Robotics: Science and Systems 2019
Freiburg im Breisgau, June 22-26, 2019

Reachable Space Characterization of Markov Decision Processes with Time Variability

Junhong Xu[†], Kai Yin[†], Lantao Liu

Abstract—**We propose a solution to a time-varying variant of Markov Decision Processes which can be used to address decision-theoretic planning problems for autonomous systems operating in unstructured outdoor environments. We explore the time variability property of the planning stochasticity and investigate the state reachability, based on which we then develop an efficient iterative method that offers a good trade-off between solution optimality and time complexity. The reachability space is constructed by analyzing the means and variances of states' reaching time in the future. We validate our algorithm through extensive simulations using ocean data, and the results show that our method achieves a great performance in terms of both solution quality and computing time.**

I. Introduction

Autonomous vehicles that operate in the air and water are easily disturbed by stochastic environmental forces such as turbulence and currents. The motion planning in such uncertain environments can be modeled using a *decision-theoretic planning* framework where the substrate is the Markov Decision Processes (MDPs) [40] and its partially observable variants [22].

To cope with various sources of uncertainty, the prevalent methodology for addressing the decision-making of autonomous systems typically takes advantage of the known characterization such as probability distributions for vehicle motion uncertainties. However, the characterization of the uncertainty can vary in time with some extrinsic factors related to environments such as the time-varying disturbances in oceans as shown in Fig. 1. Despite many successful achievements [42, 28, 19, 31, 20], existing work typically does not synthetically integrate environmental variability with respect to time into the decision-theoretic planning model.

Although it can be easily recognized that the time-varying stochastic properties represent a more general description of the uncertainty [43], addressing the related planning problems is not an incremental extension to the basic time-invariant counterpart methods. This is because many components in the basic time-invariant model become time-varying terms simultaneously, requiring substantial re-modeling work and a completely new solution design. Therefore, in this work we explore the time variability of the planning stochasticity and investigate the state reachability, and these important properties allow us to gain insights for devising an efficient approach with a good trade-off between the solution optimality and the time complexity. The reachability space is constructed essentially by analyzing the first and second moments of expected future states' reaching time. Finally, we validate our method in the scenario of navigating a marine vehicle in the ocean, and our simulation results show that the proposed approach produces results close to the optimal solution but requires much smaller computational time comparing to other baseline methods.

Fig. 1: A motivating scenario: the ocean environment is highly dynamic. Autonomous vehicle planning needs to consider the strong time-varying disturbances caused by ocean currents. (Source: NASA Scientific Visualization Studio.)

II. Related Work

Most existing methods that solve MDPs in the literature assume time-invariant transition models [40]. Such an assumption has also been used in most literature on reinforcement learning (RL) [39, 9, 24], where an agent tries to maximize accumulated rewards by interacting with its environment. A technique related to temporal analysis is temporal difference (TD) learning [38, 41]. RL extends the TD technique by constantly correcting previous predictions based on the availability of new observations [39, 5]. Unfortunately, existing TD techniques are typically built on time-invariant models. Research on time variability has also been conducted in multi-agent co-learning scenarios where multiple agents learn in the environment simultaneously and enable the transition of the environment to evolve over time [18, 44, 12]. To tackle the environmental uncertainty, multiple stationary MDPs have been utilized to obtain better environmental representations [2, 17, 8]. Policies are learned for each MDP and then switched if a context change is detected. In general, these approaches use a piecewise stationary approximation, i.e., a time-invariant MDP is applied in each time period of a multi-period horizon.

[†] Authors contributed equally.

J. Xu and L. Liu are with the School of Informatics, Computing, and Engineering at Indiana University, Bloomington, IN 47408, USA. E-mail: {xu14, lantao}@iu.edu. K. Yin is with HomeAway, Inc. E-mail: yinkai1000@gmail.com.

Instead of directly modeling time-varying transition functions, exogenous variables outside of MPDs may be leveraged to characterize the variation of transition functions [25]. Some relevant work employs partially observable MDPs to deal with non-stationary environments [7, 11], or uses Semi-Markov Decision Processes (SMDPs) to incorporate continuous action duration time in the model [37, 33, 23, 4]. However, these frameworks still essentially assume time-invariant transition models.

Recently, the time-dependent MDP has been analyzed where the space-time states are directly employed [6]. It has been shown that even under strong assumptions and restrictions, the computation is still prohibitive [34]. Relevant but different from this model, the time-varying MDP is formulated by directly incorporating time-varying transitions whose distribution can be approximated or learned from environments [27]. This method is compared with our proposed algorithm in the experiment section.

Reachability analysis has been vastly researched in the control community where the majority of work falls in non-stochastic domains. For example, an important framework utilizes Hamilton-Jacobi reachability analysis to guarantee control policies to drive the dynamical systems within some predefined safe set of states under bounded disturbances [32, 3]. Control policies can also be learned through machine learning approaches to keep the system outside of unsafe states [13, 14]. In addition, convex optimization procedures are carried out to compute reachable funnels within which the states must remain under some control policy [30]. These funnels are then used to compute robot motions online with a safety guarantee.

III. Problem Formulation

We are motivated by problems that autonomous vehicles (or manipulators) move toward defined goal states under exogenous time-varying disturbances. These problems can be modeled as a time-varying Markov Decision Process.

A. Time-Varying Markov Decision Process

We represent a time-varying Markov Decision Process (TVMDP) as a 5-tuple $(\mathbb{S}, \mathbb{T}, \mathbb{A}, \mathcal{T}, R)$. The *spatial* state space $\mathbb{S}$ and the action space $\mathbb{A}$ are finite. We also need an extra discrete *temporal* space $\mathbb{T}$ in our discussions, though our framework supports continuous time models. Because the MDP now is time-varying, an action must depend on both state and time; that is, for each state $s \in \mathbb{S}$ and each time $t \in \mathbb{T}$, we have a feasible action set $A(s,t) \subset \mathbb{A}$. The entire set of feasible state-time-action triples is $\mathbb{F} := \{(s,t,a) \in \mathbb{S} \times \mathbb{T} \times \mathbb{A}\}$. There is a probability transition law $\mathcal{T}(s,t,a;\cdot,\cdot)$ on $\mathbb{S} \times \mathbb{T}$ for all $(s,t,a) \in \mathbb{F}$. Thus, $\mathcal{T}(s,t,a;s',t')$ specifies the probability of transitioning to state s' at a future time t' given the current state s and time t $(t \leqslant t')$ under action a. The final element $R : \mathbb{F} \to \mathbb{R}^1$ is a real-valued reward function that depends on the state-time-action triple.

Comparing with the classic MDP representation, TVMDP contains an additional time space $\mathbb{T}$; the transition law $\mathcal{T}$ and the reward function depend on the state, action, and time. Therefore, the major difference between TVMDPs and classic MDPs is that the transition probability and reward function are time-dependent.

We consider the class of deterministic Markov policies [33], denoted by Π; that is, the mapping $\pi \in \Pi : \mathbb{S} \times \mathbb{T} \to \mathbb{A}$ depends on the current state and the current time, and $\pi(s,t) \in A(s,t)$ for a given state-time pair. The initial state $s_0 \in \mathbb{S}$ at time $t = 0$ and policy $\pi \in \Pi$ determine a stochastic trajectory (path) process $\tau := \{(s_t, a_t)\}_{t \geqslant 0}$. The two terms, path and trajectory, will be used interchangeably. For a given initial state s_0 and starting time t_0, the expected discounted total reward is represented as:

$$v^\pi(s_0, t_0) = \mathbb{E}^\pi_{s_0,t_0}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, t_k, a_k)\right], \tag{1}$$

where $\gamma \in [0,1)$ is the discount factor that discounts the reward at a geometric decaying rate. Our aim is to find a policy π^* to maximize the total reward from the starting time t_0 at the initial state s_0, i.e.,

$$\pi^*(s_0, t_0) = \arg\max_{\pi \in \Pi} v^\pi(s_0, t_0). \tag{2}$$

Accordingly, the optimal value is denoted by $v^*(s_0, t_0)$.

B. Passage Percolation Time

The 2D plane in which the autonomous system operates is modeled as the spatial state space. The plane is discretized into grids, and the center of each grid represents a state. Two spatial states are connected if their corresponding grids are neighbors, i.e. a vehicle in one grid is able to transit to the other grid directly without passing through any other grids. Because the vehicle motion follows physical laws (e.g., motion kinematics and dynamics), travel time is required for the vehicle to transit between two different states. Let $h(s,t,s')$ be the local transition time for a vehicle to travel from state s at time t to a connected state s'. Such a local transition time is time-dependent and is assumed deterministic.

If, however, the two states s and s' are not connected, then the transition time between them is a random variable and depends on the trajectory of the vehicle. For any finite path $\tau = \{(s, a_0), ..., (s', a_n)\}_{n>0}$ starting from time u at state s and ending with state s', we define the *Passage Percolation Time* (PPT) between s and s' to be

$$H^u_{s,s'} = \sum_{(s_k,a_k) \in \tau} h(s_k, t_k, s_{k+1}), \tag{3}$$

where $t_0 = u, t_k = t_{k-1} + h(s_{k-1}, t_{k-1}, s_k)$ and $0 < k < n$. In addition, we require $H^u_{s,s} = 0$. By definition it is the transition time (travel time) on a path from the state s at time u until firstly reaching the state s' [1, 15]. If the local transition time h does not depend on time, the definition of Eq.(3) is exactly the same as the conventional *passage time* for percolation [1]. We would like to emphasize that $H^u_{s,s'}$ is a random variable, which relies on the realized path between s and s'. Under the policy π, the mean and variance of the PPT are assumed to exist, and are denoted by $\mu^{\pi}_{s,u;s'} := \mathbb{E}^{\pi}[H^{u}_{s,s'}]$ and $(\sigma^{\pi}_{s,u;s'})^2 = \mathbb{V}^{\pi}[H^{u}_{s,s'}]$, respectively.

C. Spatiotemporal State Space Representation

One can view a TVMDP as a classic MDP by defining the product of both *spatial* and *temporal* spaces $\mathbb{S} \times \mathbb{T}$ as a new state space $\mathbb{S}'$. Namely, the state space now stands for the spatiotemporal space in our context. In this representation, one can imagine that the spatial state space $\mathbb{S}$ is duplicated on every discrete time "layer" to form a collection of spatial states along the temporal dimension as $\{\mathbb{S}_1, \mathbb{S}_2, ...\}$, where each $\mathbb{S}_k$ is the same as $\mathbb{S}$. The state-time pair (s, t) corresponds to a state in this spatiotemporal space. Transition links are added by concatenating states on different time layers, constrained by the local transition time h. Similar spatiotemporal representation is also adopted in many other fields [35, 6, 29].

We emphasize here that the discrete time intervals and transition links between states in the spatiotemporal representation can be determined by the underlying motion kinematics of autonomous vehicles via the local transition time. This is a time discretization-free mechanism and is naturally supported by our proposed TVMDP framework. We will show an example in Section V-C.

D. TVMDP Value Iteration in Spatiotemporal Space

The optimal policy π^* for TVMDPs may be conceptually achieved by the conventional value iteration approach through sweeping the entire state space $\mathbb{S}$ and time space $\mathbb{T}$. The TVMDP value iteration (VI) amounts to iterating the following state-time value function until convergence,

$$v(s,t) = \max_{a \in A(s,t)} \left\{ R(s,t,a) + \gamma \cdot \mathbb{E}[v(s',t') \mid s, a; t] \right\}, \quad (4)$$

where s' is the next state to visit at time t' from the current state s at time t, and

$$\mathbb{E}[v(s',t') \mid s, a; t] = \sum_{s' \in \mathbb{S}, t' \in \mathbb{T}} \mathcal{T}(s,t,a;s',t') \cdot v(s',t')$$

is the weighted average of the value functions of all the next possible spatiotemporal states.

The value function Eq. (4) is a modification of the conventional Bellman equation as it includes a notion of time. In addition to propagating values spatially from next states, it also backs up the value temporally from a future time. Moreover, the benefits of the spatiotemporal representation in applications are readily seen, as the solution to Eq. (4) is equivalent to applying dynamic programming directly to the spatiotemporal state space.

A typical spatiotemporal state space is very large, especially when high state resolution is needed. The naive dynamic programming-based value iteration or policy iteration involves backing up state-values not only from the spatial dimension but also from the temporal dimension. It is generally intractable to solve for the exact optimal policy due to the so-called curse of dimensionality [26].

IV. Methodology

In this section, we present tractable iterative algorithms for TVMDP by a reduction of the spatiotemporal state space in each iteration. Our approach is grounded in characterizing the most possibly reachable set of states through the first and second moments of the passage percolation time.

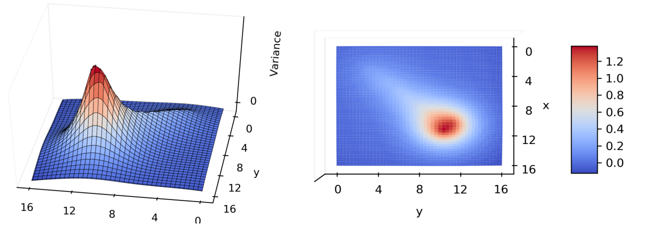
A. Overview

One of the major challenges to solving the Bellman equation (Eq. (4)) is the search in a large spatiotemporal state space. Once we are able to reduce the whole spatiotemporal space to a tractable size, it is then possible to obtain solutions by, for instance, the value iteration algorithm, within a reasonable time span. Given a policy, an initial state-time pair, and a probability transition law, for a fixed (spatial) state s, probabilities of visiting s at different times are highly likely different. If we are able to quantify the reachability of spatiotemporal states by visiting probability, and trim the whole spatiotemporal space by removing those with small reachability, it is highly likely that the optimal total reward will not be affected much (under certain restrictions on the variability of the reward and transition functions), and we can gain significant benefits from reducing the computation.

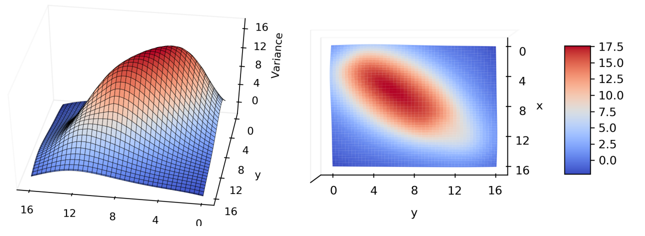
The previously introduced variable Passage Percolation Time (PPT) $H^{u}_{s,s'}$ sheds light on estimating the chances of reaching state s' by evaluating the transition time from a state s at time u. Although the exact probability distribution of PPT is generally hard to obtain, its first and second moments are relatively easy to compute. Therefore, *we use the expectation and variance of PPT between the initial spatial state s_0 and another arbitrary state s to characterize the most possible time interval to reaching s.* Following this way, we are able to find a most reachable set $\{(s,t)\}_{t \in \mathbb{T}}$ along the spatiotemporal dimensions. This reachable set is closed, allowing us to reduce the search by looking only within the enclosed spatiotemporal space.

The above procedure actually views all $H^{u}_{s_0,s}$ independently, and does not compute the probability for every trajectory. If one state-action pair in a trajectory is removed, the associated trajectory will become invalid in the reduced space. Therefore, it may eliminate too many potential trajectories (paths) τ with relatively large probability. In order to remedy this problem, we reconstruct a TVMDP on the reduced spatiotemporal space to maintain a certain correlation between connected spatial states (recall the definition of connection in section III-B). In Section IV-D, we will see this procedure is equivalent to *mapping* a removed potential trajectory to another one in the reduced search space.

The final algorithm iterates the above procedures on the whole spatiotemporal space. In each iterative step, the first and second moments of the PPT are recomputed. Intuitively, *the first moment is used to find the "backbone" that outlines the most reachable states, whereas the second moment determines the "thickness" (or volume) of the most reachable space.* The policy is then updated on the reduced space, and the actions for states outside the reduced space are mapped to



(a) (b)



(c) (d)

Fig. 2: The change in the variance under different uncertainties in the environment. The state transition is modeled as a Gaussian distribution which is referred to Sec. V. (a)(b) 3D and top down views with the variance of the transition function set to 0.1. (c)(d) A more uncertain environment with the variance set to 1.

Algorithm 1 Value Iteration with Expected Passage Percolation Time

```
Input: Time-varying transition function $\mathcal{T}$, time-varying reward $R$, spatial states $\mathbb{S}$, and $t_0 = 0$.
Output: Policy $\pi$ and the expected passage percolation time $\mu_s, \forall s \in \mathbb{S}$.
 1: Initialize $\mu_s = \tilde{\mu}_s = 0, v(s) = 0, \forall s \in \mathbb{S}$;
 2: repeat
 3:     $\delta = 0$.
 4:     for each $s \in S$ do
 5:         $V = v(s)$;
 6:         Solve Bellman equation (5);
 7:         $\delta = \max(\delta, V - v(s))$;
 8:     end for
 9:     Get policy $\pi$;
10:     Solve for $\mu_s$ by $N-1$ systems of linear equations (7);
11:     Set $\tilde{\mu}_s = \mu_s$;
12: until $\delta$ is less than a threshold or the maximum number of iteration has been reached.
```

the updated actions on nearest states in the reduced space. Comparing with [27] which only uses the first moment of PPT to characterize TVMDP, our method obtains a better approximation to the full spatiotemporal space of TVMDP.

B. Value Iteration with Expected Passage Percolation Time

This section presents the first approximate algorithm, an expected PPT-based value iteration, for TVMDPs. It serves as a burn-in procedure for the algorithm in Section IV-D.

In this first approximate algorithm, we use the transition probabilities and the reward function at the expected PPTs to approximate their time-varying counterpart. Then we approximate the TVMDP by an MDP with a properly defined action set at s. Accordingly, we can conduct value iteration on the state space $\mathbb{S}$ rather than space $\mathbb{S} \times \mathbb{T}$, and use the resulting time-independent policy for the state at any time.

Suppose $\mu_s := \mathbb{E}^\pi[H^{t_0}_{s_0,s}]$ are already obtained for all $s \in \mathbb{S}$ given the initial state s_0 and the starting time t_0. Note that we always have $\mu_{s_0} = 0$. With a new transition probability law defined as $\mathcal{T}^{\pi,\mu_s}_{s,s'} := \sum_{t'} \mathcal{T}(s, \mu_s, a; s', t')$ and a new reward function as $R^{\mu_s}(s,a) := R(s, a, \mu_s)$, one can update the policy by solving the following Bellman equation

$$v(s) = \max_{a \in A(s)} \left\{ R^{\mu_s}(s,a) + \gamma \cdot \mathbb{E}^\pi_{\mu_s}[v(s') \mid s, a] \right\}, \tag{5}$$

where $\mathbb{E}^\pi_{\mu_s}(\cdot)$ indicates the calculation under the new transition law $\mathcal{T}^{\pi,\mu_s}_{s,s'}$ and $A(s)$ is the action set at s. The output policy does not depend on time anymore, and will be used for any state all the time. Usually we only have one $t' \in \mathbb{T}$ such that $\mathcal{T}(s,u,a;s',t') > 0$ for fixed s, u, a and s' in the practical modeling framework. If $t'' \neq t'$, then $\mathcal{T}(s,u,a;s',t'') = 0$. In this case, $\mathcal{T}^{\pi,\mu_s}_{s,s'} = \mathcal{T}(s, \mu_s, a; s', t')$. Now we show how to obtain the expected PPTs. By the conditional expectation formula $\mathbb{E}^\pi[H^u_{s,s'}] = \mathbb{E}^\pi[H^u_{s'',s'} \mid s'']$, the expected PPTs satisfy the following linear system

$$\mathbb{E}^\pi[H^u_{s,s'}] = \sum_{s'',u''} \mathcal{T}(s,a,u;s',u'') \left(h(s,u,s'') + \mathbb{E}^\pi[H^{u''}_{s'',s'}] \right) \tag{6a}$$

$$\mathbb{E}^\pi[H^u_{s,s}] = 0, \tag{6b}$$

where $s \neq s'$, $u'' = u + h(s,u,s'')$, and s'' in Eq. (6a) is the next visiting state from s. The above recursive relationship is similar to the equations for estimating the mean first passage time for a Markov Chain [21, 10], except that the mean first passage time from a state to itself is a positive value.

We do not solve Eq. (6) for all time u altogether. Instead, we approximate the solutions in a recursive fashion with the aim of estimating μ_s for all s in the space. In other words, we only carry the estimates of μ_s into the next iteration. In each iteration, we approximate Eq. (6a) by the following linear system

$$\mathbb{E}^\pi[H^{\tilde{\mu}_s}_{s,s'}] = \sum_{s''} \mathcal{T}^{\pi,\tilde{\mu}_s}_{s,s''} \left(h(s,\tilde{\mu}_s,s'') + \mathbb{E}^\pi[H^{\tilde{\mu}_{s''}}_{s'',s'}] \right), \tag{7}$$

where $\tilde{\mu}_s = \mathbb{E}^{\tilde{\pi}}(H^{t_0}_{s_0,s})$ are obtained from the latest iteration with the previous policy $\tilde{\pi}$. It should be noted that one only needs to solve a linear system which consists of $(N-1)$ linear equations with $N-1$ unknown variables for one μ_s. Here, $N := |\mathbb{S}|$ denote the total number of states. Therefore, we merely need to solve $(N-1)$ linear systems to obtain all μ_s in one iteration step.

Lastly, we make a note on the numerical solution for solving the above linear system. In practice, there could be extreme large values for some estimates of $\mathbb{E}(H^{\tilde{\mu_s}}_{s,s'})$, indicating that the associated states are nearly impossible to visit. Yet these large values could result in numerical instability. To avoid this issue, we put a discount factor α <1.0 to the expected PPT on the right side of Eq. (7), i.e., replacing $\mathbb{E}^{\pi}[H^{\tilde{\mu}_{s''}}_{s'',s'}]$ by $\alpha \cdot \mathbb{E}^{\pi}[H^{\tilde{\mu}_{s''}}_{s'',s'}]$. The expected PPT-based algorithm is described in Alg. 1.

C. Variance of Passage Percolation Time and Reachable Space

Once the expectation of $H^{u}_{s,s'}$ is obtained, one is able to further derive the estimation of variance under the same policy π by the total variance formula as below [16]:

$$\mathbb{V}^{\pi}[H^{u}_{s,s'}] = \mathbb{E}^{\pi}[\mathbb{V}^{\pi}[H^{u}_{s,s'}|s'']] + \mathbb{V}^{\pi}[\mathbb{E}^{\pi}[H^{u}_{s,s'}|s'']], \quad (8)$$

where s'' can be viewed as the next state to visit from s before reaching s'. The two terms on the right hand side of the above equation are calculated as

$$\mathbb{E}^{\pi}[\mathbb{V}^{\pi}[H^{u}_{s,s'}|s'']] = \sum_{s'',\mu''} \mathcal{T}(s,u,a;s'',\mu'')\mathbb{V}^{\pi}[H^{u''}_{s'',s'}], \quad (9)$$

and

$$\begin{aligned} & \mathbb{V}^{\pi}[\mathbb{E}^{\pi}[H^{u}_{s,s'}|s'']] \\ = & \sum_{s'',u''} \mathcal{T}(s,u,a;s'',u'')(\mathbb{E}^{\pi}[H^{u}_{s,s'}|s''] - \mathbb{E}^{\pi}[H^{u}_{s,s'}])^2 \\ = & \sum_{s'',u''} \mathcal{T}(s,u,a;s'',u'') \cdot \Big(h(s,u,s'') \\ & + \mathbb{E}^{\pi}[H^{u''}_{s'',s'}] - \mathbb{E}^{\pi}[H^{u}_{s,s'}]\Big)^2 . \end{aligned} \quad (10)$$

Eq. (9) and (10) show that the variance estimation also relies on solving a linear system. Because the local transition time $h(s,u,s')$ is assumed deterministic and does not depend on policy, it can be taken out from the term $\mathbb{E}^{\pi}[H^{u}_{s,s'}|s''] = \mathbb{E}^{\pi}[h(s,u,s'') + H^{u}_{s'',s'}|s'']$ in Eq. (10). The same idea also applies to $\mathbb{V}^{\pi}[H^{u}_{s,s'}|s'']$ in Eq. (9).

Because we aim to obtain a range of time around μ_s to reach state s from the initial state t_0, we only focus on $\sigma_s^2 := \mathbb{V}^{\pi}[H^{t_0}_{s_0,s}]$. The structure of linear equations for σ_s^2 is similar to that for expectation in the previous section. Therefore, we can get the solutions in a similar recursive fashion:

$$\begin{aligned} \mathbb{V}^{\pi}[H^{\mu_s}_{s,s'}] = & \sum_{s''} \mathcal{T}^{\pi,\mu_s}_{s,s''} \Big(\mathbb{V}^{\pi}[H^{\mu_s}_{s'',s'}] \\ & + \Big(h(s,\mu_s,s'') + \mathbb{E}^{\pi}[H^{\mu_{s''}}_{s'',s'}] - \mathbb{E}^{\pi}[H^{\mu_s}_{s,s'}]\Big)^2\Big) . \end{aligned} \quad (11)$$

Similarly, an array of linear $N-1$ equations (11) with the same number of unknown variables needs to be solved for a σ_s^2. It is worth noting that the computation of expectation and variance of PPT can be computed in parallel, offering an extra boost in computational time as shown later in Section V. As an illustration, Fig. 2 shows the variance of PPT from the initial state to other states in spatial (width of top down view) and temporal (height of the 3D view) space under different environmental uncertainties.

Now we can formally define the reachable space of TVMDPs.

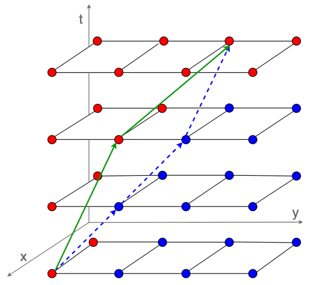


Fig. 3: Illustration of the reconstruction procedure. Four replicated layers on different time slots. Transition only between neighbored spatial states in two consecutive time slots. The red nodes are spatiotemporal states in reachable space, whereas blue ones are outside. Transition probability from the bottom left state to the next state (the end of blue arrow) is reassigned to the state two time slots away (the end of green arrow).

Definition 1. *Given a policy π for TVMDP, a state-time pair (s,t) is reachable from an initial state-time pair (s_0,t_0) if t satisfies*

$$\mu_s - m_r\sigma_s \leqslant t \leqslant \mu_s + m_r\sigma_s, \quad (12)$$

where $m_r \geqslant 1$ is a predefined regulation parameter. The set of all reachable state-time pairs is called reachable space for TVMDP, and is denoted by $(\mathbb{S}\times\mathbb{T})^{-}$.

Note that the predefined parameter m_r used to further “inflate" the reachable space envelope if there is a need.

D. TVMDP Value Iteration with Reachable Space

The reachable space describes the variability of the first time for the vehicle to visit a state. Computational cost may be reduced by iterating within the reachable space only. However, the space trimming leads to some issues while processing the states at or around the reachable space border.

Consider a random trajectory $\{\ldots,(s_k,a_k),(s_{k+1},a_{k+1}), \ldots\}$, where the vehicle reaches s_k at t_k with (s_k,t_k) in the reachable space $(\mathbb{S}\times\mathbb{T})^{-}$. To aid a clear presentation, we use the symbol $\mathcal{T}_u$ as a shorthand for $\mathcal{T}(s_k,t_k,a_k;s_{k+1},u)$ in this subsection. If (s_{k+1},t_{k+1}) is not in this reachable space, the transition probability law restricted on $(\mathbb{S}\times\mathbb{T})^{-}$ has the property $\sum_{u:(s_{k+1},u)\in(\mathbb{S}\times\mathbb{T})^{-}} \mathcal{T}_u < 1$. This places a negative impact on red the algorithms to maximize the value function. Therefore, we need to reassign the transition probability law as follows.

Let $l_{s_{k+1}} = \mu_{s_{k+1}} - m_r \cdot \sigma_{s_{k+1}}$ and $b_{s_{k+1}} = \mu_{s_{k+1}} + m_r \cdot \sigma_{s_{k+1}}$ If $t_{k+1} < l_{s_{k+1}}$, then we reassign transition probability

$$\mathcal{T}_{l_{s_{k+1}}} = \sum_{u' \leqslant l_{k+1}} \mathcal{T}_{u'} .$$

For $t_{k+1} > l_{s_{k+1}}$, we simply re-normalize the transition probability if $0 < \sum_{u:(s_{k+1},u)\in(\mathbb{S}\times\mathbb{T})^{-}} \mathcal{T} < 1$. We call such a treatment a *reconstruction* procedure for TVMDPs.

Fig. 3 illustrates an intuition of the reconstruction procedure: if there are a few states along a spatiotemporal path outside the reachable space, the entire path would not be considered

Algorithm 2 TVMDP Value Iteration with Reachable Space

```
Input: TVMDP elements $(\mathbb{S}, \mathbb{T}, \mathbb{A}, \mathcal{T}, R)$; starting state $s_0$;
    starting time $t_0 := 0$; $m_r$; numbers $k, n, i$.
Output: $\pi^*$: an approximation of the optimal policy $\mathbb{S} \times \mathbb{T}$.
 1: Burn-in: run Algorithm 1 for $k$ iterations to get $\pi_0$;
 2: for j=0...n do
 3:     Compute $\mathbb{E}^{\pi_j}[H^{t_0}_{s_0,s}]$ and $\mathbb{V}^{\pi_j}[H^{t_0}_{s_0,s}]$ for all $s \in \mathbb{S}$;
 4:     Apply Eq. (12) to obtain reachable space $(\mathbb{S} \times \mathbb{T})^-$;
 5:     Perform reconstruction procedure;
 6:     Execute value iteration for $i$ iterations to obtain $\pi_{j+1}$
        on $(\mathbb{S} \times \mathbb{T})^-$;
 7:     Map actions using Eq.(13) if $(s,t) \notin (\mathbb{S} \times \mathbb{T})^-$;
 8:     if $\pi_{j-1} = \pi_j$ and $j > 0$ then
 9:         break
10:     end if
11: end for
```

in the value function. However, the reconstruction procedure *maps* this path back to the reachable space.

Our final algorithm combines the techniques introduced in the previous sections and applies value iteration in the reachable space with reconstruction in an iterative manner. It is summarized in Alg. 2. In each iteration, there are two stages. During the first stage, the expectation and variance of PPTs are computed. These estimations allow us to determine the reachable space $(\mathbb{S} \times \mathbb{T})^-$. During the second stage, the value iteration algorithm is employed on $(\mathbb{S} \times \mathbb{T})^-$ to update the policy. Finally, if $(s,t) \notin (\mathbb{S} \times \mathbb{T})^-$, we assign the action $\pi(s,t)$ as

$$\begin{aligned} &\pi(s,t) = \pi(s',t), \\ &\text{where } s' = \arg\min_{s'}\{d(s,s') \mid (s',t) \in (\mathbb{S} \times \mathbb{T})^-\}, \end{aligned} \tag{13}$$

assuming that $\mathbb{S}$ has a metric measure $d(\cdot,\cdot)$.

V. EXPERIMENTS

We validate our proposed method in simulation. The experiments were conducted in Ubuntu 16.04 on a PC with a 4.20 GHz i7-7700k CPU and 32 GB RAM.

A. Experimental Setup

We consider a 2D ocean surface subject to time-varying ocean currents in our experiments. The surface is tessellated into a 2D grid map where the centroid of each cell represents a state. A simulated marine vehicle with a kinematics model can transit in eight directions $\{N, NE, E, SE, S, SW, W, NW\}$ except for boundary states. The task for the vehicle is to reach a designated goal state with the minimum trajectory length.

Ocean currents are modeled as a velocity vector field where each vector can represent the magnitude and the direction of currents. Three types of currents are considered:

1) Self-spinning: each disturbance vector spins at the center of each cell. The vector components are given by $v_x = A\cos(\omega t), v_y = A\sin(\omega t)$, where A is the magnitude and ω is the spinning frequency.
2) Dynamic vortex: the vector components along x-axis and y-axis are given by $v_x = x_c - x + y - y_c$, $v_y = x_c - x - y + y_c$, where (x,y) and (x_c, y_c) are coordinates of the cell and the center of the vortex, respectively. The center of the vortex translates and rotates according to the following functions $x_c = r\cos(\omega t) + c_x$, $y_c = r\sin(\omega t) + c_y$, where r is the rotating radius and (c_x, c_y) represents the rotating center.
3) Regional Ocean Model System (ROMS) data [36]: the dataset provides ocean currents data every three hours per day. This allows us to obtain the transition function in the temporal dimension. Gaussian Process Regression (GPR) is used to model and extrapolate the ocean current vector field in time and space.

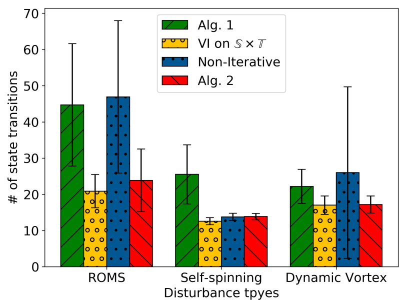


Fig. 4: Comparison of averaged number of transitions for four algorithms. The result averages over 100 runs with the same random seed for each algorithm and disturbance.

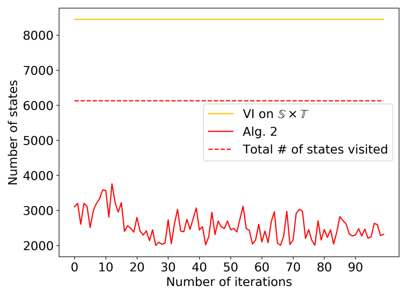
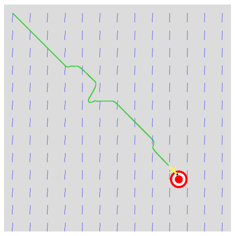
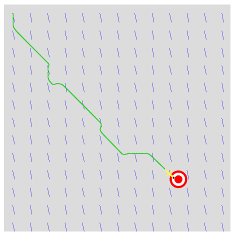


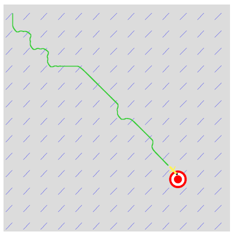
Fig. 5: Comparison between the number of states in the full space and reduced space over a hundred iterations in 13×13 grid world.

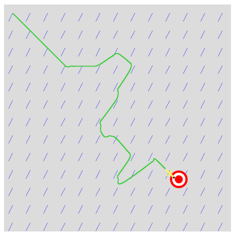
We compare our algorithm (Alg. 2) with three other methods:

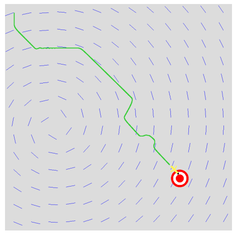
- Value Iteration in the spatiotemporal space (VI in $\mathbb{S} \times \mathbb{T}$): this method computes solutions exhaustively in the entire space, thus it is used as the baseline (only for problems with a small size state space).
- Value iteration with expected PPT (Alg. 1): it executes VI in the spatial space $\mathbb{S}$ only, where the time input for transition function is approximated by expected PPTs, i.e., expected PPTs are merely used in Bellman equation (5). This is also the main framework proposed in [27].
- Non-iterative variant of Alg. 2: it only reconstructs reachable space once (with $n = 1$). VI is executed only in this reduced space until the convergence condition is

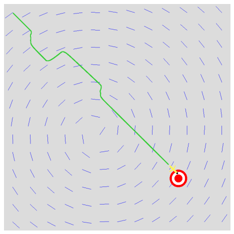

(a) VI on $\mathbb{S} \times \mathbb{T}$

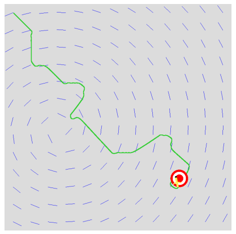

(b) Alg. 2


(c) Non-iterative


(d) Alg. 1


(e) VI on $\mathbb{S} \times \mathbb{T}$


(f) Alg. 2


(g) Non-iterative

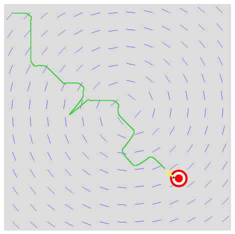
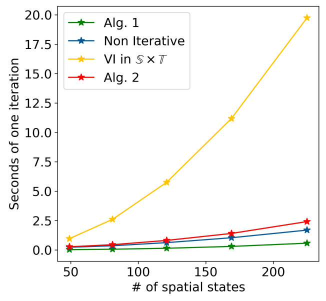
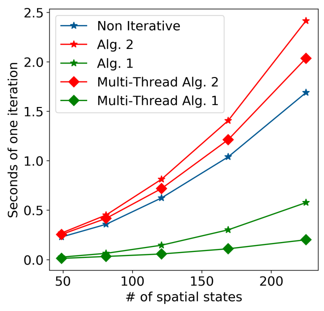

(h) Alg. 1

Fig. 6: Trajectories in the simplified environment with artificial disturbances. The top four figures demonstrate the trajectories of the four algorithms under self-spinning ocean currents. The bottom row shows the trials under dynamic vortex disturbances.



(a)



(b)

Fig. 7: Computational time comparisons between the four algorithms; (a) Single-threaded version; (b) Multi-threaded version for solving expected times and their variances.

reached. This is used to show the necessity of the iterative procedure in Alg. 2.

We choose $m_r = 2$ in Eq.(12). Because we are interested in minimizing the travel time, we impose a small reward -0.1 for each action execution except for reaching the goal and obstacle states. The rewards for reaching the goal and obstacle states are 1 and -1, respectively. Note that we do not use time-varying reward functions for experiments, but similar solutions can be obtained by exactly the same algorithm.

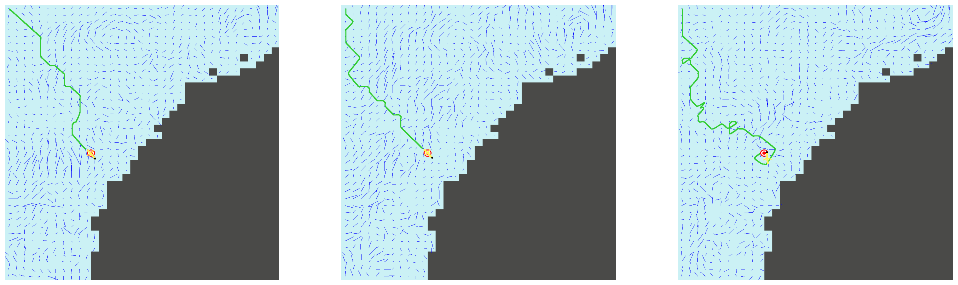
B. Simplified and Analytic Scenarios

We start with a simplified setting without considering the robot kinematics model: we set local transition time $h(s, t, s') = 1$ for all $s, s' \in \mathbb{S}$ and $t \in \mathbb{T}$. Such a simplified model allows us to obtain the optimal policy as a ground truth. Specifically, we discretize the time into 50 slots with a specified horizon of 50 transitions, and the spatial state $\mathbb{S}$ is represented by a 13×13 grid plane. Therefore, the full size of our spatiotemporal state space is $13 \times 13 \times 50 = 8450$. We use a Gaussian distribution to model the robot's state transition function where the variance is set to be 0.6. We set $A = 4, \omega = 1$ for self-spinning model. The parameters of the dynamic vortex model are set to $r = 3, \omega = 1$. For ROMS data, the same size of spatiotemporal state space is also chosen (cropped) from the raw dataset. Fig. 4 shows the average number of state transitions of the four algorithms, where the number of state transitions implies the number of hops used to reach the goal state (the smaller the number, the better the algorithm). The VI in the full space $\mathbb{S} \times \mathbb{T}$ (with dotted yellow texture) is the exhaustive search in the entire space, thus it provides the optimal solution. It is obvious that Alg. 2 (in red stripe) has a performance that is the closest to the optimal. The non-iterative method gives a worse performance, indicating that the iterative procedure helps improve results.

Fig. 5 illustrates the effectiveness of the state reduction by Alg. 2. It shows the size of the reduced state space at each iteration as well as the total number of states visited by the algorithm. One can observe that the number of states visited at each iteration is around one third of the full space.

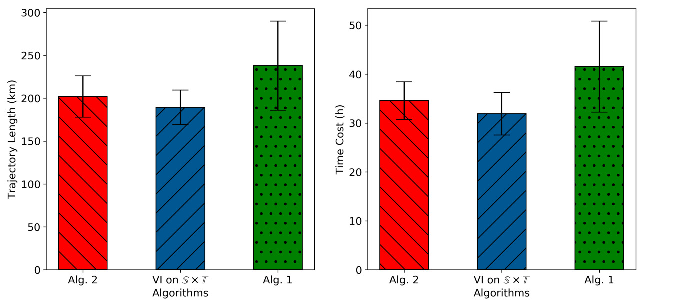
Fig. 6 shows the trajectories of the four methods in two types of disturbance vector fields, which reveal that our algorithm achieves the closest performance to the optimal solution.

Statistics on computational costs per iteration are shown in Fig. 7. Since the reconstruction causes the reachable spatiotemporal space to change at every iteration, we average the computational time over all iterations. It is obvious that our method (Alg. 2) requires much less computational time than the VI in full space. The gap between our method and non-iterative algorithms is mainly due to the overhead in computing the first and second moments of PPTs. Since the linear systems corresponding to first and second moments of PPTs are separate, multi-threading techniques can be utilized to further decrease the computational cost without affecting



(a) (b) (c)

Fig. 8: Trajectory results using the entire ROMS data. (a) Trajectory from the full spatiotemporal space; (b) Trajectory from Alg. 2; (c) Trajectory from Alg 1.



(a) (b)

Fig. 9: (a) Trajectory lengths of the three algorithms; (b) Time costs. The statistics are averaged over 100 trials.

Alg. 2	Alg. 1	VI in $\mathbb{S}\times\mathbb{T}$	Multi-Thread Alg. 2	Multi-Thread Alg. 1
3min	1min	15min	2.4min	0.3min

TABLE I: Computational cost for one iteration using the whole original ROMS data over three days.

the final result. Fig. 7(b) shows the improvement using multi-thread computation.

C. Realistic and High-Fidelity Scenarios

In this experiment, we consider the task of navigating an autonomous underwater vehicle to a goal state with a kinematics model. Specifically, a PID controller is used to follow the high-level action $a = \pi(s,t)$ generated by the policy given the current state-time pair (s,t). In this case, the local transition time $h(s,t,s')$ is no longer a constant because it is influenced by the vehicle motion dynamics and the time-varying ocean disturbances. We evaluate the algorithms using the entire region of the ROMS data, which is discretized into a 35×39 grid map with a spatial resolution of 6 km $\times$ 6 km. The vehicle's maximum velocity is set to 6 km/h.

In order to solve the first and second moments of PPTs, we need to compute the local transition times first. However, the exact form of $h(s,t,s')$ is hard to obtain due to the continuous vehicle motions, ocean currents, and the state space. To address this problem, we resort to a method that utilizes Monte Carlo trials to estimate $h(s,t,s')$.

Specifically, we want to approximate the traveling time from state s at time t to its next state s' given the vehicle motion and the current ocean disturbance estimated via GPR. We first discretize the orientation of vehicle's motion as well as the ocean disturbance direction into eight directions. Then, we sample the next state s' randomly from the transition function for each pair of discrete motion and disturbance directions. The local transition time is calculated as $h_a(s,t,s') = \frac{v_{net}}{d}$, where v_{net} is the net velocity of the vehicle after taking account of the ocean disturbance at time t and d is the distance between s and s'. To obtain an estimate of local transition times for all states, the above method iterates over the entire spatiotemporal state space with 100 trials for each state. Note that this estimation can be computed offline without adding any computational cost to the proposed algorithms.

We use the above estimation of local transition time to compute the first and second moments of PPTs in Alg. 1 and Alg. 2. Because of the high computational cost of VI in $\mathbb{S}\times\mathbb{T}$, we choose to discretize the time dimension with 30 slots with a time horizon of 50 hours. To keep the comparison fair, we choose the same discretization for Alg. 2. We exclude the non-iterative method in this experiment as it does not produce comparable performance.

Fig. 9 shows the statistics of the three algorithms in terms of trajectory length and time cost. The results reveal similar performance for scenarios described in the previous section, except that the performance gap between VI in $\mathbb{S}\times\mathbb{T}$ and Alg. 2 is slightly smaller. This is because the discrete time intervals in Alg. 2 are adjusted according to the local transition time, as mentioned in the end of Section III-C. The corresponding trajectories are shown in Fig. 8. The computational time is presented in Table I. We can see that, in this test scenario our algorithm requires only one-fifth of the time used by the VI over the full spatiotemporal space.

VI. Conclusions

We present a passage percolation time-based method to time-varying Markov Decision Processes that can be applied to autonomous systems' motion (action) planning. Our method iteratively solves the TVMDP reconstructed from a reachable space that was reduced from the original state space based on the first and second moments of passage percolation time. Our extensive simulations using ocean data show that our approach produces results close to the optimal solution but requires much smaller computational time. We will incorporate the methods of statistical learning for data into our proposed framework in the future.

References

[1] Antonio Auffinger, Michael Damron, and Jack Hanson. 50 years of first passage percolation. *arXiv preprint arXiv:1511.03262*, 2015.

[2] Taposh Banerjee, Miao Liu, and Jonathan P How. Quickest change detection approach to optimal control in Markov Decision processes with model changes. In *American Control Conference (ACC), 2017*, pages 399–405. IEEE, 2017.

[3] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pages 2242–2253. IEEE, 2017.

[4] Frederick J Beutler and Keith W Ross. Time-average optimal constrained semi-Markov Decision Processes. *Advances in Applied Probability*, 18(2):341–359, 1986.

[5] Justin Boyan. Least-squares temporal difference learning. In *In Proceedings of the Sixteenth International Conference on Machine Learning*, pages 49–56. Morgan Kaufmann, 1999.

[6] Justin A Boyan and Michael L Littman. Exact solutions to time-dependent MDPs. In *Advances in Neural Information Processing Systems*, pages 1026–1032, 2001.

[7] Samuel PM Choi, Dit-Yan Yeung, and Nevin Lianwen Zhang. An environment model for nonstationary reinforcement learning. In *Advances in neural information processing systems*, pages 987–993, 2000.

[8] Bruno C Da Silva, Eduardo W Basso, Ana LC Bazzan, and Paulo M Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd International Conference on Machine learning*, pages 217–224, 2006.

[9] Peter Dayan and Yael Niv. Reinforcement Learning: The Good, The Bad and The Ugly. *Current Opinion in Neurobiology*, 18(2):185–196, 2008.

[10] Shoubhik Debnath, Lantao Liu, and Gaurav Sukhatme. Solving markov decision processes with reachability characterization from mean first passage times. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7063–7070. IEEE, 2018.

[11] Finale Doshi-Velez, David Pfau, Frank Wood, and Nicholas Roy. Bayesian nonparametric methods for partially-observable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37 (2):394–407, 2015.

[12] Richard Everett and Stephen Roberts. Learning against non-stationary agents with opponent modelling and deep reinforcement learning. In *2018 AAAI Spring Symposium Series*, 2018.

[13] Jeremy H Gillula and Claire J Tomlin. Guaranteed safe online learning via reachability: tracking a ground target using a quadrotor. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2723–2730. IEEE, 2012.

[14] Jeremy H Gillula and Claire J Tomlin. Reducing conservativeness in safety guarantees by learning disturbances online: iterated guaranteed safe online learning. In *Robotics: Science and Systems*, volume 8, page 81, 2013.

[15] Geoffrey Grimmett. *Percolation, volume 321 of Fundamental Principles of Mathematical Sciences*. Springer-Verlag, Berlin, 1999.

[16] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford university press, 2001.

[17] Emmanuel Hadoux, Aurélie Beynier, and Paul Weng. Sequential decision-making under non-stationary environments via sequential change-point detection. In *Learning over Multiple Contexts (LMCE)*, 2014.

[18] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *International Conference on Machine Learning*, pages 1804–1813, 2016.

[19] Ruijie He, Emma Brunskill, and Nicholas Roy. Puma: Planning under uncertainty with macro-actions. In *AAAI*, 2010.

[20] Chen Huang, Kai Yin, and Lantao Liu. Learning partially structured environmental dynamics for marine robotic navigation. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–8. IEEE, 2018.

[21] Jeffrey J. Hunter. The computation of the mean first passage times for markov chains. *Linear Algebra and its Applications*, pages 100–122, 2018.

[22] Tommi Jaakkola, Satinder P Singh, and Michael I Jordan. Reinforcement learning algorithm for partially observable markov decision problems. In *Advances in neural information processing systems*, pages 345–352, 1995.

[23] L Jianyong and Z Xiaobo. On average reward semi-markov decision processes with a general multichain structure. *Mathematics of Operations Research*, 29(2): 339–352, 2004.

[24] J. Kober, J. Andrew (Drew) Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, July 2013.

[25] Zhuoshu Li, Zhitang Chen, Pascal Poupart, Sanmay Das, and Yanhui Geng. Faster policy adaptation in environments with exogeneity: A state augmentation approach. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1035–1043. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[26] Lantao Liu and Nathan Michael. An mdp-based approximation method for goal constrained multi-mav planning under action uncertainty. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 56–62. IEEE, 2016.

[27] Lantao Liu and Gaurav S Sukhatme. A solution to time-varying markov decision processes. *IEEE Robotics and Automation Letters*, 3(3):1631–1638, 2018.

[28] Yuanfu Luo, Haoyu Bai, David Hsu, and Wee Sun Lee. Importance sampling for online planning under uncertainty. *The International Journal of Robotics Research*,

page 0278364918780322, 2016.

[29] Monirehalsadat Mahmoudi and Xuesong Zhou. Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state-space-time network representations. *Transportation Research Part B: Methodological*, 89:19 – 42, 2016. ISSN 0191-2615.

[30] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[31] Ruben Martinez-Cantin, Nando de Freitas, Arnaud Doucet, and José A Castellanos. Active policy learning for robot planning and exploration under uncertainty. In *Robotics: Science and Systems*, volume 3, pages 321–328, 2007.

[32] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005.

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

[34] Emmanuel Rachelson, Patrick Fabiani, and Frédérick Garcia. TiMDPpoly: An Improved Method for Solving Time-Dependent MDPs. In *ICTAI*, pages 796–799, 2009.

[35] Pan Shang, Ruimin Li, Jifu Guo, Kai Xian, and Xuesong Zhou. Integrating Lagrangian and Eulerian observations for passenger flow state estimation in an urban rail transit network: A space-time-state hyper network-based

[43] Ward Whitt. Time-varying queues. *Queueing Models*

assignment approach. *Transportation Research Part B: Methodological*, 121:135–167, 2019.

[36] Alexander F Shchepetkin and James C McWilliams. The regional oceanic modeling system (roms): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean modelling*, 9(4):347–404, 2005.

[37] Richard Sutton, Doina Precup, and Satinder Singh. Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[38] Richard S. Sutton. Learning to predict by the methods of temporal differences. In *MACHINE LEARNING*, pages 9–44. Kluwer Academic Publishers, 1988.

[39] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

[40] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[41] Gerald Tesauro. Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68, 1995.

[42] Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012.
and Service Management, pages 79–164, 2018.

[44] Yan Zheng, Zhaopeng Meng, Jianye Hao, Zongzhang Zhang, Tianpei Yang, and Changjie Fan. A deep bayesian policy reuse approach against non-stationary agents. In *Advances in Neural Information Processing Systems*, pages 960–970, 2018.