frontiers Neurorobotics*, 1:6.

Palo, N. D., Byravan, A., Hasenclever, L., Wulfmeier, M., Heess, N. M. O., and Riedmiller, M. A. (2023). Towards A Unified Agent with Foundation Models. *Reincarnating Reinforcement Learning Workshop, International Conference on Learning Representations*.

Palo, N. D. and Johns, E. (2024). Keypoint Action Tokens Enable In-Context Imitation Learning in Robotics. *Robotics: Science and Systems Proceedings*.

Pan, J., Zhang, Y., Tomlin, N., Zhou, Y., Levine, S., and Suhr, A. (2024). Autonomous Evaluation and Refinement of Digital Agents. In *Conference on Language Modeling*.

Park, S., Choi, J., Kim, J., Lee, H., and Kim, G. (2022). Lipschitz-constrained Unsupervised Skill Discovery. In *International Conference on Learning Representations*.

Park, S., Frans, K., Eysenbach, B., and Levine, S. (2025a). OGBench: Benchmarking Offline Goal-Conditioned RL. *International Conference on Learning Representations*.

Park, S., Frans, K., Mann, D., Eysenbach, B., Kumar, A., and Levine, S. (2025b). Horizon reduction makes rl scalable. *arXiv*.

Park, S., Ghosh, D., Eysenbach, B., and Levine, S. (2024a). Hiql: Offline goal-conditioned rl with latent states as actions. *Neural Information Processing Systems*.

Park, S., Kreiman, T., and Levine, S. (2024b). Foundation policies with Hilbert representations. In *International Conference on Machine Learning*.

Park, S., Lee, K., Lee, Y., and Abbeel, P. (2023). Controllability-aware unsupervised skill discovery. In *International Conference on Machine Learning*.

Park, S., Rybkin, O., and Levine, S. (2024c). METRA: Scalable Unsupervised RL with Metric-Aware Abstraction. In *International Conference on Learning Representations*.

Parker-Holder, J., Jiang, M., Dennis, M., Samvelyan, M., Foerster, J. N., Grefenstette, E., and Rocktaschel, T. (2022). Evolving Curricula with Regret-Based Environment Design. *arXiv*.

Parr, R. and Russell, S. (1997). Reinforcement learning with hierarchies of machines. *Neural Information Processing Systems*.

Pateria, S., Subagdja, B., Tan, A.-H., and Quek, C. (2021). Hierarchical Reinforcement Learning: A Comprehensive Survey. *ACM Computing Surveys*, 54:1–35.

Paul, S., Vanbaar, J., and Roy-Chowdhury, A. (2019). Learning from trajectories via subgoal discovery. *Advances in Neural Information Processing Systems*, 32.

Pertsch, K., Lee, Y., and Lim, J. (2021). Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning*.

Pfau, D., Petersen, S., Agarwal, A., Barrett, D. G. T., and Stachenfeld, K. L. (2018). Spectral Inference Networks: Unifying Deep and Spectral Learning. In *International Conference on Learning Representations*.

Pignatelli, E., Ferret, J., Rockaschel, T., Grefenstette, E., Paglieri, D., Coward, S., and Toni, L. (2024). Assessing the Zero-Shot Capabilities of LLMs for Action Evaluation in RL. *Neural Information Processing Systems*.

Piray, P. and Daw, N. D. (2021). Linear reinforcement learning in planning, grid fields, and cognitive control. *Nature Communications*, 12(1):4942.

Pitis, S., Chan, H., Zhao, S., Stadie, B. C., and Ba, J. (2020). Maximum Entropy Gain Exploration for Long Horizon Multi-goal Reinforcement Learning. In *International Conference on Machine Learning*.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., and Zaremba, W. (2018). Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. *arXiv*.

Pong, V., Dalal, M., Lin, S., Nair, A., Bahl, S., and Levine, S. (2020). Skew-Fit: State-Covering Self-Supervised Reinforcement Learning. In *International Conference on Machine Learning*.

Portelas, R., Colas, C., Weng, L., Hofmann, K., and Oudeyer, P.-Y. (2020). Automatic Curriculum Learning For Deep RL: A Short Survey. In *International Joint Conference on Artificial Intelligence*.

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

Precup, D. and Sutton, R. S. (2000). Temporal abstraction in reinforcement learning. In *International Conference on Machine Learning*.

Precup, D., Sutton, R. S., and Singh, S. P. (2000). Eligibility Traces for Off-Policy Policy Evaluation. In *International Conference on Machine Learning*.

PrismarineJS (2013). PrismarineJS/mineflayer: Create Minecraft bots with a powerful, stable, and high level JavaScript API. https://github.com/PrismarineJS/mineflayer.

Puig, X., Undersander, E., Szot, A., Cote, M. D., Yang, T.-Y., Partsey, R., Desai, R., Clegg, A., Hlavac, M., Min, S. Y., Vondruš, V., Gervet, T., Berges, V.-P., Turner, J. M., Maksymets, O., Kira, Z., Kalakrishnan, M., Malik, J., Chaplot, D. S., Jain, U., Batra, D., Rai, A., and Mottaghi, R. (2024). Habitat 3.0: A Co-Habitat for Humans, Avatars, and Robots. In *International Conference on Learning Representations*.

Puterman, M. L. (1994). Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics.

Qiu, W. and Zhu, H. (2022). Programmatic reinforcement learning without oracles. In *International Conference on Learning Representations*.

Racanière, S., Lampinen, A. K., Santoro, A., Reichert, D. P., Firoiu, V., and Lillicrap, T. P. (2019). Automated curricula through setter-solver interactions. *International Conference on Learning Representations*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning*.

Ramesh, R., Tomar, M., and Ravindran, B. (2019). Successor Options: An Option Discovery Framework for Reinforcement Learning. In *International Joint Conference on Artificial Intelligence*.

Raparthy, S. C., Hambro, E., Kirk, R., Henaff, M., and Raileanu, R. (2023). Generalization to new sequential decision making tasks with in-context learning. *arXiv*.

Ravindran, B. (2003). SMDP homomorphisms: An algebraic approach to abstraction in semi markov decision processes. *International Joint Conference on Artificial Intelligence*.

Ravindran, B. (2004). *An algebraic approach to abstraction in reinforcement learning*. University of Massachusetts Amherst.

Ravindran, B. and Barto, A. G. (2001). *Symmetries and model minimization in markov decision processes*. PhD thesis, University of Massachusetts.

Ravindran, B. and Barto, A. G. (2002). Model minimization in hierarchical reinforcement learning. In *Abstraction, Reformulation, and Approximation: 5th International Symposium*.

Ravindran, B. and Barto, A. G. (2003). Relativized options: Choosing the right transformation. In *International Conference on Machine Learning*.

Ravindran, B. and Barto, A. G. (2004). Approximate homomorphisms: A framework for non-exact minimization in Markov decision processes.

Rawles, C., Li, A., Rodriguez, D., Riva, O., and Lillicrap, T. P. (2023). AndroidInTheWild: A Large-Scale Dataset For Android Device Control. In *Neural Information Processing Systems*.

Rezaei-Shoshtari, S., Zhao, R., Panangaden, P., Meger, D., and Precup, D. (2022). Continuous mdp homomorphisms and homomorphic policy gradient. *Neural Information Processing Systems*.

Riemer, M., Cases, I., Rosenbaum, C., Liu, M., and Tesauro, G. (2019). On the Role of Weight Sharing During Deep Option Learning. In *The Association for the Advancement of Artificial Intelligence*.

Riemer, M., Liu, M., and Tesauro, G. (2018). Learning Abstract Options. *Neural Information Processing Systems*.

Ring, M. B. (1995). *Continual learning in reinforcement environments*. PhD thesis, University of Texas.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5):465–471.

Rocamonde, J., Montesinos, V., Nava, E., Perez, E., and Lindner, D. (2024). Vision-Language Models are Zero-Shot Reward Models for Reinforcement Learning. *International Conference on Learning Representations*.

Rodriguez-Sanchez, R. and Konidaris, G. (2024). Learning Abstract World Models for Value-preserving Planning with Options. *Reinforcement Learning Journal*, 4:1733–1758.

Rovee-Collier, C. K. and Gekoski, M. J. (1979). The economics of infancy: A review of conjugate reinforcement. *Advances in child development and behavior*, 13:195–255.

Salge, C., Glackin, C., and Polani, D. (2014). Empowerment–an introduction. *Guided Self-Organization: Inception*, pages 67–114.

Salter, S., Wulfmeier, M., Tirumala, D., Heess, N., Riedmiller, M., Hadsell, R., and Rao, D. (2022). Mo2: Model-based offline options. In *Conference on Lifelong Learning Agents*.

Samvelyan, M., Khan, A., Dennis, M., Jiang, M., Parker-Holder, J., Foerster, J. N., Raileanu, R., and Rocktaschel, T. (2023). MAESTRO: Open-Ended Environment Design for Multi-Agent Reinforcement Learning. *International Conference on Learning Representations*.

Schaul, T., Borsa, D., Ding, D., Szepesvari, D., Ostrovski, G., Dabney, W., and Osindero, S. (2019). Adapting Behaviour for Learning Progress. *arXiv*.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). Universal value function approximators. In *International conference on machine learning*, pages 1312–1320.

Schmidhuber, J. (1987). Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Master's thesis, Technische Universitat Munchen.

Schmidhuber, J. (2004). Optimal ordered problem solver. *Machine Learning*, 54:211–254.

Schmidhuber, J. (2010). Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990-2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247.

Schmidhuber, J. (2011). PowerPlay: Training an Increasingly General Problem Solver by Continually Searching for the Simplest Still Unsolvable Problem. *Frontiers in Psychology*, 4.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv*.

Sharma, A., Ahn, M., Levine, S., Kumar, V., Hausman, K., and Gu, S. (2020a). Emergent Real-World Robotic Skills via Unsupervised Off-Policy Reinforcement Learning. In *Robotics: Science and Systems*.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. (2020b). Dynamics-Aware Unsupervised Discovery of Skills. In *International Conference on Learning Representations*.

Sharma, A., Sharma, M., Rhinehart, N., and Kitani, K. M. (2018). Directed-info gail: Learning hierarchical policies from unsegmented demonstrations using directed information. *International Conference on Learning Representations*.

Sheth, A. and Roy, K. (2023). Neurosymbolic Value-Inspired AI (Why, What, and How). *IEEE Intelligent Systems*.

Shi, J. and Malik, J. (2000). Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

Shu, T., Xiong, C., and Socher, R. (2018). Hierarchical and Interpretable Skill Acquisition in Multi-task Reinforcement Learning. *International Conference on Learning Representations*.

Silver, D. and Ciosek, K. (2012). Compositional Planning Using Optimal Option Models. In *International Conference on Machine Learning*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T. P., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, 550:354–359.

Silver, D., Singh, S., Precup, D., and Sutton, R. S. (2021). Reward is enough. *Artif. Intell.*, 299:103535.

Silver, T., Dan, S., Srinivas, K., Tenenbaum, J. B., Kaelbling, L., and Katz, M. (2024). Generalized planning in pddl domains with pretrained large language models. In *The Association for the Advancement of Artificial Intelligence*.

Simsek, Ö. and Barto, A. G. (2004). Using relative novelty to identify useful temporal abstractions in reinforcement learning. *International Conference on Machine Learning*.

Şimşek, Ö. and Barto, A. G. (2008). Skill Characterization Based on Betweenness. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1497–1504. Curran Associates, Inc.

Şimşek, Ö., Wolfe, A. P., and Barto, A. G. (2005). Identifying useful subgoals in reinforcement learning by local graph partitioning. In *International Conference on Machine Learning*.

Singh, S., Barto, A. G., and Chentanez, N. (2004). Intrinsically Motivated Reinforcement Learning. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pages 1281–1288.

Smith, M., van Hoof, H., and Pineau, J. (2018). An Inference-Based Policy Gradient Method for Learning Options. In *International Conference on Machine Learning*.

Sohn, S., Oh, J., and Lee, H. (2018). Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies. *Neural Information Processing Systems*.

Solway, A., Diuk, C., Córdova, N., Yee, D., Barto, A. G., Niv, Y., and Botvinick, M. M. (2014). Optimal behavioral hierarchy. *PLOS Computational Biology*, 10(8):1–10.

Sontakke, S. A., Zhang, J., Arnold, S. M. R., Pertsch, K., Biyik, E., Sadigh, D., Finn, C., and Itti, L. (2023). RoboCLIP: One Demonstration is Enough to Learn Robot Policies. *Neural Information Processing Systems*.

Sprekeler, H. (2011). On the relation of slow feature analysis and Laplacian eigenmaps. *Neural Computation*, 23(12):3287–3302.

Srinivas, A., Krishnamurthy, R., Kumar, P., and Ravindran, B. (2016). Option discovery in hierarchical reinforcement learning using spatio-temporal clustering. *arXiv*.

Stachenfeld, K. L., Botvinick, M. M., and Gershman, S. J. (2017). The hippocampus as a predictive map. *Nature neuroscience*, 20(11):1643–1653.

Stolle, M. and Precup, D. (2002). Learning Options in Reinforcement Learning. In *Symposium on Abstraction, Reformulation and Approximation*.

Stout, A. and Barto, A. G. (2010). Competence progress intrinsic motivation. In *International Conference on Development and Learning*.

Strehl, A. L. and Littman, M. L. (2008). An analysis of model-based Interval Estimation for Markov Decision Processes. *J. Comput. Syst. Sci.*, 74(8):1309–1331.

Strouse, D., Baumli, K., Warde-Farley, D., Mnih, V., and Hansen, S. (2022). Learning more skills through optimistic exploration. In *International Conference on Learning Representations (ICLR)*.

Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., and Fergus, R. (2018). Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play. In *International Conference on Learning Representations*.

Sun, S.-H., Wu, T.-L., and Lim, J. J. (2020). Program guided agent. In *International Conference on Learning Representations*.

Sutton, R. S. (1988). Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3:9–44.

Sutton, R. S. (2019). The Bitter Lesson. `http://www.incompleteideas.net/IncIdeas/BitterLesson.html`.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Sutton, R. S., Machado, M. C., Holland, G. Z., Szepesvari, D., Timbers, F., Tanner, B., and White, A. (2023). Reward-respecting subtasks for model-based reinforcement learning. *Artificial Intelligence*, 324:104001.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (1999a). Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Neural Information Processing Systems*.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *International Conference on Autonomous Agents and Multiagent Systems*.

Sutton, R. S., Precup, D., and Singh, S. (1998). Intra-Option Learning about Temporally Abstract Actions. In *International Conference on Machine Learning*.

Sutton, R. S., Precup, D., and Singh, S. (1999b). Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 112:181–211.

Sutton, R. S., Singh, S., and McAllester, D. A. (2001). Comparing Policy-Gradient Algorithms. http://incompleteideas.net/papers/SSM-unpublished.pdf.

Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D. S., Maksymets, O., Gokaslan, A., Vondruš, V., Dharur, S., Meier, F., Galuba, W., Chang, A. X., Kira, Z., Koltun, V., Malik, J., Savva, M., and Batra, D. (2021). Habitat 2.0: Training Home Assistants to Rearrange their Habitat. *Neural Information Processing Systems*.

Talvitie, E. (2017). Self-correcting models for model-based reinforcement learning. In *The Association for the Advancement of Artificial Intelligence*.

Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7).

Team, S., Raad, M. A., Ahuja, A., and et al, C. B. (2024). Scaling Instructable Agents Across Many Simulated Worlds. *arXiv*.

Tedrake, R., Manchester, I. R., Tobenkin, M. M., and Roberts, J. W. (2010). LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification. *The International Journal of Robotics Research*, 29(8):1038–1052.

Telikani, A., Tahmassebi, A., Banzhaf, W., and Gandomi, A. H. (2021). Evolutionary Machine Learning: A Survey. *Association for Computing Machinery*.

Thakoor, S., Rowland, M., Borsa, D., Dabney, W., Munos, R., and Barreto, A. (2022). Generalised Policy Improvement with Geometric Policy Composition. In *International Conference on Machine Learning*.

Thomaz, A. L., Breazeal, C., et al. (2006). Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *The Association for the Advancement of Artificial Intelligence*.

Thrun, S. and Pratt, L. Y. (1998). Learning to Learn: Introduction and Overview. In *Learning to Learn*.

Tirinzoni, A., Touati, A., Farebrother, J., Guzek, M., Kanervisto, A., Xu, Y., Lazaric, A., and Pirotta, M. (2025). Zero-Shot Whole-Body Humanoid Control via Behavioral Foundation Models. In *International Conference on Learning Representations*.

Todorov, E. (2006). Linearly-solvable Markov decision problems. In *Neural Information Processing Systems*.

Todorov, E. (2009a). Compositionality of optimal control laws. In *Neural Information Processing Systems*.

Todorov, E. (2009b). Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478–11483.

Todorov, E., Erez, T., and Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*.

Touati, A. and Ollivier, Y. (2021). Learning One Representation to Optimize All Rewards. In *Neural Information Processing Systems*.

Touati, A., Rapin, J., and Ollivier, Y. (2023). Does Zero-Shot Reinforcement Learning Exist? In *International Conference on Learning Representations*.

Trivedi, D., Zhang, J., Sun, S.-H., and Lim, J. J. (2021). Learning to synthesize programs as interpretable and generalizable policies. *Neural Information Processing Systems*.

Tse, H. T., Chandrasekar, S., and Machado, M. C. (2025). Reward-Aware Proto-Representations in Reinforcement Learning. *arXiv*.

Uesato, J., Kushman, N., Kumar, R., Song, H. F., Siegel, N. Y., Wang, L., Creswell, A., Irving, G., and Higgins, I. (2023). Solving Math Word Problems with Process-based and Outcome-based Feedback. *arXiv*.

Van Hasselt, H. and Wiering, M. A. (2009). Using continuous action spaces to solve discrete problems. In *International Joint Conference on Neural Networks*.

Varga, R. S. (2000). *Matrix Iterative Analysis*. Springer.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is All you Need. In *Neural Information Processing Systems*.

Vats, S., Likhachev, M., and Kroemer, O. (2023). Efficient Recovery Learning using Model Predictive Meta-Reasoning. In *International Conference on Robotics and Automation*.

Veeriah, V., Oh, J., and Singh, S. (2018). Many-goals reinforcement learning. *ArXiv*.

Veeriah, V., Zahavy, T., Hessel, M., Xu, Z., Oh, J., Kemaev, I., van Hasselt, H., Silver, D., and Singh, S. (2021). Discovery of Options via Meta-Learned Subgoals. In *Neural Information Processing Systems*.

Venuto, D., Islam, S. N., Klissarov, M., Precup, D., Yang, S., and Anand, A. (2024). Code as Reward: Empowering Reinforcement Learning with VLMs. *International Conference on Machine Learning*.

Verma, A., Le, H., Yue, Y., and Chaudhuri, S. (2019). Imitation-projected programmatic reinforcement learning. *Neural Information Processing Systems*.

Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri, S. (2018). Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. (2017). Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*.

Victor, B. (2011). Up and Down the Ladder of Abstraction. https://worrydream.com/LadderOfAbstraction/.

Vigorito, C. M. and Barto, A. G. (2008). Autonomous Hierarchical Skill Acquisition in Factored MDPs. In *Yale Workshop on Adaptive and Learning Systems*.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K.,

Smith, O., Schaul, T., Lillicrap, T. P., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575:350–354.

Vlastelica, M., Kolev, P., Cheng, J., and Martius, G. (2023). Diverse offline imitation via fenchel duality. *European Workshop on Reinforcement Learning*.

Wan, Y. and Sutton, R. S. (2022). Toward Discovering Options that Achieve Faster Planning. *arXiv*.

Wan, Y., Zaheer, M., White, A., White, M., and Sutton, R. S. (2019). Planning with Expectation Models. In *International Joint Conference on Artificial Intelligence*.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L. J., and Anandkumar, A. (2023a). Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv*.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2016). Learning to reinforcement learn. *arXiv*.

Wang, K., Zhou, K., Feng, J., Hooi, B., and Wang, X. (2022). Reachability-Aware Laplacian Representation in Reinforcement Learning. *International Conference on Machine Learning*.

Wang, K., Zhou, K., Zhang, Q., Shao, J., Hooi, B., and Feng, J. (2021). Towards Better Laplacian Representation in Reinforcement Learning with Generalized Graph Drawing. In *International Conference on Machine Learning*.

Wang, T., Torralba, A., Isola, P., and Zhang, A. (2023b). Optimal Goal-Reaching Reinforcement Learning via Quasimetric Learning. *International Conference on Machine Learning*.

Wang, Y., Sun, Z., Zhang, J., Xian, Z., Biyik, E., Held, D., and Erickson, Z. (2024a). RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback. In *International Conference on Machine Learning*.

Wang, Y., Xian, Z., Chen, F., Wang, T.-H., Wang, Y., Fragkiadaki, K., Erickson, Z., Held, D., and Gan, C. (2024b). RoboGen: towards unleashing infinite data for automated robot learning via generative simulation. In *International Conference on Machine Learning*.

Wang, Z. Z., Mao, J., Fried, D., and Neubig, G. (2024c). Agent Workflow Memory. *arXiv*.

Warde-Farley, D., de Wiele, T. V., Kulkarni, T. D., Ionescu, C., Hansen, S., and Mnih, V. (2019). Unsupervised Control Through Non-Parametric Discriminative Rewards. In *International Conference on Learning Representations*.

Watkins, C. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., hsin Chi, E. H., Xia, F., Le, Q., and Zhou, D. (2022). Chain of Thought Prompting Elicits Reasoning in Large Language Models. *Neural Information Processing Systems*.

Wilkes, M. V., Wheeler, D. J., Gill, S., and Corbató, F. (1958). The preparation of programs for an electronic digital computer. American Institute of Physics.

Wirth, C., Akrour, R., Neumann, G., and Fürnkranz, J. (2017). A Survey of Preference-Based Reinforcement Learning Methods. *Journal of Machine Learning Research*, 18(136):1–46.

Wiskott, L. and Sejnowski, T. J. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Wu, Y., Tucker, G., and Nachum, O. (2019). The Laplacian in RL: Learning Representations with Efficient Approximations. In *International Conference on Learning Representations*.

Wulfmeier, M., Rao, D., Hafner, R., Lampe, T., Abdolmaleki, A., Hertweck, T., Neunert, M., Tirumala, D., Siegel, N. Y., Heess, N., and Riedmiller, M. A. (2020). Data-efficient Hindsight Off-policy Option Learning. *International Conference on Machine Learning*.

Xiao, T., Chan, H., Sermanet, P., Wahid, A., Brohan, A., Hausman, K., Levine, S., and Tompson, J. (2022). Robotic Skill Acquisition via Instruction Augmentation with Vision-Language Models. *Robotics: Science and Systems Proceedings*.

Xie, T., Zhao, S., Wu, C. H., Liu, Y., Luo, Q., Zhong, V., Yang, Y., and Yu, T. (2024). Text2Reward: Automated Dense Reward Function Generation for Reinforcement Learning. *International Conference on Learning Representations*.

Xie, Z., Ji, C., and Zhang, Y. (2022). Deep Skill Chaining with Diversity for Multi-agent Systems. In *International Conference on Artificial Intelligence*.

Xu, Z., van Hasselt, H. P., and Silver, D. (2018). Meta-gradient reinforcement learning. *Neural Information Processing Systems*.

Yang, R., Chen, J., Zhang, Y., Yuan, S., Chen, A., Richardson, K., Xiao, Y., and Yang, D. (2024). SelfGoal: Your Language Agents Already Know How to Achieve High-level Goals. *Association for Computational Linguistics*.

Yang, R., Fang, M., Han, L., Du, Y., Luo, F., and Li, X. (2021a). MHER: Model-based hindsight experience replay. *Deep Reinforcement Learning Workshop, Neural Information Processing Systems*.

Yang, Y., Inala, J. P., Bastani, O., Pu, Y., Solar-Lezama, A., and Rinard, M. (2021b). Program synthesis guided reinforcement learning for partially observed environments. *Neural Information Processing Systems*.

Yu, W., Gileadi, N., Fu, C., Kirmani, S., Lee, K.-H., Arenas, M. G., Chiang, H.-T. L., Erez, T., Hasenclever, L., Humplik, J., Ichter, B., Xiao, T., Xu, P., Zeng, A., Zhang, T., Heess, N. M. O., Sadigh, D., Tan, J., Tassa, Y., and Xia, F. (2023). Language to Rewards for Robotic Skill Synthesis. *Conference on Robot Learning*.

Zahavy, T., Haroush, M., Merlis, N., Mankowitz, D. J., and Mannor, S. (2018). Learn what not to learn: Action elimination with deep reinforcement learning. *Neural Information Processing Systems*.

Zahavy, T., O'Donoghue, B., Barreto, A., Flennerhag, S., Mnih, V., and Singh, S. (2021). Discovering Diverse Nearly Optimal Policies with Successor Features. In *Unsupervised Reinforcement Learning Workshop, International Conference on Machine Learning*.

Zahavy, T., Schroecker, Y., Behbahani, F. M. P., Baumli, K., Flennerhag, S., Hou, S., and Singh, S. (2022). Discovering Policies with DOMiNO: Diversity Optimization Maintaining Near Optimality. *International Conference on Learning Representations*.

Zala, A., Cho, J., Lin, H., Yoon, J., and Bansal, M. (2024). EnvGen: Generating and Adapting Environments via LLMs for Training Embodied Agents. In *Conference on Language Modeling*.

Zhang, A., McAllister, R., Calandra, R., Gal, Y., and Levine, S. (2021a). Learning invariant representations for reinforcement learning without reconstruction. *International Conference on Learning Representations*.

Zhang, J., Lehman, J., Stanley, K. O., and Clune, J. (2024). OMNI: Open-endedness via Models of human Notions of Interestingness. *International Conference on Learning Representations*.

Zhang, J., Pertsch, K., Yang, J., and Lim, J. J. (2021b). Minimum description length skills for accelerated reinforcement learning. 2021.

Zhang, M., Tang, H., Hao, J., and Zheng, Y. (2023). Towards A Unified Policy Abstraction Theory and Representation Learning Approach in Markov Decision Processes. *International Conference on Learning Representations*.

Zhang, S. and Whiteson, S. (2019). DAC: The Double Actor-Critic Architecture for Learning Options. In *Neural Information Processing Systems*.

Zhang, Y., Abbeel, P., and Pinto, L. (2020). Automatic Curriculum Learning through Value Disagreement. *Neural Information Processing Systems*.

Zhao, Z., Samel, K., Chen, B., et al. (2021). Proto: Program-guided transformer for program-guided tasks. *Neural Information Processing Systems*.

Zheng, Q., Henaff, M., Zhang, A., Grover, A., and Amos, B. (2024). Online Intrinsic Rewards for Decision Making Agents from Large Language Model Feedback. *arXiv*.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Bisk, Y., Fried, D., Alon, U., et al. (2024). WebArena: A Realistic Web Environment for Building Autonomous Agents. *International Conference on Learning Representations*.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. (2008). Maximum Entropy Inverse Reinforcement Learning. In *The Association for the Advancement of Artificial Intelligence*.