A Solution to Time-Varying Markov Decision Processes

Lantao Liu and Gaurav S. Sukhatme

***Abstract*—We consider a decision-making problem where the environment varies both in space and time. Such problems arise naturally when considering, e.g., the navigation of an underwater robot amidst ocean currents or the navigation of an aerial vehicle in wind. To model such spatiotemporal variation, we extend the standard Markov decision process (MDP) to a new framework called the time-varying Markov decision process (TVMDP). The TVMDP has a time-varying state transition model and transforms the standard MDP that considers only *immediate* and *static* uncertainty descriptions of state transitions, to a framework that is able to adapt to future time-varying transition dynamics over some horizon. We show how to solve a TVMDP via a redesign of the MDP value propagation mechanisms by incorporating the introduced dynamics along the temporal dimension. We validate our framework in a marine robotics navigation setting using spatiotemporal ocean data and show that it outperforms prior efforts.**

***Index Terms*—AI-based methods, motion and path planning, marine robotics.**

I. Introduction

CONSIDER a scenario where an underwater vehicle navigates across an area of ocean over a period of a few weeks to reach a goal location. The ocean currents are typically strong enough to disturb the vehicle's motion, causing significantly uncertain action outcomes. Decision theoretic planning methods [1] cope with action uncertainty by stochastically modeling the action outcomes. However, the dynamic nature of the ocean currents implies that to be effective, any underlying model that describes the uncertainty associated with the vehicle's actions must also vary with time.

A popular method for addressing action uncertainty is the Markov Decision Process (MDP) [2], a decision-making framework in which the uncertainty due to actions is modeled using a stochastic state transition function. However, a limitation of this approach is that the state transition model is static, i.e., the uncertainty distribution is a "snapshot at a certain moment" [3].

Fortunately, environmental dynamics such as those associated with ocean currents can be forecast (albeit imprecisely) over a future time horizon. We exploit the idea that the forecast dynamics are time-varying functions that can be used to stochastically predict the state transition model. In this letter, we describe how to incorporate time-varying stochasticity into an MDP-style decision-making model and show how the resulting framework leads to improved planning accuracy.

Contributions: We propose a new method called the *time-varying Markov Decision Process (TVMDP)*. Different from the standard MDP which is designed for solving decision-making problems with time-invariant transition models, the TVMDP has the capability of handling a time-varying transition model and can model decision problems with stochastic state transitions that vary both spatially and temporally. The proposed TVMDP method requires neither discretization of time nor augmentation of state space. But to solve it, we show that the basic Bellman backup mechanism used for computing the MDP is not enough. Instead, we present a new mechanism based on two iterative value propagation processes, which proceed in both spatial and temporal dimensions, where the spatial process involves Bellman backup but the temporal dimension is expanded using Kolmogorov equations. This also implies that, the standard MDP is a special case of TVMDP.

II. Related Works

Time varying Markov transition dynamics have been studied previously in the context of pattern analysis of economic growth [4], (aimed at understanding the dynamics of growth based on a collection of different states corresponding to different countries), analyzing fiscal policies [5] (civilian spending/taxing), and the environment [6] (extreme temperature events). However, the models in these studies are Hidden Markov Models (HMMs), and unlike an MDP, they do not have the notion of an action to control transitions between states.

The approach proposed in this letter bears comparison to a framework called the *time-dependent Markov Decision Process* (TDMDP) [7] that includes time in the state space of the MDP. In the TDMDP the probability distribution associated with the transition model is used to describe the (uncertain) travel duration from one state to the next. This is similar to the conventional treatment of time-dependence in planning and routing in operations research [8]. In contrast, our approach (the TVMDP) follows the standard MDP convention and utilizes the transition model to describe the *uncertain transitions among states (stochastic state jumping), while assuming that the transition model itself is time-varying*. The TDMDP uses a likelihood function to control the state transitions. In order to take advantage of the standard Bellman backup mechanism to compute

Manuscript received September 10, 2017; accepted January 11, 2018. Date of publication February 2, 2018; date of current version February 27, 2018. This letter was recommended for publication by Associate Editor P. Tokekar and Editor J. Wen upon evaluation of the reviewers' comments. This work was supported by National Science Foundation under Grant 1619319. *(Corresponding author: Lantao Liu.)*

L. Liu is with the Intelligent Systems Engineering Department, Indiana University, Bloomington, IN 47408 USA (e-mail: lantao@iu.edu).

G. S. Sukhatme is with the Department of Computer Science, the University of Southern California, Los Angeles, CA 90089 USA (e-mail: gaurav@usc.edu).

Digital Object Identifier 10.1109/LRA.2018.2801479

the solution, a set of restrictions are imposed on almost every term of the TDMDP formulation, making the framework less applicable in many realistic scenarios.

Recently, the TDMDP has been analyzed [9], and it has been shown that even under strong assumptions and restrictions, the computation can still be cumbersome. The same analysis recommends means for approximating a solution using strategies such as prioritized sweeping [10]. The idea shares certain similarity to the *real-time dynamic programming* [11] where those states with larger values or value changes have a higher priority to propagate. We use this approximation scheme on TDMDP to compare with our approach in the experiment section. Differences between our approach and the TDMDP are compared more formally in the remainder of this work.

Proximal works also include the vast literature on reinforcement learning (RL) [12]–[14] wherein an agent tries to maximize accumulated rewards by interacting with the environment during its lifetime, and the system is typically formulated as an MDP. A technique related to future prediction is temporal difference (TD) learning [15], [16]. RL extends the TD technique by allowing the learned state-values to guide actions, and correct previous predictions based on availability of new observations [12], [17]. Unfortunately, existing TD techniques are based on time-invariant models.

Additionally, in order to extend the discrete-time actions, a temporal abstraction based concept (the Semi-Markov Decision Processes (SMDP)) [18] has been designed so that actions can be performed in a more flexible manner. SMDPs consider stochastic action durations, but still assume the transition model itself is time-invariant.

III. Preliminaries

A. Markov Decision Process (MDP)

Definition III.1: An MDP $\mathcal{M}$ is defined by a 4-tuple $\mathcal{M} =< S, A, T, R >$, where $S = \{s\}$ and $A = \{a\}$ represent the countable state space and action space, respectively. The stochastic transition dynamics, also known as the transition model, are given by

$$T_a(s, s') = \Pr(s_{k+1} = s' | s_k = s,\ a_k = a) \tag{1}$$

which is a probability mass function that leads the agent to succeeding state $s_{k+1} = s'$ when it executes the action $a_k = a$ from state $s_k = s$. The time step k is also known as the computing epoch, where $1 \leq k \leq K$. The fourth element $R_a(s, s')$ is a positive scalar (also called the reward) for performing action a in state s and reaching state s'. A policy is a complete mapping $\pi : S \rightarrow A$ so that the agent applies the action $a_k \in A$ in state $s_k \in S$ at step k. If the action is independent of k, the policy is called stationary and a_k is simply denoted by a.

Note that, unlike conventional formulations, here we use k instead of t to index the algorithmic iterative steps/epochs, for both stationary and non-stationary models. In other words, we regard all K non-stationary algorithm iterations as momentary events, in order to distinguish the temporal process (a.k.a. planning horizon) which will be discussed further in the remainder of this letter.

Definition III.2: The $\mathbf{Q}$-value of a state-action pair (s, a) is defined as the the one-step look-ahead value of state s if the immediate action a is performed. More formally,

$$Q(s, a) = \sum_{s' \in S} T_a(s, s') \Big(R_a(s, s') + \gamma V(s') \Big), \tag{2}$$

where $V(s')$ is the value (accumulated reward) for state s' and $\gamma \in [0, 1)$ is a discount factor for discounting future rewards at a geometric rate.

The objective is to find an optimal policy π^* satisfying

$$V_{\pi^*}(s) \equiv V^*(s) = \max_{a \in A} Q(s, a),\ \forall s \in S. \tag{3}$$

When $\gamma < 1$, there exists a stationary policy that is optimal, and the optimal policy $\pi^*(s)$ is

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} T_a(s, s') \Big(R_a(s, s') + \gamma V^*(s') \Big) \tag{4}$$

Employing Bellman's principle of optimality avoids enumerating solutions naively. In particular, the dynamic programming based value iteration (VI) and policy iteration (PI) are the most widely used strategies for solving MDPs [2], [3].

IV. Technical Approach

A conventional MDP has a static transition model $T_a(s, s')$. This means that T has no time-dependence nor does the form of T vary with time[†]. In many practical scenarios, e.g., marine vehicles in the ocean or aerial vehicles flying in the air, the transition model $T_a(s, s')$ must account for environmental disturbances that vary with time. Such a time variation property requires the control policy to also be a function of time to successfully reject dynamic disturbances and the resultant action uncertainty. This motivates us to design a decision-making framework that involves a dynamic and time-varying state transition model.

A. Upgrade From "Spatial Only" to Both Spatial & Temporal

First, the MDP state needs to be upgraded. By "upgrade" we do not mean to simply append a time variable t to the state variable, e.g., $s := < s, t >$, so that the classic MDP framework can be directly employed. Such a simple appending operation is problematic because:

- Time differs from space in its properties. Specifically, space is isotropic and symmetric. Within kinodynamic constraints, we can control an object to freely move in space. In contrast, time is asymmetric and unidirectional.
- The differing properties of space and time imply that, *reachability* in space and time are not the same. We cannot travel back in time. We have much more restricted reachability (controllability) in time than in space.

[†]We use *time-dependent* and *time-varying* to mean different things. Following the convention in operations research, in the MDP context the term *time-dependent* is used to describe stochastic travel duration between states, whereas the term *time-varying* is used to express the change in the transition model as time elapses.

The above comparison also explains why the standard solution for an MDP cannot be used straightforwardly for an MDP with a time-varying transition model. In order to enforce the time constraints, one possible way is to duplicate states into discretized slices of "time layers," and then manipulate the transition probabilities so that state hopping backwards is prevented (with a probability of 0). However, such setting requires a significant increment of state space (sometimes action space also), as well as expensive computational costs for manipulating the transition models.

In this work, we propose a framework without discretizing time, and thus the state space and action space remain the same as the basic MDP. We show with a few steps that such spatiotemporal transition model can be addressed using an exact method extended from the MDP, and the state reachability (e.g., states traveling along the time dimension) is explored along algorithmic iterations.

Correlate Space and Time: The difference in reachability in space and time indicates that, the state transitions (and value propagation) in space and in time should be treated as two separate processes along a "spatial channel" and "temporal channel," respectively.

The key idea of our work is to *evolve* the spatial process of state transition (and value propagation) along the temporal channel, where the spatial and temporal processes under two channels are coupled by the underlying real-world physics. For example, in the marine vehicle planning scenario, the two processes are correlated and coupled by the vehicle's motion dynamics under environmental constraints such as disturbances from ocean currents.

Formally, to transform an MDP to a TVMDP, the static state transition model $T_a(s, s')$ needs to be a function of time, and we re-write it as $T_a(s, s', t)$. Similarly, the reward function becomes $R_a(s, s', t)$. The value function is modified accordingly:

$$V^*(s,t) = \max_{a \in A} \sum_{s' \in S} T_a(s,s',t)\Big(R_a(s,s',t) + \gamma V^*(s',t')\Big), \tag{5}$$

where $V(s', t')$ means the value of a future state s' at future time t'. Comparing Eq. (5) with the classic MDP (4), we see that every term of Eq. (5) is a function of time. Thus, the TVMDP can be written as $\mathcal{M}(t) = <S, A, T(t), R(t)>$. Next, we show how to construct this time-varying decision-making framework in detail.

Add Real-Valued Transition Time: A major difference between the TVMDP $\mathcal{M}(t)$ and the MDP $\mathcal{M}$ lies in that, $\mathcal{M}(t)$ is built on, and thus requires estimation of, the real-valued *transition time* for the agent to travel between pairwise states, which is different from the "state hopping" in an MDP – an important property inherited from Markov Chains.

More formally, the *transition time* here denotes the observable travel time cost if the agent travels from a state s to another state s'. It is used to track and map future time moments to their corresponding policy rewards/values at those moments, along the unidirectional temporal channel.

To understand the basic idea, assume that at time t_0 the robot is in state s_0, and let $t(s_0, s)$ be the transition time from s_0 to an arbitrary state s. Since the transition model is a function of time and the time elapses during the robot's motion, thus when the robot eventually arrives at state s, the transition model that impacts the robot is given by $T_a(s, s', \tau)$ instead of $T_a(s_0, s)$ (or equivalently $T_a(s_0, s, t_0)$), where $\tau = t_0 + t(s_0, s)$ is a later time. Therefore, value evaluation/iteration for state s at the starting moment t_0 needs to look ahead and utilize the transition model $T_a(s, s', \tau)$ captured at the later time τ. Fig. 1 depicts such an idea. It is worth mentioning that, the discrete time layers in Fig. 1 are used only for the purpose of demonstrating the idea of an additional time dimension. We are showing that the proposed method does not need to discretize time.

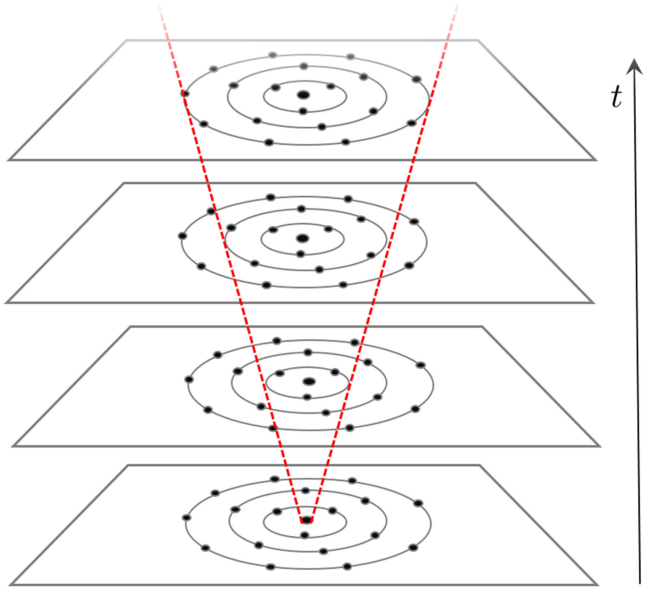


Fig. 1. Conceptual illustration of the value iteration propagation along both spatial and temporal dimensions (red dashed lines). Along the value propagation, the spatial and temporal processes need to be coupled by real world dynamics from both the robot (action/motion) and the environment (disturbance).

Also note that, here we use a perfect prediction model that assumes the estimated times $t(s_0, s)$ and τ are accurate. This however, is unrealistic due to the robot's stochastic behavior. This problem is discussed and addressed in Section IV-B.

Construct Time-Varying Transition Models: The mapping of future time moments to future policy values essentially relies on employing the correct transition models at those future moments. This is because policy values are propagated on the stochastic network that is expressed by those transition models. Hence, *the key component of a TVMDP is the time-varying transition model*. By a time-varying transition model, we mean that the state transition distribution is time-varying (e.g., it can be caused by the dynamic disturbances of the environment). This differentiates our proposed framework from the time-dependent formulations [7], in which the transition model is used to represent the stochastic travel duration between pairwise states.

Time-varying transition model can be predicted by utilizing the forecast environmental dynamics. E.g., the forecast data of ocean currents in next few days can define a predictable planning horizon, within which the extracted disturbance dynamics can be used to predict the robot's future state transition properties.

Specifically, since both the robot's action a and the external disturbance d are usually two independent forms of sources that cause the robot's (uncertain) motion, similar to the transition distribution $T_a(s, s')$ caused by the action a, we use $T_d(s, s')$ to represent the transition distribution caused by the pure disturbance d. The synergistic transition model $T_{ad}(s, s')$ caused by both the action a and the disturbance d can be obtained by unifying their two independent transition distributions, and all those distributions can be associated with time t if the action/disturbance is a function of time. Formally,

$$T_{ad}(s, s', t) = T_a(s, s', t) \oplus T_d(s, s', t) \tag{6}$$

where the operator $\oplus$ "combines" two distributions captured (or predicted) at moment t. The operations of combining probability distributions will depend on specific distribution types, e.g., see [19], [20]. In our robotics motion planning scenario, we assume both $T_a(s, s', t)$ and $T_d(s, s', t)$ follow independent Gaussian distributions.

B. Estimation of Transition Time

Before estimating the time-varying transition models corresponding to the future moments, the real-valued transition times traveling from current state to all other states need to be estimated first. Intuitively, the transition time estimation is a process of "look ahead into future." One might also imagine this as a *forward* process, as time evolves in a forward direction.

However, such a forward estimation cannot be done in a straightforward way. This is because the forward process/propagation is usually used on a deterministic acyclic graph, but in the MDP state transition topology, there are both cycles and stochasticity. The estimation from one state to another actually relies on each other, resulting in a chicken and egg problem. In addition, we need to estimate not only the time of one-step look-ahead, but also those times many steps ahead corresponding to all other distant states within the planning horizon.

Our approach consists of two steps. The first step is to estimate "local" transition time from each state $s \in S$ to its *one-hop* succeeding states $\mathcal{N}(s) \subseteq S$ that s can directly transit to. Building on this, the second step is to estimate the "global" transition time from the robot's current state s_0 to all other *multi-hop* states $S \backslash \mathcal{N}(s_0)$. Details are as follows.

Local One-Hop Transition Time Estimate: The concept of one-hop can also be interpreted as one-step look ahead, since it measures the agent's travel time from current state to the one-hop adjacent neighboring states. Note, the one-hop time estimate is *domain-specific* and may be done in multiple ways. For example, with a known transition model that essentially describes the state hopping probability distribution from the current state to neighboring states, the one-hop time can be tested and estimated offline by Monte Carlo trials. There are also other closed-form solutions by taking advantage of the known state transition probability distributions, and an application example in the marine robotic planning scenario is provided in the Appendix.

Global Multi-Hop Transition Time Estimate: Using the local one-hop transition time, we can then estimate the global multi-hop transition time $t(s, s_e)$ from an arbitrary state s to an end state $s_e \in S$ which does not immediately succeed s. Multi-hop transition time estimation is *domain-independent*.

Estimating the global multi-hop transition time is challenging because the time estimate from an arbitrary state s to another multi-hop state s_e needs to take into account many combinations of possible hopping scenarios due to the underlying cyclic nature. This causes interrelated dependence among states: estimation of arrival time at s_e travelled from s relies on a starting time at s, which essentially relies on the time estimates of all other states that directly or indirectly connect to s, including the state s_e.

One possible solution is to compute the *first passage/hitting* time at state s_e. However, the classic *Markov Chain first-passage times* method [21] does not apply here because it only considers and analyzes, the number of hops, instead of estimating the real-valued travel time.

Our solution is to formulate the problem using Kolmogorov equations [22]. For example, from s_1, the transition time $t(s_1, s_e)$ can be represented by an expectation $\mathbf{E}(t(s_1, s_j^{(1)}) + t(s_j^{(1)}, s_e))$, where $s_j^{(1)} \in \mathcal{N}(s_1)$ is s_1's one-hop succeeding states, and $t(s_j^{(1)}, s_e)$ is again a multi-hop transition time from $s_j^{(1)}$ to ending state s_e. Similarly, we can formulate an expression for the transition time $t(s_i, s_e)$ for all other $s_i \in S$ to s_e:

$$\begin{cases} t(s_1, s_e) = \sum_{s_j^{(1)} \in \mathcal{N}(s_1)} T_a\left(s_1, s_j^{(1)}, t\right)\left(t\left(s_1, s_j^{(1)}\right) + t\left(s_j^{(1)}, s_e\right)\right) \\ \qquad\qquad \vdots \\ t(s_n, s_e) = \sum_{s_j^{(n)} \in \mathcal{N}(s_n)} T_a\left(s_n, s_j^{(n)}, t\right)\left(t(s_n, s_j^{(n)}\right) + t\left(s_j^{(n)}, s_e\right)\right) \end{cases}$$

where $t(s_e, s_e) = 0$ and each $t(s_i, s_j^{(i)})$ denotes the previously obtained one-hop transition time.

We have a total of $|S|$ variables and $|S|$ equations, which form a linear system. From the current state s_0, there are $|S \backslash \mathcal{N}(s_0)|$ multi-hop non-succeeding states. Therefore, we need to solve $|S \backslash \mathcal{N}(s_0)|$ linear systems each of which specifies a different end state $s_e \in S \backslash \mathcal{N}(s_0)$. Solving (sparse) linear systems is the most time expensive part during each value iteration, and it requires a complexity of $O(|S|^{2.3})$ [23]. Thus the overall time complexity to compute all $|S|$ estimates is $O(|S| * |S|^{2.3})$ for each iteration.

C. Synergy of Spatial and Temporal Processes

The final step to solve the proposed TVMDP is to propagate policy values in order to maximize optimal actions.

After having obtained the transition time t from the robot's current state s_0 to an arbitrary state s, the transition model at state s and time t can be constructed:

$$T_{ad}(s, s', t) = T_{ad}(s, s', t(s_0, s)). \tag{7}$$

where $s' \in S$ are the possible successor states from s.

In contrast to the forward time estimation process, the value propagation is more like a *backward* process: the propagation process employs a Bellman backup mechanism and propagates the rewards from distant states in the future back to the current

Algorithm 1: Value Iteration (with agent's state s_0).

Initialize: $k \leftarrow 1$,
foreach $s \in S$ **do**
  Initialize $V_0(s) = 0$, $t(s_0, s) = 0$
/ value propagation in spatial channel */*
foreach $s \in S$ **do**
  $\pi_k^*(s) = \texttt{argmax}_{a \in A} \sum_{s' \in \mathcal{N}(s)} T_a(s, s', t(s_0, s)) \cdot$
    $\left(R_a(s, s', t(s_0, s)) + \gamma V_{k-1}(s', t(s_0, s'))\right)$
  update optimal action $a^*(s) = \pi_k^*(s)$
/ transition time estimates in temporal channel */*
foreach $s \in S$ **do**
  foreach $s_j \in \mathcal{N}(s)$ **do**
    Estimate one-hop transition time $t(s, s_j)$
foreach $s \in S \setminus \mathcal{N}(s_0)$ **do**
  Estimate multi-hop transition time $t(s_0, s)$
$k \leftarrow k + 1$, **goto** Step 5
Terminate algorithm if given tolerance is reached

state at the current moment. To improve estimation accuracy, the forward time estimate process and backward value propagation process need to alternate iteratively until a certain (preset) convergence threshold is met.

After each iteration of Bellman backup, the action policy is updated. Based on the updated policy, the transition time estimates from the current state to all other states $t(s_0, s)\ \forall s \in S$ are calculated, which are then used to update the time-varying transition model $T_{ad}(s, s', t(s_0, s))$ at s. The updated transition model is in turn utilized for next value iteration/propagation and time estimate. The corresponding pseudo-code is described in Algorithm 1. In essence, the underlying computing mechanism can be imagined as value iteration that combines both spatial "expansion" and temporal "evolution," which occur simultaneously in two separate channels. This is a solution mechanism that fundamentally differs from that of the classic MDP. In fact, the MDP's Bellman backup procedure can be regarded as the *spatial expansion* process, and thus the MDP is a special case of TVMDP.

The proposed time estimation formulations produce unique solutions (solving them does not involve iterative optimization processes). Thus after we integrated the time estimations with the MDP value iterations, the convergence behavior mainly depends on the standard value iterations. The algorithm will converge because along the algorithmic stages, both the time estimates and state values (updated by value iterations) are constantly improved and become more and more accurate. Since value iteration is known to be near-optimal depending on the convergence stopping tolerance, the solution to TVMDP is thus expected to be near-optimal depending on the accuracy of time estimation. Formally, the solution can become "suboptimal" only when the time estimates are inaccurate compared to ground truth (e.g., local time estimate model is imperfect, forecast data and dynamics are inaccurate, vehicle state and control are noisy, see Appendix).

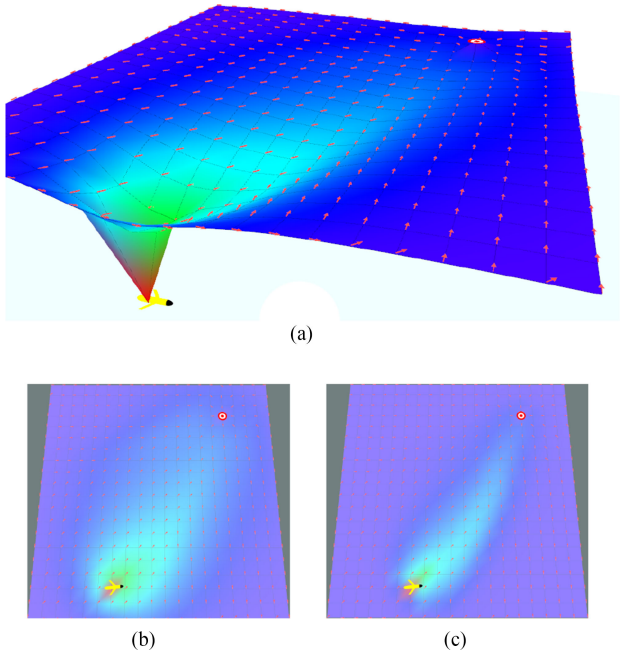


Fig. 2. (a) TVMDP action policy (red arrows) on a funnel-like surface where the height of funnel represents estimated transition time from the robot's state (funnel bottom); (b)(c) Policy maps projected onto 2D spatial dimensions. The colormap implies reachability in space. Different magnitudes of action uncertainty result in differing areas of reachable regions. The red target symbol is the goal state to reach.

V. Experiments

We validated our method in an ocean monitoring scenario, where the ocean currents vary both spatially and temporally. An underwater glider simulator written in C++ was built in order to test the proposed decision-making framework. The simulation environment was constructed as a two dimensional ocean surface, and we tessellated the environment into a grid map. We represent the center of each cell/grid as a state, where each non-boundary state can transit in eight directions (N, NE, E, SE, S, SW, W, NW) and a ninth idle action (returning to itself). Time-varying ocean currents are external disturbances for the robot and are represented as a vector field.

We first investigated the policy patterns generated from the proposed framework with time-varying transition models. Fig. 2(a) shows a policy map (red arrows) on a funnel-like surface where the height of the funnel is a measure of estimated transition time from the robot state (bottom of the funnel). Fig. 2(b) and (c) are projected policy maps onto a 2D plane. A brighter region implies a larger chance of being visited by the robot, and the difference of brighter regions in two figures reveals differing "magnitudes" of action uncertainty (Fig. 2(b) has larger action uncertainty than Fig. 2(c)).

To evaluate the proposed method, we compare it with other relevant approaches including the standard MDP, the discrete-time MDP (DTMDP), as well as a modified version of time-dependent MDP (TDMDP) [7] since it shares certain similarity with our framework.

We start by comparing TVMDP with MDP and DTMDP. The DTMDP extends from MDP by adding a time dimension, i.e.,

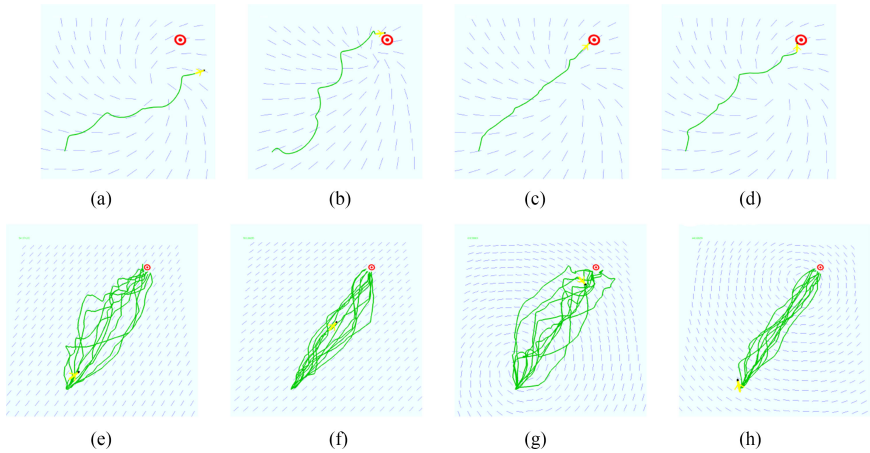


Fig. 3. The top four pictures demonstrate trajectories from (a) MDP; (b) DTMDP with low time-discretization resolution; (c) DTMDP with high time-discretization resolution; (d) TVMDP. The disturbance is a dynamic vortex-like vector field. The bottom four pictures show accumulated trajectories of many trials resulted from ATMDP and TVMDP, in spinning and vortex vector fileds, respectively. (a) MDP, (b) DTMDP low, (c) DTMDP high, (d) TVMDP, (e) ATMDP, (f) TVMDP, (g) ATMDP, and (h) TVMDP.

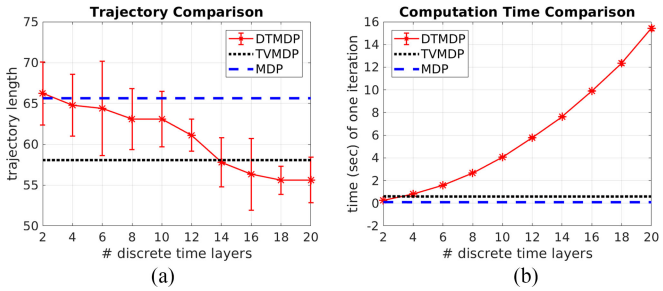


Fig. 4. Performance comparisons between TVMDP, MDP, and DTMDP. The two dashed lines are averaged values from MDP and TVMDP, respectively. The grid map has a dimension of 10×10, thus the number of states is 100 for MDP and TVMDP, and is $100d_t$ for DTMDP where dt is the number of time layers.

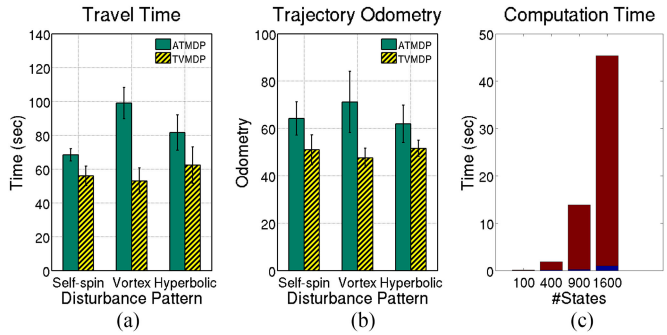


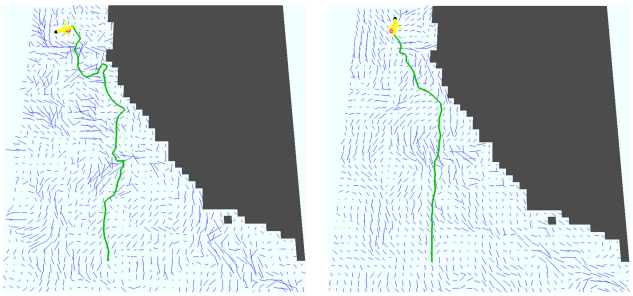
Fig. 5. Performance comparisons between ATMDP and the TVMDP, under differing disturbance patterns. (a) Trajectory odometry (lengths); (b) Overall travel time; (c) Computational time required by TVMDP to generate solutions. The red parts are the time used for solving linear systems.

duplicating MDP states into a series of discretized time layers. This method relies on a pre-defined time horizon as well as discretization resolution. Fig. 3(a)–(d) demonstrate trajectories from these methods under a dynamic vortex-like disturbance field. We can observe that, DTMDP is very sensitive to the resolution of time discretization. Specifically, the method generates a good result when the resolution is high enough, but produces a bad performance when the resolution is low. Statistics in Fig. 4 reveal more details. We can see from Fig. 4(a) that, the trajectory length/cost of DTMDP has an obvious decreasing trend as the number of time layers (time-discretization resolution) grows, and can actually result in a better performance than TVMDP if the time resolution is high enough. The reason for the sub-optimality of TVMDP is due to imperfect time estimates (see approximation method in Appendix). Note, however, an important drawback of DTMDP lies in its prohibitive computational cost. Fig. 4(b) shows computational costs for a 10×10 grid map and only one value iteration. This implies that the prohibitive cost of DTMDP makes it less useful for most of application scenarios.

We also compared TVMDP with a modified version of time-dependent MDP (TDMDP) and we call it *approximate time-varying MDP (ATMDP)*. (The TDMDP is not straightforwardly applicable in our scenario due to the reasons we analyzed earlier. We modified it by employing an approximation method of *prioritized sweeping* [9] which essentially propagates the largest value function changes in priority through the state space, such that the dilemmas caused by the stochastic and cyclic topology are mitigated and a look-ahead time estimate solution is approximated.) Fig. 3(e)–(h) show trajectories produced from the ATMDP and TVMDP. We can see that, the trajectories of our method are much smoother and shorter.

Statistics with regard to trajectory lengths and time costs for TVMDP and ATMDP are provided in Fig. 5(a) and (b). These results indicate that the TVMDP method leads to smaller travel distances and shorter travel times. We use the `Eigen` iterative sparse matrix solver to compute linear systems for estimating multi-hop transition time. Fig. 5(c) shows that our method requires $\sim$15 seconds for $\sim$1000 states and $\sim$50 seconds for $\sim$2000 states (on a desktop with 3.30 GHz Intel $i7$ CPU).

Finally, we also tested the algorithms using real ocean current data. The simulator is able to read and process data from the Regional Ocean Model System (ROMS) [24] which predicts/forecasts ocean currents up to 72 hours in advance. This allows us to utilize these ocean predictions to model the



(a) (b)

Fig. 6. Trajectory results from running on ROMS data. (a) Trajectory from ATMDP; (b) Trajectory from TVMDP. (a) ATMDP and (b) TVMDP.

temporal dimensional transition dynamics. However, ROMS provides eight datasets for one day (every three hours). This means the data is not time-continuous. To address this, we use Gaussian Processes Regression (GPR) to interpolate and extrapolate the missing parts. Fig. 6 shows results from running the ATMDP and the TVMDP. We can observe that, comparing with TVMDP, only the beginning part of ATMDP trajectory is good. This is because ATMDP uses an approximation method that can only provide very rough time estimation results, and the approximation errors are propagated as the horizon grows, which eventually leads to poor policies for the later part of the trajectory.

VI. Conclusion

We presented a time-varying MDP framework called TVMDP which is able to handle transition models that vary both spatially and temporally. We show that in order to solve the TVMDP, the basic Bellman backup mechanism used for computing the MDP is not enough. Instead, we developed mechanisms to estimate the temporal parameter as well as integrate the time-varying stochasticity. Our proposed mechanism consists of two iterative value propagation processes that proceed in both spatial and temporal dimensions, where the spatial process involves Bellman backup but the temporal dimension is expanded using Kolmogorov equations. Finally we validated our method with various dynamic disturbances including those from real ocean data. The results show that comparing to the conventional methods, our approach produces high quality solution with a low computational cost.

Appendix

A. Marine Robotic Motion Planning Example

The purpose of this appendix is to demonstrate a possible way of estimating the real-valued one-hop transition time described in Section IV-B.

Since the estimation of any real-valued transition time is based on continuous state space, we first need to convert the conventional discrete MDP state space to a continuous one. There are many methods to do so (e.g., [25], [26]). To facilitate the description of the main algorithm, we adopt a simple strategy that maps from/to the other within certain resolution. The continuous form of action and external disturbance are also defined accordingly:

- State $\mathbf{x}$ is the counterpart of MDP state s but in continuous state space. When $\mathbf{x}$ coincides exactly at s, we denote it as $\mathbf{x}(s)$ in continuous space. We can also map $\mathbf{x}$ back to discrete space: $\mathbf{x} \mapsto s$ if $||\mathbf{x} - \mathbf{x}(s)||$ is less than discrete space partition resolution;
- Local control/action reference $\mathbf{a}(s)$ at s is a directional vector, where the vector direction is defined by the discrete action $a \in A$, and the vector magnitude is defined by the robot's actual control inputs.
- Vector $\mathbf{d}(s)$ expresses external disturbance at s.

Consequently, if the transition from state s to a succeeding state s' is deterministic, the transition time can be approximated by $||\mathbf{x}(s) - \mathbf{x}(s')||/v$, or simply $||\mathbf{x} - \mathbf{x}'||/v$, assuming the robot's speed v is a constant in a static environment.

We assume the outcomes of $\mathbf{a}(s)$ and $\mathbf{d}(s)$ are stochastic and specifically they follow independent Gaussian distributions. After a travel time $\mathcal{T}$, the robot's movement translation $\mathbf{x}$ (denoting arriving state) after applying $\mathbf{a}(s)$ and being disturbed by $\mathbf{d}(s)$ also follows a Gaussian distribution:

$$f_a(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\right) \quad (8)$$

where μ and Σ are the mean and covariance of $\mathbf{x}$, respectively. It is worth mentioning that the MDP may produce multiple optimal actions (with equal optimal value) at some state. In such a case, a *mixture distribution* can be used,

$$f_{\{a_1,\ldots,a_k\}}(\mathbf{x}) = \sum_{i=1}^{k} w_i f_{a_i}(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} f_{a_i}(\mathbf{x}), \quad (9)$$

where the weighting parameter w_i for component PDFs are identical as actions have the same optimal value.

Let $\{\mathbf{a}^*(s)\}$ be the set of optimal actions at state s and at time t, i.e., s_t for short, the time-varying transition model thus can be expressed as

$$T_{ad}(s, s', t) = \Pr\big(s' \mid s_t, \{\mathbf{a}^*(s_t)\}, \mathbf{d}(s_t)\big). \quad (10)$$

In practice, such discrete probability mass function is approximated by integrating Eq. (8) over discretized volumes.

With the above model, we can proceed to compute the local one-hop transition time estimate.

Due to the stochastic nature of the transition model in Eq. (10), the robot in state s may eventually arrive at any one-hop succeeding state $s' \in \mathcal{N}(s)$. However, the estimation of transition time $t(s, s')$ is based on the assumption that the robot will reach a designated next state s' with probability 1. To satisfy this assumption, we need to choose an action $\tilde{\mathbf{a}}(s)$ with which the robot motion is exactly toward s'. Let $\tilde{\mathbf{r}}(s) = \tilde{\mathbf{a}}(s) + \mathbf{d}(s)$ denote the resultant of such selected action and environmental disturbance at s.

To simplify the calculation, one way is to transform the coordinate system such that the robot's motion direction $\frac{\tilde{\mathbf{r}}(s)}{||\tilde{\mathbf{r}}(s)||}$ is exactly on an arbitrary coordinate basis, built on which the multivariate PDF (Eq. (10)) can be approximated by a univariate PDF by marginalization.

Fig. 7 shows an illustration with robot state s located at the coordinate origin. θ is the angle between $\tilde{\mathbf{r}}(s)$ and a basis x_1.

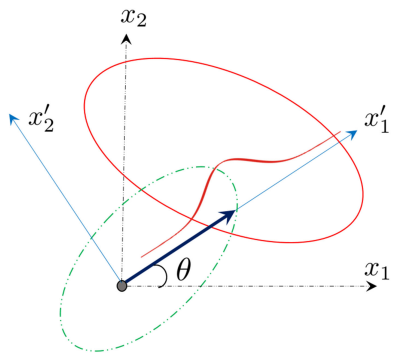


Fig. 7. Transition time estimation in 2D. The black dot at the origin represents the robot's current state s; the red solid ellipse denotes the state continuous distribution $f(\mathbf{x})$ under $\tilde{\mathbf{r}}(s) = \tilde{\mathbf{a}}(s) + \mathbf{d}(s)$ (the bold black arrow); the green dashed ellipse represents a contour of $\mathbf{r}(s) = \mathbf{a}(s) + \mathbf{d}(s)$ for all allowable $\mathbf{a}(s)$ given that $\mathbf{d}(s)$ is fixed.

To transform the coordinate system such that $\tilde{\mathbf{r}}(s)$ is on x_1, the coordinate needs to rotate with corresponding rotation matrix $R(\theta)$. More generally, with a rotation matrix R and replacing $\mathbf{x} = R\tilde{\mathbf{x}}$ in Eq. (8), we obtain a distribution in the transformed coordinate system:

$$f(\tilde{\mathbf{x}}) \propto \exp\left(-\frac{1}{2}(\tilde{\mathbf{x}} - R^{-1}\mu)^T R^T \Sigma^{-1} R(\tilde{\mathbf{x}} - R^{-1}\mu)\right)$$
$$\propto \exp\left(-\frac{1}{2}(\tilde{\mathbf{x}} - R^T\mu)^T (\tilde{\Sigma})^{-1}(\tilde{\mathbf{x}} - R^T\mu)\right)$$

where $\tilde{\mu} = R^T\mu$ and $\tilde{\Sigma} = R^T\Sigma R$ after transformation.

Next we can calculate the *expectation* of resultant states in the robot's motion direction. Let x_i denote a selected i-th basis (variable) in the direction of motion in the *new coordinate system*, and $\mathbf{x}_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$ represent all other variables. Then the conditional expectation of x_i can be calculated by

$$\mathbf{E}(x_i|\mathbf{x}_{-i} = \mathbf{0}) = \int x_i f(x_i|\mathbf{x}_{-i}) I_{\{\mathbf{x}_{-i}=\mathbf{0}\}}(\mathbf{x}_{-i})\mathrm{d}\mathbf{x}$$
$$= \int x_i \frac{f(x_i, \mathbf{x}_{-i})}{f(\mathbf{x}_{-i})} I_{\{\mathbf{x}_{-i}=\mathbf{0}\}}(\mathbf{x}_{-i})\mathrm{d}\mathbf{x}$$
$$= \int_0^{\tau} \int x_i \frac{f(x_i, \mathbf{x}_{-i})}{f(\mathbf{x}_{-i})} I_{\{\mathbf{x}_{-i}=\mathbf{0}\}}(\mathbf{x}_{-i})\mathrm{d}\mathbf{x}_{-i}\mathrm{d}x_i$$

where indicator variable $I_{\{\mathbf{x}_{-i}=\mathbf{0}\}}(\mathbf{x}_{-i}) = 1$ if $\mathbf{x}_{-i} = \mathbf{0}$ and 0 otherwise. τ can be either $+\infty$ (motion in the direction of x_i) or $-\infty$ (motion against the direction of x_i).

Since states $\tilde{\mathbf{x}}$ are obtained by applying $\tilde{\mathbf{r}}(s)$ for a fixed time $\mathcal{T}$, the local one-hop transition time $t(s, s')$ is approximated by

$$t(s, s') = \frac{||\mathbf{x} - \mathbf{x}'||}{(\mathbf{E}(x_i|\mathbf{x}_{-i} = \mathbf{0})/\mathcal{T})} \tag{11}$$

where $||\mathbf{x} - \mathbf{x}'||$ represents the translation defined as before and the denominator is the estimated velocity given that the destination state is s'.

REFERENCES

[1] C. Boutilier, T. Dean, and S. Hanks, "Decision-theoretic planning: Structural assumptions and computational leverage," *J. Artif. Intell. Res.*, vol. 11, pp. 1–94, 1999.

[2] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Belmont, MA, USA: Athena Scientific, 2000.

[3] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. Hoboken, NJ, USA: Wiley, 1994.

[4] B. Morier and V. Teles, "A time-varying markov-switching model for economic growth," Escola de Economia de São Paulo, Getulio Vargas Foundation, Brazil, Textos para discussão 305, 2011.

[5] L. Agnello, G. Dufrénot, and R. Sousa, "Using time-varying transition probabilities in Markov switching processes to adjust us fiscal policy for asset prices," *Econ. Modell.*, vol. 34, pp. 25–36, 2013.

[6] R. Hosseini, N. Le, and J. Zidek, "Time-varying Markov models for binary temperature series in Agrorisk management," *J. Agricultural, Biol., Environ. Statist.*, vol. 17, no. 2, pp. 283–305, 2012.

[7] J. A. Boyan and M. L. Littman, "Exact solutions to time-dependent mdps," in *Proc. Adv. Neural Inf. Process. Syst*. MIT Press, Cambridge, MA, USA, 2000, pp. 1026–1032.

[8] A. V. Donati, R. Montemanni, N. Casagrande, A. E. Rizzoli, and L. M. Gambardella, "Time dependent vehicle routing problem with a multi ant colony system," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1174–1191, 2008.

[9] E. Rachelson, P. Fabiani, and F. Garcia, "Timdppoly: An improved method for solving time-dependent mdps," in *Proc. Int. Conf. Tools Artif. Intell.*, 2009, pp. 796–799.

[10] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Mach. Learn.*, vol. 13, no. 1, pp. 103–130, 1993.

[11] B. Bonet and H. Geffner, "Labeled RTDP: Improving the convergence of real-time dynamic programming," in *Proc. Proc. 13th Int. Conf. Automated Plan. Scheduling*, 2003, pp. 12–21.

[12] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[13] P. Dayan and Y. Niv, "Reinforcement learning: The good, the bad and the ugly," *Curr. Opin. Neurobiol.*, vol. 18, no. 2, pp. 185–196, 2008.

[14] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Jul. 2013.

[15] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, pp. 9–44, 1988.

[16] G. Tesauro, "Temporal difference learning and td-gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[17] J. Boyan, "Least-squares temporal difference learning," in *Proc. 16th Int. Conf. Mach. Learn*. Morgan Kaufmann, San Mateo, CA, USA, 1999, pp. 49–56.

[18] R. Sutton, D. Precup, and S. Singh, "Between MDPS and semi-MDPS: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, pp. 181–211, 1999.

[19] R. Ranjan and T. Gneiting, "Combining probability forecasts," *J. Roy. Statist. Soc. B, Statist. Methodol.*, vol. 72, no. 1, pp. 71–91, 2010.

[20] R. L. Winkler, "Combining probability distributions from dependent information sources," *Manage. Sci.*, vol. 27, no. 4, pp. 479–488, 1981.

[21] A. J. Siegert, "On the first passage time probability problem," *Phys. Rev.*, vol. 81, no. 4, 1951, Art. no. 617.

[22] A. N. Kolmogorov, "Über die analytischen Methoden in der Wahrscheinlichkeitsrechnung," *Math. Annal.*, vol. 104, no. 1, pp. 415–458, 1931.

[23] G. H. Golub and C. F. Van Loan, *Matrix Comput.*, 4th ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 1996.

[24] A. F. Shchepetkin and J. C. McWilliams, "The regional oceanic modeling system (ROMS): A split-explicit, free-surface, topography-following-coordinate oceanic model," *Ocean Modell.*, vol. 9, no. 4, pp. 347–404, 2005.

[25] M. Toussaint and A. Storkey, "Probabilistic inference for solving discrete and continuous state Markov decision processes," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 945–952.

[26] S. Sanner, K. V. Delgado, and L. N. de Barros, "Symbolic dynamic programming for discrete and continuous state MDPs," in *Proc. Twenty-Seventh Conf. Uncertainty Artif. Intell.*, F. Cozman and A. Pfeffer, Eds. AUAI Press, Arlington, VA, USA, 2012, pp. 643–652.