The Termination Critic

Anna Harutyunyan
DeepMind

Will Dabney
DeepMind

Diana Borsa
DeepMind

Nicolas Heess
DeepMind

Rémi Munos
DeepMind

Doina Precup
DeepMind

Abstract

In this work, we consider the problem of autonomously discovering behavioral abstractions, or options, for reinforcement learning agents. We propose an algorithm that focuses on the *termination condition*, as opposed to – as is common – the policy. The termination condition is usually trained to optimize a control objective: an option ought to terminate if another has better value. We offer a different, information-theoretic perspective, and propose that terminations should focus instead on the *compressibility* of the option's encoding – arguably a key reason for using abstractions. To achieve this algorithmically, we leverage the classical options framework, and learn the option transition model as a "critic" for the termination condition. Using this model, we derive gradients that optimize the desired criteria. We show that the resulting options are non-trivial, intuitively meaningful, and useful for learning and planning.

1 Introduction

Autonomous discovery of meaningful behavioral abstractions in reinforcement learning has proven to be surprisingly elusive. One part of the difficulty perhaps is the fundamental question of *why* such abstractions are needed, or useful. Indeed, it is often the case that a primitive action policy is sufficient, and the overhead of discovery outweighs the potential speedup advantages. In this work, we adopt the view that abstractions are primarily useful due to their ability to *compress* information, which yields speedups in learning and planning. For example, it has been argued in the neuroscience literature that behavior hierarchies are optimal if they induce plans of minimum description length across a set of tasks [Solway et al., 2014].

Despite significant interest in the discovery problem, only in recent years have there been algorithms that tackle it with minimal supervision. An important example is the option-critic [Bacon et al., 2017], which focuses on options [Sutton et al., 1999] as the formal framework of temporal abstraction and learns both of the key components of an option, the policy and the termination, end-to-end. Unfortunately, in later stages of training, the options tend to collapse to single-action primitives. This is in part due to using the advantage function as a training objective of the termination condition: an option ought to terminate if another option has better value. This, however, occurs often throughout learning, and can be simply due to noise in value estimation. The follow-up work on option-critic *with deliberation cost* [Harb et al., 2018] addresses this option collapse by modifying the termination objective to additionally penalize option termination, but it is highly sensitive to the associated cost parameter.

In this work, we take the idea of modifying the termination objective to the extreme, and propose for it to be completely independent of the task reward. Taking the compression perspective, we suggest that this objective should be information-theoretic, and should capture the intuition that it would be useful to have "simple" option encodings that focus termination on a small set of states. We show that such an objective correlates with the planning performance of options for a set of goal-directed tasks.

Our key technical contribution is the manner in which the objective is optimized. We derive a result that relates the gradient of the option transition model to the gradient of the termination condition, allowing one to express objectives in terms of the option model, and

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

optimize them directly via the termination condition. The model hence acts as a "critic", in the sense that it measures the quality of the termination condition in the context of the objective, analogous to how the value function measures the quality of the policy in the context of reward. Using this result, we obtain a novel policy-gradient-style algorithm which learns terminations to optimize our objective, and policies to optimize the reward as usual. The separation of concerns is appealing, since it bypasses the need for sensitive trade-off parameters. We show that the resulting options are non-trivial, and useful for learning and planning.

The paper is organized as follows. After introducing relevant background, we present the termination gradient theorem, which relates the change in the option model to the change in terminations. We then formalize the proposed objective, express it via the option model, and use the termination gradient result to obtain the online actor-critic termination-critic (ACTC) algorithm. Finally, we empirically study the learning dynamics of our algorithm, the relationship of the proposed loss to planning performance, and analyze the resulting options qualitatively and quantitatvely.

2 Background and Notation

We assume the standard reinforcement learning (RL) setting [Sutton and Barto, 2017] of a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{X}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{X}$ is the set of states, $\mathcal{A}$ the set of discrete actions; $p : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$ is the transition model that specifies the environment dynamics, with $p(x'|x, a)$ denoting the probability of transitioning to state x' upon taking action a in x; $r : \mathcal{X} \times \mathcal{A} \to [-r_{\max}, r_{\max}]$ the reward function, and $\gamma \in [0, 1]$ the scalar discount factor. A policy is a probabilistic mapping from states to actions. For a policy π, let the matrix p^π denote the dynamics of the induced Markov chain: $p^\pi(x'|x) = \sum_{a\in\mathcal{A}} \pi(a|x)p(x'|x, a)$, and r^π the reward expected for each state under π: $r^\pi(x) = \sum_{a\in\mathcal{A}} \pi(a|x)r(x, a)$.

The goal of an RL agent is to find a policy π that produces the highest expected cumulative reward:

$$J(\pi) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t | x_0, \pi\right], \tag{1}$$

where $R_t = r(X_t, A_t)$ is the random variable corresponding to the reward received at time t, and x_0 is the initial state of the process. An *optimal* policy is one that maximizes J. A related instrumental quantity in RL is the action-value (or Q-) function, which measures J for a particular state-action pair:

$$q^\pi(x, a) = \mathbb{E}\left[\sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k} | X_t = x, A_t = a, \pi\right]. \tag{2}$$

The simplest way to optimize the objective (1) is directly by adjusting the policy parameters θ_π (assuming π is differentiable) [Williams, 1992]. The *policy gradient (PG)* theorem [Sutton et al., 2000] states that:

$$\nabla_{\theta_\pi} J(\pi) = \sum_x d^\pi(x) \sum_a \nabla_{\theta_\pi} \pi(x, a) q^\pi(x, a)$$

where d_π is the stationary distribution induced by π. Hence, samples of the form $\nabla_{\theta_\pi} \log \pi(x, a) q^\pi(x, a)$ produce an unbiased estimate of the gradient $\nabla_{\theta_\pi} J(\pi)$.

The final remaining question is how to estimate q^π. A standard answer relies on the idea of *temporal-difference (TD) learning*, which itself leverages sampling of the following recursive form of Eq. (2), known as the Bellman Equation [Bellman, 1957]:

$$q^\pi(x, a) = r(x, a) + \gamma \sum_{x'} p(x'|x, a) \sum_{a'} \pi(a'|x') q^\pi(x', a').$$

Iterating this equation from an arbitrary initial function q produces the correct q^π in expectation (e.g. [Puterman, 1994]).

2.1 The Options Framework

Options provide the standard formal framework for modeling temporal abstraction in RL [Sutton et al., 1999]. An option o is a tuple $(\mathcal{I}^o, \beta^o, \pi^o)$. Here, $\mathcal{I}^o \subseteq \mathcal{X}$ is the initiation set, from which option o may start (as in other recent work, we take $\mathcal{I}^o = \mathcal{X}$ for simplicity), $\beta^o : \mathcal{X} \to [0, 1]$ is the termination condition, with $\beta^o(x)$ denoting the probability of option o terminating in state x; and π^o is the internal policy of option o. As is common, we assume that options eventually terminate:

Assumption 1. *For all $o \in \mathcal{O}$ and $x \in \mathcal{X}$, $\exists x_f$ that is reachable by π^o from x, s.t. $\beta^o(x_f) > 0$.*

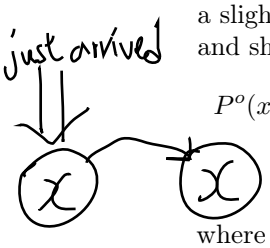
Analogously to the one-step MDP reward and transition models r and p, options induce *semi*-MDP [Puterman, 1994] reward and transition models:

$$P^o(x_f|x_s) = \gamma\beta^o(x_f)p^{\pi^o}(x_f|x_s) + \gamma \sum_x p^{\pi^o}(x|x_s)(1 - \beta^o(x))P^o(x_f|x),$$

$$R^o(x_s) = r^{\pi^o}(x_s) + \gamma \sum_x p^{\pi^o}(x|x_s)(1 - \beta^o(x))R^o(x).$$

That is: the option transition model $P^o(\cdot|x_s)$ outputs a sub-probability distribution over states, which, for

each x_f captures the discounted probability of reaching x_f from x_s (in any number of steps) *and* terminating there. In this work, unless otherwise stated, we will use a slightly different formulation, which is undiscounted and shifted backwards by one step:



$$P^o(x_f|x_s) = \beta^o(x_f)\mathbb{I}_{x_f=x_s} + (1-\beta^o(x_s))\sum_x p^{\pi^o}(x|x_s)P^o(x_f|x), \quad (3)$$

where $\mathbb{I}$ denotes the indicator function. The lack of discounting is convenient because it leads to P^o being a probability (rather than sub-probability) distribution over x_f so long as Assumption 1 holds.[1] The backwards time shifting replaces the first $\beta^o(x_f)p^{\pi^o}(x_f|x_s)$ term with $\beta^o(x_f)\mathbb{I}_{x_f=x_s} = \beta^o(x_s)\mathbb{I}_{x_f=x_s}$, which conveniently considers β^o of the same state, and will be important for our derivations.

We will use μ to denote the policy over options, and define the option-level Q-function as in [Sutton et al., 1999].

3 The Termination Gradient Theorem

The starting point for our work is the idea of optimizing the option's termination condition independently and with respect to a different objective than the policy. How can one easily formulate such objectives? We propose to leverage the relationship of the termination condition to the option model P^o, which is a distribution over the final states of the option, a quantity that is relevant to many natural objectives. The classical idea of terminating in "bottleneck" states is an example of such an objective.

To this end, we first note that the definition (3) of P^o has a formal similarity to a Bellman equation, in which P^o is akin to a value function and β^o influences both the immediate reward *and* the discount. Hence, we can express the gradient of P^o through the gradient of β^o. This will allow us to express high-level objectives through P^o and optimize them through β^o. The following general result formalizes this relationship, while the next section uses it for a particular objective. The theorem concerns a single option o, and we write θ_β to denote the parameters of β^o.

Theorem 1. *Let $\beta^o(x) \in (0,1), \forall x \in \mathcal{X}, o \in \mathcal{O}$ be parameterized by θ_β. The gradient of the option model w.r.t. θ_β is:*

$$\nabla_{\theta_\beta} P^o(x_f|x_s) = \sum_x P^o(x|x_s)\nabla_{\theta_\beta}\log\beta^o(x)r^o_{x_f}(x),$$

where the "reward" term is given by:

$$r^o_{x_f}(x) \stackrel{def}{=} \frac{\mathbb{I}_{x_f=x} - P^o(x_f|x)}{1-\beta^o(x)}$$

The proof is given in appendix. In the following, we will drop the superscript and simply write θ_β. Note that in the particular (common) case when β is parameterized with a sigmoid function, the expression takes the following simple shape:

$$\nabla_{\theta_\beta} P^o(x_f|x_s) = \sum_x P^o(x|x_s)\nabla_{\theta_\beta}\ell_{\beta^o}(x)(\mathbb{I}_{x_f=x} - P^o(x_f|x)),$$

where $\ell_{\beta^o}(x)$ denotes the logit of $\beta^o(x)$. The pseudo-reward $\mathbb{I}_{x_f=x} - P^o(x_f|x)$ has an intuitive interpretation. In general, it is always positive at x_f (and so $\beta^o(x_f)$ should always increase to maximize $P^o(x_f|x_s)$), and negative at all other states $x \neq x_f$ (and so $\beta^o(x)$ should always decrease to maximize $P^o(x_f|x_s)$). The amount of the change depends on the dynamics $P^o(x_f|x)$.

- In terminating states x_f, if $P^o(x_f|x_f)$ – the likelihood of terminating in x_f immediately *or* upon a later return – is low, then the change in $\beta^o(x_f)$ is high: an immediate termination needs to occur in order for $P^o(x_f|x_s)$ to be high. If $P^o(x_f|x_f)$ is high, then $P^o(x_f|x_s)$ may be high without β^o needing to be high.[2]
- In non-terminating states $x \neq x_f$, the higher $P^o(x_f|x)$, or the likelier it is for the desired terminating state x_f to be reached from x, the more the termination probability at x is reduced, to ensure that this happens. If x_f is not reachable from x no change occurs at x to maximize $P^o(x_f|x_s)$.

The theorem can be used as a general tool to optimize any differentiable objective of P^o. In the next section we propose and justify one such objective, and derive the complete algorithm.

4 The Termination Critic

We now propose the specific idea for the actor-critic termination critic (ACTC) algorithm. We first formulate our objective, then use Theorem 1 to derive the gradient of this objective as a function of the gradient of β^o. Finally, we formulate the online algorithm that estimates all of the necessary quantities from samples.

4.1 Predictability Objective

We postulate that desirable options are "targeted" and have small terminating regions. This can be expressed

[1] In particular, if we sum over x_f we obtain the Poisson binomial (or sequential independent Bernoulli) probability of one success out of infinite trials, which is equal to 1.

[2] E.g. if there is a single terminating state that is guaranteed to be reached, any $\beta^o(x_f) > 0$ will do.

as having a low entropy distribution of final states. We hence propose to explicitly minimize this entropy as an objective:

$$J(P^o) = H(X_f|o) \tag{4}$$

where H denotes entropy, and X_f is the random variable denoting a terminating state. We call this objective *predictability*, as it measures how predictable an option's outcome (final state) is. Note that the entropy is marginalized over all starting states. The marginalization allows for *consistency* – without it, the objective can be satisfied for each starting state individually without requiring the terminating states to be the same. We will later show that this objective indeed correlates with planning performance (Section 6.3).

In the exact, discrete setting the objective (4) is minimized when an option terminates at a single state x_f. Note that the this is the case irrespective of the choice of x_f. The resulting x_f will hence be determined by the learning dynamics of the particular algorithm. We will show later empirically that the gradient algorithm we derive is attracted to x_f-s that are most visited, as measured by P^o (Section 6.1).

The objective is reminiscent of other recent information-theoretic option-discovery approaches (e.g. [Gregor et al., 2016, Florensa et al., 2017, Hausman et al., 2018]), but is focused on the shape of each option in isolation. Similar to those works, one may wish to explicitly optimize for *diversity* between options as well, so as to encourage specialization. This can be attained for example by maximizing the complete mutual information $I(X_f|O) = H(X_f) - H(X_f|O)$ as the objective (rather than only its second term). In this work we choose to keep the objective minimalistic, but will observe some diversity occur under a set of tasks, due to the sampling procedure, so long as the policy over options is non-trivial.

The manner in which we propose to optimize this objective is novel and entirely different from existing work. We leverage the analytical gradient relationship derived in Theorem 1, and so instead of estimating the entropy term directly, we will express it through the option model, and estimate this option model. The following proposition expresses criteria (4) via the option model.

Proposition 1. *Let μ denote the policy over options, and let $d^\mu(\cdot|o)$ be option o's starting distribution induced by μ. Let $P^o_\mu(x_f) \overset{def}{=} \sum_{y_s} d^\mu(y_s|o)P^o(x_f|y_s)$. The criteria (4) can be expressed via the option model as follows:*

$$J(P^o) = -\mathbb{E}_{x_s}\Big[\sum_{x_f} P^o(x_f|x_s)\log P^o_\mu(x_f)\Big]$$

The proof is mainly notational and given in appendix.

This proposition implies that for a particular start state x_s, our global entropy loss can be interpreted as a cross-entropy loss between the distributions $P^o(\cdot|x_s)$ of final states from a particular state x_s and $P^o_\mu(\cdot)$ of final states from all start states.

A note on Assumption 1. Finally, we note that the entropy expression is only meaningful if P^o is a probability distribution, otherwise it is for example minimized by an all-zero P^o. In turn, P^o is a distribution if Assumption 1 holds, but if the option components are being learned, this may be tricky to uphold, and a trivial solution may be attractive. We will see empirically that the algorithm we derive does not collapse to this trivial solution. However, a general way of ensuring adherence to Assumption 1 remains an open problem.

4.2 The Predictability Gradient

We are now ready to express the gradient of the overall objective from Proposition 1 in terms of the termination parameters θ_β. This result is analogous to the policy gradient theorem, and similarly, for the gradient to be easy to sample, we need for a certain distribution to be independent of the termination condition.

Assumption 2. *The distribution $d^\mu(\cdot|o)$ over the starting states of an option o under policy μ is independent of its termination condition β^o.*

This is satisfied for example by any fixed policy μ. Of course, in general μ often depends on the value function over options q which in turn depends on β^o, but one may for example apply two-timescale optimization to make q appear quasi-fixed (e.g. [Borkar, 1997]). Our experiments show that even when not doing this explicitly, the online algorithm converges reliably. The following theorem derives the gradient of the objective (4) in terms of the gradient of β^o.

Theorem 2. *Let Assumption 2 hold. We have that:*

$$\nabla_{\theta_\beta} J(P^o) = -\sum_{x_s} d^\mu(x_s|o)\sum_x \frac{P^o(x|x_s)}{\beta^o(x)}\frac{\nabla_{\theta_\beta}\beta^o(x)}{1-\beta^o(x)}\times$$
$$\Bigg[\underbrace{\log P^o_\mu(x) - \sum_{x_f} P^o(x_f|x)\log P^o_\mu(x_f)}_{\text{reachability advantage } A^o_P(x)}$$
$$+\underbrace{1 - \sum_{x_f} P^o(x_f|x)\frac{P^o(x_f|x_s)P^o_\mu(x)}{P^o_\mu(x_f)P^o(x|x_s)}}_{\text{trajectory advantage } A^o_\tau(x|x_s)}\Bigg]$$

The proof is a fairly straightforward differentiation, and is given in appendix. The loss consists of two terms. The *termination advantage* measures how likely option

o is to reach and terminate in state x, as compared to other alternatives x_f. The *trajectory advantage* measures the desirability of state x in context of the starting state x_s – if x is a likely termination state *in general*, but not for the particular x_s, this term will account for it. Appendix D studies the effects of these two terms on learning dynamics in isolation.

Now, we would like to derive a sample-based algorithm that produces unbiased estimates of the derived gradient. Substituting the expression for the gradient of the logits, we can rewrite the overall expression as follows:

$$\nabla_{\theta_\beta} J(P^o) = -\mathbb{E}_{x_s}\mathbb{E}_x \nabla_{\theta_\beta} \ell_{\beta^o}(x)\Big(A^o_P(x) + A^o_\tau(x|x_s)\Big)$$

where the two expectations are w.r.t. the distributions written out in Theorem 2. We will sample these expectations from trajectories of the form $\tau = x_s, \dots, x, \dots, x_f$, and base our updates on the following corollary.

Corollary 1. *Consider a sample trajectory $\tau = x_s, \dots, x, \dots, x_f$, and let β^o be parameterized by a sigmoid funciton. For a state x, the following gradient is an unbiased estimate of $\nabla_{\theta_\beta} J(P^o)$:*

$$-\nabla_{\theta_\beta} \ell_{\beta^o}(x)\beta^o(x)\Big(\tilde{A}^o_P(x, x_f) + \tilde{A}^o_\tau(x, x_f|x_s)\Big), \quad (5)$$

$$\tilde{A}^o_P(x, x_f) = \log P^o_\mu(x) - \log P^o_\mu(x_f)$$

$$\tilde{A}^o_\tau(x, x_f|x_s) = 1 - \frac{P^o(x_f|x_s)P^o_\mu(x)}{P^o_\mu(x_f)P^o(x|x_s)}$$

where $\ell_{\beta^o}(x)$ denotes the logit of β^o at state x, and $\tilde{A}^o_P(x, x_f)$, $\tilde{A}^o_\tau(x, x_f|x_s)$ are samples from the corresponding advantages for a particular x_f.

The $\beta^o(x)$ factor in Eq. (5) is akin to an importance correction necessary due to not actually having terminated at x (and hence not having sampled $P^o(x|x_s)$). Note that the state x_f itself never gets updated directly, because the sampled advantages for it are zero (although the underlying state still may get updates when not sampled as final). The resulting magnitude of termination values at chosen final states hence depends on the initialization. To remove the dependence, we will deploy a baseline in the complete algorithm.

5 Algorithm

We now give our algorithm based on Corollary 1. In order to do so, we need to simultaneously estimate the transition model P^o. Because for a given final state, it is simply a value function, we can readily do this with temporal difference learning. Furthermore, we need to estimate $P^o_\mu(x_f)$, which we do simply as an empirical average of $P^o(x_f|x_s)$ over all experienced x_s. Finally, we deploy a per-option baseline B^o that tracks the empirical average update and gets subtracted from the updates at all states. The complete end-to-end algorithm is summarized in Algorithm 1. Similarly to e.g. A3C [Mnih et al., 2016], our algorithm is online up to the trajectory length. One can potentially be fully online by instead of relying on the sampled terminating state x_f, "bootstrapping" with the value of $P^o(x_f|x)$, but this requires an accurate estimate of P^o.

Algorithm 1 Actor-critic termination critic (ACTC)

Given: Initial Q-function q_0, step-sizes $(\alpha_k)_{k\in\mathbb{N}}$
for $k = 1, \dots$ **do**
  Sample a trajectory $x_s, \dots, x_i, \dots, x_f$ for T_k steps
  for all x_i in the trajectory **do**
    Update $\beta(x_i)$ with TG via (5)
    Update $P^o(x_f|x_i)$ with TD via (3)
    Update $P^o_\mu(x_f)$ with MSE towards $P^o(x_f|x_i)$
    Update B^o with MSE towards (5)
    Update $\pi^o(a_i|x_i)$ with PG via the multi-step advantage w.r.t. Q
    Update $Q(x_i, o)$ with TD in a standard way
  end for
end for

6 Experiments

We will now empirically support the following claims:

- Our algorithm directs termination into a small number of frequently visited states;
- The resulting options are intuitively appealing and improve learning performance; and
- The predictability objective is related to planning performance and the resulting options improve both the objective and planning performance.

We will evaluate the latter two in the classical Four Rooms domain [Sutton et al., 1999] (see Figure 6 in appendix). But before we begin, let us consider the learning dynamics of the algorithm on a small example.

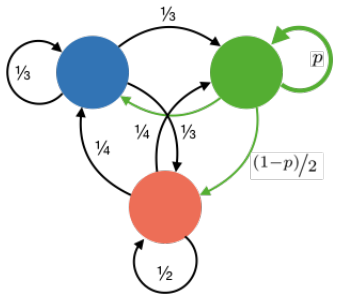


Figure 1: Example MDP

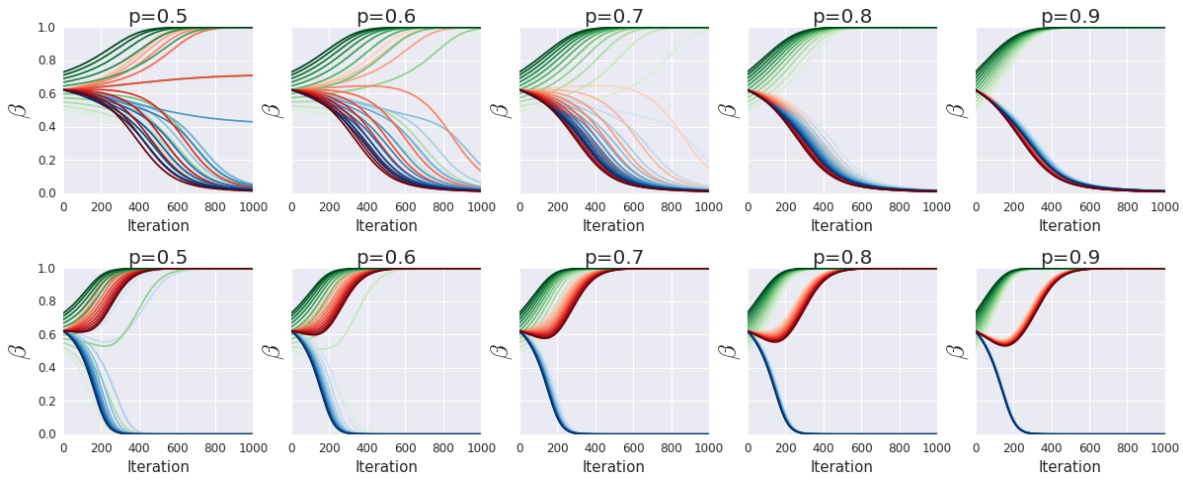


Figure 2: Learning dynamics. The color groups correspond with the states of the MDP from Fig. 1, while different lines correspond to different initial values of $\beta(green)$ (a lighter color depicts a lower value). So there are three lines for each starting value of β, one of each color. **First row:** Termination-critic. **Second row:** Naive reachability. We see that when the β-initialization is not too low, termination critic correctly concentrates termination on the attractor state and that state only, while the naive version saturates two of the states.

6.1 Learning Dynamics

The predictability criterion we have proposed is minimized by any termination scheme that deterministically terminates in a single state. Then, what solution do we expect for our algorithm to find? We hypothesize that the learning dynamics favor states that are visited more frequently, that is: are more reachable as measured by P^o, up to the initial disbalance of β. In this section we validate this hypothesis on a small example.

Consider the MDP in Fig. 1. The green state is a potential attractor, with its attraction value being determined by a parameter p. Let the initial $\beta(blue) = \beta(red) = 0.5$. In this experiment, we investigate the resulting solution as a function of the initial $\beta(green)$ and p. Following the intuition above, we expect the algorithm to increase $\beta(green)$ more when these values are higher. We compare the results with another way of achieving a similar outcome, namely by simply using the marginal value P^o_μ as the objective ("naive reachability"). As expected, we find that the full predictability objective is necessary for concentrating in a *single* most-visited state. See Fig. 2.

We further performed an ablation on the two advantage terms that comprise our loss (the reachability advantage and the trajectory advantage); these results are given in Figure 8. We find that neither term in isolation is sufficient, or as effective as the full objective.

6.2 Option Discovery

We now ask how well the termination critic and the predictability objective are suited as a method for end-to-end option discovery. As discussed, option discovery remains a challenging problem and end-to-end algorithms, such as the option-critic, often have difficulties learning non-trivial or intuitively appealing results.

We evaluate both ACTC and option-critic with deliberation cost (A2OC) on the Four Rooms domain using visual inputs (a top-down view of the maze passed through a convolutional neural network). The basic task is to navigate to a goal location, with zero per-step reward and a reward of 1 at the goal. The goal changes randomly between a fixed set of eight locations every 20 episodes, with the option components being oblivious to the location, but the option-level value function being able to observe the location of the goal.

6.2.1 Architecture

We build on the neural network architecture of A2OC with deliberation cost [Harb et al., 2018], which in turn is almost identical to the network for A3C [Mnih et al., 2016]. Specifically, the A3C network outputs a policy π and action-value function Q, whereas our network outputs $\{\pi^o\}_{o\in\mathcal{O}}$, and $\{Q(\cdot, o)\}_{o\in\mathcal{O}}$ given the input state. We use an additional network that takes in two input states, x_s and x_f, and outputs $\{P^o(x_f|x_s)\}_{o\in\mathcal{O}}$, and $\{\beta^o(x_f)\}_{o\in\mathcal{O}}$, as well as some useful auxiliary predictions: a marginal $\{P^o_\mu(x_f)\}_{o\in\mathcal{O}}$ and the update baseline $\{B^o\}_{o\in\mathcal{O}}$. This network uses a convolutional network with identical structure as used in A3C, that feeds into a single shared fully-connected hidden layer, followed by a specialized fully-connected layer for each of the outputs.

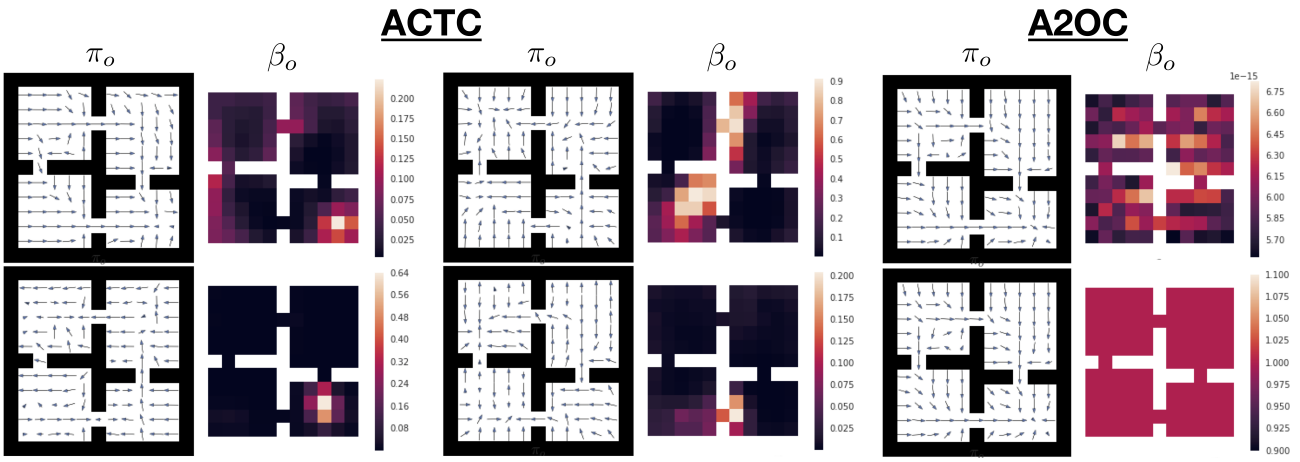


Figure 3: Example resulting options from ACTC (left) and A2OC (right). Each option is depicted via its policy and termination condition. ACTC concentrates termination probabilities around a small set of states while A2OC, with deliberation cost, tends to saturate on constant zero or constant one termination probability.

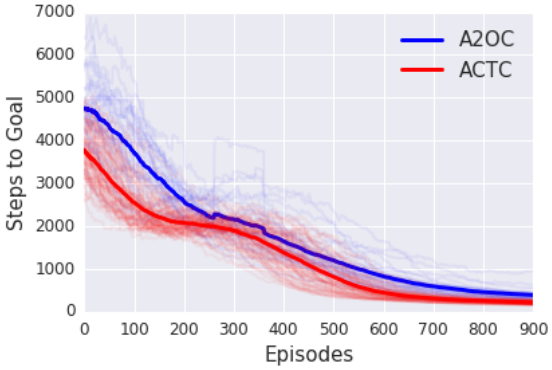


Figure 4: The learning performance of the two algorithms on the the Four Rooms task with switching goals. We plot the entire suite of hyperparameters, which for A2OC includes various deliberation costs, and for ACTC different learning rates for the β-network. We see ACTC exhibit better learning performance.

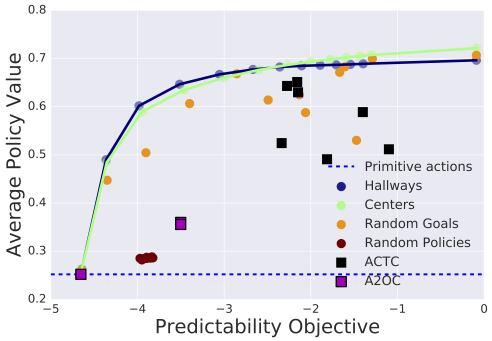


Figure 5: Investigating correlation between predictability and planning performance. Average policy value plotted against predictability objective (negative of the loss). A2OC options generalize poorly to unseen goals and have unpredictable terminations. ACTC optimizes the predictability objective leading to reusable options.

6.2.2 Learning and Planning in Four Rooms

We first depict the options qualitatively with an example termination profile shown in Figure 3. We see that ACTC leads to tightly concentrated regions with high termination probability and low probability elsewhere, whereas A2OC even with deliberation cost tends to converge to trivial termination solutions. Although ACTC does not always converge to terminating in a single region, it leads to distinct options with characteristic behavior and termination profiles.

Next, in Figure 4 we compare the online learning performance between ACTC and A2OC with deliberation cost. The traces indicate separate hyper-parameter settings and seeds for each algorithm and the bold line gives their average. ACTC enjoys better performance throughout learning.

6.3 Correlation with Planning Performance

Finally, we investigate the claim that more directed termination leads to improved planning performance. To this end, we generate various sets ($n = 4$) of goal-directed options in the Four Rooms domain by systematically varying the option-policy goal location and concentration of termination probability around the goal location. We evaluate these options, combined with primitive actions, by averaging the policy value during ten iterations of value iteration and all possible goal locations (see appendix for more details).

We compare this average policy value as a function of the predictability objective of each set of options in Figure 5. "Hallways" corresponds to one option for each hallway, "Centers" – to the centers of each room, "Random Goals" – to each option selecting a unique random goal location, and "Random Options" – to both the policies and termination probabilities being uniformly random. We observe, as has previously been reported, that even random options improve planning over primitive actions alone (shown by the dashed-line). Additionally, we confirm that generally as the predictability objective increases towards zero the average policy value increases, corresponding to faster convergence of value iteration.

Finally, we plot (with square markers) the performance of options learned from the previous section using ACTC and A2OC with deliberation cost. Due to A2OC's option collapse, its advantage over primitive actions is small, while ACTC performs similarly to the better (more deterministic) random goal options.

7 Related Work

The idea of decomposing behavior into reusable components or abstractions has a long history in the reinforcement learning literature. One question that remains largely unanswered, however, is that of suitable criteria for identifying such abstractions. The option framework itself [Sutton et al., 1999, Bacon et al., 2017] provides a computational model that allows the implementation of temporal abstractions but does in itself not provide an objective for option induction. This is addressed partially in [Harb et al., 2018] where long-lasting options are explicitly encouraged.

A popular means of encouraging specialization is via different forms of information hiding, either by shielding part of the policy from the task goal (e.g. [Heess et al., 2016]) or from the task reward (e.g. [Vezhnevets et al., 2017]). [Frans et al., 2018] combine information hiding with meta-learning to learn options that are easy to reuse across tasks.

Information-theoretic regularization terms that encourage mutual information between the option and features of the resulting trajectory (such as the final state) have been used to induce diverse options in an unsupervised or mixed setting (e.g. [Gregor et al., 2016, Florensa et al., 2017, Eysenbach et al., 2019]), and they can be combined with information hiding (e.g. [Hausman et al., 2018]). Similar to our objective they can be seen to encourage predictable outcomes of options. [Co-Reyes et al., 2018] have recently proposed a model that directly encourages the predictability of trajectories associated with continuous option embeddings to facilitate mixed model-based and model-free control.

Unlike our work, approaches described above consider options of fixed, predefined length. The problem of learning option boundaries has received some attention in the context of learning behavioral representations from demonstrations (e.g. [Daniel et al., 2016, Lioutikov et al., 2015, Fox et al., 2017, Krishnan et al., 2017]). These approaches effectively learn a probabilistic model of trajectories and can thus be seen to perform trajectory compression.

Finally, another class of approaches that is related to our overall intuition is one that seeks to identify "bottleneck" states and construct goal-directed options to reach them. There are many definitions of bottlenecks, but they are generally understood to be states that connect different parts of an environment, and are hence visited more often by successful trajectories. Such states can be identified through heuristics related to the pattern of state visitation [McGovern and Barto, 2001, Stolle and Precup, 2003] or by looking at between-ness centrality measures on the state-transition graphs. Our objective is based on a similar motivation of finding a small number of states that give rise to a compressed high-level decision problem which is easy to solve. However, our algorithm is very different (and cheaper computationally).

8 Discussion

We have presented a novel option-discovery criterion that uses predictability as a means of regularizing the complexity of behavior learned by individual options. We have applied it to learning meaningful *termination conditions* for options, and have demonstrated its ability to induce options that are non-trivial and useful.

Optimization of the criterion is achieved by a novel policy gradient formulation that relies on learned option models. In our implementation we choose to decouple the reward optimization from the problem of learning where to terminate. This particular choice allowed us to study the effects of meaningful termination in isolation. We saw that even if the option policies optimize the same reward objective, non-trivial terminations prevent option collapse onto the same policy.

This work has focused entirely on goal-directed options, whose purpose is to reach a certain part of the state space. There is another class, often referred to as skills, which are in a sense complementary, aiming to abstract behaviors that apply *anywhere* in the state space. One exciting direction for future work is to study the relation between the two and to design option induction criteria that can interpolate between different regimes.

References

[Bacon et al., 2017] Bacon, P.-L., Harb, J., and Precup, D. (2017). The option-critic architecture. In *Proceedings of 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 1726–1734.

[Bellman, 1957] Bellman, R. (1957). *Dynamic programming*. Princeton University Press.

[Borkar, 1997] Borkar, V. S. (1997). Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5):291–294.

[Co-Reyes et al., 2018] Co-Reyes, J. D., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., and Levine, S. (2018). Self-Consistent Trajectory Autoencoder: Hierarchical Reinforcement Learning with Trajectory Embeddings. In *Proceedings of the 35th International Conference on Machine Learning (ICML-18)*.

[Daniel et al., 2016] Daniel, C., van Hoof, H., Peters, J., and Neumann, G. (2016). Probabilistic inference for determining options in reinforcement learning. *Machine Learning, Special Issue*, 104(2):337–357.

[Eysenbach et al., 2019] Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. (2019). Diversity is all you need: Learning skills without a reward function. In *Proceedings of the 7th International Conference on Learning Representations (ICLR-19)*.

[Florensa et al., 2017] Florensa, C., Duan, Y., and Abbeel, P. (2017). Stochastic neural networks for hierarchical reinforcement learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR-17)*.

[Fox et al., 2017] Fox, R., Krishnan, S., Stoica, I., and Goldberg, K. (2017). Multi-level discovery of deep options. *CoRR*, abs/1703.08294.

[Frans et al., 2018] Frans, K., Ho, J., Chen, X., Abbeel, P., and Schulman, J. (2018). Meta learning shared hierarchies. In *Proceedings of the 6th International Conference on Learning Representations (ICLR-18)*.

[Gregor et al., 2016] Gregor, K., Rezende, D. J., and Wierstra, D. (2016). Variational intrinsic control. *CoRR*, abs/1611.07507.

[Harb et al., 2018] Harb, J., Bacon, P.-L., Klissarov, M., and Precup, D. (2018). When waiting is not an option: Learning options with a deliberation cost. In *Proceedings of 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*.

[Hausman et al., 2018] Hausman, K., Springenberg, J. T., Wang, Z., Heess, N., and Riedmiller, M. (2018). Learning an embedding space for transferable robot skills. In *Proceedings of the 6th International Conference on Learning Representations (ICLR-18)*.

[Heess et al., 2016] Heess, N., Wayne, G., Tassa, Y., Lillicrap, T. P., Riedmiller, M. A., and Silver, D. (2016). Learning and transfer of modulated locomotor controllers. *CoRR*, abs/1610.05182.

[Krishnan et al., 2017] Krishnan, S., Fox, R., Stoica, I., and Goldberg, K. (2017). Ddco: Discovery of deep continuous options for robot learning from demonstrations. In *Proceedings of the 1st Conference on Robot Learning (CoRL-17)*, pages 418–437.

[Lioutikov et al., 2015] Lioutikov, R., Neumann, G., Maeda, G., and Peters, J. (2015). Probabilistic segmentation applied to an assembly task. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 533–540.

[McGovern and Barto, 2001] McGovern, A. and Barto, A. G. (2001). Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*.

[Mnih et al., 2016] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*, pages 1928–1937.

[Puterman, 1994] Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York, NY, USA, 1st edition.

[Solway et al., 2014] Solway, A., Diuk, C., Córdova, N., Yee, D., Barto, A. G., Niv, Y., and Botvinick, M. M. (2014). Optimal behavioral hierarchy. *PLoS computational biology*, 10(8):e1003779.

[Stolle and Precup, 2003] Stolle, M. and Precup, D. (2003). Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223.

[Sutton and Barto, 2017] Sutton, R. S. and Barto, A. G. (2017). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 2nd edition.

[Sutton et al., 2000] Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function

approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063.

[Sutton et al., 1999] Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211.

[Vezhnevets et al., 2017] Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. (2017). Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML-17)*.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

A The Option Transition Process

It will be convenient to consider the option transition process:

$$\begin{aligned}
P^{(0)}(x_f|x_s) &= \Pr(x_t = x_f|x_t = x_s) = \mathbb{I}_{x_s=x_f} \\
P^{(1)}(x_f|x_s) &= \Pr(x_{t+1} = x_f|x_t = x_s) \\
&= (1-\beta^o(x_s))p^{\pi^o}(x_f|x_s) \\
&\dots \\
P^{(k)}(x_f|x_s) &= \Pr(x_{t+k} = x_f|x_t = x_s) \\
&= \sum_x P^{(1)}(x|x_s)P^{(k-1)}(x_f|x)
\end{aligned}$$

We can then rewrite P^o from (3) as:

$$\begin{aligned}
P^o(x_f|x_s) &= \beta^o(x_f)\left(P^{(0)}(x_f|x_s) + P^{(1)}(x_f|x_s) + \dots\right) \\
&= \beta^o(x_f)\sum_{k=0}^{\infty} P^{(k)}(x_f|x_s) \qquad (6)
\end{aligned}$$

B Omitted Proofs

B.1 Proof of Theorem 1

Proof. We have:

$$\begin{aligned}
&\nabla_{\theta_\beta} P^o(x_f|x_s) \\
&= \nabla_{\theta_\beta}\beta^o(x_s)\mathbb{I}_{x_f=x_s} + \nabla_{\theta_\beta}(1-\beta^o(x_s))\sum_x p^{\pi^o}(x|x_s)P^o(x_f|x) \\
&= \nabla_{\theta_\beta}\beta^o(x_s)\mathbb{I}_{x_f=x_s} + \sum_x p^{\pi^o}(x|x_s)\Big(\nabla_{\theta_\beta}P^o(x_f|x) \\
&\quad - \nabla_{\theta_\beta}\Big(\beta^o(x_s)P^o(x_f|x)\Big)\Big) \\
&= \nabla_{\theta_\beta}\beta^o(x_s)\mathbb{I}_{x_f=x_s} + \sum_x p^{\pi^o}(x|x_s)\Big(\nabla_{\theta_\beta}P^o(x_f|x) \\
&\quad - \nabla_{\theta_\beta}\beta^o(x_s)P^o(x_f|x) - \beta^o(x_s)\nabla_{\theta_\beta}P^o(x_f|x)\Big) \\
&= \nabla_{\theta_\beta}\beta^o(x_s)\Big(\mathbb{I}_{x_f=x_s} - \sum_x p^{\pi^o}(x|x_s)P^o(x_f|x)\Big) \\
&\quad + (1-\beta^o(x_s))\sum_x p^{\pi^o}(x|x_s)\nabla_{\theta_\beta}P^o(x_f|x). \qquad (7)
\end{aligned}$$

And so what we have is a $(1-\beta^o(x_i))$-discounted value function, whose reward is $\nabla_{\theta_\beta}\beta^o(x_i)r^o_{x_f}(x_i)$, where

$$r^o_{x_f}(x_i) = \mathbb{I}_{x_f=x_i} - \sum_{x_{i+1}} p^{\pi^o}(x_{i+1}|x_i)P^o(x_f|x_{i+1})$$

Now, from Eq. (3) and if $\beta^o(x) \neq 1$, we have:

$$\begin{aligned}
\sum_x p^{\pi^o}(x|x_s)P^o(x_f|x) &= \frac{P^o(x_f|x_s) - \beta^o(x_s)\mathbb{I}_{x_f=x_s}}{1-\beta^o(x_s)} \\
r^o_{x_f}(x_s) &= \mathbb{I}_{x_f=x_s} - \frac{P^o(x_f|x_s) - \beta^o(x_s)\mathbb{I}_{x_f=x_s}}{1-\beta^o(x_s)} \\
&= \frac{\mathbb{I}_{x_f=x_s} - P^o(x_f|x_s)}{1-\beta^o(x_s)} \qquad (8)
\end{aligned}$$

Using this notation, and recalling the transition process from Eq. (6), we can rewrite (7) as:

$$\begin{aligned}
&\nabla_{\theta_\beta}P^o(x_f|x_s) \\
&= \nabla_{\theta_\beta}\beta^o(x_s)r^o_{x_f}(x_s) + \sum_x P^{(1)}(x|x_s)\nabla_{\theta_\beta}P^o(x_f|x) \\
&= \sum_x\sum_{k=0}^{\infty} P^{(k)}(x|x_s)\nabla_{\theta_\beta}\beta^o(x)r^o_{x_f}(x) \\
&= \sum_x \frac{P^o(x|x_s)}{\beta^o(x)}\nabla_{\theta_\beta}\beta^o(x)r^o_{x_f}(x) \\
&= \sum_x P^o(x|x_s)\nabla_{\theta_\beta}\log\beta^o(x)r^o_{x_f}(x)
\end{aligned}$$

Where the third equality follows from (6) and requires for $\beta^o(x)$ to not be 0.

□

B.2 Proof of Proposition 1

Proof. Let $\Pr(x|o)$ denote the probability of a state x being terminal for an option o. By definition of entropy we have:

$$\begin{aligned}
H(X_f|o) &= -\sum_{x_f}\Pr(x_f|o)\log\Pr(x_f|o) \\
&= -\sum_{x_f}\sum_{x_s}\Pr(x_s|o)\Pr(x_f|x_s,o) \\
&\qquad \times \log\sum_{x_s}\Pr(x_s|o)\Pr(x_f|x_s,o) \\
&= -\sum_{x_f}\sum_{x_s} d^\mu(x_s|o)P^o(x_f|x_s) \\
&\qquad \times \log\underbrace{\sum_{y_s} d^\mu(y_s|o)P^o(x_f|y_s)}_{\text{marginal } P^o_\mu(x_f)} \\
&= -\sum_{x_s} d^\mu(x_s|o)\sum_{x_f} P^o(x_f|x_s)\log P^o_\mu(x_f)
\end{aligned}$$

□

B.3 Proof of Theorem 2

Proof.

$$\begin{aligned}
\nabla_{\theta_\beta}J(P^o) &= -\nabla_{\theta_\beta}\underbrace{\sum_{x_s} d^\mu(x_s|o)}_{\mathbb{E}_{x_s}}\sum_{x_f} P^o(x_f|x_s)\log P^o_\mu(x_f) \\
&= -\mathbb{E}_{x_s}\Big[\sum_{x_f}\Big(\nabla_{\theta_\beta}P^o(x_f|x_s)\log P^o_\mu(x_f)
\end{aligned}$$

$$
\begin{aligned}
&+ P^o(x_f|x_s)\frac{\nabla_{\theta_\beta} P^o_\mu(x_f)}{P^o_\mu(x_f)}\Big)\Big] \\
&= -\mathbb{E}_{x_s}\Big[\sum_{x_f}\Big(\sum_x P^o(x|x_s) r^o_{x_f}(x)\nabla_{\theta_\beta}\log\beta^o(x)\log P^o_\mu(x_f) \\
&\quad + \frac{P^o(x_f|x_s)}{P^o_\mu(x_f)}\underbrace{\sum_{y_s} d^\mu(y_s|o)\sum_x P^o(x|y_s)\, r^o_{x_f}(x)\nabla_{\theta_\beta}\log\beta^o(x)}_{\sum_x P^o_\mu(x)}\Big)\Big] \\
&= -\mathbb{E}_{x_s}\Big[\sum_x P^o(x|x_s)\nabla_{\theta_\beta}\log\beta^o(x)\sum_{x_f} r^o_{x_f}(x) \\
&\quad \times\Big(\log P^o_\mu(x_f) + \frac{P^o(x_f|x_s)}{P^o_\mu(x_f)}\frac{P^o_\mu(x)}{P^o(x|x_s)}\Big)\Big] \\
&= -\mathbb{E}_{x_s}\Big[\sum_x P^o(x|x_s)\frac{\nabla_{\theta_\beta}\beta^o(x)}{\beta^o(x)}\sum_{x_f}\frac{\mathbb{I}_{x_f=x} - P^o(x_f|x)}{1-\beta^o(x)} \\
&\quad \times\Big(\log P^o_\mu(x_f) + \frac{P^o(x_f|x_s)}{P^o_\mu(x_f)}\frac{P^o_\mu(x)}{P^o(x|x_s)}\Big)\Big] \\
&= -\underbrace{\sum_{x_s} d^\mu(x_s|o)}_{\text{sample}}\quad\underbrace{\sum_x \frac{P^o(x|x_s)}{\beta^o(x)}}_{\text{sample (continuation)}}\quad\frac{\nabla_{\theta_\beta}\beta^o(x)}{1-\beta^o(x)} \\
&\quad \times\Big[\Big(\log P^o_\mu(x)+1\Big) \\
&\quad -\underbrace{\sum_{x_f} P^o(x_f|x)}_{\text{sample}}\Big(\log P^o_\mu(x_f) + \frac{P^o(x_f|x_s)P^o_\mu(x)}{P^o_\mu(x_f)P^o(x|x_s)}\Big)\Big]
\end{aligned}
$$

Sampling the highlighted expectations, and noting that if ℓ are the logits of β^o,

$$\nabla_{\theta_\beta}\ell_{\beta^o}(x) = \frac{\nabla_{\theta_\beta}\beta^o(x)}{\beta^o(x)(1-\beta^o(x))},$$

we have our result.

□

C Correlation with Planning Performance

The policies considered in these experiments consist of some set of four options combined with the set of primitive actions. Planning performance, for a single goal-directed task, is evaluated as the average policy value over all states at the end of each of ten iterations of value iteration. Consider Figure 7 which shows the value iteration performance curve for a single task, comparing policies of primitive actions, options, and their combination. The planning performance is the average of this curve for ten iterations, further averaged over all possible goal-directed tasks in Four Rooms. This measures how quickly value iteration, using this set of option policies and terminations, is able to plan.

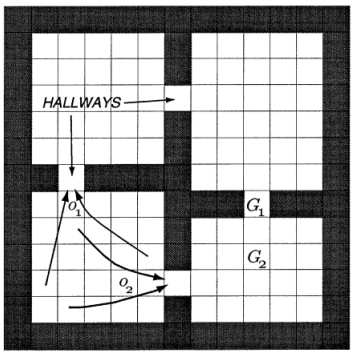


Figure 6: The Four Rooms domain map.

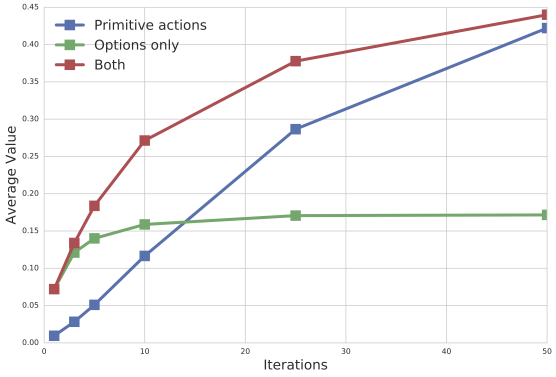


Figure 7: Example of planning performance using options, primitive actions, and both options and primitive actions.

D Learning Dynamics

Fig. 8 further studies the learning dynamics induced by the different components of the algorithm. We compare the previous two variants from Fig. 2 with only including the reachability advantage term (Row 3), and only including the trajectory advantage term (Row 4). The former does not focus on a single state, while the latter does not concentrate at all for many values of β.

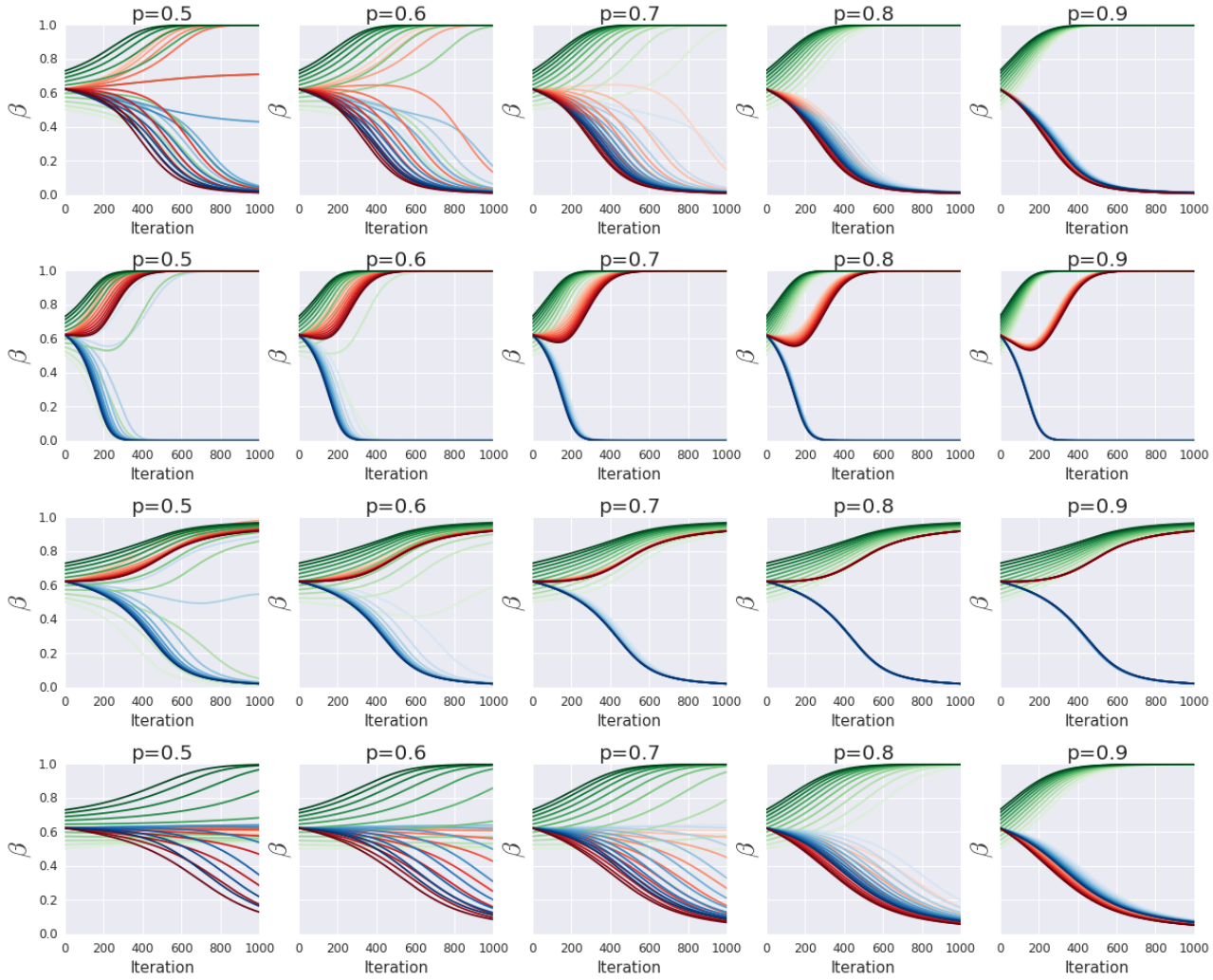


Figure 8: Learning dynamics. The color groups correspond with the states of the MDP from Fig. 1, while different lines correspond to different initial values of $\beta(green)$ (a lighter color depicts a lower value), **First row:** Termination-critic. **Second row:** Naive reachability. **Third row:** Only termination score advantage. **Fourth row:** Only relative termination advantage. We see that when the β-initialization is not too low, termination critic correctly concentrates termination on the attractor state and that state only, while the naive version saturates two of the states. The two ablations show the reachability advantage having similar behavior to naive reachability, while the trajectory advantage is not concentrating enough when the attraction values are low.