Learning and Planning with Timing Information in Markov Decision Processes

Pierre-Luc Bacon, Borja Balle, and **Doina Precup**
Reasoning and Learning Lab,
School of Computer Science, McGill University
{*pbacon,bballe,dprecup*}*@cs.mcgill.ca*

Abstract

We consider the problem of learning and planning in Markov decision processes with temporally extended actions represented in the options framework. We propose to use predictions about the duration of extended actions to represent the state and show that this leads to a compact predictive state representation model independent of the set of primitive actions. Then we develop a consistent and efficient spectral learning algorithm for such models. Using just the timing information to represent states allows for faster improvement in the planning performance. We illustrate our approach with experiments in both synthetic and robot navigation domains.

1 INTRODUCTION

Modelling the dynamics of an agent embedded in a large, complex environment is key to building good planning algorithms for such agents. In most practical applications, models are carefully designed by hand, and the agent's "state" is given by measurements which are understandable by the designer of the system (such as spatial location and velocity, in the case of a robot). However, learning dynamical models for such states from data, as well as planning with them can be quite tricky. An alternative idea is to use models that are "subjective", centered on the agent's own perception and action capabilities. For example, affordances [Gibson, 1977] describe "state" through the courses of action that are enabled. Similarly, in robotics, subjective representations have been used to model dynamics, e.g. [Bowling *et al.*, 2005; Stober *et al.*, 2011]. Such models are appealing from a psychological point of view, but run into computational problems in very large observation spaces.

In this paper, we focus on a special class of subjective models, *timing models*, which arise from restricting the observations available to the agent to just information about the duration of certain courses of action. Timing of events is understood to be crucial to animal learning [Machado *et al.*, 2009]. The goal of this paper, however, is not learning of the timing of external events, but rather to learn the duration of courses of action that an agent might take. The ensemble of such durations will constitute the agent's *state*, which will be maintained as new data is received. We use the framework of options [Sutton *et al.*, 1999] to model extended courses of actions, and we present an approach for learning a predictive model of option durations.

Our models over durations can be viewed as affordances if we consider an option to be available when its predicted duration is within some reasonable bounds. Note that these models are much simpler than full option models, which provide joint information on the timing as well as the state or observation in which the option will terminate, e.g. [Wolfe and Singh, 2006]. We emphasize that timing information is very cheap to measure and process, even with simple hardware. Hence, a major area of application for this type of approach is on devices in which processing can be costly, such as simple, Roomba-like robots (as in our experiments) or cellphones (on which using a lot of sensors or computation drains the battery).

Our approach can also be interpreted as a computationally and statistically efficient way of exploiting prior information about useful courses of action provided in the form of options. The size of our models is independent of the number of possible primitive actions in the underlying system, which is very useful in large or continuous action spaces. Moreover, because we are able to learn feature representations for states using timing information only, our method can be applied to observable settings with high-dimensional, complex observations, as well as to partially observable settings, where building a full model for planning would be too data and computation-hungry. Examples include investment problems (where the best strategy may include information about past performance, past trades, news etc, which is too hard to process) or robotics (where sensors may produce a lot of data, but this may not be easy to process in real-time).

Of course, the utility of timing models depends on the nature of the task to be solved by the agent, as well as on the "quality" of the options available to the agent. The simplest example in which option duration models are beneficial is that of minimum time to goal problems, in which an agent receives a fixed penalty per time step until its task is completed. In this case, knowing the duration of an option immediately provides the reward model, so the option duration model has direct value for a planner. More generally, option duration models are beneficial as a form of localization. If you imagine a robot that has models for options that move radially out from the current position, this would allow localizing with respect to all neighboring walls. Finally, consider a problem in which a financial agent is holding stocks, and options which hold a particular stock while it is above a certain value, and sell under that value. In this case, timing models tell us exactly when stocks would be crossing certain barriers. It is clear in this case that, even though we are estimating only durations, the model encodes important state information (because of the way in which the options are defined).

In this paper, we analyze the capacity of option duration models to represent states in a Markov Decision Process (MDP). We propose a spectral algorithm for learning option duration models which builds on existing work for learning transformed predictive state representations [Rosencrantz *et al.*, 2004]. Finally we evaluate the quality of learning and planning with this type of models in experiments with discrete and continuous MDPs.

2 PRELIMINARIES AND NOTATION

We use bold letters to denote vectors $\mathbf{v} \in \mathbb{R}^d$ and matrices $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$. For matrix $\mathbf{M}$, $\mathbf{M}^+$ denotes its Moore–Penrose pseudo-inverse. Sometimes we name the columns and rows of a matrix using ordered index sets $\mathcal{I}$ and $\mathcal{J}$, so $\mathbf{M} \in \mathbb{R}^{\mathcal{I} \times \mathcal{J}}$ denotes a matrix of size $|\mathcal{I}| \times |\mathcal{J}|$ with rows and columns indexed by $\mathcal{I}$ and $\mathcal{J}$ respectively.

Let Σ be a finite set of symbols. We use $\Sigma^\star$ to denote the set of all finite strings over Σ and λ for the empty string. Given two strings $u, v \in \Sigma^\star$, $w = uv$ is their concatenation, in which case u is called a prefix of w, and v is a suffix of w. Given two sets of strings $\mathcal{P}, \mathcal{S} \subseteq \Sigma^\star$, $\mathcal{PS}$ is the set obtained by taking every string of the form uv with $u \in \mathcal{P}$ and $v \in \mathcal{S}$.

Given a predicate $\mu(x)$ we denote by $\mathbb{I}[\mu(x)]$ the indicator function which is one when $\mu(x)$ is true and zero otherwise.

2.1 Markov Decision Processes and Temporally Extended Actions

A *Markov decision process* (MDP) is a tuple $M = \langle S, A, P, R \rangle$ where S is the state set, A is the action set, $P : S \times A \to (S \to [0, 1])$ defines a probability distribution over next states, and $R : S \times A \to \mathbb{R}$ is the expected reward function (see [Puterman, 1994] for a review). We refer to probability distributions on S by α, but sometimes use $\boldsymbol{\alpha}$ to stress that we view them as vectors in $\mathbb{R}^S$. Suppose α is a distribution over S and $\pi : S \times A \to [0, 1]$ is a stochastic action policy which, given state s, chooses action a with probability $\pi(s, a)$. The environment then returns a state sampled from P; and the resulting state distribution α' is given by:

$$\alpha'(s') = \sum_{s \in S} \alpha(s) \sum_{a \in A} \pi(s, a) P(s, a)(s') \ .$$

Temporal abstraction in MDPs has been used as a tool to speed up learning and planning algorithms. We adopt the framework of options [Sutton *et al.*, 1999], with the goal of learning state representations based on option timing models. An *option* is a tuple $\omega = \langle I_\omega, \pi_\omega, \beta_\omega \rangle$ where $I_\omega \subseteq S$ is the set of initial states, $\pi_\omega : S \times A \to [0, 1]$ is the option's stochastic action policy, and $\beta_\omega : S \to [0, 1]$ is the option termination probability for each state. We will drop the option's subscript and write $\omega = \langle I, \pi, \beta \rangle$ when there is no risk of confusion.

We say that an agent *interacts with an MDP via a set of options* Ω if at all times the actions performed by the agent follow the policy specified by some option $\omega \in \Omega$, which is executed until termination.

Each option has an associated reward model $R(s, \omega)$ and an associated transition model $P(s, \omega)(s')$, which can be computed using the MDP model and the definition of the options (see [Sutton *et al.*, 1999] for details). Given a set of options Ω, the optimal option-value function satisfies the following Bellman equation:

$$Q^*_\Omega(s, \omega) = R(s, \omega) + \sum_{s' \in S} P(s, \omega)(s') \max_{\omega' \in \Omega} Q^*_\Omega(s', \omega')$$

Planning with options aims to find the optimal policy over options π^*_Ω, which is achieved most often by estimating Q^*_Ω. This does not require a model to be known; instead, it can be done using samples, in similar fashion as Q-learning. If the state space is large or continuous, function approximation can used to represent states. The option duration model that we develop in this paper will be used as a form of state representation in order to learn an approximation to Q^*_Ω.

2.2 Predictive State Representations

A *predictive state representation* is a model of a dynamical system in which the current state is represented as a set of predictions about the future behavior of the system [Littman *et al.*, 2002; Singh *et al.*, 2004]. We use a particular instantiation of this general idea, the so-called *transformed linear predictive state representation* [Rosencrantz *et al.*, 2004], abbreviated as PSR.

Let Σ be a finite set of symbols. A PSR over Σ is a tuple $\mathcal{A} = \langle \boldsymbol{\alpha}_\lambda, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}_{\sigma\in\Sigma}\rangle$ where $\boldsymbol{\alpha}_\lambda, \boldsymbol{\alpha}_\infty \in \mathbb{R}^n$ are the initial and final weights respectively, and $\mathbf{A}_\sigma \in \mathbb{R}^{n\times n}$ are the transition weights. The dimension n of these vectors and matrices is the *number of states* of the PSR. Though PSR is the usual name for this type of model in the reinforcement learning literature, they are also called *weighted finite automaton* (WFA) [Droste and Vogler, 2009] or *observable operator models* (OOM) [Jaeger, 2000]. This distinction reflects the fact that this same parametrization can be used to define models with different semantics, depending on the meaning associated with the values computed by the model.

A PSR $\mathcal{A}$ computes a function $f_\mathcal{A} : \Sigma^\star \to \mathbb{R}$ that assigns a number to each string $x = x_1 x_2 \cdots x_t \in \Sigma^\star$ as follows:

$$f_\mathcal{A}(x) = \boldsymbol{\alpha}_\lambda^\top \mathbf{A}_{x_1}\mathbf{A}_{x_2}\cdots\mathbf{A}_{x_t}\boldsymbol{\alpha}_\infty = \boldsymbol{\alpha}_\lambda^\top \mathbf{A}_x \boldsymbol{\alpha}_\infty \ .$$

In a PSR, a string $x \in \Sigma^\star$ represents a sequence of events produced by a dynamical system, and $f_\mathcal{A}(x)$ is the probability that the system produces x. In many cases of interest, $\Sigma = A \times O$, where A is a set of actions and O a set of observations, so any $x \in \Sigma^\star$ represents a sequence of action-observation pairs. The vector $\boldsymbol{\alpha}_\lambda$ represents the initial state of the system.

If a history $u \in \Sigma^\star$ has already been observed, the conditional probability of observing a sequence of events $v \in \Sigma^\star$ after u is:

$$f_{\mathcal{A},u}(v) = \frac{f_\mathcal{A}(uv)}{f_\mathcal{A}(u)} = \frac{\boldsymbol{\alpha}_\lambda^\top \mathbf{A}_u\mathbf{A}_v\boldsymbol{\alpha}_\infty}{\boldsymbol{\alpha}_\lambda^\top \mathbf{A}_u\boldsymbol{\alpha}_\infty} = \frac{\boldsymbol{\alpha}_u^\top \mathbf{A}_v\boldsymbol{\alpha}_\infty}{\boldsymbol{\alpha}_u^\top\boldsymbol{\alpha}_\infty} \ .$$

Hence, given some history u, the initial state $\boldsymbol{\alpha}_\lambda$ can be updated to a new state $\boldsymbol{\alpha}_u/(\boldsymbol{\alpha}_u^\top\boldsymbol{\alpha}_\infty)$, which allows computing probabilities of future observations conditioned on the current history. Because the partition of a sequence of observations x into a history u and a future v (also called *test*) yields $x = uv$, we sometimes call histories *prefixes* and futures *suffixes*.

2.3 Hankel and System Dynamics Matrices

The behavior of a stochastic dynamical system producing observations in a finite set Σ can be entirely characterized by the function $f : \Sigma^\star \to \mathbb{R}$ giving the probability $f(x)$ of observing each possible sequence of observations x. A convenient algebraic way to summarize all the information conveyed by f is with its *Hankel matrix*, a bi-infinite matrix $\mathbf{H}_f \in \mathbb{R}^{\Sigma^\star\times\Sigma^\star}$ with rows and columns indexed by strings in $\Sigma^\star$. Strings indexing rows and columns are interpreted as prefixes and suffixes respectively. The entries in $\mathbf{H}_f$ are given by $\mathbf{H}_f(u,v) = f(u,v)$ for every $u, v \in \Sigma^\star$.

Although $\mathbf{H}_f$ is an infinite matrix, in some cases it can have finite rank. In particular, a well-known result states that $\mathbf{H}_f$ has rank at most n if and only if there exists a PSR $\mathcal{A}$ with n states satisfying $f_\mathcal{A} = f$ [Carlyle and Paz, 1971; Fliess, 1974]. This result is the basis of recently developed spectral learning algorithms for PSRs [Boots *et al.*, 2011], which we review in Sec. 4.

Instead of looking at the full Hankel matrix, algorithms usually work with finite sub-blocks of this matrix. A convenient way to specify such blocks is to give the "names" to the rows and columns. Specifically, given a finite set of prefixes $\mathcal{P} \subset \Sigma^\star$ and a finite set of suffixes $\mathcal{S} \subset \Sigma^\star$, the pair $\mathcal{B} = (\mathcal{P}, \mathcal{S})$ is a *basis* defining the sub-block $\mathbf{H}_\mathcal{B} \in \mathbb{R}^{\mathcal{P}\times\mathcal{S}}$ of $\mathbf{H}_f$, whose entries are given by $\mathbf{H}_\mathcal{B}(u,v) = \mathbf{H}_f(u,v)$. Note that every sub-block built in this way satisfies $\mathrm{rank}(\mathbf{H}_\mathcal{B}) \leq \mathrm{rank}(\mathbf{H}_f)$; when equality is attained, the basis $\mathcal{B}$ is *complete*.

Sometimes it is also convenient to look at one-step shifts of the finite Hankel matrices. Let $\mathbf{H} \in \mathbb{R}^{\mathcal{P}\times\mathcal{S}}$ be a finite sub-block of $\mathbf{H}_f$ specified by a basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$. Then, for every symbol $\sigma \in \Sigma$, we define the sub-block $\mathbf{H}_\sigma \in \mathbb{R}^{\mathcal{P}\times\mathcal{S}}$ whose entries are given by $\mathbf{H}_\sigma(u,v) = \mathbf{H}_f(u,\sigma v)$. For a fixed basis, we also consider the vectors $\mathbf{h}_\mathcal{S} \in \mathbb{R}^\mathcal{S}$ with entries given by $\mathbf{h}_\mathcal{S}(v) = \mathbf{H}_f(\lambda, v)$ for every $v \in \mathcal{S}$, and $\mathbf{h}_\mathcal{P} \in \mathbb{R}^\mathcal{P}$ with $\mathbf{h}_\mathcal{P}(u) = \mathbf{H}_f(u,\lambda)$.

The Hankel matrix $\mathbf{H}_f$ is tightly related to the *system dynamics matrix* (SDM) of the stochastic process described by f [Singh *et al.*, 2004], but while the entries of the Hankel matrix represent *joint* probabilities over prefixes and suffixes, the corresponding entry in the SDM is the *conditional* probability of observing a suffix given the prefix. In particular, the SDM of f is the matrix $\bar{\mathbf{H}}_f \in \mathbb{R}^{\Sigma^\star\times\Sigma^\star}$ given by $\mathbf{H}_f$ with the rows scaled by the probability of prefixes. The SDM of f shares a lot of properties with its Hankel counterpart. In particular, spectral learning algorithms for f can be derived in terms of sub-blocks of both $\mathbf{H}_f$ and $\bar{\mathbf{H}}_f$. We will work with the Hankel matrices of f, but all our algorithms can easily be adapted to work with the SDM instead.

3 OPTION DURATION MODELS

We are interested in the dynamics of an agent interacting with an MDP M via a set of options Ω. Recall that in this setting the agent is not allowed to perform primitive actions, and options must be executed until termination. We are interested in situations in which the *duration* of an option is an informative statistic about the state of the MDP. That is, we would like to identify a state in M with the distribution over the durations of a sequence of options executed starting from that state. This section introduces a special PSR for this purpose that we call the *option duration model* (ODM). This is basically a dynamical model of an MDP viewed through the lens of a fixed set of options, in which the state is represented by a vector of parameters sufficient for predicting the duration of future options. In the following sections we present a spectral algorithm for

learning ODM from just information about the duration of options in an MDP, and explore the possibilities the state representation given by this ODM in order to plan with options in the original MDP.

The goal of this section is to understand what form the probability distributions over option durations take, and how to write these compactly in order to allow efficient learning from just duration information. To begin, assume we can reset the MDP M to a fixed state s_0 and explore the environment from there by performing a sequence of t options $\omega_1, \ldots, \omega_t$, e.g. chosen uniformly at random. From the point of view of the MDP dynamics, this process will generate a sequence of of state-action trajectories like the following:

$$\begin{aligned}
&\omega_1 : (s_0, \pi_{\omega_1}(s_0), s_1, \pi_{\omega_1}(s_1), \ldots, s_{T_1-1}, \pi_{\omega_1}(s_{T_1-1}), s_{T_1}) \\
&\omega_2 : (s_{T_1}, \pi_{\omega_2}(s_{T_1}), s_{T_1+1}, \pi_{\omega_2}(s_{T_1+1}), \ldots, \pi_{\omega_2}(s_{T_2-1}), s_{T_2}) \\
&\ldots : \ldots \\
&\omega_t : (s_{T_{t-1}}, \pi_{\omega_t}(s_{T_{t-1}}), \ldots, \pi_{\omega_t}(s_{T_t-1}), s_{T_t}) \ ,
\end{aligned}$$

where $\pi_\omega(s)$ denotes an action chosen according to the distribution given by ω's stochastic policy. Note that in this trace, the sequence of visited states, the actions executed, and the option stopping times form a stochastic dynamical process whose distribution can be computed in terms of the parameters of the MDP M, the parameters defining the options in Ω, and the sequence of options $\omega_1, \ldots, \omega_t$.

If one is only interested in the duration of the options executed in the above trace, then the interaction can be summarized in terms of termination and continuation events for the options being executed. That is, at each time step, we can ignore the current observed state and the action being chosen, and just focus on whether the current option terminates in the current state. We will use the symbol $\sharp$ (*sharp*) to denote continuation, and $\perp$ (*bottom*) to denote termination. Thus, at a given time step, we use $(\omega, \sharp)$ to denote option ω is being executed and does not terminate, and $(\omega, \perp)$ if it terminates. Using this notation, the "duration information" in the previous trajectory can be expressed as:

$$(\omega_1, \sharp)^{d_1-1}(\omega_1, \perp)(\omega_2, \sharp)^{d_2-1}(\omega_2, \perp)\ldots(\omega_t, \sharp)^{d_t-1}(\omega_t, \perp) \ ,$$

where the durations are given by $d_i = T_i - T_{i-1}$, $T_0 = 0$. In more compact form, we can write the duration information in a trace as a sequence of option-duration pairs: $(\omega_1, d_1)(\omega_2, d_2)\cdots(\omega_t, d_t)$, with $\omega_i \in \Omega$ and $d_i \in \mathbb{N} = \{1, 2, \ldots\}$. Both notations will be convenient in our derivations.

Broadly speaking, an option duration model for an MDP M and a set of options Ω is a mapping that associates with every state s in M a conditional probability distribution

$$\mathbb{P}_s[\bar{d} \mid \bar{\omega}] = \mathbb{P}_s[d_1, \ldots, d_t \mid \omega_1, \ldots, \omega_t]$$

that represents the stochastic process of stopping events induced by executing the options $\bar{\omega} = \omega_1, \ldots, \omega_t$ in M starting from s. More formally, let $\mathcal{D}_t$ denote the set of all[1] probability distributions over $\mathbb{N}^t$, and for every state s and $\bar{\omega} \in \Omega^t$ let $D^s_{\bar{\omega}} \in \mathcal{D}_t$ denote the distribution $\mathbb{P}_s[\cdot \mid \bar{\omega}]$. Now, by letting $\bar{\omega}$ run over all possible sequence of options in Ω^+ we can associate with every s a mapping $D^s_\bullet : \Omega^+ \to \cup_{t>0}\mathcal{D}_t$. We say that the set of options Ω is *rich* for MDP M if the maps $D^s_\bullet$ are different for all s, that is, if the functional $\Delta(s) = D^s_\bullet$ is injective, so for every $s, s' \in S$ there exists some sequence of options $\bar{\omega} \in \Omega^+$ such that $D^s_{\bar{\omega}} \neq D^{s'}_{\bar{\omega}}$. Clearly, in this case, $\Delta(s)$ uniquely identifies the state s, indicating that the duration over options provides a unique representation for each state.

If Δ is not injective, such models can still be sufficient for planning in special circumstances. For example, consider minimum-time-to-goal problems, in which fixed negative rewards are attributed per time step, and suppose the agent intends to plan using only options. In this case, states for which $\Delta(s) = \Delta(s')$ will also have the same optimal value function V^*_Ω (a result that follows directly from the way in which option models are defined [Sutton *et al.*, 1999]).

In next section we give a concrete realization of this abstract ODM by showing how to encode the collection of distributions $\Delta(s)$ using a PSR.

3.1 Representing Option Duration Models with PSR

We now show that the probability distributions of the timing of options can be compactly represented in the form of a PSR. Given $s \in S$ and an option $\omega \in \Omega$, we denote by $\delta(s, \omega)$ the random variable counting the number of steps until termination when following ω from s. Note that s might be an initial state for ω, but also a state traversed during the execution of ω, in which case $\delta(s, \omega)$ is the remaining number of steps until termination. Let $s_0 \in S$, $\omega = \langle I, \pi, \beta \rangle$, and $d > 0$ be an integer. We start by considering the probability $\mathbb{P}[\delta(s_0, \omega) = d]$, which is given by:

$$\sum_{\bar{s} \in S^d} \sum_{\bar{a} \in A^d} \mathbb{P}[s_0, a_0, s_1, a_1, \cdots, a_{d-1}, s_d, \perp] \ ,$$

where $\bar{s} = s_1 \cdots s_d$ is the sequence of states traversed by ω, $\bar{a} = a_0 \cdots a_{d-1}$ is a sequence of actions chosen by π_ω, and $\perp$ denotes the option termination. Note the appearance of $\perp$ at the end explicitly requires the option to last for exactly d steps; state trajectories which stop earlier are not

[1] We could be more precise and restrict ourselves to *discrete phase-type distributions* because these are enough to model the duration until absorption events in Markov chains; but this does not simplify our arguments at this point. The class of distributions we consider will become clear in Section 3.1

considered. By the Markov property,

$$\begin{aligned}
&\mathbb{P}[s_0, a_0, \cdots, s_{d-1}, a_{d-1}, s_d, \perp] \\
&= \left(\prod_{i=0}^{d-1} \mathbb{P}[a_i, s_{i+1} | s_i] \cdot \mathbb{P}[\sharp | s_{i+1}]\right) \cdot \frac{\mathbb{P}[\perp | s_d]}{\mathbb{P}[\sharp | s_d]} \\
&= \left(\prod_{i=0}^{d-1} \pi(s_i, a_i)\mathbb{P}(s_i, a_i, s_{i+1})(1 - \beta(s_{i+1}))\right) \cdot \frac{\beta(s_d)}{1 - \beta(s_d)} \ .
\end{aligned}$$

With some algebra, it can be shown that summing this expression over $\bar{s}$ and $\bar{a}$ yields:

$$\mathbb{P}[\delta(s_0, \omega) = d] = \mathbf{e}_{s_0}^\top \mathbf{A}_{\omega,\sharp}^{d-1} \mathbf{A}_{\omega,\perp} \mathbf{1} \ , \tag{1}$$

where $\mathbf{e}_{s_0} \in \mathbb{R}^S$ is an indicator vector with $\mathbf{e}_{s_0}(s) = \mathbb{I}[s = s_0]$, $\mathbf{A}_{\omega,\sharp} \in \mathbb{R}^{S \times S}$ is a matrix with $\mathbf{A}_{\omega,\sharp}(s, s') = \sum_{a \in A} \pi(s, a) P(s, a, s')(1 - \beta(s'))$, $\mathbf{A}_{\omega,\perp} \in \mathbb{R}^{S \times S}$ is a matrix with $\mathbf{A}_{\omega,\perp}(s, s') = \sum_{a \in A} \pi(s, a) P(s, a, s')\beta(s')$, and $\mathbf{1} \in \mathbb{R}^S$ is a vector of ones.

Using the ODM notation introduced in the previous section, Equation (1) can be written as

$$\mathbb{P}_{s_0}[d \mid \omega] = \mathbf{e}_{s_0}^\top \mathbf{A}_{\omega,\sharp}^{d-1} \mathbf{A}_{\omega,\perp} \mathbf{1} \ .$$

Now we would like to extend this expression to sequences containing more than one option. Using a similar derivation, for any $t > 0$, $\bar{d} \in \mathbb{N}^t$, and $\bar{\omega} \in \Omega^t$, we can compute $\mathbb{P}_{s_0}[\bar{d} \mid \bar{\omega}]$ as follows:

$$\mathbf{e}_{s_0}^\top \mathbf{A}_{\omega_1,\sharp}^{d_1-1} \mathbf{A}_{\omega_1,\perp} \mathbf{A}_{\omega_2,\sharp}^{d_2-1} \mathbf{A}_{\omega_2,\perp} \cdots \mathbf{A}_{\omega_t,\sharp}^{d_t-1} \mathbf{A}_{\omega_t,\perp} \mathbf{1} \ .$$

Note that the same reasoning applies if we consider distributions over states. That is, instead of a fixed initial state s_0, we consider a state sampled according to some distribution α_0 over S. In that case, we define the random variable $\delta(\alpha_0, \omega)$ representing the duration of option ω when started from $s_0 \sim \alpha_0$. By the same argument as above,

$$\mathbb{P}_{\alpha_0}[d \mid \omega] = \mathbb{P}[\delta(\alpha_0, \omega) = d] = \boldsymbol{\alpha}_0^\top \mathbf{A}_{\omega,\sharp}^{d-1} \mathbf{A}_{\omega,\perp} \mathbf{1} \ ,$$

where $\boldsymbol{\alpha}_0 \in \mathbb{R}^S$ is the vector representation of α_0. Again, this can be extended in the same way to sequences with more than one option. Therefore, we have proved the following result.

Theorem 1. *Let M be an MDP with n states, Ω a set of options, and $\Sigma = \Omega \times \{\sharp, \perp\}$. For every distribution α_0 over the states of M, there exists a PSR $\mathcal{A} = \langle \boldsymbol{\alpha}_0, \mathbf{1}, \{\mathbf{A}_\sigma\} \rangle$ with at most n states that computes the distributions over durations of options executed from a state sampled according to α_0.*

Note that the MDP transition kernel and the options' stochastic policies are coupled inside the transition matrices of PSR $\mathcal{A}$ representing the ODM. This coupling is the reason why we can model the timing of options in an MDP via a process with observation on the set $\Omega \times \{\sharp, \perp\}$ whose size is *independent* of the set of primitive actions A, and whose transitions operators have size at most $|S|$. Note that this can be particularly interesting in settings where the number of possible actions is very large but a small number of options is enough to specify the "useful" modes of operation of an agent.

3.2 ODM for Explorable Trajectories

Note that the ODM representation $\mathcal{A}$ given by Theorem 1 can be used to query the probability of observing any trajectory in $(\Omega \times \{\sharp, \perp\})^\star$ starting from a state sampled from $\boldsymbol{\alpha}$. In principle, this includes trajectories which are not valid for an agent interacting with an MDP via options – e.g. we can query the probability of trajectories of the form $(\omega_1, \sharp)^{d_1}(\omega_2, \sharp)^{d_2} \ldots$, where ω_1 was interrupted before termination. Note that this type of trajectories will never be observed by an agent that explores an environment by interacting with it only via options executed in call-and-return fashion. In particular, an agent does not need to learn about these trajectories if the goal is to plan via options only, and cannot hope to learn about these trajectories if it only explores the environment via options. We now show that a PSR representation for an ODM can be restricted to produce non-zero probabilities for legal trajectories only, without increasing the size of the model too much.

Recall that $\Sigma = \Omega \times \{\sharp, \perp\}$. When options are always executed to termination, valid trajectories belong to a subset of $\Sigma^\star$:

$$V = \left(\bigcup_{\omega \in \Omega} \{(\omega, \sharp)^\star (\omega, \perp)\}\right)^\star \ .$$

That is, each sequence of option-continuation events must end with an option-termination event before a new option can be started.

Now we want to modify the model in Theorem 1 so that it only assigns non-zero probabilities to trajectories in V. Let f be the function computed by $\mathcal{A}$. Then we want another PSR computing the function given by $\tilde{f}(x) = f(x)$ for $x \in V$ and 0 otherwise. We now show that $\tilde{f}$ can also be computed by a PSR with size close to that of $\mathcal{A}$.

Theorem 2. *If $\mathcal{A}$ has n states, then there exists a PSR $\tilde{\mathcal{A}}$ with at most $(|\Omega| + 1)n$ states computing $\tilde{f}$.*

Proof. By the Carlyle–Paz–Fliess theorem, it suffices to show that $\operatorname{rank}(\tilde{\mathbf{H}}) \leq (|\Omega| + 1)n$, where $\tilde{\mathbf{H}} \in \mathbb{R}^{\Sigma^\star \times \Sigma^\star}$ is the bi-infinite Hankel matrix associated with $\tilde{f}$. Let $\chi_V : \Sigma^\star \to \mathbb{R}$ be the characteristic function of the regular language V, which is given by $\chi_V(x) = 1$ if $x \in V$ and $\chi_V(x) = 0$ otherwise. Note that it is easy to construct a DFA with $|\Omega| + 1$ states recognizing V. Such a DFA can also be written as a PSR with $|\Omega| + 1$ states computing the function χ_V. Thus, the rank of the Hankel matrix $\mathbf{H}_V$ associated with χ_V is at most $|\Omega| + 1$. Now note that we

can write $\tilde{\mathbf{H}} = \mathbf{H} \circ \mathbf{H}_V$, where $\mathbf{H}$ is the Hankel matrix of f and $\circ$ denotes entry-wise multiplication (also known as Hadamard matrix product). Finally, we use an elementary bound on the rank of Hadamard products to conclude that

$$\text{rank}(\tilde{\mathbf{H}}) \leq \text{rank}(\mathbf{H}) \cdot \text{rank}(\mathbf{H}_V) = (|\Omega| + 1)n \ . \quad \square$$

Note that the bound on the number of states of $\tilde{\mathcal{A}}$ is in general not tight. For example, in our experiments with a grid-like environment, we observed that starting with an MDP with n states the true rank of $\tilde{\mathbf{H}}$ is on the order of $O(\sqrt{n})$. So the PSR representation of ODM can lead to more compact models than the underlying MDP, and can potentially lead to faster learning and planning algorithms. This issue should be investigated further in future work.

4 SPECTRAL LEARNING OF ODM

The PSR representation of ODM given by the theorems in previous section can be computed explicitly if the underlying MDP is known exactly. In this section we explore the possibility of learning such a representation from data about option durations, without exploiting any knowledge about the underlying MDP. The rationale behind this approach is that it can lead to more compact representations that do not depend on the set of primitive actions, and learn only the parts of the MDP's state space which are reachable via options.

Because an option duration model over valid trajectories can be represented with a PSR of moderate size, we can use the spectral learning algorithm in [Boots *et al.*, 2011] to estimate an ODM from a set of trajectories in $\Sigma^\star$ produced by the agent exploring the MDP using a fixed policy over options. For each trajectory, the initial state is sampled according to a fixed distribution α. We assume that the options executed by the agent are selected according to some fixed open-loop policy. This is important if we want to use the sampled trajectories to learn a model of the environment which is independent of the exploration policy.

The algorithm takes as input Σ and a basis $\mathcal{B}$ in $\Sigma^\star$, uses them to estimate the corresponding Hankel matrices, and then recovers a PSR by performing singular value decomposition and linear algebra operations on these matrices. Although the method works almost out-of-the-box, in practice the results tend to be sensitive to the choice of basis. Thus, after briefly recapitulating how the spectral learning algorithm proceeds, we will devote the rest of the section to describe a procedure for building a basis which is tailored for the case of learning option duration models.

Suppose the basis $\mathcal{B}$ is fixed and the desired number of states n is given. Suppose that a set of sampled trajectories was used to estimate the Hankel matrices $\mathbf{H}, \mathbf{H}_\sigma \in \mathbb{R}^{\mathcal{P} \times \mathcal{S}}$ and vectors $\mathbf{h}_\mathcal{P} \in \mathbb{R}^\mathcal{P}$, $\mathbf{h}_\mathcal{S} \in \mathbb{R}^\mathcal{S}$ defined in Sec. 2.3. The algorithm starts by taking the truncated SVD $\mathbf{U}_n \mathbf{D}_n \mathbf{V}_n^\top$ of $\mathbf{H}$, where $\mathbf{D}_n \in \mathbb{R}^{n \times n}$ contains the first n singular values of $\mathbf{H}$, and $\mathbf{U}_n \in \mathbb{R}^{\mathcal{P} \times n}$ and $\mathbf{V}_n \in \mathbb{R}^{\mathcal{S} \times n}$ contain the first left and right singular vectors respectively. Finally, we compute the transition operators of a PSR as $\mathbf{A}_\sigma = \mathbf{D}_n^{-1} \mathbf{U}_n^\top \mathbf{H}_\sigma \mathbf{V}_n$, and the initial and final weights as $\boldsymbol{\alpha}_\lambda^\top = \mathbf{h}_\mathcal{S}^\top \mathbf{V}_n$ and $\boldsymbol{\alpha}_\infty = \mathbf{D}_n^{-1} \mathbf{U}_n^\top \mathbf{h}_\mathcal{P}$. This yields a PSR with n states. It was proved in [Boots *et al.*, 2011] this algorithm is statistically consistent: if the population Hankel matrices are known and the basis $\mathcal{B}$ is complete, then the learned PSR is equivalent to the one that generated the data.

4.1 Basis Selection

Finding a complete basis for a given function $f : \Sigma^\star \to \mathbb{R}$ is in general a hard combinatorial problem: although if $\mathbf{H}_f$ has rank n, there exists a complete basis with n prefixes and suffixes, the search space where these lie is doubly-exponential in n. To overcome this problem, randomized and greedy basis selection heuristics have been proposed in the past [Wiewiora, 2005; Balle *et al.*, 2012]. Greedy selection is in general computationally expensive because it requires re-learning the model every time a new element is added to the basis. Randomized search does not work in our case because it cannot take into account the fact that we have syntactic constraints on the valid trajectories. Therefore, we designed a procedure that takes the structure of our problem into account.

Our procedure is parametrized by various quantities depending on the environment and the observed trajectories: the maximum possible duration T_ω of each option $\omega \in \Omega$ in our training dataset, and $T = \max_\omega T_\omega$; an upper bound $K_r \geq 1$ on the number of option executions needed to reach every possible state in M when initial states are sampled from α; and, an upper bound $K_d \geq 1$ on the number of option executions needed to distinguish/disambiguate every pair of states in M in terms of option duration information.

We assume these quantities are given, either because they can be derived from prior knowledge of the application domain, or because they are cross-validated. The definitions imply that T and K_r are a function of the geometry of the environment and how this relates to the available options. On the other hand, K_d measures the statistical similarity between states in M when seen through the lens of Ω. The intuition is simple: we want to ensure that we have enough prefixes in the Hankel matrix to get to all reachable states in M, and enough suffixes to make sure each of these states can be distinguished from each other. The following construction formalizes this intuition.

We obtain the set $\mathcal{P}$ of prefixes in the basis as the union of

two disjoint sets. The first set is:

$$\mathcal{P}_\perp = \big\{x_1 x_2 \cdots x_t \;\big|\; 1 \le t \le K_r, x_i = (\omega_i, \sharp)^{d_i}(\omega_i, \perp), \omega_i \in \Omega, 0 \le d_i \le T_{\omega_i}\big\} \quad .$$

These are trajectories with at most K_r option executions, containing all possible sequences of options, and all possible option durations allowed by the model. A simple calculation shows that $|\mathcal{P}_\perp| = O(K_r T^{K_r} |\Omega|^{K_r})$. Note that each trajectory in $\mathcal{P}_\perp$ terminates with a symbol $(\omega, \perp)$. If we remove this last symbol for each trajectory, we obtain a disjoint set of prefixes $\mathcal{P}_\sharp = \{x \mid x(\omega, \perp) \in \mathcal{P}_\perp\}$. Then we take $\mathcal{P} = \mathcal{P}_\perp \cup \mathcal{P}_\sharp$. Note that again we have $|\mathcal{P}| = O(K_r T^{K_r} |\Omega|^{K_r})$.

Similarly, the set of suffixes will be obtained as the union of two sets. These are defined as follows:

$$\mathcal{S}_\sharp = \big\{x_1 x_2 \cdots x_t \;\big|\; 1 \le t \le K_s, x_i = (\omega_i, \sharp)^{d_i}(\omega_i, \perp), \omega_i \in \Omega, 0 \le d_i \le T_{\omega_i}\big\} \quad ,$$

$$\mathcal{S}_\perp = \{(\omega, \perp)x \mid x \in \mathcal{S}_\sharp, \omega \in \Omega\} \quad .$$

The suffixes in $\mathcal{S}_\sharp$ are obtained like the prefixes in $\mathcal{P}_\perp$, with the only difference that the number of option executions is now upper bounded by K_s instead of K_r. Suffixes in $\mathcal{S}_\perp$ are obtained by prefixing each string in $\mathcal{S}_\sharp$ with each possible option termination symbol $(\omega, \perp)$. The whole set of suffixes is $\mathcal{S} = \mathcal{S}_\perp \cup \mathcal{S}_\sharp$, and we have $|\mathcal{S}| = O(K_s T^{K_s} |\Omega|^{K_s+1})$.

Note that not every string in $\mathcal{P}\mathcal{S}$ defines a valid trajectory. This is required for the Hankel matrices $\mathbf{H}_\sigma$ to be different from zero; otherwise the operators $\mathbf{A}_\sigma$ cannot be correctly recovered. To see why this is the case, consider the basis $\mathcal{B}' = (\mathcal{P}_\perp, \mathcal{S}_\sharp)$. By construction we have $\mathcal{P}_\perp \mathcal{S}_\sharp \subset V$. However, if we consider a system where some ω never lasts for just one step, then every trajectory in $\mathcal{P}_\perp\{(\omega, \perp)\}\mathcal{S}_\sharp$ has probability zero. In particular, in such a system the matrix $\mathbf{H}_\sigma$ over the basis $\mathcal{B}'$ is exactly zero. To work around this problem, it is necessary to introduce non-valid trajectories in the basis, to ensure that $\mathbf{H}$ will contain some sub-blocks filled with zeros, but the $\mathbf{H}_\sigma$ will contain some non-zero sub-blocks.

4.2 Scalability

Though at first glance the exponential dependence on K_r and K_s in the size of the basis may seem prohibitive, in practice the situation is much better. On one hand, these are worst-case bounds that assume all possible sequences of options and durations need to be considered. However, in practice some of those might never be observed — e.g. because the duration of certain options has little variance, or because some options are not available in many configurations. On the other hand, as already mentioned, we have prior knowledge indicating that a significant fraction of the entries appearing in the resulting Hankel matrices will be zero because they correspond to non-valid trajectories. This will lead to highly sparse matrices which can benefit from sparse representations, sparse singular value decompositions, and sparse numerical linear algebra in general. In those cases, the complexity of the operations involved will not depend on the size of these matrices, but instead on the number of non-zero entries. This quantity will in practice be much smaller that the size of the Hankel matrices given by our choice of basis, and we have observed such benefits in our experiments.

5 PLANNING WITH ODM

In this section we want to briefly investigate the potential utility of ODM representations for planning with options. Specifically, we want to show how to use ODMs as a form of state representation for value function learning. In a nutshell, our main result states that nothing is lost by planning with options using a PSR representation of an ODM instead of the original MDP representation.

The first important observation is that the construction in Theorem 1 (and in consequence, the construction in Theorem 2) leads to a PSR with identical transition operators $\{\mathbf{A}_\sigma\}$ regardless of which initial distribution over the states of M is chosen. That is, the only thing that needs to be modified when we change the initial distribution over the states is the initial state in the PSR. Thus, given an MDP M and a set of options Ω, we can fix the transition structure of a PSR representation for the corresponding ODM – either over all trajectories or only valid trajectories (the distinction is not important for the purpose of this section). These will be given by $\{\mathbf{A}_\sigma\}$ for $\sigma \in \Sigma = \Omega \times \{\sharp, \perp\}$, and the corresponding final weights $\boldsymbol{\alpha}_\infty$. We denote by n' the dimension of this PSR. Then we can map each possible distribution α over the states of M to a PSR state $\boldsymbol{\theta}_\alpha \in \mathbb{R}^{n'}$. Note that although this mapping is straightforward for the construction in Theorem 1, this is not the case for the construction in Theorem 2, and even less for the PSR recovered by the learning algorithm in Section 4 because such a PSR is only equivalent to the original up to conjugation of the transition weights by an invertible matrix [Boots *et al.*, 2011]. In addition to distributions over initial states, we can also consider PSR states obtained by the state-update procedure. That is, if we are given a valid trajectory $u \in V$ representing a history of options and associated continuation and termination events, then we can obtain the updated state

$$\boldsymbol{\theta}_{\alpha,u}^\top = \frac{\boldsymbol{\theta}_\alpha^\top \mathbf{A}_u}{\boldsymbol{\theta}_\alpha^\top \mathbf{A}_u \boldsymbol{\alpha}_\infty} \quad .$$

This represents another probability distribution over the states of M. Note that by construction we have $\boldsymbol{\theta}_{\alpha,\lambda} = \boldsymbol{\theta}_\alpha$. With this notation we can show that the state-option value function of any policy over options is linear in the PSR state.

Theorem 3. *Let $\pi_\Omega : S \times \Omega \to [0,1]$ be a stochastic stationary policy over options on the MDP M. For every $\omega \in \Omega$ there exists a vector $\boldsymbol{\rho}_\omega \in \mathbb{R}^{n'}$ so that for every distribution α over states in M and every history $u \in V$, we have $\mathbb{E}_s[Q^{\pi_\Omega}(s,\omega)] = \boldsymbol{\theta}_{\alpha,u}^\top \boldsymbol{\rho}_\omega$, where the expectation is over states s sampled from the distribution induced by observing u after starting in a state drawn from α.*

The above theorem includes as a special case the situation in which the agent is at a fixed state in M; i.e. no uncertainty. The proof follows along the lines of a similar result known for PSRs with reward structure induced by a corresponding POMD [James *et al.*, 2004]. The key difference is that we need to take into account the semi-Markov nature of option termination events, which make the computations involved in the discount factors and extended transition kernels more involved. Though the details are not complicated, the algebra is too lengthy to be included in this version and will be presented in full in an extended version.

Although planning with PSRs has been studied using methods akin to POMDP planning [James *et al.*, 2004; Izadi and Precup, 2008; Boots *et al.*, 2011], we chose a more efficient alternative, the Fitted-Q Iteration (FQI) algorithm of Ernst *et al.* [2005]. Similar uses of FQI with PSR states have been proposed in Ong *et al.* [2012]; Hamilton *et al.* [2013]. FQI learns a policy through an iterative re-fitting process over batches of sampled transitions of the form: $\langle s, a, r, s' \rangle$. At every iteration, an ensemble of extremely randomized trees (ExtraTrees) [Geurts *et al.*, 2006] is fit to $Q(s,a)_t = r + \gamma \max_{a'} Q_{t-1}(s', a')$ over all quadruples s, a, r, s'. In order to ensure convergence, the tree structure must be kept fixed across iterations. Hence, we fit the regressor in the first iteration and only refreshed the values and not the structure on subsequent iterations. We plan directly over the ODM state vector updated at each step with the corresponding operator (continuation or termination). So, in the first step, we compute $\boldsymbol{\theta}_0 = \boldsymbol{\alpha}_\lambda^\top \mathbf{A}_{\sigma_0}$, then update $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1}^\top \mathbf{A}_{\sigma_t}$. The ODM state vector is then normalized in order to prevent difficulties with the regressor. This approach is leveraging a well-known planning approach, so it is straightforward. It is possible that the structure of ODM can be exploited further to design more efficient planning algorithms. We leave this topic for future work.

6 EXPERIMENTS

We first assess the performance of the spectral learning algorithm with our data-driven basis construction method for $K_r = 2$ and $K_s = 1$. We use a 4-connected grid with four actions representing the cardinal directions (NEWS). Unless the current state is a "wall" each action moves the agent one step in the specified direction with probability 0.9, and maintains the current state with probability 0.1. We also define one option for each cardinal direction. These options take as many steps as possible in the specified direction until they hit a wall, at which point the option terminates. A uniformly random exploration policy is used for sampling 10000 episodes in which five options are executed to termination. We also collected a test set consisting of 10000 trajectories of 8 options sequences. We evaluate the prediction accuracy by computing the relative error over the estimated remaining number of steps in the currently executing option. For each test trajectory, we picked a time index uniformly at random and conditioned the learned ODM on the history up to this point. These random split points were then kept fixed throughout all evaluations. Figure 1b shows that the prediction accuracy increases with the dimension of the ODM. More samples also allow for better predictions. Since the prediction task is inherently stochastic, even the true ODM cannot achieve 0 relative error.

6.1 Planning in a grid world

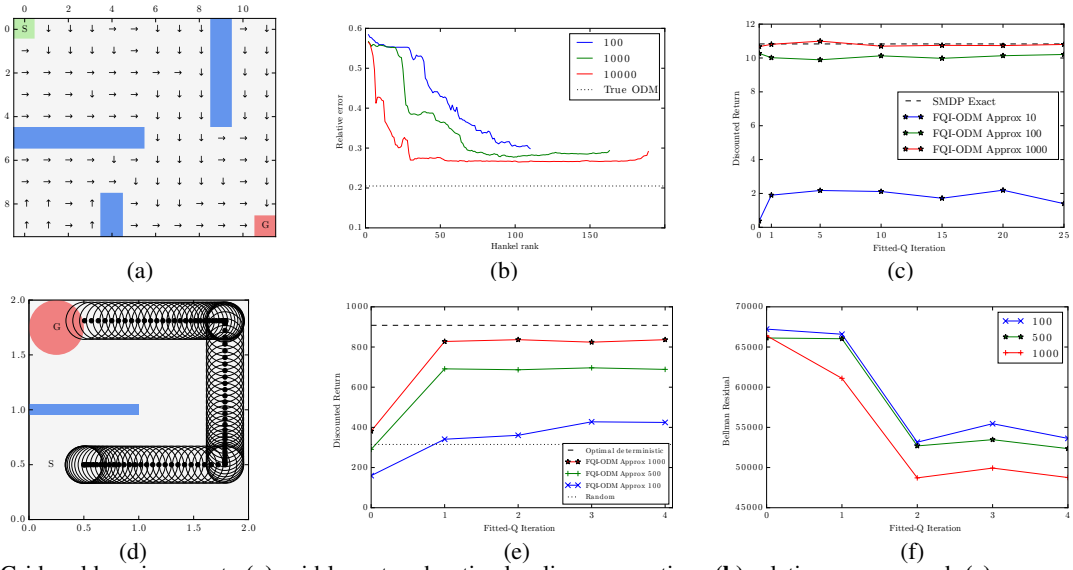
We formulating the grid-world problem by giving a reward of +100 for entering the bottom-right state and -10 per collision, with discount factor 0.9. A dataset of 1000 trajectories of 8 options each was collected with a uniformly random policy over options. For each curve in figure 1, we use our dataset to simultaneously learn an ODM and plan over it. We evaluate the performance of the greedy policy by taking 100 Monte-Carlo estimates in the simulated environment. Given the true underlying MDP and a set of options, we compute the resulting Semi-Markov Decision Process (SMDP) (see p. 26 of Sutton *et al.* [1999]) and solve it using value iteration, to obtain the baseline (optimal achievable return) for our evaluation. We extended the ExtraTrees implementation of Pedregosa *et al.* [2011] with the freezing mechanism required for FQI. We used all available features for splitting internal nodes and fitted an ensemble of 10 trees with maximum depth of 5. Figure 1c shows that an optimal policy can be obtained using 1000 trajectories and one planning step.

6.2 Planning with a simulated robot

Using the same methodology, we experimented with the proposed ODM learning and planning approach in a simulated robot environment with continuous state space, nonlinear dynamics, and actions that correspond to real actuators of the robot. We leverage the simulation capabilities of the 2D physics engine Box2D [2] to obtain realistic accelerations, collisions and friction effects.

We set the density of a circular differential wheeled robot to 3.0 and its radius to 0.17 cm. A primitive action in this domain consists in the application of force vector of $(0.1, 0.1)$ on both wheels every $1/10$ of a simulated second. At the beginning of every episode, the robot is initialized at position $(0.5, 0.5)$ of a 2x2 meters environment.

[2] `http://box2d.org`



(a) (b) (c) (d) (e) (f)

Figure 1: Gridworld environment: **(a)** grid layout and optimal policy over options **(b)** relative error vs rank **(c)** average discounted cumulative return. Simulated robot environment: **(c)** trajectory of an optimal policy **(d)** average discounted cumulative reward **(f)** mean square Bellman residual

A 10cm thick wall separates the robot from the goal location at $(0.25, 0.75)$. The episode finishes when the robot is within 25cm of the target.

As in the grid-world problem, we define a set of options over four radial directions. The termination condition triggers when the front-facing distance sensor detects an obstacle 6cm ahead. Because of the nonlinear dynamics, the hand-defined options controller does not always stop exactly within this distance. Given enough acceleration, the robot might sometimes touch the walls. This additional source of noise due to an imperfect controller is interesting from an experimental point of view. A stochastic component also makes the forward action ineffective 5% of the time. The rewards are 1000 for reaching the goal region and -10 by collision. Taking a primitive action incurs no cost but there is a discount factor $\gamma = 0.999$.

We collected 2000 trajectories of at most 10 options with a uniformly random policy over options. Due to the size of the environment, we found that trajectories of at most 5 options were insufficient to consistently learn good policies. We used $K_r = 2, K_s = 1$ for the basis, considered all features for node splitting and used 10 trees of maximum depth 8. The results presented in Figure 1e were obtained by resampling 10 times the original dataset without replacement, to characterize the stability of the learning algorithm. The greedy policy derived from FQI was then evaluated with 10 Monte-Carlo rollouts. Since the underlying MDP is unknown, we could not compute the exact optimal policy, so we upper-bounded the achievable return through the policy in Figure 1d under 100% success rate for the forward command. We used the average return of a uniform random policy as baseline. As expected, the mean square Bellman residual [Ernst *et al.*, 2005] decreases for larger sample sizes and over longer planning horizon (Fig. 1f). While 100 episodes yield ODM policies slightly better than random, 1000 episodes are sufficient to recover a close-to-optimal solution (Fig. 1e).

7 DISCUSSION

The approach we presented learns a predictive model for option durations, and we illustrates its use in robot navigation tasks. As discussed, timing models are simple, yet in many problems they are sufficient for good quality planning. To see why having simpler models might be useful, consider the prevalence of bandit problems in ad placement; the task could be done better with full MDPs, data efficiency is more important for the application. Similarly, we believe that being able to exploit simple yet efficient timing models is interesting and important. The use of this type of model in planning algorithms should be investigated further. Models of states in terms of what happens after executing certain actions are known to be useful, e.g. Dayan [1993]. But our models are simplified, hence different both from action-respecting embeddings Bowling *et al.* [2005, 2007] and predictive state representations with options Wolfe and Singh [2006], which aim to learn full observation models conditioned on extended actions, in order to characterize the current state. Timing models get around the problem of both large action spaces (by using a finite set of options) and the problem of large observation spaces (by focusing only on continuation and termination). A theoretical analysis of the error of planning with timing models instead of true transition models is left for future work. We also aim to study the sample efficiency of this approach.

References

B. Balle, A. Quattoni, and X. Carreras. Local loss optimization in operator models: A new insight into spectral learning. *International Conference on Machine Learning (ICML)*, 2012.

B. Boots, S. Siddiqi, and G. Gordon. Closing the learning planning loop with predictive state representations. *International Journal of Robotic Research*, 2011.

M. Bowling, A. Ghodsi, and D. Wilkinson. Action respecting embedding. *International Conference on Machine Learning (ICML)*, 2005.

M. Bowling, D. Wilkinson, A. Ghodsi, and A Milstein. Subjective localization with action respecting embedding. *Robotics Research*, 2007.

J. W. Carlyle and A. Paz. Realizations by stochastic finite automata. *Journal of Computer Systems Science*, 1971.

P. Dayan. Improving generalisation for temporal difference learning: The successor representation. *Neural Computation*, 1993.

W. Droste, M. Kuich and H. Vogler. *Handbook of weighted automata*. Springer, 2009.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *J. Mach. Learn. Res.*, 6:503–556, December 2005.

M. Fliess. Matrices de Hankel. *Journal de Mathématiques Pures et Appliquées*, 1974.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Mach. Learn.*, 63(1):3–42, April 2006.

J. J. Gibson. The theory of affordances. In R. Shaw and J. Bransford, editors, *Perceiving, Acting, and Knowing*, 1977.

William L. Hamilton, Mahdi M. Fard, and Joelle Pineau. Modelling sparse dynamical systems with compressed predictive state representations. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 178–186, 2013.

Masoumeh T. Izadi and Doina Precup. Point-based planning for predictive state representations. In *Advances in Artificial Intelligence*, volume 5032 of *Lecture Notes in Computer Science*, pages 126–137. Springer Berlin Heidelberg, 2008.

H. Jaeger. Observable operator models for discrete stochastic time series. *Neural Computation*, 2000.

Michael R James, Satinder Singh, and Michael L Littman. Planning with predictive state representations. In *Machine Learning and Applications, 2004. Proceedings. 2004 International Conference on*, pages 304–311. IEEE, 2004.

M.L. Littman, R.S. Sutton, and S. Singh. Predictive representations of state. *Neural Information Processing Systems (NIPS)*, 2002.

A. Machado, M. T. Malheiro, and W. Erlhagen. Learning to time: A perspective. *Journal of the Experimental Analysis of Behavior*, 2009.

Sylvie C.W. Ong, Yuri Grinberg, and Joelle Pineau. Goal-directed online learning of predictive models. In Scott Sanner and Marcus Hutter, editors, *Recent Advances in Reinforcement Learning*, volume 7188 of *Lecture Notes in Computer Science*, pages 18–29. Springer Berlin Heidelberg, 2012.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

M. L. Puterman. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

M. Rosencrantz, G. Gordon, and S. Thrun. Learning low dimensional predictive representations. *International Conference on Machine Learning (ICML)*, 2004.

S. Singh, M. R. James, and M. R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. *Uncertainty in Artificial Intelligence (UAI)*, 2004.

J. Stober, R. Miikkulainen, and B. Kuipers. Learning geometry from sensorimotor experience. *Joint Conference on Development and Learning and Epigenetic Robotics*, 2011.

R. S Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999.

E. Wiewiora. Learning predictive representations from a history. *International Conference on Machine Learning (ICML)*, 2005.

B. Wolfe and S. Singh. Predictive state representations with options. *International Conference on Machine Learning (ICML)*, 2006.