UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ALINE WEBER

Identifying Reusable Early-Life Options

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Bruno Castro da Silva

Porto Alegre
December 2020

“Don’t let anyone rob you of your imagination, your creativity, or your curiosity.
It’s your place in the world; it’s your life.
Go on and do all you can with it, and make it the life you want to live.”
— MAE JEMISON

ABSTRACT

We introduce a method for identifying short-duration reusable motor behaviors, which we call *early-life options*, that allow robots to perform well even in the very early stages of their lives. This is important when agents need to operate in environments where the use of poor-performing policies (such as the random policies with which they are typically initialized) may be catastrophic. Our method augments the original action set of the agent with specially-constructed behaviors that maximize performance over a possibly infinite family of related motor tasks. These are akin to primitive reflexes in infant mammals—agents born with our early-life options, even if acting randomly, are capable of producing rudimentary behaviors comparable to those acquired by agents that actively optimize a policy for hundreds of thousands of steps. We also introduce three metrics for identifying useful early-life options and show that they result in behaviors that maximize both the option's expected return while minimizing the risk that executing the option will result in extremely poor performance. We evaluate our technique on three simulated robots tasked with learning to walk under different battery consumption constraints and show that even random policies over early-life options are already sufficient to allow for the agent to perform similarly to agents trained for hundreds of thousands of steps.

Keywords: Reinforcement Learning. Options. Early-Life Options. Primitive Reflexes.

Identificando *Early-Life Options* Reutilizáveis

RESUMO

Neste trabalho, introduzimos um método para identificar comportamentos motores reutilizáveis e de curta duração, que chamamos de *early-life options*. Esses comportamentos permitem com que robôs tenham boa performance mesmo nos momentos iniciais de suas vidas. Isso é importante quando agentes precisam interagir em ambientes nos quais o uso de políticas ruins (por exemplo, as políticas aleatórias com as quais os agentes geralmente são inicializados) pode ser catastrófico. Nosso método estende o conjunto de ações original do agente com comportamentos especialmente construídos para maximizar a performance em uma família possivelmente infinita de tarefas motoras relacionadas. Esses comportamentos são similares a reflexos primitivos em mamíferos, presentes no início de suas vidas. Agentes que iniciam suas vidas com a possibilidade de utilizar *early-life options*, mesmo quando agindo aleatoriamente, são capazes de produzir comportamentos rudimentares comparáveis a comportamentos de agentes que otimizaram suas políticas por centenas de milhares de passos. Nós introduzimos três métricas para identificar *early-life options* úteis e mostramos que elas resultam em comportamentos que maximizam o retorno esperado da *option*, ao mesmo tempo em que minimizam o risco de obter performance significativamente baixa ao executá-la. Nós avaliamos o método proposto em três robôs simulados, cuja tarefa é aprender a caminhar sob diferentes restrições de consumo de bateria. Nós mostramos que mesmo políticas aleatórias sobre o conjunto de *early-life options* já são suficiente para que o agente tenha performance similar a de agentes que foram treinados por centenas de milhares de passos.

Palavras-chave: Aprendizado por Reforço. *Options*. *Early-Life Options*. Reflexos Primitivos.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Using Reinforcement Learning (RL) algorithms to solve high-dimensional control problems may require a number of samples that is prohibitively large. Solving Atari games, for instance, often requires an agent to interact with its environment for hundreds of millions of timesteps. This is in sharp contrast with the level of performance achieved by humans and other animals when interacting with new tasks. One of the reasons why RL algorithms cannot yet achieve these performance levels is that they solve each new problem *tabula rasa*; then, if an agent is faced with the same or similar problem, many times throughout its life, it has to repeatedly learn to solve it from scratch. Humans, on the other hand, have a multitude of prior knowledge, either innate or acquired throughout their lifetimes, that allow for more rapidly learning to solve new tasks.

Developmental psychologists have studied the prior knowledge that humans often use when interacting with their environments, both in terms of visual biases [Spelke 1990] and in terms of developmental processes for acquiring reusable motor skills, such as reaching or grasping [Berthier and Keen 2006]. From a computational perspective, previous works have investigated the different ways in which the performance of RL agents is hurt due to the lack of prior knowledge—e.g., lack of innate visual biases and biases towards exploring objects [Doshi-Velez and Ghahramani 2011, Dubey et al. 2018].

In the RL community, a common way of equipping agents with motor priors for accelerating learning is through the use of *options*, or temporally-extended actions [Sutton, Precup and Singh 1999]. Options are reusable behaviors defined in terms of primitive actions or other options. One of the motivating principles underlying this idea is that subproblems recur, so that options can be reused when solving a variety of related tasks throughout an agent's lifetime. As an example, consider an agent tasked with driving. During this task, an agent has to repeatedly execute particular types of actions, or sequences of actions, such as signaling right/left when turning or changing lanes. It would be useful, then, to have the complete behavior (sequence of actions) necessary for signaling encoded as one extended action—an option. That way, at each turn, the agent would be able to select the corresponding option—a single decision—instead of having to individually select all primitive actions that are required to describe the signaling behavior.

Most of the existing state-of-the-art methods for learning options focus on identifying (*during an agent's lifetime*) useful recurring behaviors that help it to acquire optimal policies to solve a task more rapidly. Popular approaches are based on two main

ideas. The first class of methods is based on creating options for reaching states deemed to be important, such as subgoals. Several different approaches to how these subgoals can be achieved and leveraged in the learning process have been proposed. Goals could be defined, for instance, by finding commonalities between different paths to a solution [McGovern and Barto 2001] or by analyzing properties of the transition graph [Bacon 2013, Şimşek, Wolfe and Barto 2005]. Different methods also explore the concept of relative novelty in order to find subgoals [Şimşek and Barto 2004]. More recently, the use of Proto-Value Functions, which capture the geometry of the state space and are capable of finding bottlenecks, has been explored [Machado, Bellemare and Bowling 2017].

Other techniques for defining options are based on directly optimizing the parameters of an option's policy, so that an agent can more efficiently solve a given task. In [Bacon, Harb and Precup 2017] the authors derive policy gradient theorems that allow an agent to learn both intra-option policies and policies over options. This method has been extended to take into account the concept of deliberation costs—the costs of changing from one option to another while planning [Harb et al. 2018]. More recently, this family of methods has been combined with the idea of Proto-Value Functions, as in [Liu et al. 2017], in order to exploit properties of the geometry of the state space. Other approaches that have been explored include techniques that aim to generate a set of options that are as diverse as possible [Eysenbach et al. 2018], and approaches that make use of compression techniques to find the best set of reusable options [Garcia, Silva and Thomas 2019].

The methods discussed above are certainly relevant contributions to the problem of learning reusable behaviors. However, they focus on the problem of learning options that are best at improving the agent's performance throughout its entire lifetime. **In this work, by contrast, we do not wish to identify options that help an agent to perform well throughout its entire lifetime. We, instead, wish to identify options that allow robots to perform well in the *very early stages* of their lives**. This is important whenever agents may need to operate in environments where the use of poor-performing policies (such as the random policies with which they are typically initialized) may be catastrophic. We call these behaviors *early-life options* and consider them to be similar to primitive reflexes—such as the sucking reflex or the Moro reflex—in infant mammals. The Moro reflex [Berk 2009] is a particularly relevant example to the setting we tackle: it is present at birth and causes an infant's legs and head to extend, while the arms jerk up, whenever the infant experiences sudden shifts in its head position. In human evolutionary

history, this reflex may have helped infants to hold on to their mothers while being carried around. Importantly, this reflex is useful only in the very early stages of an infant's life and disappears after about six months.

The Moro reflex, briefly discussed above, is an example of a safety and survival early-life behavior, but other types of primitive reflexes exist. One relevant early-life reflex, present in some mammals, is the walking reflex—a type of reflex that assists not with safety or survival, but which is a type of learning bias. This reflex makes an infant attempt to walk by putting one foot in front of the other whenever placed standing on a surface. Similarly to the Moro reflex, this one also disappears after some months, when the infant actually starts to walk. This reflex is part of a sequence of innate behaviors present during the developmental stages that infants go through when learning to walk. First, a child learns to sit without support; then, to stand without assistance, to crawl, to walk with assistance, and so on. Importantly, this sequence of developmental stages implies that it is necessary for the child to acquire primitive/low-level capabilities before being able to learn more complex ones. Without such developmental stages and its corresponding sequence of increasingly more complex behaviors (built over previously-acquired simpler behaviors), it would be extremely hard to directly learn how to walk. This is the biological inspiration and intuitive motivation that underlies the definition of early-life options: simple behaviors that *(i)* act as the basis for making it possible to learn more complex ones; and *(ii)* are no longer necessary once the agent makes sufficient learning progress.

In this work, we introduce a method for identifying short-duration reusable early-life options that allow robots to perform well in the very early stages of their lives. Our method augments the original action set of the agent with specially-constructed options akin to primitive reflexes in mammals, so that agents equipped with them are capable of achieving high performance on a possibly infinite family of related motor tasks—i.e., they are reusable across many tasks. We propose an offline optimization process for generating, evaluating, and selecting candidate options that maximize specially-constructed performance metrics. We propose three metrics for evaluating the usefulness of candidate options and show that they identify behaviors that maximize both the expected return of an option while also minimizing the risk that executing it will result in extremely poor performance. We evaluate our technique on three simulated robots tasked with learning to walk under different battery consumption constraints and show that even random policies over early-life options are already sufficient to allow for the agent to perform similarly to agents that are trained for hundreds of thousands of steps.

The following chapters are organized as follows: Chapter 2 discusses the technical background relevant to our work. Chapter 3 explores related works in the literature and discusses similarities and differences with respect to our proposed approach. Chapter 4 introduces the proposed method for learning early-life options. Experiments are presented in Chapter 5. Finally, in Chapter 6 we present our conclusions and ideas for future work.

2 BACKGROUND

In this chapter, we provide a discussion on the techniques and ideas necessary to understand our proposed method. In Section 2.1 we discuss Reinforcement Learning and a few selected learning methods. In Section 2.2 we introduce the options framework. Finally, in Section 2.3 we discuss primitive reflexes in mammals and discuss the reasons why they are a biologically-inspired motivation underlying the idea of the early-life options.

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a computational approach to goal-directed learning and decision making. It is different from other machine learning approaches given its emphasis on learning based on the interactions of an agent with its environment, without requiring any type of supervision or complete models of the environment. Consider an agent in a gridworld, where the agent is tasked with arriving at a specific location. In RL, the only information made available to the agent are numerical reward signals that it receives after each action—these could, for instance, be positive when the agent reaches the goal position and zero otherwise. To define agent-environment interactions in terms of states, actions, and rewards, the problem is typically modeled as a Markov Decision Process (MDP).

An MDP M is a tuple $(\mathcal{S}, \mathcal{A}, r, p, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a function returning (expected) scalar rewards for executing action a in state s, p is a transition function specifying the probability $p(s'|s, a)$ of transitioning to s' after taking action a in state s, and $\gamma \in [0, 1)$ is a discount factor. A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a mapping specifying the probability $\Pr(a|s)$ of selecting action a when in state s. The goal of an RL agent is to learn a policy that accumulates as much reward as possible. Let the reward received at time t be the random variable R_t and the cumulative reward (or *return*) from time t be the random variable $G_t \doteq \sum_{i=0}^{T-1} \gamma^i R_{t+i}$, where T is a time horizon. A value function $v_\pi(s)$ is defined as the expected returned achieved when following policy π and starting in state s: $v_\pi(s) \doteq \mathbb{E}[G_t|S_t = s]$. An action-value function $q_\pi(s, a)$ is defined as the expected return achieved when following policy π, starting in state s, and taking a particular action a: $q_\pi(s, a) \doteq \mathbb{E}[G_t|S_t = s, A_t = a]$. Solving an MDP M consists of finding a policy π^* that maximizes the agent's expected return.

Several methods have been proposed in the literature to solve this optimization problem. There are two main approaches: tabular RL methods, and function approximation RL methods. The former are simpler methods that typically assume environments with state and action spaces small enough to represent value or action-value function using tables, and can often find optimal solutions. The latter approximate value functions using some class of approximators (e.g., linear value function approximators), and can typically find approximate solutions to the optimal-policy problem. They can, however, be applied to much larger problems. In what follows, we will present a few sample model-free methods of each category. These are called model-free since they do not require prior knowledge of the reward and transition functions.

2.1.1 Tabular Methods

A well-known tabular method for reinforcement learning is Q-learning [Watkins 1989]. This method is a temporal-difference learning method, where the agent learns from raw experiences and updates its estimate of an action-value function based on the experiences themselves and on its current estimate of that function. This is known in the literature as *bootstrapping*. Another characteristic of Q-Learning is that it is off-policy—the agent can improve its behavior towards an optimal policy, π^*, even if it behaves and collects samples according to any other policy[1].

It is known that every optimal policy for an MDP shares the same action-value function, called the optimal action-value function, q^*:

$$q^*(s, a) = \max_{\pi} q_\pi(s, a), \tag{2.1}$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$. From the optimal action-value function, it is straightforward to find the optimal action when in a given state: for any state s, the agent can simply choose any action $a \in \mathcal{A}(s)$ that maximizes $q^*(s, a)$.

The objective of Q-learning is to learn a (typically tabular) estimator $Q(s, a)$ for the optimal action-value function. In the tabular case, Q is represented by a $|\mathcal{S}| \times |\mathcal{A}|$ matrix, called a Q-table. In order to update the matrix, at each step of the episode, the agent chooses an action a based on its current state, s, using a policy derived from Q. Then, the agent executes the action and observes the next state, s', and the reward, r.

[1] Under mild assumptions regarding the exploratory process induced by the policy.

With that, it is possible to update the corresponding $Q(s, a)$ term in the matrix according to a learning rule based on the Bellman equation[2]:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a')). \tag{2.2}$$

Given sufficient timesteps and the possibility of updating all state-action pairs of the Q-table, the update rule given by Eq. 2.2 will converge to q^*.

2.1.2 Approximate-Solution Methods

Tabular methods often perform well when dealing with small discrete environments. However, for many tasks in which we may like to apply RL, the state and/or action spaces are very large or even infinite. In such cases, it is not possible to find π^* using tabular methods. To address this limitation, one wishes to find approximate representations of the action-value functions (or of the policy), which allow the agent to generalize, across states, estimates of the return achievable from different situations. This generalization process is often called function approximation—it allows the agent to collect samples from a given function of interest (e.g. a value function) and construct an approximation of the true underlying function based on those samples. One approach to deal with function approximation is through policy gradient methods. These methods directly approximate the optimal policy for a given problem, and may or may not depend on also estimating (approximate) value or action-value functions. Methods that learn and update both a policy and a value function are often called Actor-Critic methods. Policy gradient methods are based on computing or estimating the gradient of the expected return of the agent with respect to policy parameters, and then updating a parameterized policy in the direction of the gradient; i.e., in the direction that maximizes the expected return.

A recently-proposed policy-gradient algorithm is called Proximal Policy Optimization (PPO) [Schulman et al. 2017]. This method is based on the idea of using a history of agent experiences to determine the largest policy improvement step possible, while ensuring that such a step will not be too large and overshoot the target policy parameters that maximize performance. The policy update involves a constraint representing how different the new and old policies are allowed to be, in order to avoid overshooting. In particular, the update rule of PPO involves a clipping procedure that keeps the policy

[2]For an historical introduction to this concept, see [Bellman 1957, Bellman 1957].

from changing too fast, and therefore removes incentives for the new updated policy to get too far from the current one.

The PPO learning rule for updating a policy π_θ, parameterized by θ, is the following:

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}[L(s, a, \theta_k, \theta)|s, a \sim \pi_{\theta_k}], \tag{2.3}$$

where θ_k are the parameters of the current policy, θ_{k+1} are the parameters of the updated policy, and where the maximization problem on the right-hand side of Eq. 2.3 is usually solved by taking steps of stochastic gradient descent to maximize the objective, L. Here, L is defined as:

$$L(s, a, \theta_k, \theta) = \min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A_{\pi_{\theta_k}}(s, a), \text{clip}\Big(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon \Big) A_{\pi_{\theta_k}}(s, a) \right), \tag{2.4}$$

where ϵ is a hyperparameter that specifies how different the new policy (π_θ) is allowed to be with respect to the current one (π_{θ_k}), and where $A_{\pi_{\theta_k}}(s, a)$ is the advantage function given by $A_{\pi_{\theta_k}}(s, a) = Q_{\pi_{\theta_k}}(s, a) - V_{\pi_{\theta_k}}(s)$. Note that the clipping procedure, in Eq. 2.4, serves as a regularizer by removing incentives for the policy to change dramatically. Most implementations of PPO estimate the advantage function via the Generalized Advantage Estimation procedure [Schulman et al. 2016]. This often requires representing the value-function using a parameterized approximator—e.g., a neural network trained to minimize the mean-squared error between estimated state values and empirical returns.

In our experiments (Chapter 5), which involve high-dimensional continuous states and actions, value function approximation and policy gradient methods are required. In all of our experiments, we use the PPO algorithm to learn near-optimal policies.

2.2 Options

As previously discussed, in this work we introduce a method for identifying reusable behaviors that help a robot to perform well in the early stages of its life. A standard way of representing reusable, temporally-extended behaviors in RL is via the *Options framework* [Sutton, Precup and Singh 1999]. This framework describes a set of formalisms for learning and using temporally-extended actions, or *options*. Intuitively, options represent sequences of primitive actions (or other options) that encode high-level behaviors that

may be reusable by the agent in different situations. They are akin to motor skills.

One of the motivating principles underlying this idea is that subproblems recur, so that options can be reused while solving a task or even in a variety of similar tasks. As an example, consider the task of walking from one room to another in a given building. In order to leave the first room, the agent needs to open a door. Then, when the agent arrives at the other room, it needs to open a second door. In this case, opening doors is a recurrent behavior that could be represented by an option. Once this option has been acquired by the agent, the entire corresponding temporally-extended behavior is defined and made available to the agent, which keeps the agent from having to repeatedly re-learn (from scratch) such a skill every time that a new door is encountered. This facilitates and accelerates both the exploration process executed by the agent and the policy-learning process. As previously-mentioned, options can be defined in terms of primitive actions or other options. This leads to the possibility of creating hierarchies of options, where, for example, the behavior of opening a door could be defined in terms of other options—options for grabbing the doorknob, turning the doorknob, and pushing a door.

In the Options framework, a Markovian option o consists of three components: *(1)* a policy $\pi_o(s, a)$ describing the probability of taking action a while executing option o in state s; *(2)* an initiation set I_o specifying the states $s \in S$ in which the option can be initiated; and *(3)* a termination condition $\beta_o(s)$ specifying the probability of the option terminating at a state s. Let us continue discussing the example of an option for opening a door. The initiation set of the option would include all states in which it is possible to reach the doorknob; it would make no sense, by contrast, for the option to be available from states where the agent is in the middle of a park. The policy of the option would correspond to a mapping from states to actions that would move the agent's hand and arm in a reasonable manner in order to open the door. Finally, the termination condition of the option would be set to one in all states where the door is opened, and zero otherwise. When the agent chooses an option for execution, it repeatedly selects and executes primitive actions according to the option's policy, until the option's termination condition is reached. Option policies can be represented in an open-loop manner (in which case they are known as macros). This involves a fixed sequence of actions. Alternatively, options policies can be closed-loop, in which case the particular actions that are executed at a time are based on the current state of the environment.

When options are available to the agent, the standard formalism of MDPs can be extended to a more sophisticated framework known as Semi-Markov Decision Processes,

or SMDPs [Sutton, Precup and Singh 1999]. In an SMDP, the policy of an option can rely on the complete history of experiences since the option was initiated. The SMDP formalism allows for agents to use the same learning and planning techniques as those used when only primitive actions are available. This is achieved by defining a corresponding Bellman equation for options. As an example, the Q-learning method can be extended to allow for options to be executed, in which case option-action values need to be learned [McGovern and Sutton 1998]. Under this method, when updating the q-value of primitive actions, the update is the same as that of standard Q-learning (Section 2.1). When updating options, by contrast, the update is given by:

$$Q(s,o) \leftarrow (1-\alpha)Q(s,o) + \alpha(r_o + \gamma^n \max_{a' \in \mathcal{A}_{s_{t+n}}} Q(s_{t+n}, a')), \tag{2.5}$$

where $r_o = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + +\gamma^{n-1} r_{t+n}$, and where n is the number of steps during which the option o was executed. Furthermore, s is the state where the option was initiated, s_{t+n} is the state where the option terminated, and $\mathcal{A}_{s_{t+n}}$ is the set of actions available in state s_{t+n}. We provide further details on how the options framework is used, in our work, in Section 4.1.

2.3 Early-Life Options in Mammals

The proposed idea of this work—to learn early-life options helpful in the very early stages of an agent's life—is inspired by the observation of similar behaviors in many infant mammals, where such behaviors were optimized/discovered by natural selection. Such behaviors are often called *primitive reflexes*: innate reactions that happen automatically in response to a given stimulus. These reflexes can be of various types, ranging from survival behaviors to parenting reflexes (that help create a connection between an infant and their parents) and learning-bias reflexes that accelerate the acquisition of learned behaviors [Berk 2009].

An example of a primitive reflex that holds a survival value is the rooting reflex, which helps an infant find the mother's nipple in order to breastfeed. This reflex is only activated when babies are hungry and are touched by another person. The Moro reflex [Berk 2009] is another example of a survival-based reflex. It causes an infant's legs and head to extend, while the arms jerk up, whenever the infant experiences sudden shifts in its head position. In human evolutionary history, this reflex may have helped infants to

hold on to their mothers while being carried around.

Other primitive reflexes help to establish a connection between the infant and their parents, such as the sucking reflex, which is linked to the rooting reflex, and that assists in breastfeeding. Another relevant reflex is the palmar grasp reflex, which is often encountered in infant mammals: it causes a child to close their hands around any objects placed on their palms. The palmar grasp reflex encodes one of the first readily recognizable fine motor skills that are crucial to the normal development of a child. Other than having survival value, both of these reflexes encourage parents to interact with, and react to an infant's actions, responding with love and affection and allowing parents to comfort children in case of distress.

Besides providing survival value, primitive reflexes can also provide learning biases that accelerate the acquisition of learned behaviors. One example is the walking reflex, which is present at birth and provides a bias to help to learn walking gaits—even though infants, at this young age, cannot yet support their own weight. With this reflex, when the soles of an infant's feet touch a flat surface, they attempt to walk by placing one foot in front of the other [Siegler, DeLoache and Eisenberg 2003].

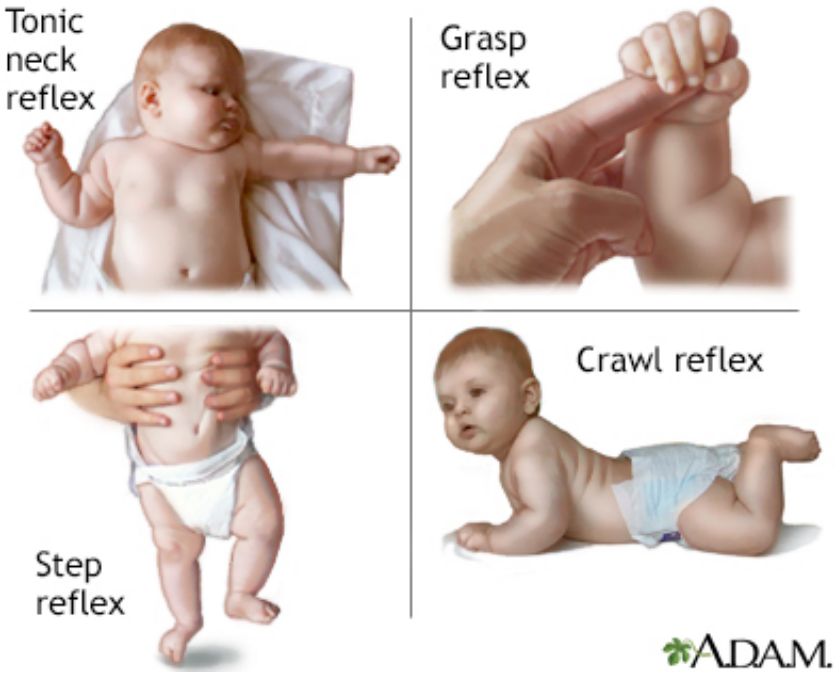
One important characteristic of the above-mentioned reflexes (Figure 2.1) is that they are only useful at the very beginning of an infant's lifetime. Most of them disappear by the age of six months. The walking reflex, for instance, becomes weaker around 5-6 months of age, as the infant starts to attempt to walk.

Primitive reflexes are inherently related to another inspiration for our work: the observation of motor developmental stages in mammals. These are changes in motor behavior that occur over the lifespan of a child and that represent the sequential, age-related processes that an infant undergoes. Figure 2.2 illustrates the milestone achievements that a child undergoes while learning to walk. It illustrates that a given task—learning to talk—may be too complex to be tackled directly. Instead, natural selection facilitates the acquisition of this behavior by imposing a sequence of ever more complex motor capabilities that a child needs to acquire, sequentially. Without such developmental stages, it would be extremely hard for (some types of) mammals to directly learn to walk.

Primitive reflexes in mammals, and the existence of motor development stages, illustrate how learning can often be seen as a hierarchical process, starting from simpler innate behaviors[3] and converging to more complex learned ones. These, therefore, are the biological inspirations and motivation for the definition of early-life options (ELOs).

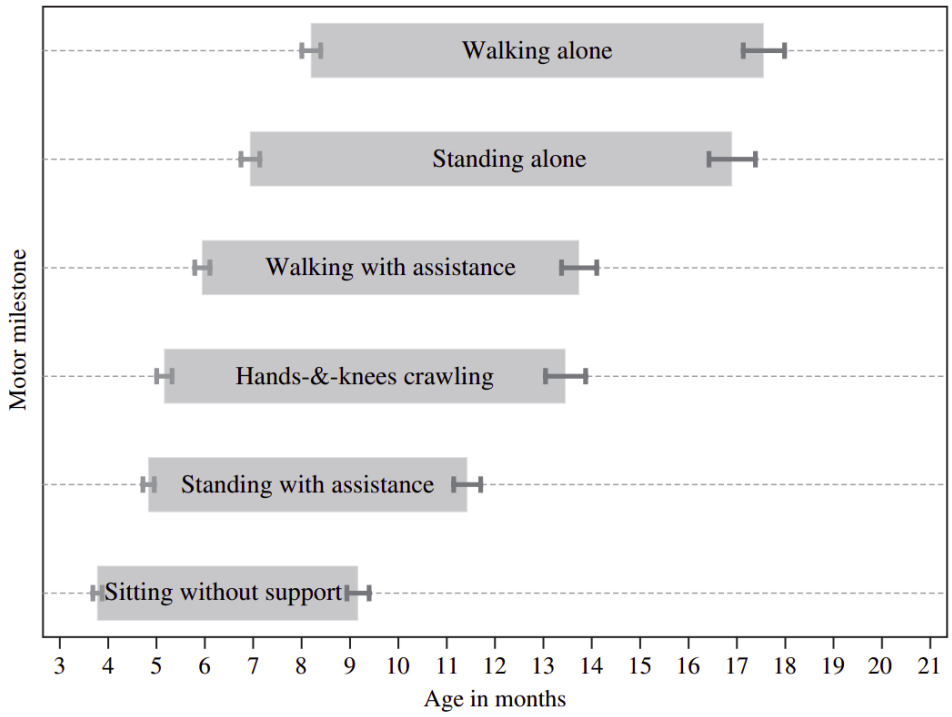
[3] Which disappear when no longer necessary.

Figure 2.1: Sample primitive reflexes in mammals.



Source: [A.D.A.M. 2019 (accessed 11/13/2020)]

Figure 2.2: Developmental motor stages that a child undergoes when learning to walk, as a function of age.



Source: [Onis 2006]

ELOs, as previously discussed, are simple behaviors that act as the basis for making it possible for an agent to learn more complex ones. They are also removed from an agent's repertoire when no longer needed—i.e., after the agent makes sufficient learning progress.

3 RELATED WORK

In this chapter, we discuss existing methods related to option learning. There are two main strategies for identifying and learning options. The first one, briefly discussed in Section 3.1, is based on finding useful subgoal states for solving a particular task, and then creating an option for reaching each subgoal. The second strategy, discussed in Section 3.2, is based on directly learning option policies that result in maximal return, without having to identify subgoals.

Notice that all methods presented in this chapter are related, but orthogonal to our proposed objective. Both our method, and the related techniques discussed here, intend to discover and optimize options. Existing related methods, however, are designed to identify options that are useful over the entire lifetime of an agent. We, by contrast, wish to identify options that are useful during the early moments of the agent's lifetime. Because the option-discovery techniques discussed here are orthogonal to our objective, they can be combined with our method to provide useful options once the agent has acquired sufficient basic knowledge about how to interact with its environment. As an example, it would be possible to use our technique to identify survival behaviors and reflexes, and to use the methods discussed in this chapter to build upon early-life options to construct more sophisticated behaviors, useful in the latter parts of an agent's lifetime.

3.1 Goal-Based Options

A common heuristic for defining useful options is to create options specialized in reaching particular subgoal states. In the RL literature, subgoal states are often defined as *bottleneck states*. Intuitively, these are states considered important for solving a particular task since they allow the agent to move between well-connected regions of the state space. Consider, for instance, a problem where the agent has to move from one room to another. In this environment, doors are examples of bottleneck states: no matter where within a given room the agent is, or where it wishes to go, all solutions need to pass through the door state.

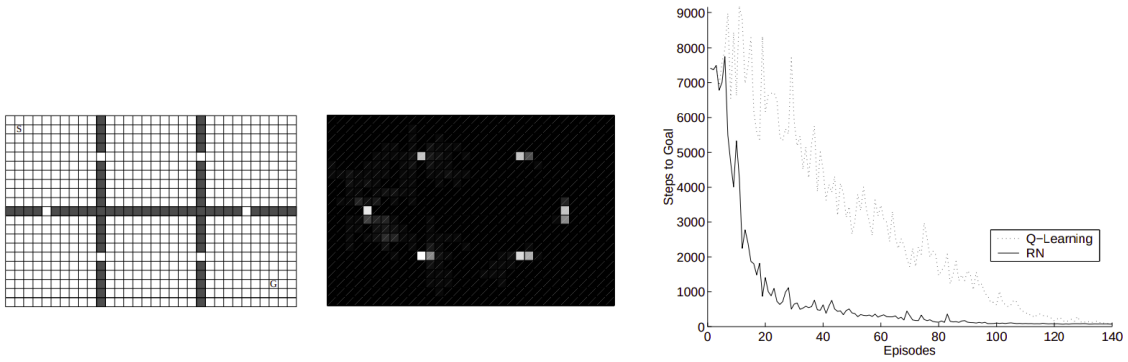
In [McGovern and Barto 2001], the authors proposed one of the first methods to automatically discover subgoals in RL. The proposed technique is based on identifying similarities between different paths that the agent may take to a given solution. In particular, the idea involves finding bottleneck states from an ensemble of trajectories

accumulated by the agent while interacting with the environment. To find such commonalities, the authors pose option discovery as a multiple-instance learning problem: the agent needs to classify each trajectory as positive (reached the goal) or negative (did not reach the goal). The objective, then, is to find the region of states that appears in every successful trajectory but that does not appear in unsuccessful ones. The authors use the concept of diverse density to find such solutions. The process of identifying subgoal states occurs online, while the agent learns to solve a task. After each new subgoal state is identified, an option to reach that state is created and added to the set of actions available to the agent. By using this technique, the agent is capable of learning faster, when compared to the setting with no options. Furthermore, this method also allows the agent to transfer and reuse knowledge of the learned options in order to more rapidly solve different, but related tasks.

A different approach to identify subgoal states was introduced by Bacon [Bacon 2013]. Here, the author proposed analyzing a graph representation of the state space in order to define which states are bottlenecks. The assumption underlying this technique is that desirable subgoals lie on paths that connect different densely-connected regions of the state space. The author constructed a graph based on sample trajectories, where states are nodes and where a transition between states produces an edge. Each edge has a weight corresponding to the number of times that the corresponding transition occurred. The author then used a community detection method, where a community is defined as a group of nodes that is heavily connected within themselves, but that has sparse links connecting it with other communities. Subgoal states are defined as the edges that connect different communities. Each option's policy is learned using Q-learning. The initiation set of each option is defined as the set of states within the corresponding community, and the termination probability is one at the subgoal state and zero otherwise. This approach results in a more principled way of defining initiation sets, compared to the diverse density-based techniques discussed previously. However, it is heavily dependent on setting hyperparameters of the underlying community detection mechanism.

Similarly, in [Şimşek and Barto 2004] the authors argue that subgoals should be defined as states that allow for the transition between different regions of the state space. The authors proposed using a method based on the concept of *relative novelty* in order to identify such states. The novelty of a state is defined as how frequently the state is visited; the relative novelty of a state is the ratio between the novelty of states that follow it in a given trajectory, and the states that precede it. The task of finding subgoals, then,

Figure 3.1: Relative novelty goal discovery: (a) The Six-room gridworld environment. (b) Subgoals identified by the method. (c) Mean steps to the goal.



(a) (b) (c)

Source: [Şimşek and Barto 2004]

can be formulated as a classification problem, where the objective is to classify a state as a subgoal or not based on its relative novelty score. This method allows for subgoals to be identified without requiring access to the reward function, which implies that it can be used even if the agent is following an exploratory policy and even if the agent does not complete the task. Figure 3.1 depicts the subgoals identified by this method when applied to a gridworld environment, as well as its performance when compared with Q-learning.

In [Şimşek, Wolfe and Barto 2005], the authors also investigate a method based on analyzing a graph of transitions, and make similar assumptions that subgoals should connect different regions of the state space. Here, however, the authors used local partitioning techniques and applied them to graphs constructed based only on the most recent experiences of the agent. After constructing such a graph, the method finds bottleneck states by identifying cuts of the graph. In particular, it identifies recurring low-probability edges that connect subgroups of nodes of the graph. Subgoals are defined as the states that are endpoints of such edges. This technique is capable of identifying subgoals based on local information (i.e., given only information about the states that surround them) instead of requiring metrics that are defined over the entire state space.

A different goal-based technique was introduced by Machado et al. [Machado, Bellemare and Bowling 2017], where the authors argued that options can be defined by analyzing the Laplacian of the transition graph associated with an MDP. The Laplacian of a graph is known to capture geometric information about the underlying state space, such as its symmetries and bottlenecks. Based on this idea, the authors compute Proto-Value Functions, or PVFs [Mahadevan 2005], which encode information about the graph's Laplacian and can be used to specify intrinsic reward functions called eigenpur-

poses. Such functions, when used to define rewards given to the agent, incentivize it to traverse the state space, and explore it, by following the directions of the PVF. From each eigenpurpose, a corresponding eigenbehavior can be computed: an eigenbehavior is the policy of the option that maximizes the reward given by the corresponding eigenpurpose reward function. Since these options are defined by taking into account only the transition dynamics of the environment, they can arguably be reused when solving different tasks defined over the same state space, where each task is associated with a particular reward function.

More recently, the method of Successor Options was introduced by [Ramesh, Tomar and Ravindran 2019]. This method leverages the idea of *successor representations* [Dayan 1993]. A successor representation (SR) assigns a new set of features to each state, where the features encode information about which future states are expected to follow from the state, given a particular policy. Since nearby states are expected to have similar successors, their SRs are also similar. Importantly, since SRs reflect information about the expected trajectories that may follow from a given state, they can be used to build a model of the state space and to capture the temporal structure between states in a graph. Based on computing success representations, subgoals can be defined as states that have maximally different SRs; i.e., states from which the agent is expected to visit very different regions of the state space. The overall method consists of building a successor representation of the state space, identifying subgoals, building policies for reaching each subgoal, and then using such options to solve a collection of related tasks. A related technique was proposed in [Goel and Huber 2003]. In this approach, a subgoal state is defined as a state that can be reached by trajectories originating from many different states, and such that its successors do not have this property.

Even though the methods discussed so far allow agents to autonomously identify options, we emphasize that they differ with respect to our objective in two main ways: *(1)* we do not require the identification of subgoals to define early-life options; and *(2)* we do not wish to discover options that are useful in the long-term, throughout an agent's lifetime. By contrast, we wish to identify one particular set of options—those that help to maximize return in the very early stages of an agent's lifetime, when it operates under a nearly random initial policy.

3.2 Direct Option Policy Optimization

In contrast to the related work introduced in the previous section, existing literature also considers identifying options by directly optimizing option policies that result in high return. This is akin to directly discovering reusable behaviors by tuning their parameters (typically via gradient ascent), instead of assuming that useful behaviors are those that necessarily lead to one or more subgoal states.

In [Bacon, Harb and Precup 2017], for instance, the authors derived policy gradient theorems for learning options. In particular, they introduced the *Option-Critic architecture*, which allows agents to learn intra-option policies, termination conditions, and also policies over options, without ever requiring the definition of intrinsic reward functions or subgoals. The proposed method aims to optimize the option-value function $Q_\Omega(s_0, o_0)$, where s_0 and o_0 denote the agent's initial state and option. The option-value function $Q_\Omega(s, o)$ is defined as:

$$Q_\Omega(s, o) = \sum_a \pi_{o,\theta}(a|s) Q_U(s, o, a) \tag{3.1}$$

where $\pi_{o,\theta}(a|s)$ is the policy of option o, parameterized by θ, and $Q_U(s, o, a)$ is the value of executing an a action in the context of a given state-option pair:

$$Q_U(s, o, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) \big[(1 - \beta_{o,\vartheta}(s')) Q_\Omega(s', o) + \beta_{o,\vartheta}(s') V_\Omega(s')\big] \tag{3.2}$$

where $\beta_{o,\vartheta}(s)$ is the termination function of option o, parameterized by ϑ. From these equations, the authors were able to obtain the gradient of expected return with respect to θ and ϑ. This allows agents to perform gradient ascent in order to directly optimize option policies and option termination conditions. The Option-Critic architecture requires that the user set the number of desired options. This architecture has shown good performance in several different environments, ranging from gridworlds to Atari games.

In [Harb et al. 2018], the authors proposed an important extension to the Option-Critic architecture. This work was motivated by empirical results that indicate that the original Option-Critic method often converges to options that are equivalent to executing only one primitive action. In order to minimize the probability of single-action options, the authors assumed a cost associated with switching between options, so that the optimization algorithm is incentivized to discover longer, more temporally-extended be-

haviors. In particular, the authors introduced a deliberation cost that is incurred upon switching to a new option. They then formulate the problem by extending the Option-Critic architecture in a way that subtracts deliberation costs from the rewards achieved by the agent. The authors derived gradient-based learning algorithms to optimize this new objective. This approach resulted in good performance in many different domains and is capable of discovering longer-lasting options that—unlike in the original Option-Critic work—do not shrink over time.

Another work for option discovery that extends the Option-Critic architecture was proposed in [Liu et al. 2017]. Here, the authors introduce the idea of eigenoptions—related to eigenbehaviors, discussed in the previous section. The resulting algorithm is called the Eigenoption-Critic algorithm. This technique modifies the original reward function and combines it with an intrinsic reward derived from the eigenpurposes functions introduced in [Machado, Bellemare and Bowling 2017]. This mixed reward signal is used to update an option's policy; the policy over options is updated based only on the original extrinsic reward. The benefits of this extension of the Option-Critic architecture is that it can better deal with non-stationary environments, and that it often results in a more diverse set of options.

Yet another technique for directly optimizing option policies was introduced in [Eysenbach et al. 2018]. The authors work under the hypothesis that a set of skills is useful if it maximizes the coverage over the set of possible behaviors that an agent may need to execute. The proposed method learns options by maximizing an objective function under a maximum entropy policy. The intuition underlying this method is that its objective encodes the idea that skills should be as *diverse* as possible—they should be able to consistently take the agent to a large set of different states. This implies that the set of states that are reachable by each skill is what distinguishes the (maximally diverse) behaviors discovered by this method.

Another approach for discovering options was proposed in [Garcia, Silva and Thomas 2019]. Here, the authors proposed creating options by identifying recurrent action patterns in trajectories drawn from well-performing policies. They introduced a three-step framework that begins by sampling trajectories from near-optimal policies. The method then identifies sequences of actions that recur in the trajectories by compressing them. Intuitively, this associates a symbol to each recurring subsequence of actions. Each such symbol, and its corresponding sequences of actions, is then used to define an option. Notice that, by construction, the options that allow for the maximum compression of tra-

jectories are also maximally reusable options: chunks of behaviors that appear in many trajectories observed while executing well-performing policies. After identifying a set of candidate recurring options, the algorithm performs option evaluation: the value of an option is defined as its expected Q-value over all states in a given set of tasks/MDPs. The last step performed by the algorithm is option selection, where options that lead to higher rewards (and that are dissimilar from the other discovered options) are selected. This method has a similar structure to the technique that we introduce in Chapter 4. However, it uses different underlying techniques (e.g., compression algorithms) and has a different objective—to optimize options for the entire lifetime of the agent, instead of optimizing early-life behaviors in particular.

The methods discussed in this section offer a different perspective to option discovery: to directly optimize the policies of a set of options in order to maximize return. As previously mentioned, they are related to our goal—to discover reusable options—but they differ with respect to our objective in that they aim at identifying options that are useful throughout an agent's lifetime. We, by contrast, are interested in discovering early-life options that mimic the types of safety and learning-bias behaviors often observed in infants' primitive reflexes.

4 LEARNING EARLY-LIFE OPTIONS

In this chapter, we introduce our proposed method for learning early-life options. This chapter is organized as follows: Section 4.1 presents the setting of our problem. In Section 4.2 we elaborate on the mathematical objectives of our proposed technique. We then introduce, in Section 4.3, three metrics based on which early-life options with different characteristics may be optimized. Finally, in Section 4.4 we provide details about a complete algorithm to identify sets of reusable early-life options.

4.1 Setting

We assume an RL agent that needs to solve not a single problem (task), but that may be presented with a *sequence* of tasks drawn from some task distribution. This is the setting typically tackled by methods dealing with learning options in multi-task problems [Kober et al. 2012, Silva, Konidaris and Barto 2012, Stulp et al. 2013]. Each task is modeled as an MDP, and we assume that the MDPs have dynamics and reward functions similar enough so that they can be considered variations of the same task. A family of similar and related tasks could be constructed, for instance, by assuming the same transition dynamics, but by associating different reward functions with different tasks. In Chapter 5 we expand on this point and introduce an infinite family of related MDPs corresponding to motor problems where robots need to learn to walk efficiently while operating under different power consumption constraints.

Let Ψ be the set of possible tasks that an agent may need to solve. Each element of this space is an MDP which we assume can be compactly described by a vector τ of parameters. Learning to grasp a particular object, for instance, is a task that may be compactly characterized by parameters specifying the object's shape and weight. Assume, furthermore, that problems in Ψ occur in an agent's lifetime with probabilities given by some distribution P.

Similarly to how we defined the value $v_\pi(s)$ of a state s (see Section 2.1), let us now define the value $v_{d_0,P}(\pi)$ of a policy π when evaluated over a distribution P of tasks and different initial states from which it may be deployed—where initial states are drawn

from some distribution d_0:

$$v_{d_0,P}(\pi) \doteq \int P(\tau) \sum_{s \in \mathcal{S}} d_0(s) v_\pi(s,\tau) d\tau. \tag{4.1}$$

Here, $v_\pi(s,\tau)$ is defined similarly to $v_\pi(s)$ but makes it explicit that the policy is being evaluated in a particular task, τ. In what follows we omit the dependence on d_0 and P to simplify the notation, and refer to the performance of a policy over a distribution of tasks and initial states simply as $v(\pi)$. Importantly, note that $v(\pi)$ is an *expected value*. In this work, however, we will be concerned not only with optimizing the expected performance (return) of a policy or option, but with optimizing more sophisticated properties of its *distribution* of its possible returns. This will be achieved by introducing three option-evaluation metrics (see Section 4.3 for more details).

Let a context $C \doteq (\tau, s_0)$ be a random variable denoting a tuple containing a task τ, drawn from P, and an initial state s_0, drawn from d_0. Let $V(\pi)$ be the random variable denoting the possible returns obtained by executing π in a random context. It should be clear, then, that $v(\pi) = \mathbb{E}_{d_0,P}[V(\pi)]$. Our method requires evaluating not only the performance of individual early-life options, but of *sets* of options. In what follows we abuse notation and extend the definition of the value of an option, $V(\pi_o)$, to the value of a set of options, $V(\pi_{o_1}, \ldots, \pi_{o_K})$, where K is the number of options in the set. This is a random variable denoting the average of the corresponding options' returns:

$$V(\pi_{o_1}, \ldots, \pi_{o_K}) \doteq \frac{1}{K} \sum_{i=1}^{K} V(\pi_{o_i}). \tag{4.2}$$

In our setting, to keep our formalism simple, we assume that options can be initiated from any state (i.e., $I_o = \mathcal{S}$), and assume that they last a short pre-defined number of timesteps, T, so that their termination condition $\beta_o = 1$ iff the option has been executed for T steps. This latter decision is justified by the observation that while learning with longer options may be more sample-efficient, if a given option set is not well-fitted for a particular task (e.g., it does not allow for optimal policies over primitive actions to be represented exactly), then shorter options are more flexible and may result in better solutions [Harutyunyan et al. 2017]. Due to these assumptions, we henceforth refer to an option o simply by its policy π_o and leave its other components implicit.

4.2 Mathematical Objective

Consider an agent tasked with learning to solve a family of tasks that require executing slightly different walking gaits. Instead of learning each walking pattern from scratch—which could be hard—we argue that such an agent would benefit from being born with primitive behaviors similar to the walking reflex in infant mammals. Such a reflex, or early-life option, would serve as a learning bias that leads the agent to visit states and to perform actions that facilitate learning correct walking behaviors. Access to such a learning bias would allow for more rapid policy acquisition when compared to starting the learning process completely from scratch. Additionally, if we assume that the agent of interest is a physical robot—which may suffer critical hardware damage by falling—then the importance of innate balancing and primitive walking reflexes is made even clearer.

Our goal is to identify a set $O^* = \{\pi_{o_1}, \ldots, \pi_{o_K}\}$ of options that can be used to augment $\mathcal{A}$, the set of actions, so that the resulting agent has access to behaviors that are akin to primitive reflexes in infant mammals. In particular, agents born with such options, even when acting randomly in the early stages of their lives, should be capable of producing rudimentary behaviors with performance comparable to that of agents that are allowed to optimize a policy for hundreds of thousands of steps. We call these *early-life options*, since their goal is to guarantee good performance in the early stages of an agent's life where the use of initial near-random policies may be catastrophic.

One way of characterizing this set is by identifying a set of options whose expected return, when evaluated over a distribution of possible tasks and initial states (contexts) is maximal:

$$O^* \doteq \underset{\pi_{o_1},\ldots,\pi_{o_K}}{\arg\max} \quad \frac{1}{K}\sum_{i=1}^{K} \mathbb{E}[V(\pi_{o_i})]. \tag{4.3}$$

This definition makes an important assumption: it evaluates a *set* of candidate options based on their individual performances. Usually, however, options cannot be evaluated by the individual returns that they provide, but by the benefits that they jointly provide to an agent throughout its entire lifetime (e.g., [Bacon, Harb and Precup 2017, Machado, Bellemare and Bowling 2017]). This is *not* our objective, however. We, by contrast, wish to identify options that help agents in the *early stages* of their lives, possibly when they are still operating under near-random initial policies. The criterion expressed in Eq. 4.3 models this worst-case scenario precisely: it represents the average return achieved by an agent acting under a random policy over options, as the returns come from executing

each option in random tasks and random states. A set of options with a high expected return even in this case—i.e., when the agent is not learning and is acting under an initial random policy—is considered to be a good set of early-life options. Such a set of options enables the agent to maximize performance even when executing a near-random policy, and before it acquires sufficient knowledge about how to complete a particular task.

4.3 Quantifying the Performance of Early-Life Options

Eq. 4.3 defines an optimal set of options as one that, if used by a near-random agent, results in maximal expected return. We are interested, however, in defining more sophisticated option evaluation metrics that take into account not only the option's mean return, but also properties of the tails of its return distribution. This is useful, for instance, so that we can identify ELOs that maximize expected performance while minimizing the probability that they may produce risky behaviors. In what follows, we introduce three metrics to evaluate a candidate early-life option, π_o, based on more general properties of its return distribution:

1. the *Maximum-Mean* metric $\psi_{\mu}(\pi_o)$. This is the simplest way of evaluating an option π_o. It directly estimates the expected value of the option's return distribution. This metric does not take into account return variance or the negative tails of its distribution (i.e., the risks associated with executing the option). This metric is useful in situations where there are no risks for the agent and the only objective is to obtain as much reward as possible. Consider, for instance, an agent tasked with learning how to ride a bike, but assume that the bike has training wheels. As the risk of falling is low, the agent is allowed to follow somewhat riskier policies (but which may result in faster learning) as long as those policies yield higher expected returns.
2. the *Negative Tail-Averse* metric $\psi_{-}(\pi_o)$. This metric takes into account not only the expected (mean) performance of an option, but also the area under the curve of the negative tail of its return distribution. This metric favors options with both high expected return and with a low probability of producing significantly poor (possibly catastrophic) returns. It is useful in situations where there are non-negligible risks that the agent wishes to avoid. Consider, once again, an agent tasked with learning how to ride a bike, but now assume that the bike has no training wheels. Even though we would like the agent to make fast learning progress (i.e., achieving high

expected return), we also wish to prevent, as much as possible, that the agent falls and gets damaged—thereby achieving low return. This setting requires maximizing expected return while minimizing the probability of risky behaviors, which yield low return, such as those associated with the negative tail of the option's return distribution.

3. the *Positively-Skewed* metric $\psi_+(\pi_o)$. This metric takes into account not only the expected performance of an option, but also the area under the curve to the right of the mean of the option's return distribution. It favors options with a high expected return and that maximize the probability that they will produce behaviors with above-average quality—even if at the risk of sometimes producing behaviors with subpar performance. This metric is useful in situations where it is acceptable that the agent executes risky actions that, if successful, result in high-performance behaviors. Consider an agent tasked with learning to ride a bike under a given time constraint. We would like the agent to make as much progress as possible towards learning a high-performing policy, even if that implies that the agent may sometimes fall and get hurt.

Based on the definitions of these metrics, we can now re-write Eq.4.3 in a more general form so that the value of each option can be defined with respect to a selected metric, ψ, and not necessarily with respect to its expected return. The set of optimal options O^*_ψ with respect to a given metric, ψ, then, is:

$$O^*_\psi \doteq \underset{\pi_{o_1},\ldots,\pi_{o_K}}{\arg\max} \quad \frac{1}{K}\sum_{i=1}^{K} \psi(V(\pi_{o_i})). \tag{4.4}$$

Note that Eq. 4.4 is equivalent to Eq. 4.3 if the *Maximum-Mean metric* ψ_μ is used.

As previously discussed, the *Maximum-Mean metric* ψ_μ directly measures the expected value of an option's return distribution, and is thus defined as:

$$\psi_\mu(\pi_o) \doteq \mathbb{E}[V(\pi_o)]. \tag{4.5}$$

Given this definition, it is possible to see that $O^*_{\psi_\mu}$ is simply the set of K options with the highest average return. Let us denote the mean and standard deviation of this option set as $\mu^* \doteq \mathbb{E}[V(O^*_{\psi_\mu})]$ and $\sigma^* \doteq (\text{Var}[V(O^*_{\psi_\mu})])^{\frac{1}{2}}$, respectively. Note, again, that these statistics are defined with respect to the set of early-life options selected solely to maximize *expected return*, but without taking into account any risks associated with executing

the options. To account for this possibility, we introduce two risk-aware metrics for evaluating the performance of candidate early-life options, so as to quantify both how well they perform over a wide range of possible contexts (tasks and initial states), as well as the possible *risks* involved with executing them.

A common way of incorporating the notion of risk, when evaluating policies, is via the Markowitz mean-variance model [Markowitz 1959]. According to this model, one should prefer policies π that maximize $\mathbb{E}[V(\pi)] - \beta \text{Var}[V(\pi)]$, where $\beta \in \mathbb{R}$ regulates the penalty on return variability. This criterion imposes a trade-off that penalizes expected return in favor of policies with a lower variance. It does not care, however, whether the variance is equally caused by above-average and below-average returns, or whether, e.g., most of the variance results from extremely positive (above-average) returns. In this latter case, it should be intuitively clear that return variance is desirable and should not be penalized. To more carefully model the different ways in which return variability may positively or negatively affect the desirability of a candidate option, we introduce two novel metrics for evaluating their performances:

$$\psi_{-}(\pi_o) = \mathbb{E}[V(\pi_o)] - k\text{Pr}\big(V(\pi_o) < (\mu^* - \alpha\sigma^*)\big), \tag{4.6}$$

which we call the *Negative Tail-Averse* metric; and

$$\psi_{+}(\pi_o) = \mathbb{E}[V(\pi_o)] + k\text{Pr}(V(\pi_o) > \mu^*), \tag{4.7}$$

which we call the *Positively-Skewed* metric. The Negative Tail-Averse metric ψ_{-} (Eq. 4.6) takes into account both the mean return of an option and the probability that its execution will result in returns that are α standard deviations below the mean of $O^*_{\psi_\mu}$. Intuitively, it first characterizes the negative tail of the return distribution of the options set constructed greedily solely based on options' mean returns (via the ψ_μ criterion). Then, it trades-off between achieving a high expected return while minimizing the probability that the returns of the option may fall in that tail. This metric favors early-life options that are similar in nature to the ones identified by ψ_μ (in terms of large expected returns) but that also are risk-aware—they would only get selected by Eq. 4.4 if they are unlikely to produce extremely poor performances. The Positively-Skewed metric ψ_{+} (Eq. 4.7), by contrast, favors early-life options that have both a large expected return and whose returns tend (with high probability) to be situated above the mean of $O^*_{\psi_\mu}$. Intuitively, it cares about the mean return of an option and also about maximizing the probability that

its execution will, *most of the time*, produce better-than-expected returns—even if at the risk of sometimes (with low probability) producing behaviors with subpar performances. These are risk-seeking early-life options since they favor behaviors that tend to generate extremely positive returns while accepting the risk of a possibly longer negative tail. This metric results in options with a return distribution that is positively-skewed—thus its name. The preference for actions with positively-skewed returns has been extensively studied in Prospect Theory. Evidence exists, for instance, that when losses are costlier than gains, investors tend to favor stocks with positively-skewed returns [Kumar, Motahari and Taffler 2018].

4.4 Constructing Early-Life Option Sets

We approximate the solution to Eq. 4.4 using a three-step procedure: **(a)** generation of a set containing N candidate early-life options; **(b)** approximation of each candidate option's return distributions by evaluating its return over Z different random contexts; and **(c)** selecting the top K highest-ranking options according to a given metric ψ. The generation of candidate options is done by sampling N possible random contexts $C_i = (\tau, s_0)$ in which the agent could have to perform in the early stages of its life. We place the agent in such a context and record a short trajectory of optimal actions drawn from a near-optimal policy for τ. The action trajectory is then used to construct a candidate option's policy π_o capable of (approximately) reproducing the behavior observed in the trajectory. This can be achieved via imitation learning algorithms or standard supervised learning techniques. The process of approximating a candidate option's return distribution is simpler: for each candidate option π_o, we generate a large number Z of possible random contexts and execute the option in each one. We then use the resulting Z return observations of π_o to construct an approximation of the return distribution of the option; this can be achieved, e.g., by using kernel density estimation techniques over the returns. Finally, the process of selecting the top K best options according to a metric ψ requires only that we use the estimated return distribution of each option π_o to compute its corresponding metric $\psi(\pi_o)$. The K highest-ranking candidate options w.r.t. ψ can then be used to define the set O^*_ψ.

Notice that the process above requires solving at most Z sample tasks, which may be costly. However, these costs are amortized in the long-term. In particular, after a set of early-life options has been defined, it can be used, without any additional costs, in order

to optimize policies for *any* other tasks drawn from a possibly infinite task distribution. Furthermore, the identified early-life options are, by construction, task-agnostic: they result in reflexes that can prevent the agent from executing poor-performing behaviors at the early stages of its lifetime, even when used in a wide range of possible tasks and situations. For more details on the process of constructing early-life options, see Algorithm 1.

Algorithm 1 Construction of Early-Life Option Sets

(a) Generate Candidate Options
1. Draw a small set W of tasks $\{\tau_1, \ldots, \tau_M\}$ from P
2. Compute a near-optimal policy $\pi^*_{\tau_i}$ for each task in W
for i from $1 \ldots, N$ **do**
  - Sample a random context $C_i = (\tau, s_0)$
    $\cdot$ (τ is uniformly drawn from W; s_0 is drawn from d_0)
  - Execute π^*_τ for T steps, starting in s_0
  - Record the resulting action trajectory $h_i = (a_1, \ldots, a_T)$
  - Define an option π_{o_i} that reproduces the behavior in h_i
end for
3. Return the set of candidate options $\{\pi_{o_1}, \ldots, \pi_{o_N}\}$

(b) Estimate Return Distribution of Candidate Options
for each candidate option π_{o_i} **do**
  for j from $1 \ldots, Z$ **do**
    - Sample a random context $C_j = (\tau, s_0)$
      $\cdot$ (τ drawn uniformly from W; s_0 is drawn from d_0)
    - Execute π_{o_i} for T steps, starting in s_0
    - Record the return R_j achieved by π_{o_i} in T steps
  end for
  - Let $R(i) = \{R_1, \ldots, R_Z\}$ be the returns of π_{o_i}
  - Use $R(i)$ to approximate the return distribution of π_{o_i}
end for
- Return the approximate return distribution of each option

(c) Select Top K Candidate Early-Life Options
1. Let ψ be the option evaluation metric of interest
2. Compute $\psi(\pi_{o_i})$ for each candidate option π_{o_i}
3. Return the set O^*_ψ with the K highest-ranking options

5 EXPERIMENTS

In this chapter, we empirically show that our method succeeds in identifying a set of reusable early-life options that generalize across many tasks, and that these options allow for the agent to perform well in the early stages of their lives, even when no learning is taking place. We evaluate our proposed method in three simulated robotics tasks selected due to their sensitivity to poor-performing initial policies (see Section 5.1). We also show, in Section 5.2, that an agent equipped with a set of optimized early-life behaviors, even if acting randomly, is capable of reproducing the behavior of agents that were allowed to train for hundreds of thousands of steps.

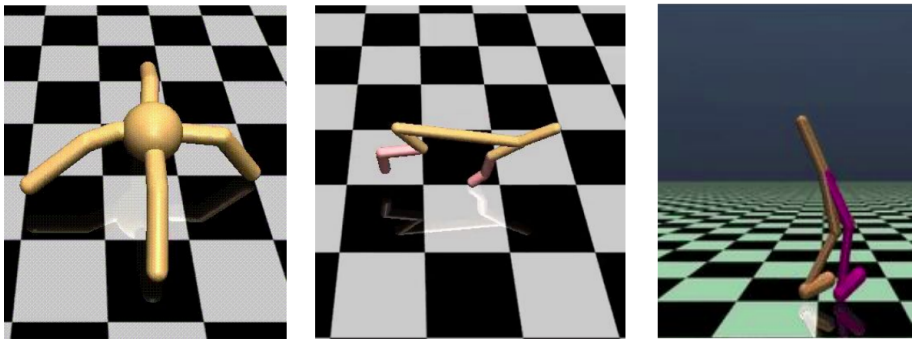
5.1 Setting

We evaluate our method, as previously mentioned, on three simulated robots. Ant is a quadruped robot with 13 rigid links, including four legs and a torso, along with 8 actuated joints. Half-Cheetah is a planar biped robot composed of two legs, a torso, and 6 actuated joints. Walker2D is a planar biped robot consisting of two legs and a torso and with 6 actuated joints. They are illustrated, respectively, in Figure 5.1. The state space of all robots includes information about its current position and velocity; the action space corresponds to decisions about joint torques. The reward functions of all robots incentivize proper walking while consuming as little energy as possible. More specifically, the reward function is a combination of five terms that balance different objectives: not falling down; making progress towards walking forward; keeping electricity costs under control; keeping joints within their allowed safe configurations; and limiting the costs associated with hitting the ground. In our experiments, we are interested, in particular, in defining different a family of walking tasks, each one corresponding to the problem of learning to walk under a given electricity cost. Let $A = [a_1, \ldots, a_J]$ be the set of robot actions, which in this case corresponds to the torques applied to each of the J actuated joints. Let Υ_i be the velocity of the i-th joint. Then, the electricity cost, EC, at a given timestep, is defined as:

$$EC = \text{e\_cost}\frac{\sum_{i=1}^{J}|a_i\Upsilon_i|}{J} + \text{st\_cost}\frac{\sum_{i=1}^{J}(a_i)^2}{J}, \tag{5.1}$$

where *e_cost* is a constant representing the cost for applying a torque to a given joint/motor (defined, by default, as -2.0); and where *st_cost* is a constant representing the cost for

Figure 5.1: From left to right: the Ant robot; the Half-Cheetah robot; and the Walker2D robot.



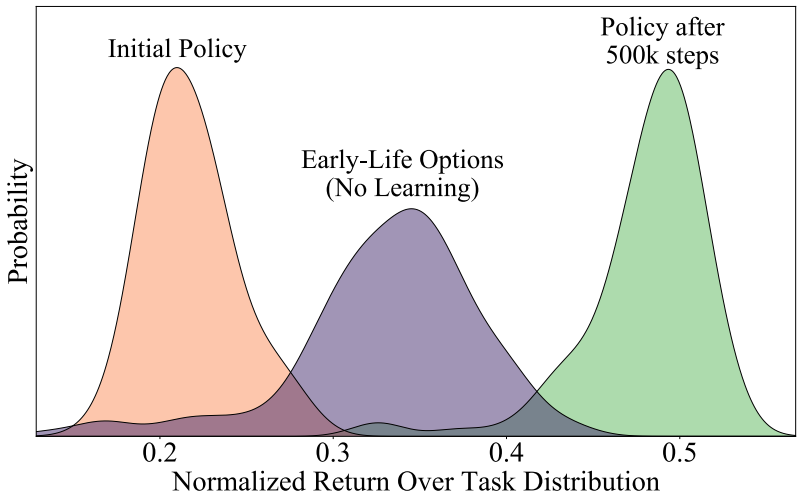
running an electric current through a motor/joint (defined, by default, as -0.1).

In our experiments, we define the family of tasks Ψ that each robot may face by defining an infinite number of MDP variations; these were obtained by modifying the reward function of each robot by varying the constant *e_cost* along the continuous range of $[-2.4, -1.4]$. Higher costs keep the robot from moving efficiently, while lower costs often create unstable walking movements since the robot has no incentive to pursue parsimonious movements. We train all agents using the Proximal Policy Optimization algorithm, described in Section 2.1. We define the task distribution P to be uniform over the range of possible electricity costs and the initial state distribution d_0 to be the one that results from initializing the agent in a standard pose and running a random policy over primitive actions for a random amount t of steps, where $t \sim U[0, 100]$. In all experiments, we generated $N = 600$ candidate early-life options and estimated their return distributions by evaluating each option in $Z = 300$ possible contexts. Option policies were open-loop sequences lasting $T = 200$ steps and were constructed in order to directly mimic each sampled action trajectory, h_i. Return distributions in our analyses were obtained via kernel density estimation over the set of Z returns collected for each option. All experiments consider the case where we optimize option sets of size $K = 5$.

5.2 Results

Our first experiment aims at demonstrating that our method is capable of learning behaviors akin to primitive reflexes in infant mammals: our "infant agent", born with early-life options, is capable of directly producing rudimentary behaviors with performances comparable to those acquired by learning agents optimizing a policy for hundreds

Figure 5.2: [Ant Robot] Return distribution achieved with early-life options vs. two learning agents at different moments in their lifetimes.

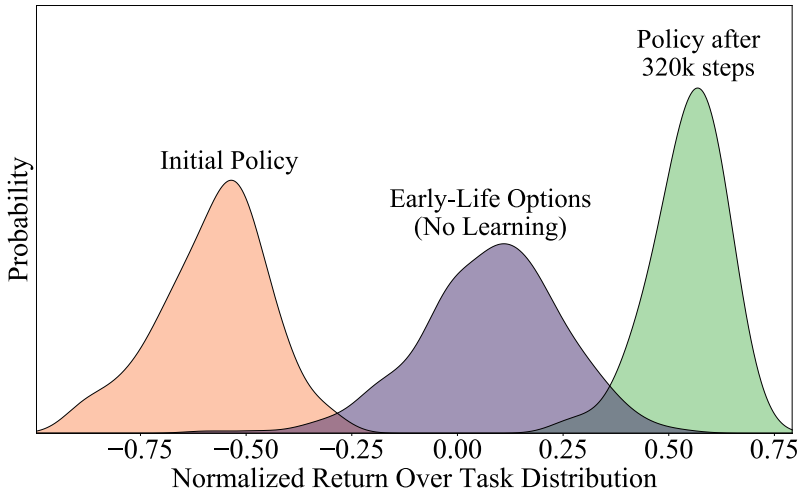


Source: The Authors

of thousands of steps. Figures 5.2 and 5.3 depict the distribution of returns achieved by different agents when evaluated over $Z = 300$ random contexts, where each context specifies a random task (a random setting of the electricity cost) and a random initial state. Figure 5.2 shows the performance achieved by our Ant agent, when equipped with early-life options identified via the Maximum-Mean metric ψ_{μ}, and when evaluated during the hardest period of its lifetime: immediately after it is initialized, when it has not yet learned a policy, and when its behavior is still essentially a random policy over primitive actions and options. We compare the distribution of returns achieved by our agent with the distributions achieved by two other agents: a learning agent operating under its initial random policy over primitive actions; and a learning agent acting under a policy acquired after 500k training steps. The mean performance of our agent (which is *not learning*, but merely selecting early-options at random) is comparable to that of an agent trained with PPO for approximately 200k steps.

Figure 5.3 presents a similar analysis but for the Half-Cheetah robot. We again observe that our optimized early-life options allow the agent—even when selecting uniformly at random from the options set—to perform similarly to an agent trained with PPO for approximately 190k steps. As expected, early-life options allow the agents to perform well (even before any learning has taken place) in the very early stages of their lifetimes. Here, we once again emphasize that our objective is *not* to construct options that accelerate the acquisition of an optimal policy, but to identify options that provide a type of motor bias that allows agents to perform well and to avoid catastrophic failures at the early stages of their lifetimes.

Figure 5.3: [Cheetah Robot] Return distribution achieved with early-life options vs. two learning agents at different moments in their lifetimes.



Source: The Authors

Our second experiment analyzes the properties of different metrics for evaluating early-life options: the Negative Tail-Averse metric, ψ_{-}, and the Positively-Skewed metric, ψ_{+}. Figure 5.4 compares the return distribution achieved by the best option set w.r.t. ψ_{μ}, and the best option set w.r.t. ψ_{-}. It highlights, in particular, the improvement (decrease) in the area under the curve (AUC) of the distribution's left tail that results from using ψ_{-}. This confirms that ψ_{-} is capable of identifying options with both high mean return and with a smaller negative tail—thereby producing behaviors that minimize the probability of extremely poor performances. Detailed numerical results regarding improvements to the negative tail's AUC are shown in Table 5.1. Interestingly, the use of a metric that trades off mean return and negative tail minimization often resulted in options with a *higher* mean return, compared to that achieved by greedily constructing options based on ψ_{μ}.

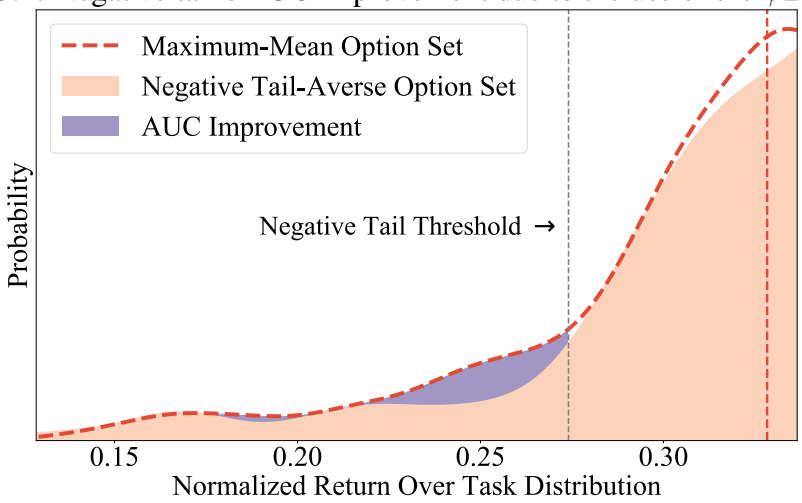
Table 5.1: Mean return & negative tail's AUC under ψ_{μ} and ψ_{-}.

	Return under ψ_{μ}		**Return under ψ_{-}**		
	Mean	AUC	Mean	AUC	AUC Improvement
Ant	0.328	0.133	**0.332** ($k = 0.05$)	**0.097**	**27.0%**
Cheetah	0.082	0.143	**0.079** ($k = 0.75$)	**0.127**	**11.1%**
Walker	0.122	0.243	**0.115** ($k = 0.75$)	**0.192**	**20.9%**

Source: The Authors

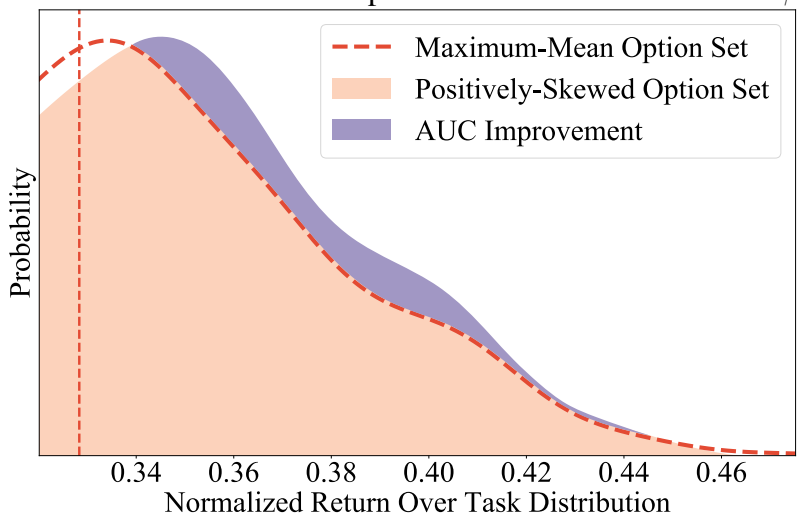
Figure 5.5 presents a similar analysis but regarding the use of the Positively-Skewed metric, ψ_{+}, for selecting options. As previously discussed, ψ_{+} cares not only about the mean return of an option, but also about maximizing the probability that its exe-

Figure 5.4: Negative tail's AUC improvement due to the use of the ψ_- metric.



Source: The Authors

Figure 5.5: Above-the-mean AUC improvement due to the use of the ψ_+ metric.



Source: The Authors

cution will, *most of the time*, produce better-than-expected returns. This figure highlights the improvement (increase) in the area under the curve (AUC) to the right of the distribution's mean. It confirms that the use of this metric is capable of identifying options with both high mean return and that favors behaviors that tend to generate positive returns while accepting the risk of a possibly longer negative tail. Detailed numerical results of the above-the-mean AUC improvement resulting from ψ_+ are shown in Table 5.2. Once again, the use of a risk-aware metric often resulted in options with *higher* mean return, compared to that achieved by greedily constructing options based on ψ_μ.

Finally, we study the distribution of values generated by one of our metrics (ψ_+). Figure 5.6 shows that very few of the candidate options have large metric values: only 0.67% of all options have performance within 10% of the performance of the best op-

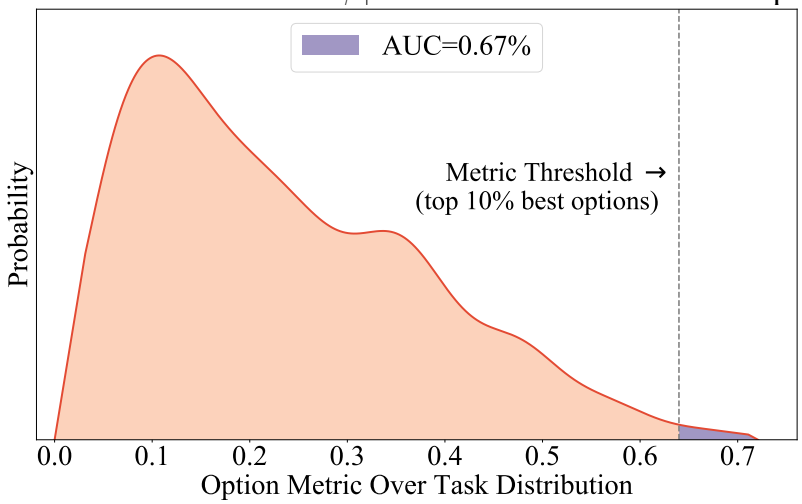
Table 5.2: Mean return & above-the-mean AUC under ψ_μ and ψ_+.

	Return under ψ_μ		**Return under ψ_+**		
	Mean	AUC	Mean	AUC	AUC Improvement
Ant	0.328	0.533	**0.334** ($k = 0.05$)	**0.597**	**12.0%**
Cheetah	0.082	0.550	**0.082** ($k = 0.05$)	**0.550**	0.0%
Walker	0.122	0.563	**0.125** ($k = 0.10$)	**0.595**	**5.7%**

Source: The Authors

tion. This is surprising since all option policies were generated by sampling directly from optimal policies for each given task (Algorithm 1). This observation implies that it is unlikely that such options, if selected randomly from the set of candidates, would directly generalize over a wide range of contexts, and further reinforces the need for a careful optimization process for identifying efficient early-life options, such as the one performed by our algorithm.

Figure 5.6: Distribution of the ψ_+ metric values over candidate options.



Source: The Authors

6 DISCUSSION

In this chapter we discuss our conclusions (Section 6.1) and then point out future research directions (Section 6.2).

6.1 Conclusions

We have introduced a method for identifying short-duration reusable motor behaviors—*early-life options*—that allow robots to perform well in the very early stages of their lives. Most of the existing work in the area of option generation focuses on identifying options that help an agent throughout its entire lifetime [Bacon, Harb and Precup 2017, Machado, Bellemare and Bowling 2017, McGovern and Barto 2001, Harb et al. 2018]. We, by contrast, are motivated by the observation that many infant mammals have primitive reflexes that are key to guarantee their safety and to facilitate learning in the early stages of their lives.

In this work, we introduced a method capable of generating early-life options by optimizing different performance metrics that take into account both an option's mean return and the potential risks that its execution may cause. We introduced three performance metrics after which ELOs may be optimized: one that only takes into account the expected performance, one that tries to maximize the expected performance while minimizing the risk of poor performance, and one that tries to maximize the expected performance while maximizing the chance of having better-than-expected performance.

Although identifying early-life options may incur additional costs (in terms of sample complexity), such costs are amortized in the long-term. In particular, after a set of early-life options has been defined, it can be used, without any additional costs, in order to optimize policies for *any* other tasks drawn from a possibly infinite task distribution. Furthermore, the identified early-life options are, by construction, task-agnostic: they result in reflexes that can prevent the agent from executing poor-performing behaviors at the early stages of its lifetime, even when used in a wide range of possible tasks and situations.

We evaluated our option-discovery technique on three simulated robots operating under different battery consumption constraints. We showed that random policies over learned early-life options are sufficient to produce performances similar to those of policies trained for hundreds of thousands of steps. We empirically observed that our

technique is capable of discovering meaningful early-life options: short-duration reusable behaviors that generalize across tasks and that, once identified, can ensure performance (expected return) similar to that of agents which were allowed to train for hundreds of thousands of steps.

6.1.1 Publication and Awards

The method introduced in this work was published in the 9th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-Epirob) 2019 [Weber et al. 2019]. It also received two university-wide awards, both for Best Artificial Intelligence Undergraduate Research Project and Best Natural Sciences Undergraduate Research Project.

6.2 Future Work

A natural future research direction involves extending our method to generate early-life options expressed in terms of closed-loop policies, so that they can represent not only fixed, short-duration reflexes, but also reflexes that can adapt to the current state of the environment. In this section, we briefly discuss an initial idea (which we are currently investigating) to achieve this objective. We argue that modifying our optimization setting to one that supports closed-loop policies will naturally allow us to consider, and optimize for, additional properties that are desirable in early-life options:

- to result in good performance (expected return) even when the agent deploys an initial nearly-random policy over options. In other words, we wish to maintain our original definition of useful early-life behaviors: behaviors that, by construction, are helpful in the very beginning of an agent's life, before any kind of learning has taken place;
- to be reusable across tasks drawn from a family of different but related problems;
- to have initiation sets and termination conditions optimized to ensure that each early-life option is available only in specific situations, thereby ensuring that each option will represent a different and specialized behavior. Consider, for instance, an early-life option/reflex for moving the agent's hand away from fire. This ELO should be executable if and only if the agent is in a corresponding situation where

it is touching a hot surface;

- to have bounded minimum and maximum complexity, where complexity is measured as the average number of timesteps involved in executing a particular option. In particular, we wish to be able to discover ELOs with minimum and maximum expected durations. This would ensure that the discovered early-life options would not converge to single-action options (as often occurs, e.g., under the Option-Critic framework), neither would converge to policies that solve an entire task (since these, by definition, would not be reusable across different tasks).

Having intuitively defined our new objectives, we propose to incorporate them in a cost function adapted from the work introduced in [Khetarpal et al. 2020]. In this work, the authors present an extension to the Option-Critic architecture (discussed in Section 3.2) where agents learn not only option policies, termination conditions, and policies over options, but also *interest functions*, which encode initiation sets of options. The proposed introduced in [Khetarpal et al. 2020], however, does not directly result in options with the properties that we outlined above. First, it only optimizes options that are useful over the entire lifetime of an agent, while our objective is to learn early-life options. Secondly, it does not allow for the agent to optimize behaviors that are reusable across many tasks; they only consider the single-task setting. Finally, it does not explicitly enforce minimum and maximum option complexity bounds, which is necessary to guarantee that the resulting technique will avoid learning degenerate or non-reusable options.

Let $I_o : S \rightarrow \mathbb{R}$ be an *interest function* associated with option o, which indicates the preference of the agent for initiating option o when in state s. This is a soft representation of an initiation set. In what follows, we argue that the objectives outlined in the beginning of this section can be captured by the following cost function:

$$J(\theta) = \sum_{\tau \in \Psi} \sum_{s \in S} \sum_{o \in O} \pi_I(o|s) Q^{\theta}(s, o|\tau) - \lambda \phi(\epsilon), \tag{6.1}$$

where $\tau \in \Psi$ is a possible task that the agent may have to solve; $s \in \mathcal{S}$ is a possible state; $o \in O$ is an early-life option; and $\theta = [\theta_I, \theta_\pi, \epsilon]$ are the parameters being optimized. In particular, let $\theta_I = [\theta_{I_{o_1}}, \ldots, \theta_{I_{o_K}}]$ be the parameters encoding the interest function I_{o_i} associated with each option o_i; let $\theta_\pi = [\theta_{\pi_{o_1}}, \ldots, \theta_{\pi_{o_K}}]$ be the parameters that represent the policy of each policy o_i; and let ϵ be a real-valued parameter associated with the termination condition of options. Furthermore, let $Q^{\theta}(s, o|\tau)$ be the expected return

achievable by option o, when initiated in state s, while solving task τ. Finally, let λ and ϕ be regularization terms related to bounding to the complexity of each option; we discuss these later.

In this setting, a stochastic policy over options, $\pi_I(o|s)$, can be defined as:

$$\pi_I(o|s) = \frac{\pi(o|s)I_o(s)}{\sum\limits_{o' \in O} \pi(o'|s)I_{o'}(s)} \tag{6.2}$$

where $I_o : S \rightarrow \mathbb{R}$ is the interest function indicating the preference of the agent for initiating the option o in state s, and where (by our definition of early-life options) we assume that $\pi(o|s)$ is an uniform random policy. Hence, $\pi_I(o|s)$ can be rewritten as:

$$\pi_I(o|s) = \frac{I_o(s)}{\sum\limits_{o' \in O} I_{o'}(s)}. \tag{6.3}$$

We define $Q^\theta(s, o|\tau)$ similarly to the corresponding formulation of expected option return introduced in the Option-Critic paper:

$$Q^\theta(s, o|\tau) = \sum_{a \in A} \pi_o(a|s) Q_U^\theta(s, o, a|\tau) \tag{6.4}$$

where $Q^\theta(s, o|\tau)$ is the option-value function of option o, when in state s, while executing task τ; where $\pi_o(a|s)$ is the policy of option o (defined over primitive actions); and where $Q_U^\theta(s, o, a|\tau)$ is the action-value function associated with action a, when executing a particular early-life option o in the state s of task τ:

$$Q_U^\theta(s, o, a|\tau) = r + \gamma \sum_{s' \in S} p(s'|s, a)((1 - \epsilon)Q^\theta(s', o|\tau) + \epsilon V^\theta(s'|\tau)) \tag{6.5}$$

Here, r is the reward for the current timestep, $p(s'|s, a)$ is a transition probability function, and $V^\theta(s'|\tau)$ is the expected return from state s' while solving task τ.

Finally, recall that, as outlined previously, we wish to constrain the minimum and maximum expected duration of the early-life options being optimized. In order to do that, we define ϵ as the (fixed) per-timestep termination probability of any early-life option. Notice that this is one of the parameters in θ, the vector of parameters being optimized by our cost function (Eq. 6.1). If ϵ is the probability that an ELO will terminate at any given timestep, then the random duration of each option follows a geometric distribution with the expected value given by $(1 - \epsilon)/\epsilon$. Based on this observation, we can define ϕ to be

a problem-specific, duration-dependent regularization function that penalizes too-short or excessively long ELOs. Let us assume, for instance, that we wish to guarantee that early-life options will have an average duration of ρ timesteps. Then, a trivial definition of ϕ would be:

$$\phi(\epsilon) = (\rho - ((1 - \epsilon)/\epsilon))^2, \tag{6.6}$$

where this option-complexity cost can be incorporated as part of our objective function (Eq. 6.1) and weighted by a regularization term λ.

We are currently designing gradient-based methods capable of efficiently optimizing the above-described mathematical objective, which we hope will allow agents to learn reusable closed-loop early-life options. We believe that this is a non-trivial extension of the method introduced in Chapter 4, and hope that it will contribute to advancing the state-of-art in option-discovery techniques. We expect to submit this novel idea for publication within the next few months.

REFERENCES

A.D.A.M. **Infant Reflexes (Animated Dissection of Anatomy for Medicine)**. 2019 (accessed 11/13/2020). Available from Internet: <http://thnm.adam.com/content.aspx?productid=617&pid=1&gid=003292>.

BACON, P.; HARB, J.; PRECUP, D. The option-critic architecture. In: **Proceedings of the 31st AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2017.

BACON, P.-L. **On the bottleneck concept for options discovery**. Dissertation (Master) — McGill University, 2013.

BELLMAN, R. **Dynamic Programming**. 1. ed. Princeton, NJ, USA: Princeton University Press, 1957.

BELLMAN, R. A markovian decision process. **Journal of mathematics and mechanics**, JSTOR, p. 679–684, 1957.

BERK, L. **Child Development**. [S.l.]: Allyn & Bacon/Pearson, 2009. (Child Development).

BERTHIER, N.; KEEN, R. Development of reaching in infancy. **Experimental Brain Research**, v. 169, p. 507–518, 2006.

DAYAN, P. Improving generalization for temporal difference learning: The successor representation. **Neural Computation**, MIT Press, v. 5, n. 4, p. 613–624, 1993.

DOSHI-VELEZ, F.; GHAHRAMANI, Z. A comparison of human and agent reinforcement learning in partially observable domains. In: **Proceedings of the 33th Annual Meeting of the Cognitive Science Society**. [S.l.: s.n.], 2011.

DUBEY, R. et al. Investigating human priors for playing video games. In: **Proceedings of the 35th International Conference on Machine Learning**. [S.l.: s.n.], 2018. p. 1349–1357.

EYSENBACH, B. et al. Diversity is all you need: Learning skills without a reward function. In: **Proceedings of the 6th International Conference on Learning Representations**. [S.l.: s.n.], 2018.

GARCIA, F. M.; SILVA, B. C. da; THOMAS, P. S. A compression-inspired framework for macro discovery. In: **Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems**. [S.l.: s.n.], 2019. p. 1973–1975.

GOEL, S.; HUBER, M. Subgoal discovery for hierarchical reinforcement learning using learned policies. In: **Proceedings of the 16th International Florida Artificial Intelligence Research Society Conference**. [S.l.: s.n.], 2003. p. 346–350.

HARB, J. et al. When waiting is not an option: Learning options with a deliberation cost. In: **Proceedings of the 32nd AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2018.

HARUTYUNYAN, A. et al. Learning with options that terminate off-policy. **Computing Research Repository (CoRR)**, 2017.

KHETARPAL, K. et al. Options of interest: Temporal abstraction with interest functions. In: **Proceedings of the 34th AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2020.

KOBER, J. et al. Reinforcement learning to adjust parametrized motor primitives to new situations. **Autonomous Robots**, Springer Netherlands, 2012. ISSN 0929-5593.

KUMAR, A.; MOTAHARI, M.; TAFFLER, R. **Skewness Preference and Market Anomalies**. [S.l.], 2018.

LIU, M. et al. The eigenoption-critic framework. In: **NIPS Workshop on Hierarchial Reinforcement Learning**. [S.l.: s.n.], 2017.

MACHADO, M. C.; BELLEMARE, M. G.; BOWLING, M. A laplacian framework for option discovery in reinforcement learning. In: **Proceedings of the 34th International Conference on Machine Learning-Volume 70**. [S.l.: s.n.], 2017. p. 2295–2304.

MAHADEVAN, S. Proto-value functions: Developmental reinforcement learning. In: **Proceedings of the 22nd International Conference on Machine Learning**. [S.l.: s.n.], 2005. p. 553–560.

MARKOWITZ, H. M. Portfolio selection: Efficient diversification of investment. **The Journal of Finance**, v. 15, 12 1959.

MCGOVERN, A.; BARTO, A. Automatic discovery of subgoals in reinforcement learning using diverse density. In: **Proceedings of the 18th International Conference on Machine Learning**. [S.l.: s.n.], 2001.

MCGOVERN, A.; SUTTON, R. S. Macro-actions in reinforcement learning: An empirical analysis. **Computer Science Department Faculty Publication Series**, p. 15, 1998.

ONIS, M. WHO motor development study: windows of achievement for six gross motor development milestones. **Acta paediatrica**, Wiley Online Library, v. 95, p. 86–95, 2006.

RAMESH, R.; TOMAR, M.; RAVINDRAN, B. Successor options: an option discovery framework for reinforcement learning. In: AAAI PRESS. **Proceedings of the 28th International Joint Conference on Artificial Intelligence**. [S.l.], 2019. p. 3304–3310.

SCHULMAN, J. et al. High-dimensional continuous control using generalized advantage estimation. In: **Proceedings of the 4th International Conference on Learning Representations**. [S.l.: s.n.], 2016.

SCHULMAN, J. et al. Proximal policy optimization algorithms. **Computing Research Repository (CoRR)**, 2017.

SIEGLER, R. S.; DELOACHE, J. S.; EISENBERG, N. **How children develop**. [S.l.]: Macmillan, 2003.

SILVA, B. da; KONIDARIS, G.; BARTO, A. Learning parameterized skills. In: **Proceedings of the 29th International Conference on Machine Learning**. [S.l.: s.n.], 2012.

ŞIMŞEK, Ö.; BARTO, A. G. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In: **Proceedings of the 21st International Conference on Machine Learning**. [S.l.: s.n.], 2004. p. 95.

ŞIMŞEK, Ö.; WOLFE, A. P.; BARTO, A. G. Identifying useful subgoals in reinforcement learning by local graph partitioning. In: **Proceedings of the 22nd International Conference on Machine Learning**. [S.l.: s.n.], 2005. p. 816–823.

SPELKE, E. S. Principles of object perception. **Cognitive Science**, v. 14, n. 1, 1990.

STULP, F. et al. Learning compact parameterized skills with a single regression. In: **Proceedings of the IEEE-RAS International Conference on Humanoid Robots**. [S.l.: s.n.], 2013.

SUTTON, R.; PRECUP, D.; SINGH, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. **Artificial Intelligence**, v. 112, p. 181–211, 1999.

WATKINS, C. J. C. H. **Learning from delayed rewards**. Thesis (PhD) — King's College, Cambridge, 1989.

WEBER, A. et al. Identifying reusable early-life options. In: **Proceedings of the 9th Joint IEEE International Conference on Development and Learning and Epigenetic Robotics**. [S.l.: s.n.], 2019.