Eligibility Traces for Options

Ayush Jain
McGill University
Montreal
ayush.jain@mail.mcgill.ca

Doina Precup
McGill University
Montreal
dprecup@cs.mcgill.ca

ABSTRACT

Temporally extended actions not only represent knowledge in the hierarchical setup in reinforcement learning, they also improve exploration while reducing the complexity of choosing actions. The option framework provides a concrete way to implement and reason about temporal abstraction. This work attempts to test the utility of eligibility traces with options and find good ways of doing multi-step intra-option updates. Three algorithms, based on off-policy methods - importance sampling, tree-backup and retrace, are proposed for using eligibility traces with options.

KEYWORDS

Temporal abstraction; Option framework; Off-policy methods; Intra-option updates; Eligibility traces

ACM Reference Format:
Ayush Jain and Doina Precup. 2018. Eligibility Traces for Options. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 9 pages.

1 INTRODUCTION

We humans cope with the extraordinary complexity of the real world in part by thinking hierarchically. Our decision making routinely involves a choice among high-level and low-level behaviors over a broad range of time scales. Consider a person deciding to drink coffee. This highest level task can be broken down into: first, the person getting to the pantry area; and second, actually making coffee. To go to the pantry area, the person, for example, would get to the hallway, then walk down the hallway, and then turn left to enter. These high-level steps will be interleaved with low-level series of muscle twitches. Having reached the pantry area, the person has to choose a coffee maker and beans at the high level. Medium level steps would be grinding coffee beans and boiling water. Lower level steps would be the hand movements to add coffee to filter. The above example illustrates how humans regularly think and plan hierarchically, switch between different level behaviors and strategize about objectives at a broad time scale. The concept of hierarchical learning and planning has a strong appeal to humans, and so does to artificial intelligence research community. Faculty to manage hierarchical and sequential tasks is realized in autonomous agents through temporal abstraction, a concept where abstract actions represent sequences of lower-level actions.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden. © 2018 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Temporal abstraction has been explored in AI at least since the early 1970s [9, 10, 12, 14, 17, 21, 24, 29, 35]. It has also been a focus and an appealing aspect of qualitative modeling approaches to AI [4, 8, 15, 30] and has been explored in robotics and control engineering [1, 5, 6, 19]. Temporally extended actions have been shown to generate shorter plans, reduce the complexity of choosing actions, increase robustness against misspecified models, and improve exploration. In reinforcement learning, *Options* [34] provide an intuitive framework to represent, reason about and plan with temporally extended actions. Interest in temporal abstraction has recently increased substantially after successful methods of defining and creating such abstractions [2, 7, 16, 18, 20]. The values of the options and their models are learned by looking *within* the options - intra-option learning algorithms [33]. In this paper, we aim to significantly speedup the learning process.

Eligibility traces have been shown to speed up reinforcement learning and make it more robust [13]. They implement multi-step returns in a way to obtain computational advantages. To extend learning methods for options with eligibility traces, off-policy [32] nature of the algorithm is important. We want to learn about all consistent options while behaving according to a particular option.

In this work, off-policy eligibility traces for intra-option learning [33] methods have been proposed.

2 OPTIONS

The *options* framework [34] provides a concrete way to incorporate temporal abstraction into reinforcement learning with no change to the existing setup. The framework naturally represents temporal abstractions and allows for the integration of learning and planning. An option O is defined as a triple $(\mathcal{I}_o, \pi_o, \beta_o)$ comprising initiation set $\mathcal{I}_o \subseteq \mathcal{S}$, stationary internal policy π_o which defines a distribution over all possible primitive actions from a given state, and termination function β_o defining the probability of terminating in a given state.

An option o is *feasible* in a state s $\equiv$ s $\in \mathcal{I}_o$. Given a set of options, their initiation sets define a set of available options O_s $\forall s \in \mathcal{S}$. A hierarchical Markov policy $h(\cdot)$ defines a distribution over O_s. In call-and-return execution model, once an option o is selected for execution from $h(s_t)$, internal policy π_o determines the sequence of actions until termination condition $\beta_o(s_{t+k})$ is met, then, a new option o' is selected according to $h(s_{t+k})$. This process is repeated until the end of the episode. $\mathcal{I}$ and β limit an option's applicability to a part of state space $\mathcal{S}$ and hence policy π needs to be defined for only that section of the state space.

As is clear from the structure of an option, actions can be considered a special case of options - *primitive options*. Each primitive option is available whenever corresponding action is available, it lasts exactly one step and selects that particular action w.p. 1. The term *options* is a generalization for primitive as well as temporally extended actions.

Active option o_t, at time t, depends on the current state s_t, and on whether previously active option o_{t-1} terminated in s_t or not. The process of selecting active option can be made easier by folding the termination conditions β in the hierarchical distribution over options $h(\cdot)$. After folding, the new hierarchical policy over options μ can be described as:

$$\mu(o|s_t, o_{t-1}) = \begin{cases} (1-\beta_{o_{t-1}}(s_t))+\beta_{o_{t-1}}(s_t)h(o_{t-1}|s_t) & if\, o=o_{t-1} \\ \beta_{o_{t-1}}(s_t)\ h(o|s_t) & else \end{cases} \quad (1)$$

Overlaying a set of options over a base MDP leads to a semi-Markov decision process (SMDP) [27]. The time difference in occurrence of decision points in the induced SMDP is a random variable. Based on the optimal Bellman equations, state-option values and option models can be learned using the SMDP learning methods [3, 25]. For example, option values can be learned using SMDP Q-learning as -

$$Q(s,o) \longleftarrow Q(s,o) + \alpha[r + \gamma^k max_{o' \in O} Q(s', o') - Q(s,o)], \quad (2)$$

where s' is the state in which option o terminates, k denotes the random time difference between states s and s', and r denotes the cumulative discounted reward during this time.

2.1 Intra-Option Methods

SMDP learning methods treat options as black boxes- an algorithm based on SMDP methods would execute an option to completion, accumulating rewards at each step and then at termination would make one single update to the executing option. An interesting idea is to take advantage of each fragment of the experience.

If an option is Markov, in some sense, it is initiated at every time step. By looking at experience during an option's execution, more efficient Intra-Option methods [33] have been devised. For Markov options, after each time step, experience can be used to update all O_s options that could take the consistent action with some non-zero probability. Intra-Option methods are off-policy learning methods because updates can be applied to all relevant options while behaving according to a particular option's internal policy. A single step intra-option update to learn option values can be performed as -

$$\begin{aligned} Q(s_t, o) \longleftarrow Q(s_t, o) + \alpha[r_{t+1} + \gamma(1-\beta_o(s_t))Q(s_{t+1}, o) \\ + \gamma\beta_o(s_{t+1}) \max_{o' \in O} Q(s_{t+1}, o') - Q(s_t, o)] \end{aligned} \quad (3)$$

Similar updates can be used to learn the reward and transition models of the set of options. These estimates of true options values and option models have been shown to converge to optimal values under normal Q-learning constraints [33].

3 ELIGIBILITY TRACES FOR OPTIONS

TD methods are 1-step algorithms which bootstrap from the value of subsequent states. Monte Carlo methods, on the other hand, wait for the exact return at the end of an episode. TD(λ) methods, i.e. TD augmented with *eligibility traces*, produce a family of methods spanning a spectrum that has Monte Carlo on one end and 1-step TD at another. In this sense, eligibility traces interpolate between TD and Monte Carlo methods.

A more mechanistic way of seeing eligibility traces is to interpret these traces as a short term fading memory of occurrence of events. The trace marks the parameters associated with an event with it's *eligibility* for updates. When an update is to be made, only eligible states get credit or blame for the error weighted according to their eligibility. Eligibility of events decreases as they get temporally far.

Eligibility traces implement multi-step returns in a way to obtain computational advantages. Reinforcement learning algorithms are usually formulated as updates based on events that follow an action over multiple future time steps. The n-step TD methods require all of the intermediate rewards and the state *n* steps after the state being updated. Such formulations, based on looking forward from the updated state, are called *forward views*. Forward views are complex to implement because the update depends on later events/rewards that are not available at the time. However similar updates can be achieved with an algorithm that uses the current TD error and looks backward to recently visited states using an eligibility trace. These alternate ways of implementing learning algorithms are called *backward views*.

While extending options to multi-step return targets, the algorithm needs to learn about the options values and the reward and the transition models of all the different option parallelly. We want to apply updates to all consistent options which assign non-zero probability to the realized action. Therefore, off-policy [32] nature of such an algorithm is important. There are several methods for doing off-policy learning: Importance Sampling [26, 28], Off-policy $Q^\pi(\lambda)$ and $Q^*(\lambda)$ [11], Tree-backup [26] and Retrace(λ) [23]. Following operator $\mathcal{R}$ [23] represents all of the above off-policy methods -

$$\mathcal{R}Q(x,a) = Q(x,a) + \mathbb{E}_\mu\Big[\sum_{t\geq 0} \gamma^t \Big(\prod_{s=1}^{t} c_s\Big)\Big(r_t + \gamma\mathbb{E}_\pi Q(x_{t+1}, \cdot)\Big)\Big], \quad (4)$$

where c_s is some non-negative coefficient specific to an algorithm. The coefficient c_s can be thought of as the trace of the operator $\mathcal{R}$. Table 1 provides the value of c_s and summarizes the constraints and properties of off-policy learning algorithms. In this work, we focused on the Importance Sampling, the Tree-backup, and the Retrace(λ) methods.

Table 1: Summary of off-policy algorithms defined in terms of the Eq. (4), where π is the target policy and b is the behavior. Columns represent algorithm coefficient, variance in the estimates, condition of convergence and use of full return respectively.

	c_s	**Var**	**Convergence**	**Full retn.**
IS	$\frac{\pi(a_s\|x_s)}{b(a_s\|x_s)}$	High	for any π, b	yes
$Q^\pi(\lambda)$	λ	Low	for π close to b	yes
$TB(\lambda)$	$\lambda\pi(a_s\|x_s)$	Low	for any π, b	no
$Retrace(\lambda)$	$\lambda \min(1, \frac{\pi(a_s\|x_s)}{b(a_s\|x_s)})$	Low	for any π, b	yes

In the following subsections, forward views are derived using different off-policy techniques. Then, the derived forward views are transformed in computationally advantageous backward views.

These transformations are exactly equivalent and provide online update methods.

3.1 Importance Sampling

The primary method of doing off-policy learning is to correct expectation of the update from behavior policy to target policy - the Importance Sampling (IS) technique [28]. IS is a classical method of handling mismatch between two different distributions. [26] proposed Per Decision Importance Sampling (PDIS) algorithm, an eligibility trace mechanism for off-policy evaluation using importance sampling technique. The method uses a product of importance sampling ratios to correct for the trajectory probabilities.

Assuming $s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1}, a_{t+1}, r_{t+1}, s_{t+2}, \ldots$ as a trajectory over states, options, actions and rewards, updates $\forall o \in O_{s_t}$ consistent with action a_t can be performed after correcting for the difference in probability of trajectories. An update term for an option $o \neq o_t$ assumes: action a_t was selected according to $\pi_o(\cdot|s_t)$, transition is made to state s_{t+1}, and then option o_{t+1} is used to select action a_{t+1} with probability $\pi_{o_{t+1}}(\cdot|s_{t+1})$. μ, as defined in Eq 1, gives the distribution over all the options available in state s_{t+1} taking into consideration termination probability of previously running option. Importance Sampling ratio ρ correcting for the transition to option o_{t+1} from option o instead of o_t is:

$$\begin{aligned}\rho_{t+1} &= \frac{\mu(o_{t+1}|s_{t+1}, o)\pi_{o_{t+1}}(a_{t+1}|s_{t+1})}{\mu(o_{t+1}|s_{t+1}, o_t)\pi_{o_{t+1}}(a_{t+1}|s_{t+1})} \\ &= \frac{\mu(o_{t+1}|s_{t+1}, o)}{\mu(o_{t+1}|s_{t+1}, o_t)}\end{aligned} \tag{5}$$

Using the backup diagram for Importance Sampling, as seen in Fig 1, n-step returns can be written as:

$$\begin{aligned}G_t^1 &= r(s_t, a_t) + \gamma\rho_{t+1}Q(s_{t+1}, o_{t+1}, a_{t+1}) \\ G_t^2 &= r(s_t, a_t) + \gamma\rho_{t+1}r(s_{t+1}, a_{t+1}) \\ &\quad + \gamma^2\rho_{t+1}\rho_{t+2}Q(s_{t+2}, o_{t+2}, a_{t+2})\end{aligned} \tag{6}$$

Further steps can be derived in a similar fashion. λ-Return, a compound return combining all the n-step return targets, can be used as a target for the learning updates. TD error $\forall o \in O_{s_t}$ can be computed using G_t^λ as:

$$\begin{aligned}\Delta Q(s_t, o, a_t) &= G_t^\lambda - Q(s_t, o, a_t) \\ &= -Q(s_t, o, a_t) \\ &\quad + (1-\lambda)\Big[r(s_t, a_t) + \gamma\rho_{t+1}Q(s_{t+1}, o_{t+1}, a_{t+1})\Big] \\ &\quad + (1-\lambda)\lambda\Big[r(s_t, a_t) + \gamma\rho_{t+1}r(s_{t+1}, a_{t+1}) \\ &\qquad + \gamma^2\rho_{t+1}\rho_{t+2}Q(s_{t+2}, o_{t+2}, a_{t+2})\Big] \\ &\quad + (1-\lambda)\lambda^2\Big[r(s_t, a_t) + \gamma\rho_{t+1}r(s_{t+1}, a_{t+1}) \\ &\qquad + \gamma^2\rho_{t+1}\rho_{t+2}r(s_{t+2}, a_{t+2}) \\ &\qquad + \gamma^3\rho_{t+1}\rho_{t+2}\rho_{t+3}Q(s_{t+3}, o_{t+3}, a_{t+3})\Big] \\ &\quad + \ldots\end{aligned} \tag{7}$$

The summation can be simplified by the usual trick through the geometric series. Rearranging and summing the reward terms at every time step, distributing $(1-\lambda)\lambda^i$ over the bootstrapped future reward (action-option value function) term inside the parenthesis, pulling the γ and λ out and re-arranging the terms to form TD errors:

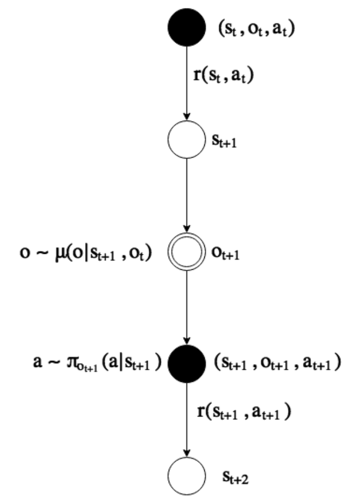

(s_t, o_t, a_t)
$r(s_t, a_t)$
s_{t+1}
$o \sim \mu(o|s_{t+1}, o_t)$
o_{t+1}
$a \sim \pi_{o_{t+1}}(a|s_{t+1})$
$(s_{t+1}, o_{t+1}, a_{t+1})$
$r(s_{t+1}, a_{t+1})$
s_{t+2}

Figure 1: Backup Diagram for Importance Sampling method for options. Filled circles represent nodes with state, option and action selected. Hollow circle represents state, and a double circle stands for option.

$$\begin{aligned}&\Delta Q(s_t, o, a_t) = \\ &\Big[r(s_t, a_t) + \gamma\rho_{t+1}Q(s_{t+1}, o_{t+1}, a_{t+1}) - Q(s_t, o, a_t)\Big] \\ &+ \gamma\lambda\rho_{t+1}\Big[r(s_{t+1}, a_{t+1}) + \gamma\rho_{t+2}Q(s_{t+2}, o_{t+2}, a_{t+2}) \\ &\quad - Q(s_{t+1}, o_{t+1}, a_{t+1})\Big] \\ &+ \gamma^2\lambda^2\rho_{t+1}\rho_{t+2}\Big[r(s_{t+2}, a_{t+2}) \\ &\quad + \gamma\rho_{t+3}Q(s_{t+3}, o_{t+3}, a_{t+3}) - Q(s_{t+2}, o_{t+2}, a_{t+2})\Big] \\ &+ \gamma^3\lambda^3\rho_{t+1}\rho_{t+2}\rho_{t+3}\Big[r(s_{t+3}, a_{t+3}) \\ &\quad + \gamma\rho_{t+4}Q(s_{t+4}, o_{t+4}, a_{t+4}) - Q(s_{t+3}, o_{t+3}, a_{t+3})\Big] \\ &+ \ldots \\ &= \sum_{i=t}^{\infty}\delta_i\prod_{k=t+1}^{i}\gamma\lambda\rho_k\end{aligned} \tag{8}$$

Eq (8) gives the generalized forward view using importance sampling technique when modifying an MDP with options. An important caveat in Eq (8) is the assumption that option executing at $time = t$ is o instead of actual option executed o_t. This affects the importance sampling corrections ρ_t, and the TD error δ_t at time t. Notation can be cleared using separate terms for importance sampling correction ζ and TD error ν in the case of the first update at $time = t$. The forward view in Eq (8) can be implemented incrementally using eligibility trace as shown in Algorithm 1.

Algorithm 1 Online, Eligibility Trace algorithm using Importance Sampling method for Options

```
$e, z$ : Eligibility traces initialized with all zeros
$\delta_t$ : TD error at $time = t$
$\nu_{o,t}$ : TD error assuming option $o$ executed at $time = t$
$\rho_t$: Importance Sampling correction at $time = t$
$\zeta_{o,t}$: Importance Sampling correction assuming option $o$ executed at $time = t$

1: Update eligibility traces $\forall(s, o, a)$:
      $e_t(s, o, a) = \lambda\gamma \left\{ \rho_t e_{t-1} + \zeta_{o,t} z_{t-1} \right\}$
      $z_t(s, o, a) = \begin{cases} 1 & if(s, a) = (s_t, a_t) \\ 0 & otherwise \end{cases}$
   where $\lambda \in [0, 1]$ is the trace decay factor.

2: Calculate TD errors: $\delta_t, \nu_{o,t} \forall(s, o, a)$ triples as:
      $\delta_t = r(s_t, a_t) + \gamma\rho_{t+1} Q(s_{t+1}, o_{t+1}, a_{t+1}) - Q(s_t, o_t, a_t)$
      $\nu_{o,t} = r(s_t, a_t) + \gamma\zeta_{o,t+1} Q(s_{t+1}, o_{t+1}, a_{t+1}) - Q(s_t, o, a_t)$

3: Apply updates to values of (s,o,a) triples:
      $Q_{t+1}(s, o, a) \longleftarrow Q_t(s, o, a) + \alpha\delta_t e_t(s, o, a)$
                    $+ \alpha\nu_{o,t} z_t(s, o, a) \quad \forall(s, o, a)$
```

3.2 Tree Backup

Previously explained Importance Sampling method has two shortcomings - first, it requires the behavior policy to be known, and second, if the target policy is too different from executing behavior policy, importance sampling ratios can be very large and may introduce high variance in the estimates. The motivation behind the Tree backup algorithm, as proposed in [26], is to overcome these two limitations.

Main idea behind tree backup method is to use expectation across all the branches of the backup diagram instead of sampling a branch as in Per Decision Importance Sampling algorithm. At each step along the trajectory, there are several possible choices of actions according to target policy. Tree backup algorithm forms target values combining value estimates of actions not taken and the new value estimate of the action taken, each weighted by corresponding action's probability under target policy. The only constraint on the behavior policy is to assign non-zero probability to all possible actions in a given state.

In the following derivation, the idea of tree backup is extended to the case of options. According to the backup diagram for Option Tree Backup in Fig 2, the target value for an update to option-action values is formed using expectation of updated future returns across all the consistent option branches and sub-branches for actions.

Assuming a trajectory over states, options, actions and rewards as in previous subsection, updates $\forall o \in O_{s_t}$ consistent with action a_t can be written using experience generated during the execution of option o_t at time step t. Update for an option $o \neq o_t$ can be formed assuming: action a_t was selected according to $\pi_o(\cdot|s_t)$, environment

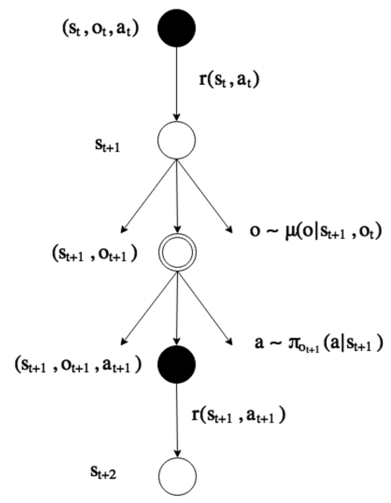

(s_t, o_t, a_t)
$r(s_t, a_t)$
s_{t+1}
(s_{t+1}, o_{t+1})
$o \sim \mu(o|s_{t+1}, o_t)$
$(s_{t+1}, o_{t+1}, a_{t+1})$
$a \sim \pi_{o_{t+1}}(a|s_{t+1})$
$r(s_{t+1}, a_{t+1})$
s_{t+2}

Figure 2: Backup Diagram for Option Tree Backup. Filled circles represent nodes with state, option and action selected. Hollow circle represents state, and a double circle stands for option.

transitions to state s_{t+1}, then option o_{t+1} is used to select action a_{t+1} with probability $\pi_{o_{t+1}}(\cdot|s_{t+1})$ and the process continues so on. μ, as defined in Eq (1), gives the distribution over all the options available in state s_{t+1} taking into consideration termination probability of previously running option. 1-step return for an option o using tree backup would be -

$$\begin{aligned} G_t^1 &= r(s_t, a_t) \\ &\quad + \gamma \sum_{o' \neq o_{t+1}} \mu(o'|s_{t+1}, o) \sum_a \pi_{o'}(a|s_{t+1}) Q(s_{t+1}, o', a) \\ &\quad + \gamma\mu(o_{t+1}|s_{t+1}, o) \sum_a \pi_{o_{t+1}}(a|s_{t+1}) Q(s_{t+1}, o_{t+1}, a) \end{aligned} \tag{9}$$

In Eq (9), the second term is the expected return from all non-realized options and the third term is the new estimate of expected return from the trajectory of the realized option. Proceeding in a similar way, the 2-step return with the additional step at $t + 2$ can be written as:

$$\begin{aligned} G_t^2 &= r(s_t, a_t) \\ &\quad + \gamma \sum_{o' \neq o_{t+1}} \mu(o'|s_{t+1}, o) \sum_a \pi_{o'}(a|s_{t+1}) Q(s_{t+1}, o', a) \\ &\quad + \gamma\mu(o_{t+1}|s_{t+1}, o) \sum_{a \neq a_{t+1}} \pi_{o_{t+1}}(a|s_{t+1}) Q(s_{t+1}, o_{t+1}, a) \\ &\quad + \gamma\mu(o_{t+1}|s_{t+1}, o)\pi_{o_{t+1}}(a_{t+1}|s_{t+1}) r(s_{t+1}, a_{t+1}) \\ &\quad + \gamma^2\mu(o_{t+1}|s_{t+1}, o)\pi_{o_{t+1}}(a_{t+1}|s_{t+1}) \\ &\qquad \times \sum_{o'' \neq o_{t+2}} \mu(o''|s_{t+2}, o_{t+1}) \sum_a \pi_{o''}(a|s_{t+2}) Q(s_{t+2}, o'', a) \\ &\quad + \gamma^2\mu(o_{t+1}|s_{t+1}, o)\pi_{o_{t+1}}(a_{t+1}|s_{t+1}) \\ &\qquad \times \mu(o_{t+2}|s_{t+2}, o_{t+1}) \sum_a \pi_{o_{t+2}}(a|s_{t+2}) Q(s_{t+2}, o_{t+2}, a) \end{aligned} \tag{10}$$

Further steps can be derived in a similar fashion. TD error using the G_t^λ $\forall o \in \mathcal{O}_{s_t}$ can be computed as:

$$\begin{aligned}
\Delta Q(s_t, o, a_t) &= G_t^\lambda - Q(s_t, o, a_t) \\
&= -Q(s_t, o, a_t) \\
&\quad + (1-\lambda)\Big\{ r(s_t, a_t) \\
&\qquad + \gamma \sum_{o' \neq o_{t+1}} \mu(o'|s_{t+1}, o) \sum_a \pi_{o'}(a|s_{t+1}) Q(s_{t+1}, o', a) \\
&\qquad + \gamma \mu(o_{t+1}|s_{t+1}, o) \sum_a \pi_{o_{t+1}}(a|s_{t+1}) Q(s_{t+1}, o_{t+1}, a) \Big\} \\
&\quad + (1-\lambda)\lambda \Big\{ r(s_t, a_t) \\
&\qquad + \gamma \sum_{o' \neq o_{t+1}} \mu(o'|s_{t+1}, o) \sum_a \pi_{o'}(a|s_{t+1}) Q(s_{t+1}, o', a) \\
&\qquad + \gamma \mu(o_{t+1}|s_{t+1}, o) \sum_{a \neq a_{t+1}} \pi_{o_{t+1}}(a|s_{t+1}) Q(s_{t+1}, o_{t+1}, a) \\
&\qquad + \gamma \mu(o_{t+1}|s_{t+1}, o) \pi_{o_{t+1}}(a_{t+1}|s_{t+1}) \Big(r(s_{t+1}, a_{t+1}) \\
&\qquad\quad + \gamma \sum_{o'' \neq o_{t+2}} \mu(o''|s_{t+2}, o_{t+1}) \sum_a \pi_{o''}(a|s_{t+2}) Q(s_{t+2}, o'', a) \\
&\qquad\quad + \gamma \mu(o_{t+2}|s_{t+2}, o_{t+1}) \sum_a \pi_{o_{t+2}}(a|s_{t+2}) Q(s_{t+2}, o_{t+2}, a) \Big) \Big\} \\
&\quad + \ldots
\end{aligned} \tag{11}$$

The expression in Eq (11) can again be simplified by the usual geometric series trick. Rearranging and summing the reward terms at every time step, distributing $(1-\lambda)\lambda^i$ over the bootstrapped future reward (action-option value function) term from the realized option trajectory, pulling the γ, λ and other probability terms out, and re-arranging the terms to form TD errors:

$$\begin{aligned}
\Delta Q(s_t, o, a_t) &= G_t^\lambda - Q(s_t, o, a_t) \\
&= \Big\{ r(s_t, a_t) + \gamma \sum_{o'} \mu(o'|s_{t+1}, o) \sum_a \pi_{o'}(a|s_{t+1}) Q(s_{t+1}, o', a) \\
&\qquad - Q(s_t, o, a_t) \Big\} \\
&\quad + \lambda\gamma\mu(o_{t+1}|s_{t+1}, o)\pi_{o_{t+1}}(a_{t+1}|s_{t+1}) \Big\{ r(s_{t+1}, a_{t+1}) \\
&\qquad + \gamma \sum_{o''} \mu(o''|s_{t+2}, o_{t+1}) \sum_a \pi_{o''}(a|s_{t+2}) Q(s_{t+2}, o'', a) \\
&\qquad - Q(s_{t+1}, o_{t+1}, a_{t+1}) \Big\} \\
&\quad + \ldots \\
&= \sum_{i=t}^{\infty} \delta_i \prod_{k=t+1}^{i} \lambda\gamma\mu(o_k|s_k, o_{k-1})\pi_{o_k}(a_k|s_k)
\end{aligned} \tag{12}$$

Eq (12) gives the generalized forward view using tree-backup targets in a MDP with options. As with Importance Sampling method, in the derivation of Eq (12) we assume that option executing at $time = t$ is o instead of actual executed option o_t. This in turn affects the TD errors δ_t at time t. Using consistent notation, TD error at first time step update assuming option o at $time = t$ is represented by $v_{o,t}$. Algorithm 2 gives a simple incremental implementation using eligibility trace for the Tree Backup method.

Algorithm 2 Online, Eligibility Trace version of Tree Backup for Options

```
    $e, z$ : Eligibility traces initialized with all zeros
    $\delta_t$ : TD error at $time = t$
    $v_{o,t}$ : TD error assuming option $o$ executed at $time = t$
1: Update eligibility traces $\forall(s, o, a)$:
       $e_t(s,o,a) = \lambda\gamma\{\mu(o_t|s_t, o_{t-1})\pi_{o_t}(a_t|s_t)e_{t-1} + \mu(o_t|s_t, o)\pi_{o_t}(a_t|s_t)z_{t-1}\}$
       $z_t(s,o,a) = \begin{cases} 1 & if(s,a) = (s_t, a_t) \\ 0 & otherwise \end{cases}$
   where $\lambda \in [0, 1]$ is the trace decay factor.
2: Calculate TD errors: $\delta_t, v_{o,t} \forall(s, o, a)$ triples as:
       $\delta_t = r(s_t, a_t) + \gamma \sum_{o'} \mu(o'|s_{t+1}, o_t) \sum_{a'} \pi_{o'}(a'|s_{t+1}) Q(s_{t+1}, o', a') - Q(s_t, o_t, a_t)$
       $v_{o,t} = r(s_t, a_t) + \gamma \sum_{o'} \mu(o'|s_{t+1}, o) \sum_{a'} \pi_{o'}(a'|s_{t+1}) Q(s_{t+1}, o', a') - Q(s_t, o, a_t)$
3: Apply updates to values of (s,o,a) triples:
       $Q_{t+1}(s,o,a) \leftarrow Q_t(s,o,a) + \alpha\delta_t e_t(s,o,a) + \alpha v_{o,t} z_t(s,o,a) \quad \forall(s,o,a)$
```

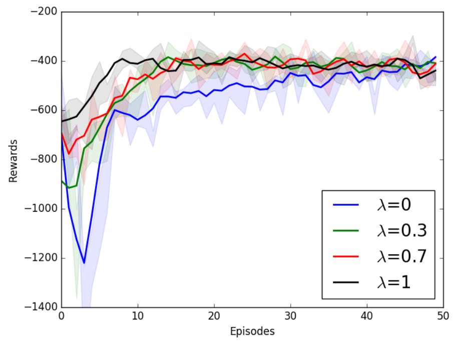
3.3 Retrace

Table 1 summarized the constraints of popular off-policy algorithms. Importance sampling method is susceptible to high variance in the estimates and requires the behavior to be known. Tree backup method improves on these limitations but suffers from the issue of premature cutting of traces blocking learning from future returns. Retrace(λ) method overcomes the shortcomings of the above two methods. It has low variance, can be used with any behavior policy and is efficient in its use of full returns.

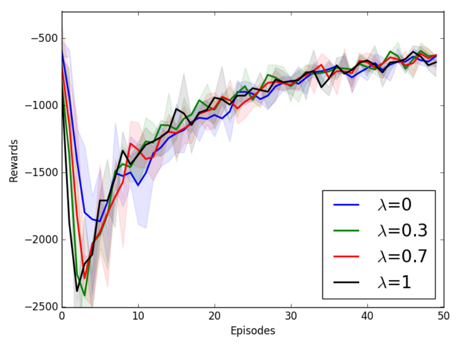
Derivation of eligibility traces using Retrace(λ) is exactly same as the derivation for Importance Sampling with the only caveat being different trace correction factor as seen in Table 1. In Algorithm 1, replacing correction factor (ρ) with $\min\left(1, \frac{\pi(a_s|x_s)}{b(a_s|x_s)}\right)$ gives a simple incremental implementation using eligibility trace for the Retrace(λ) method.

4 EXPERIMENTS

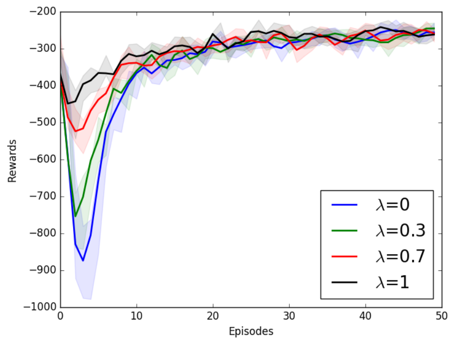
In our experiments, the option policies and termination conditions are assumed to be given ahead of time. Options are formed using different combinations of internal policies and termination conditions. There are two reasons for this. First, it allows us to look



(a) Importance Sampling



(b) Tree Backup



(c) Retrace

Figure 3: Performance of algorithms as a function of λ in random walk.

at the impact of the termination, which is a special construct for options. Second, it allows a mix of scenarios where the target goes from being very close to the behavior (same internal policy) to very different (different internal policies). The goal is to learn option values and the policy over options and assess the utility of eligibility traces by varying the trace coefficient λ.

4.1 Random Walk

We first consider random walk over MDP with 99 non-terminal states ($s_0 - s_{98}$) and one terminal state (s_{99}). An episode starts with the agent in state s_0 and terminates when it reaches s_{99}. In each non-terminal state, two deterministic actions are possible: to go *left* and to go *right*. The agent receives a reward of -1 for every time step except on reaching s_{99} when it receives a reward of 0. Discount factor γ is set to 0.9 for all the random walks.

The options are setup with the same internal policy- agent moves right with the probability of 0.7 and left otherwise. Options have different termination probabilities $\beta \in \{0.9, 0.7, 0.5, 0.3, 0.1\}$. In addition to these 5 temporally extended actions, agent also selects from 2 primitive actions: to move left and move right. Options are selected stochastically using an ϵ-greedy policy ($\epsilon = 0.15$) and are executed in call-and-return manner. Behavior is selected using the currently active option's internal policy. The learning rate for each curve was optimized using grid search and Table 3 in the appendix shows the parameter settings used in the experiments. The results for each λ value has been averaged over five independent trials.

Fig 3 shows the performance of the traces as a function of varying trace coefficient λ. Results for Importance sampling and Retrace methods clearly show that the performance steadily gets better as trace coefficient λ is increases, with the best performance being observed for $\lambda = 1$. On the other hand, difference in the performance of eligibility trace with tree backup with varying trace coefficient λ is not very evident. This behavior can again be explained by the premature cutting of trace with tree backup method which prevents use of full returns. Additional multiplication of hierarchical distribution probabilities in the trace update makes this worse in case of Tree backup option trace. Table 2 proves this hypothesis showing that only very few elements in the trace have significant eligibility values for a meaningful update.

Table 2: Premature cutting of eligibility traces for options with tree backup. Columns represent portion of elements in the eligibility trace with value more than the specified column value for a particular λ row value.

λ **values**	10^{-7}	10^{-5}	10^{-3}	10^{-1}
0	0%	0%	0%	0%
0.3	0.106%	0.072%	0.035%	0.0006%
0.7	0.147%	0.102%	0.055%	0.0015%
1	0.153%	0.109%	0.059%	0.0016%

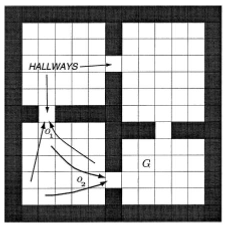


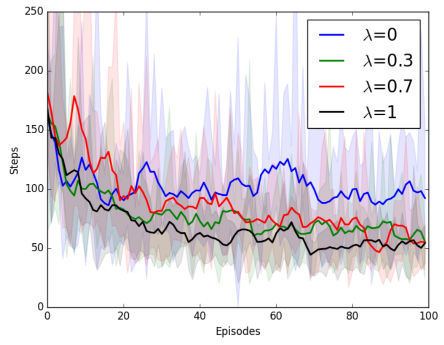
Figure 4: Four rooms domain

4.2 Four-rooms Domain

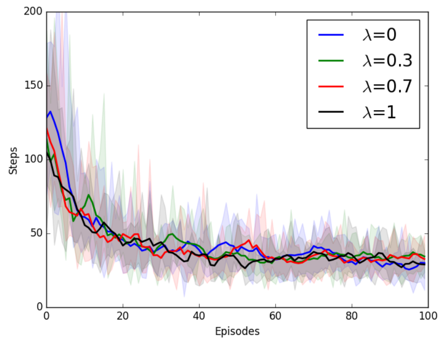
Next we consider the navigation task in four-rooms domain [32] as seen in Figure 4. The agent can move up, down, left or right with a stochastic effect. With probability 2/3, the actions cause the agent to move one cell in the corresponding direction, and with probability 1/3, the agent moves instead in one of the other three directions. Rewards are zero for all transitions except ones which lead to the goal state G. Discount factor γ is set to 0.9

In addition to the four primitive actions, there are four sets of options, each taking the agent to a different hallway. Each set of options has same internal policy but different termination probabilities $\beta \in [0.9, 0.7, 0.5, 0.3, 0.1]$. Options are selected in call-and-return manner from an ϵ-greedy policy ($\epsilon = 0.15$). Current option determines the behavior of the agent. Grid search was done to optimize learning rates for different settings as seen in Table 4.

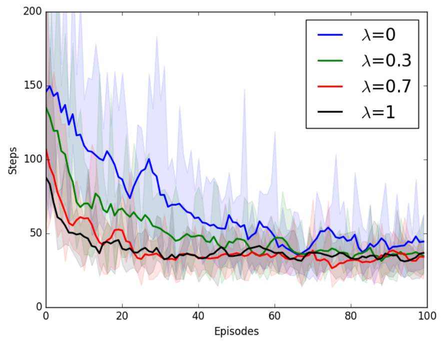
Fig 5 shows the performance of the three methods as a function of varying λ and Fig 6 compares the algorithms based on number of steps to episode termination.



(a) Importance Sampling



(b) Tree Backup



(c) Retrace

Figure 5: Performance of algorithms as a function of λ in four rooms domain. Each curve is averaged over 25 runs.

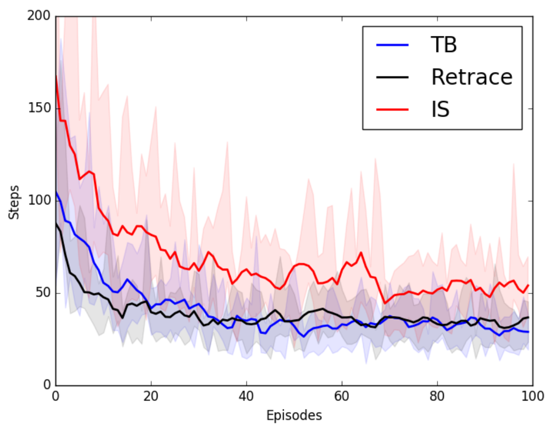


Figure 6: Comparison of IS, TB and Retrace algorithms with best λs in four rooms domain. Each curve is averaged over 25 runs.

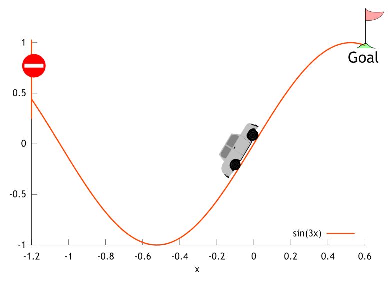


Figure 7: Mountain Car environment

As can be seen in Fig 5, algorithms perform better with increasing trace coefficient λ values. The number of steps to episode termination reduces faster as λ value increases for Importance sampling and Retrace methods, with minor effect on Tree backup method. Traces in the case of Tree backup option method are cut aggressively because of the hierarchical and internal policies. This leads to the method to perform similar with different λ values, as seen in results here. Importance sampling method has higher variance in its estimates of option values and takes more time to converge compared to Tree backup and Retrace methods as seen in Fig 6.

4.3 Mountain Car

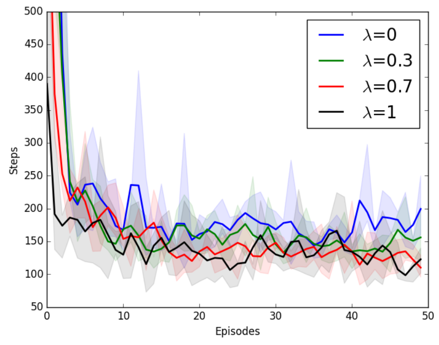
Finally we consider linear function approximations with the eligibility trace algorithms for options. In Mountain-Car problem [22, 31], as seen in Figure 7, an underpowered car has to climb a one-dimensional hill to reach a target. Agent can only reach the target position by going back and forth in the valley and building momentum. Three discrete actions: push left, no push and push right are available to the agent to move and build momentum. Agent gets a reward of -1 for every time step except when it transitions to the target and episode ends. The two dimensional state space comprises continuous position and velocity of the car. Discount factor γ is set to 0.9

Options have been formulated with internal policies to reach different positions: -0.2, 0, 0.2 and 0.4. There are multiple options with their termination probabilities $\beta \in [0.9, 0.7, 0.5, 0.3, 0.1]$ for a internal policy. In addition to these 20 options, agent can also choose from 3 primitive actions. Options are again selected in an ϵ-greedy fashion on their values with a usual call-and-return execution model ($\epsilon = 0.15$). In our experiments, state space is discretized using tile coding [32]. 9 dimensional features (1 for position of the car, and 8 for the joint space of position and velocity) is used to estimate values of option-action pairs. Table 5, in the appendix, shows the optimized parameter value for learning rates used in the experiment.

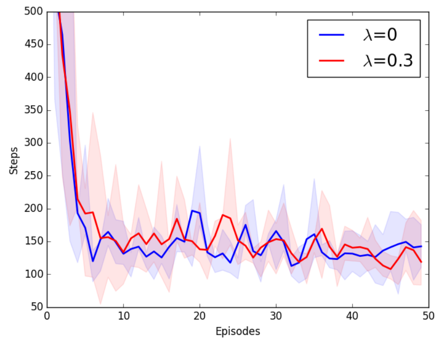
Fig 8 shows the performance of the traces as a function of varying trace coefficient λ. Performance steadily gets better as λ increases for Importance sampling and Retrace methods. Tree Backup method converges for $\lambda = 0$ and $\lambda = 0.3$. Higher λ values don't converge. Recently, Tree Backup and Retrace(λ) algorithms have been shown to be theoretically unstable with linear function approximation [36].

5 CONCLUSION

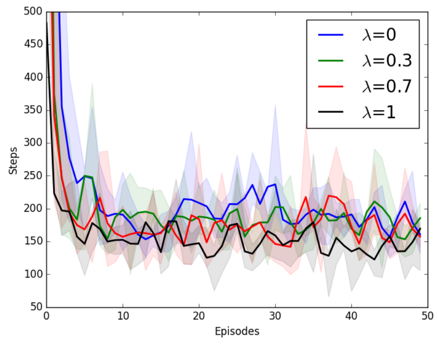
In this work, we highlight the benefits of using eligibility traces with options and searched for good ways of doing multi-step intra-option learning updates. Furthermore, three algorithms based on



(a) Importance Sampling



(b) Tree Backup



(c) Retrace

Figure 8: Performance of algorithms as a function of λ in Mountain-Car environment. Each curve is averaged over 5 runs.

off-policy learning methods were proposed for using eligibility traces with options. Even though the derivations are shown only for learning option values in this paper, same updates can be applied to learn option reward and transition models for planning with options.

The binary decision to either continue or terminate at option decision boundaries makes it mathematically challenging to formulate multi-step updates. To overcome this challenge, we treated such decisions probabilistically in the analysis of multi-step intra-option learning updates. This provides an effective way of handling option decision boundaries mathematically.

The empirical results presented in this work clearly highlight the utility of eligibility traces in hierarchical reinforcement learning. Results show that multi-step intra-option update significantly speeds up the learning process. The problem of high variance is common in importance sampling methods, and the importance sampling option traces from Algorithm 1 is also susceptible to it, though to a lesser degree as seen empirically. The known drawback with tree-backup of premature cutting of traces seems to be amplified a bit with Tree backup option trace. This is justifiable given the additional hierarchical distribution term in the updates. Retrace(λ) algorithm provides a low variance off-policy learning method which does not cut eligibility traces excessively and can be used with any target and behavior policies. Retrace Option trace performs exemplary with lower variance in its estimations and better convergence values which is supported empirically in our results.

Our experiments with the mountain car environment successfully demonstrate that the proposed multi-step intra-option learning methods using eligibility traces can be used with linear function approximation.

A future extension to this work includes incorporating eligibility traces to the option-critic architecture [2]. The current work can be used directly in the option evaluation step. Classical Tree Backup and Retrace(λ) algorithms have been shown to be theoretically unstable with linear function approximation [36]. Another direction for extension of our work is to consider adapting to the convergent Tree Backup and Retrace algorithms as proposed in [36].

APPENDIX

Parameter settings used in the experiments.

Table 3: Optimized learning rate for Random Walk from grid search

Learning	λ			
Rate (α)	0	0.3	0.7	1
`Importance Sampling`	0.05	0.05	0.05	0.05
`Tree Backup`	0.05	0.05	0.05	0.05
`Retrace(`λ`)`	0.064	0.064	0.064	0.064

Table 4: Learning rates used for Four-rooms experiments.

Learning	λ			
Rate (α)	0	0.3	0.7	1
`Importance Sampling`	0.002	0.005	0.002	0.002
`Tree Backup`	0.4	0.4	0.4	0.2
`Retrace(`λ`)`	0.4	0.4	0.4	0.4

Table 5: Learning rates used in mountain car environment.

Learning	λ			
Rate (α)	0	0.3	0.7	1
`Importance Sampling`	0.0008	0.0008	0.0008	0.0005
`Tree Backup`	0.0008	0.0008		
`Retrace(`λ`)`	0.0008	0.0008	0.0008	0.0008

REFERENCES

[1] Elizeth G Araujo and Roderic A Grupen. 1996. Learning control composition in a complex environment. In *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. 333–342.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The Option-Critic Architecture.

[3] Steven J Bradtke and Michael O Duff. 1995. Reinforcement learning methods for continuous-time Markov decision problems. In *Advances in neural information processing systems*. 393–400.

[4] Ronen I. Brafman and Moshe Tennenholtz. 1997. Modeling agents as qualitative decision makers. *Artificial Intelligence* 94, 1 (1997), 217 – 268. https://doi.org/10.1016/S0004-3702(97)00024-6 Economic Principles of Multi-Agent Systems.

[5] Roger W Brockett. 1993. Hybrid models for motion control systems. *Progress in Systems and Control Theory* 14 (1993), 29–29.

[6] Marco Colombetti, Marco Dorigo, and Giuseppe Borghi. 1996. Behavior analysis and training-a methodology for behavior engineering. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26, 3 (1996), 365–380.

[7] Christian Daniel, Herke Van Hoof, Jan Peters, and Gerhard Neumann. 2016. Probabilistic inference for determining options in reinforcement learning. *Machine Learning* 104, 2-3 (2016), 337–357.

[8] Gerald F. DeJong. 1994. Learning to plan in continuous domains. *Artificial Intelligence* 65, 1 (1994), 71 – 141. https://doi.org/10.1016/0004-3702(94)90038-8

[9] Gary L. Drescher. 1991. *Made-up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge, MA, USA.

[10] Richard E Fikes, Peter E Hart, and Nils J Nilsson. 1972. Learning and executing generalized robot plans. *Artificial Intelligence* 3 (1972), 251 – 288. https://doi.org/10.1016/0004-3702(72)90051-3

[11] Anna Harutyunyan, Marc G Bellemare, Tom Stepleton, and Rémi Munos. 2016. Q (\lambda) with Off-Policy Corrections. In *International Conference on Algorithmic Learning Theory*. Springer, 305–320.

[12] Glenn A Iba. 1989. A heuristic approach to the discovery of macro-operators. *Machine Learning* 3, 4 (1989), 285–317.

[13] Michael J Kearns and Satinder P Singh. 2000. Bias-Variance Error Bounds for Temporal Difference Updates.. In *COLT*. 142–147.

[14] Richard E Korf. 1983. *Learning to solve problems by searching for macro-operators*. Technical Report. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.

[15] BJ Kuipers. 1979. Commonsensense knowledge of space: Learning from experience. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence. Stanford, CA: Stanford Computer Science Department*.

[16] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*. 3675–3683.

[17] John E Laird, Paul S Rosenbloom, and Allen Newell. 1986. Chunking in Soar: The anatomy of a general learning mechanism. *Machine learning* 1, 1 (1986), 11–46.

[18] Marlos C Machado, Marc G Bellemare, and Michael Bowling. 2017. A Laplacian Framework for Option Discovery in Reinforcement Learning. *arXiv preprint arXiv:1703.00956* (2017).

[19] Pattie Maes and Rodney A Brooks. 1990. Learning to Coordinate Behaviors.. In *AAAI*, Vol. 90. 796–802.

[20] Daniel J Mankowitz, Timothy A Mann, and Shie Mannor. 2016. Adaptive Skills Adaptive Partitions (ASAP). In *Advances in Neural Information Processing Systems*. 1588–1596.

[21] S. Minton. 1988. *Learning Search Control Knowledge: An Explanation-Based Approach*. Kluwer Academic, Dordrecht.

[22] Andrew William Moore. 1990. Efficient memory-based learning for robot control. (1990).

[23] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. 2016. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*. 1054–1062.

[24] Allen Newell, Herbert Alexander Simon, et al. 1972. *Human problem solving*. Vol. 104. Prentice-Hall Englewood Cliffs, NJ.

[25] Ronald Parr and Stuart Russell. 1998. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems* (1998), 1043–1049.

[26] Doina Precup, Richard S Sutton, and Satinder Singh. 2000. Eligibility traces for off-policy policy evaluation. In *In Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00)*. Citeseer.

[27] Martin L Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[28] Reuven Y. Rubinstein. 1981. *Simulation and the Monte Carlo Method* (1st ed.). John Wiley & Sons, Inc., New York, NY, USA.

[29] Earl D Sacerdoti. 1974. Planning in a hierarchy of abstraction spaces. *Artificial intelligence* 5, 2 (1974), 115–135.

[30] AC Cem Say and Selahattin Kuru. 1996. Qualitative system identification: deriving structure from behavior. *Artificial Intelligence* 83, 1 (1996), 75–141.

[31] Satinder P Singh and Richard S Sutton. 1996. Reinforcement learning with replacing eligibility traces. *Recent Advances in Reinforcement Learning* (1996), 123–158.

[32] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[33] Richard S. Sutton and Doina Precup. 1998. Intra-option learning about temporally abstract actions. In *In Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufman, 556–564.

[34] Richard S. Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112, 1 (1999), 181 – 211. https://doi.org/10.1016/S0004-3702(99)00052-1

[35] Milind Tambe, Allen Newell, and Paul S. Rosenbloom. 1990. The problem of expensive chunks and its solution by restricting expressiveness. *Machine Learning* 5, 3 (01 Aug 1990), 299–348. https://doi.org/10.1007/BF00117107

[36] Ahmed Touati, Pierre-Luc Bacon, Doina Precup, and Pascal Vincent. 2017. Convergent Tree-Backup and Retrace with Function Approximation. *arXiv preprint arXiv:1705.09322* (2017).