Supplemental Material: A Laplacian Framework for Option Discovery in Reinforcement Learning

Marlos C. Machado [1] **Marc G. Bellemare** [2] **Michael Bowling** [1]

This supplementary material contains details omitted from the main text due to space constraints. The list of contents is below:

- Supporting lemmas and their respective proofs, as well as a more detailed proof of Theorem 3.1;
- Description of how to easily compute the *diffusion time* in tabular MDPs;
- The options leading to bottleneck states (doorways) we used in our experiments;
- Performance comparisons between eigenoptions and options generated to reach randomly selected states;
- Demonstration of the applicability of eigenoptions in multiple tasks with a new set of experiments;
- Further details on the empirical setting used in the Arcade Learning Environment.

A. Lemmas and Proofs

Lemma 4.1. *Suppose* $(I + A)$ *is a non-singular matrix, with* $||A|| \leq 1$. *We have:*

$$||(I + A)^{-1}|| \leq \frac{1}{1 - ||A||}.$$

Proof. [1]

$$\begin{aligned}
(I + A)(I + A)^{-1} &= I \\
I(I + A)^{-1} + A(I + A)^{-1} &= I \\
(I + A)^{-1} &= I - A(I + A)^{-1} \\
||(I + A)^{-1}|| &= ||I - A(I + A)^{-1}|| \\
&\leq ||I|| + ||A(I + A)^{-1}|| && \text{because } ||A + B|| \leq ||A|| + ||B|| \\
&\leq 1 + ||A||||(I + A)^{-1}|| && \text{because } ||AB|| \leq ||A|| \cdot ||B|| \\
||(I + A)^{-1}|| - ||A||||(I + A)^{-1}|| &\leq 1 \\
(1 - ||A||)||(I + A)^{-1}|| &\leq 1 \\
||(I + A)^{-1}|| &\leq \frac{1}{1 - ||A||} && \text{if } ||A|| \leq 1.
\end{aligned}$$

□

Lemma 4.2. *The induced infinity norm of* $(I - \gamma T)^{-1}T$ *is bounded by*

$$||(I - \gamma T)^{-1}T||_\infty \leq \frac{1}{(1 - \gamma)}.$$

[1]University of Alberta [2]Google DeepMind. Correspondence to: Marlos C. Machado <machado@ualberta.ca>.

[1]Our proof follows closely the proof of Parnell in lecture notes available at http://www-solar.mcs.st-and.ac.uk/~clare/Lectures/num-analysis.html.

Proof.

$$||(I-\gamma T)^{-1}T||_\infty \leq ||(I-\gamma T)^{-1}||_\infty ||T||_\infty \qquad \text{because } ||AB||_\infty \leq ||A||_\infty \cdot ||B||_\infty$$
$$||(I-\gamma T)^{-1}T||_\infty \leq \frac{1}{1-||-\gamma T||_\infty}||T||_\infty \qquad \text{Lemma 3.1}$$
$$||(I-\gamma T)^{-1}T||_\infty \leq \frac{1}{1-\gamma||T||_\infty}||T||_\infty \qquad \text{because } ||\lambda B|| = |\lambda|||B||$$
$$||(I-\gamma T)^{-1}T||_\infty \leq \frac{1}{(1-\gamma)}$$

□

Theorem 4.1 (Option's Termination)**.** *Consider an eigenoption $o = \langle \mathcal{I}_o, \pi_o, \mathcal{T}_o \rangle$ and $\gamma < 1$. Then, in an MDP with finite state space, $\mathcal{T}_o$ is nonempty.*

Proof. This proof is more detailed than the one presented in the main paper. We can write the Bellman equation in the matrix form: $\mathbf{v} = \mathbf{r} + \gamma T\mathbf{v}$, where $\mathbf{v}$ is a *finite* column vector with one entry per state encoding its value function. From equation (1) in the main paper we have $\mathbf{r} = T\mathbf{w} - \mathbf{w}$ with $\mathbf{w} = \boldsymbol{\phi}(s)^\top \mathbf{e}$, where $\mathbf{e}$ denotes the eigenpurpose of interest. Therefore:

$$\begin{aligned}
\mathbf{v} &= T\mathbf{w} - \mathbf{w} + \gamma T\mathbf{v} \\
\mathbf{v} + \mathbf{w} &= T\mathbf{w} + \gamma T\mathbf{v} \\
&= T\mathbf{w} + \gamma T\mathbf{v} + \gamma T\mathbf{w} - \gamma T\mathbf{w} \\
&= (1-\gamma)T\mathbf{w} + \gamma T(\mathbf{v}+\mathbf{w}) \\
\mathbf{v} + \mathbf{w} - \gamma T(\mathbf{v}+\mathbf{w}) &= (1-\gamma)T\mathbf{w} \\
(I-\gamma T)(\mathbf{v}+\mathbf{w}) &= (1-\gamma)T\mathbf{w} \\
\mathbf{v} + \mathbf{w} &= (1-\gamma)(I-\gamma T)^{-1}T\mathbf{w} \qquad (I-\gamma T)^{-1} \text{ is guaranteed to be nonsigular because } ||T|| \leq 1, \text{ where } ||T|| = \sup_{\mathbf{v}:||\mathbf{v}||_\infty=1} ||T\mathbf{v}||_\infty. \text{ By Neumann series we have } (I-\gamma T)^{-1} = \sum_{n=0}^{\infty} \gamma^n T^n \\
||\mathbf{v}+\mathbf{w}||_\infty &= (1-\gamma)||(I-\gamma T)^{-1}T\mathbf{w}||_\infty \qquad \text{using the induced norm} \\
||\mathbf{v}+\mathbf{w}||_\infty &\leq (1-\gamma)||(I-\gamma T)^{-1}T||_\infty ||\mathbf{w}||_\infty \qquad \text{because } ||A\mathbf{x}|| \leq ||A|| \cdot ||\mathbf{x}|| \\
||\mathbf{v}+\mathbf{w}||_\infty &\leq (1-\gamma)\frac{1}{(1-\gamma)}||\mathbf{w}||_\infty \qquad \text{Lemma 3.2} \\
||\mathbf{v}+\mathbf{w}||_\infty &\leq ||\mathbf{w}||_\infty
\end{aligned}$$

We can shift $\mathbf{w}$ by any finite constant without changing the reward, *i.e.* $T\mathbf{w} - \mathbf{w} = T(\mathbf{w}+\boldsymbol{\delta}) - (\mathbf{w}+\boldsymbol{\delta})$ because $T\mathbf{1}\boldsymbol{\delta} = \mathbf{1}\boldsymbol{\delta}$ since $\sum_j T_{i,j} = 1$. Therefore, we can assume $\mathbf{w} \geq \mathbf{0}$. Let $s^* = \arg\max_s \mathbf{w}_{s^*}$, so that $\mathbf{w}_{s^*} = ||\mathbf{w}||_\infty$. Clearly $\mathbf{v}_{s^*} \leq \mathbf{0}$, otherwise $||\mathbf{v}+\mathbf{w}||_\infty \geq |\mathbf{v}_{s^*} + \mathbf{w}_{s^*}| = \mathbf{v}_{s^*} + \mathbf{w}_{s^*} > \mathbf{w}_{s^*} = ||\mathbf{w}||_\infty$, arriving at a contradiction.

□

Lemma 5.1. *In the tabular case, if all transitions in the MDP have been sampled once, $T^\top T = 2L$.*

Proof. Let t_{ij} and tt_{ij} denote the entries in the i-th row and j-th column of matrices T and $T^\top T$. We can write tt_{ij} as:

$$tt_{ij} = \sum_k t_{ik} \times t_{jk}. \tag{1}$$

In the tabular case, t_{ij} has three possible values:

- $t_{ij} = +1$, meaning that the agent arrived in state j at time step i,
- $t_{ij} = -1$, meaning that the agent left state j at time step i,
- $t_{ij} = 0$, meaning that the agent did not arrive nor leave state j at time step i.

We decompose $T^\top T$ in two matrices, K and Z, such that $T^\top T = K + Z$. Here Z is a diagonal matrix such that $z_{ii} = tt_{ii}$, for all i; and K contains all elements from $T^\top T$ that lie outside the main diagonal.

When computing the elements of Z we have $i = j$. Thus $z_{ii} = \sum_k t_{ik}^2$. Because we square all elements, we are in fact summing over all transitions leaving (-1^2) and arriving (1^2) in state i, counting the node's degree twice. Thus, $Z = 2D$.

When not computing the elements in the main diagonal, for the element tt_{ij}, we add all transitions that leave state i arriving in state j (-1×1), and those that leave state j arriving in state i (1×-1). We assume each transition has been sampled once, thus:

$$tt_{ij} = \begin{cases} -2, & \text{if the transition between states } i \text{ and } j \text{ exists,} \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, we have $K = -2W$ and $T^\top T = K + Z = 2(D - W)$. □

B. Diffusion Time Computation

In the main paper we introduced *diffusion time* as a new metric to evaluate exploration, but we did not discuss how it can be computed. Diffusion time encodes the expected number of time steps required to navigate between any two states in the MDP when following a random walk. In tabular domains, we can easily compute the diffusion time with dynamic programming. To do so we define a new MDP such that the value function of a state s, under a uniform random policy, encodes the expected number of steps required to navigate between state s and a chosen goal state. We can then compute the expected number of steps between any two states by averaging, for each possible goal, the value of all other states.

The MDP in which the value function of state s encodes the expected number of time steps from s to a goal state has $\gamma = 1$ and a reward function where the agent observes $+1$ at every time step in which it is not in the goal state. Policy evaluation in this case encodes the expected number of time steps the agent will take before arriving to the goal state. To compute the diffusion time we iterate over all possible states, defining them as terminal states, and averaging the value function of the other states in that MDP.

C. Options Leading to Doorways in the 4-room Domain

Figure 1 depicts the four options we refer to in Section 4 as the options leading to bootleneck states, *i.e.*, doorways. Each option is defined in a room and it moves the agent toward the closest doorway. These options were inspired by Solway et al. (2014)'s discussion about the optimal options discovered by their algorithm.

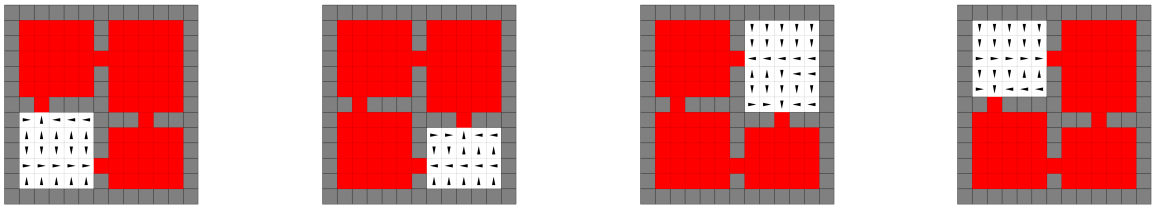
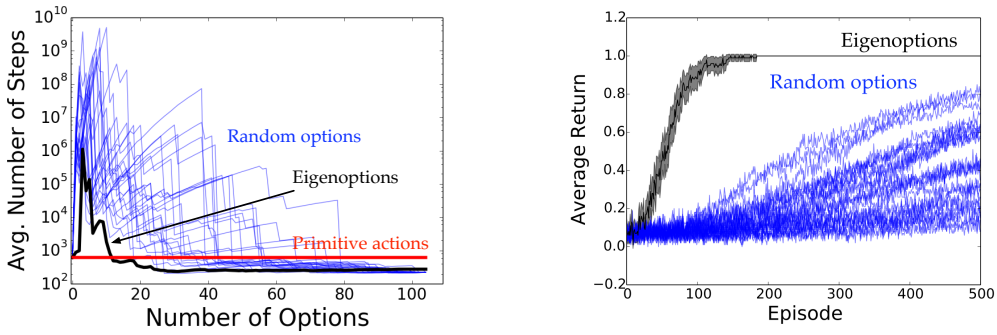


Figure 1. Options leading to bottleneck states. Each option is defined in a single room, moving the agent to the closest doorway.



(a) Diffusion time. The y-axis is in logarithmic scale.

(b) Learning curve using 64 options.

Figure 2. Diffusion time and learning performance of eigenoptions and of random options in the 4-room domain.

D. Comparison to Random Options

In this section we show the importance of using information about diffusion in the environment to define the option's purposes. This information impacts the sequence of subgoal locations the options' seek after, as well as the time scales they operate at. The ordering in which the eigenoptions are discovered and the different time scales they operate at can have a major impact on the agents' performance.

We demonstrate the importance of using the environment's diffusion information by comparing our approach to *random options*, a simple baseline that does not use such information. This baseline defines an option to be the policy, defined in the whole state space, that terminates in a randomly selected state of the environment. We performed our experiments in the tabular case because it is not clear how we can extend this baseline to settings in which states cannot be enumerated.

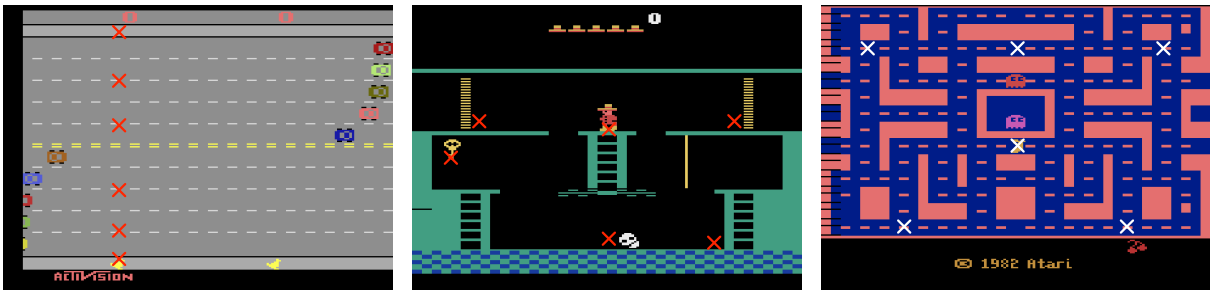
Figure 2a depicts the diffusion time (*c.f.* Section B) of random options and eigenoptions in the 4-room domain. We used the same method described in Section 4.2 to obtain the eigenoptions' performance. For the random options results, we added them incrementally to the agent's action set until having added all possible options. We repeated this process 24 times to verify the impact of adding random options in a different order. Each blue line represents the performance of one of the evaluated sequences. The results clearly show that eigenoptions do more than going to a randomly selected state. Most of the obtained sequences of random options fail to reduce the agent's diffusion time. They increase it by several orders of magnitude (notice the y-axis is in logarithmic scale) until having enough options available to the point that the graph is almost fully connected, that is, when the agent basically has an option leading it to each possible state in the MDP.

Figure 2b was generated following the protocol described in Section 4.3. It depicts the learning curve of agents equipped with eigenoptions and of agents equipped with random options. As before, the blue lines indicate the agent's performance in individual runs. We can see that no individual run is competitive to eigenoptions. When fewer options are used (not shown), the variance across individual runs is even larger, depending on whether one of the random options terminates near the goal state. In some runs the agent never even learns to reach the goal. Therefore, as in the diffusion time, on average, random options are not competitive to eigenoptions, demonstrating the importance of the diffusion model we use.

D. Empirical Evaluation of the Agent's Performance in Multiple Tasks

In Section 4 we argued that eigenoptions are useful for multiple tasks, based on results showing that eigenoptions allow us to find and to accumulated rewards faster. Here we explicit demonstrate the uselfuness of eigenoptions to multiple tasks. We evaluate the agents' performance for different starting and goal states in the 4-room domain. As in Section 4.3, we use Q-Learning ($\alpha = 0.1, \gamma = 0.9$) to learn a policy over primitive actions. The behavior policy chooses uniformly over primitive actions and options, following them until termination. Episodes were 100 time steps long, and we learned for 250 episodes. For clarity, we zoom in the plots on the interval in which agents are still learning.

Figure 5 depicts, after learning for a pre-determined number of episodes, the average over 100 trials of the agents' final performance, as well as the starting (S) and goal (G) states. Based on our previous results, we fixed the number of used eigenoptions to 64 (32 options and their negations). In this set of experiments we also compare our approach to traditional



(a) FREEWAY (b) MONTEZUMA'S REVENGE (c) MS PAC-MAN

Figure 3. Pre-defined start states in Atari 2600 games.

bottleneck options (Figure 1).

The obtained results show that switching the positions of the starting and goal states have no effect in the performance of our algorithm. Also, in almost all settings, the agents augmented by eigenoptions outperfom those equipped only with primitive actions. The comparison between eigenoptions and options that look for bottleneck states is more subtle. As expected, agents equipped with eigenoptions outperform agents equipped with options leading to bottleneck states in settings in which the goal state is far from the doorways, as discussed in the main paper. In scenarios where the goal state is closer to bottleneck states, the options leading to doorways are more competitive. Importantly, this analysis is based on the results when using 64 eigenoptions, which may not encode all options required to go to a specific region of the state space.

E. Experimental Setup in the Arcade Learning Environment

We defined six different starting states in each Atari 2600 game, letting the agent take random actions from that point until termination. The agent follows a pre-determined sequence of actions leading it to each starting state. We store the observed transitions leading the agent to the start states as well as those obtained from the random actions. In the main paper we provided results for FREEWAY and MONTEZUMA'S REVENGE. In this section we also provide results for MS PAC-MAN. The starting states for all three games are depicted in Figure 3.

The agent plays rounds of six episodes, with each episode starting from a different start state, until it observes at least 25,000 new transitions. The final incidence matrix in which we ran the SVD had 25,000 rows, which we sampled uniformly from the set of observed transitions. The agent used the deterministic version of the Arcade Learning Environment (ALE), the games' minimal action set and, a frame skip of 1.

We used three games to evaluate the options we discover in the sample-based setting with linear function approximation. We discussed the results for FREEWAY and MONTEZUMA'S REVENGE in the main paper. The results we obtained in MS. PAC-MAN are similar to those we already discussed. MS. PAC-MAN is a game in which the agent needs to navigate through a maze eating pellets while avoiding ghosts. As in the other games, the agent has the clear intent of reaching particular positions in the screen, such as corners and intersections. Figure 4 depicts the positions in which agents tend to spend most of their time on. A video of the highlighted options can be found online.[2]

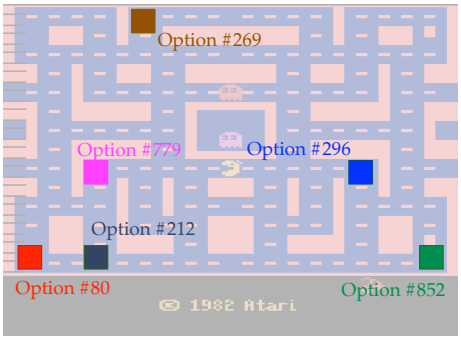


Figure 4. Options in MS. PAC-MAN (*c.f.* text for details).

[2] https://youtu.be/2BVicx4CDWA

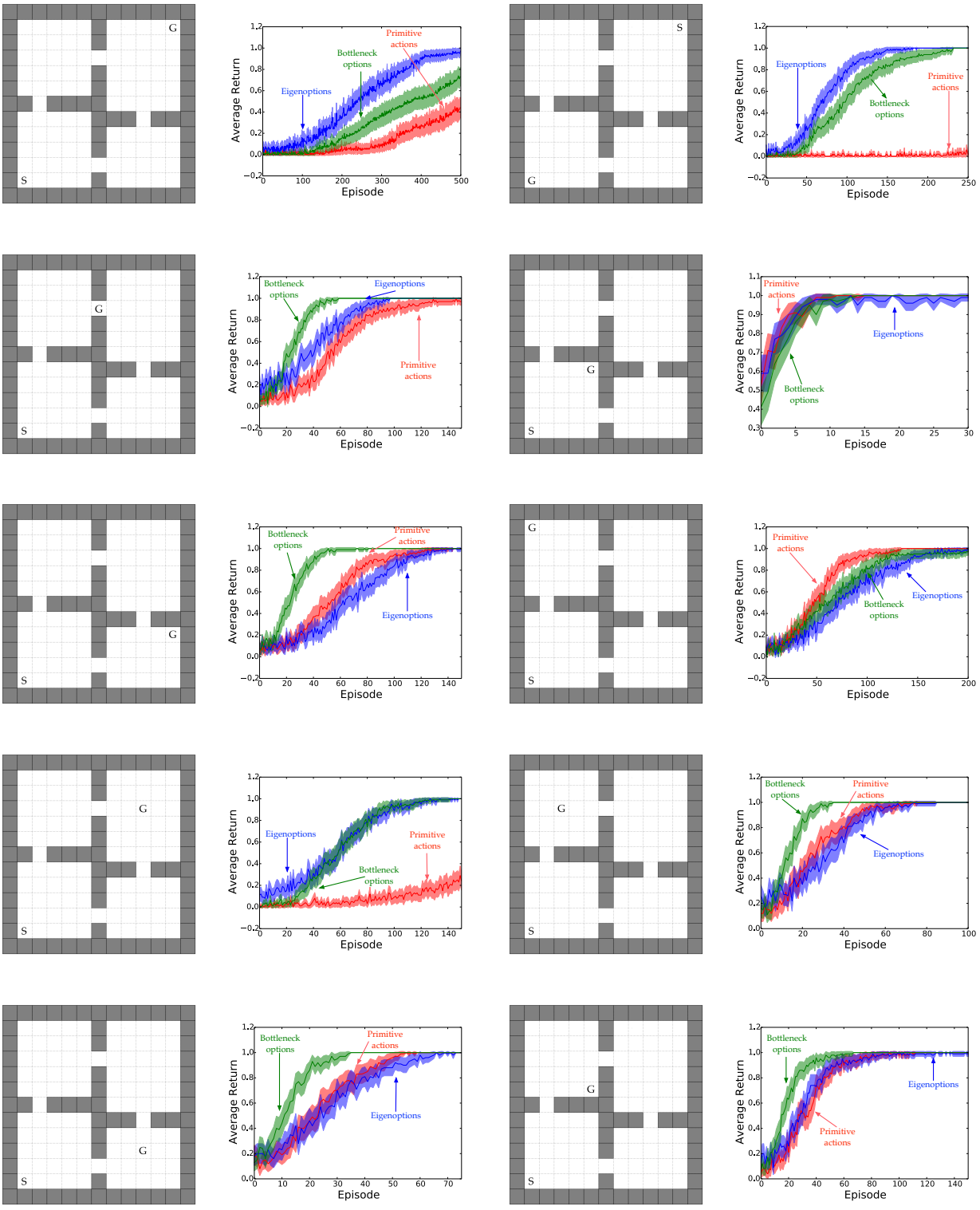


Figure 5. Agents performance in different tasks when using eigenoptions, bottleneck options, and primitive actions.