informs
https://pubsonline.informs.org/journal/mnsc

MANAGEMENT SCIENCE
Articles in Advance, pp. 1–19
ISSN 0025-1909 (print), ISSN 1526-5501 (online)

Frozen-State Value Iteration: Faster Reinforcement Learning by Freezing Slow States

Yijia Wang,[a] Daniel R. Jiang[a,b,*]

[a] University of Pittsburgh, Pittsburgh, Pennsylvania 15260; [b] Meta Platforms, Menlo Park, California 94025
*Corresponding author
Contact: yiw94@pitt.edu, https://orcid.org/0000-0001-8606-4846 (YW); danielrjiang@gmail.com, https://orcid.org/0000-0002-5388-8061 (DRJ)

Received: January 2, 2023
Revised: April 6, 2025; July 13, 2025
Accepted: September 4, 2025
Published Online in Articles in Advance: March 18, 2026

https://doi.org/10.1287/mnsc.2023.00012

Abstract. We study infinite-horizon Markov decision processes (MDPs) with fast–slow structure, in which some state variables evolve rapidly (fast states), whereas others change more gradually (slow states). This structure commonly arises in practice when decisions must be made at high frequencies over long horizons and when slowly changing information still plays a critical role in determining optimal actions. Examples include inventory control under slowly changing demand indicators or dynamic pricing with gradually shifting consumer behavior. Modeling the problem at the natural decision frequency leads to MDPs with discount factors close to one, making them computationally challenging. We propose a novel approximation strategy that freezes slow states during phases of lower level planning and subsequently applies value iteration to an auxiliary upper level MDP that evolves on a slower timescale. Freezing states for short periods of time leads to easier to solve lower level problems, whereas a slower upper level timescale allows for a more favorable discount factor. On the theoretical side, we analyze the regret incurred by our frozen-state approach, and this leads to simple insights on how to trade off regret versus computational cost. Empirically, we benchmark our new frozen-state methods on three domains: (i) inventory control with fixed order costs, (ii) a grid world problem with spatial tasks, and (iii) dynamic pricing with reference price effects. We demonstrate that the new methods produce high-quality policies with significantly less computation, and we show that simply omitting slow states is often a poor heuristic.

History: Accepted by J. George Shanthikumar, data science.
Funding: This research is based upon work supported by the U.S. National Science Foundation [Grant 1807536].
Supplemental Material: The online appendix and data files are available at https://doi.org/10.1287/mnsc.2023.00012.

Keywords: reinforcement learning • approximate dynamic programming • value iteration • hierarchical reinforcement learning

1. Introduction

Markov decision processes (MDPs) offer a foundational framework for modeling sequential decision making under uncertainty. In many real-world applications, decisions must be made at high frequencies over extended time horizons. From a modeling standpoint, this translates into MDPs with discount factors close to one (which captures the long-term nature of the objective). However, such long-horizon problems can pose computational challenges: standard algorithms such as value iteration (VI), which is a core building block in many reinforcement learning (RL) algorithms, rely on the contraction property of the Bellman operator to guarantee convergence. As the discount factor approaches one, this contraction weakens, leading to slow convergence and degraded performance within a given computational budget (Bertsekas and Tsitsiklis 1996, Jiang et al. 2015).

In a wide range of applications, state variables naturally evolve at different timescales. For example, in inventory control problems, the inventory level may fluctuate rapidly from one period to the next, whereas exogenous factors that influence demand may evolve more gradually. In dynamic pricing problems, the (endogenous) price history can change frequently, but user behavior shifts more slowly. We refer to this as fast–slow structure, in which the state decomposes into fast-evolving and slow-evolving components. Although slow states may evolve gradually, they often encode critical contextual information, and omitting them can result in suboptimal policies and high regret.

Solving MDPs that fully capture both fast and slow dynamics at the natural decision frequency typically requires long effective horizons, which can pose serious computational burdens. To manage complexity, practitioners often resort to simplifying assumptions, such as fixing or omitting slow states altogether. Whereas this reduces computational burden, it can degrade policy quality by discarding important information. This paper introduces and formalizes the notion of fast–slow

MDPs and studies the computational implications of a novel approximation strategy called frozen-state value iteration (FSVI).

1.1. Main Contributions

In this paper, we propose an approximation strategy that uses two levels of planning, aiming to provide a middle ground between solving the full MDP and completely ignoring slow states. In our approach, we first temporarily freeze the slow state during lower level planning. Concretely, we compute policies based on finite-horizon MDPs that are conditioned on a fixed slow state. Subsequently, we apply value iteration to an auxiliary upper level MDP that evolves on a slower timescale by leveraging the lower level policies. This approach makes the lower level problems more tractable, whereas the slower upper level dynamics enable the use of a more favorable (i.e., smaller) effective discount factor. Specifically, we make the following contributions:

i. We first formally define a fast–slow MDP and provide an (exact) reformulation into an MDP with hierarchical structure. The upper level is a slow-timescale infinite-horizon MDP, and the lower level is a fast-timescale finite-horizon MDP with T periods. One period of the upper level problem is composed of a complete lower level problem.

ii. We propose a frozen-state approximation to the reformulated MDP along with an associated FSVI algorithm, in which the slow state is frozen in the lower level problem, whereas each period in the upper level problem releases the slow state. Computational benefits arise in several ways: (i) frozen states simplify the dynamics of the lower level MDP (fewer successor states); (ii) the discount factor in the upper level problem is more favorable because of the reuse of the lower level policy, which is computed only once; (iii) the lower level MDP, thus, becomes separable into independent MDPs, opening the door to speedups via parallel computation. Solving the frozen-state approximation gives a policy that switches between one action from the upper level policy and $T-1$ actions from the lower level policy. We give a theoretical analysis that upper bounds the expected regret from applying this policy compared with the optimal policy.

iii. We adapt our frozen-state approach to a more practical reinforcement learning setting in which a generative model (or simulator) is available, but the MDP is not fully known. Specifically, we develop frozen-state fitted value iteration (FSFVI), a version of a fitted value iteration (FVI) (Munos and Szepesvári 2008) that incorporates our two-level planning strategy. FSFVI can learn effective policies from sampled transitions, making our method applicable to model-free or simulation-based environments. We also provide a theoretical regret bound for a special case of FSFVI using linear function approximation (see Online Appendix C).

iv. Lastly, we perform a systematic empirical study on three problem settings: (i) inventory control with fixed costs, (ii) grid world with spatial tasks, and (iii) dynamic pricing with reference effects. We show that our proposed algorithms (based on the frozen-state approximation) quickly converge to good policies using significantly less computation compared with standard methods. Notably, our results show that ignoring the slow state, that is, modeling the system as if the true state variable only includes the fast component,[1] leads to poor results.

2. Related Work

In this section, we provide a brief review of related literature. First, there exists a stream of literature focused on sequential decision-making problems with exact hierarchical, multitimescale structure. Chang et al. (2003) study multitimescale MDPs, which are composed of M different decisions that are made on M different discrete timescales. The authors consider the impact of upper level states and actions on the transition of the lower levels, an idea that is also present in our fast–slow framework. Multitimescale MDPs are often applied in supply chain problems, including production planning in semiconductor fabrication (Panigrahi and Bhatnagar 2004, Bhatnagar and Panigrahi 2006), hydropower portfolio management (Zhu et al. 2006), and strategic network growth for reverse supply chains (Wongthatsanekorn et al. 2010). Wang et al. (2018) propose a row generation–based algorithm to solve a linear programming formulation of the multitimescale MDP. Jacobson et al. (1999) consider piecewise stationary MDPs, in which the transition and reward functions are renewed every $N+1$ periods, motivated by problems in which routine decisions are periodically interrupted by higher level decisions. For the case of large renewal periods, they propose a policy called the "initially stationary policy," which uses a fixed decision rule for some number of initial periods in each renewal cycle. Our fast–slow model focuses on a novel fast–slow structure present in many MDPs and, unlike the above work, does not assume any natural or exact hierarchical structure. Instead, we focus on how a particular type of (frozen-state) hierarchical structure can be used as an approximation to the true MDP. However, we note that many MDPs with a natural two-timescale structure can also fit into our framework, and therefore, given that perspective, our model can be viewed as a generalization.

Our proposed frozen-state algorithms are also related to literature on hierarchical reinforcement learning, which are methods that artificially decompose a complex problem into smaller subproblems (Barto and Mahadevan

2003). Approaches include the options framework (Sutton et al. 1999), the hierarchies of abstract machines approach (Parr and Russell 1998), and MAXQ value function decomposition (Dietterich 2000).

Out of these three approaches, the options framework is most closely related to this paper. A Markov option (also called a macroaction or temporally extended action) is composed of a policy, a termination condition, and an initiation set (Sutton et al. 1999, Precup 2000). One of the biggest challenges is to automatically construct options that can effectively speed up reinforcement learning. A large portion of work in this direction is based on subgoals: states that might be beneficial to reach (Digney 1998, McGovern and Barto 2001, Jonsson and Barto 2005, Ciosek and Silver 2015, Wang et al. 2023). The subgoals are identified by utilizing the learned model of the environment (Menache et al. 2002, Mannor et al. 2004, Şimşek and Barto 2004, Şimşek et al. 2005) or through trajectories without learning a model (McGovern and Barto 2001, Stolle and Precup 2002).

The options (and subgoals) framework is largely motivated by robotics and navigation-related tasks, whereas we are particularly interested in solving problems that arise in the operations research and operations management domains. The problems that we study do not decompose naturally into subgoals, leading us to identify and focus on the fast–slow structure, which does indeed arise naturally for many problems of interest.

Another work that is related to the options framework is Song and Xu (2022), who divide finite-horizon MDPs into two subproblems along the time horizon and concatenate their optimal solutions to generate an overall solution. Our paper also, in a sense, divides MDPs along the time horizon, but our work is quite different from that of Song and Xu (2022) in that we work on infinite-horizon problems and convert them into auxiliary problems that operate on a slower timescale, which takes advantage of reusable lower level policies. More importantly, the various methods we propose all build on the idea of freezing certain states to reduce computational cost; this is unique to our approach, and to our knowledge, this is a novel direction that has not been proposed before.

Finally, our work contributes to the broader literature on leveraging MDP structure to improve reinforcement learning and approximate dynamic programming algorithms. Prior research exploits structural properties of the value function, such as convexity (Pereira and Pinto 1991, Philpott and Guan 2008, Nascimento and Powell 2009, Benjaafar et al. 2022) and monotonicity (Papadaki and Powell 2002, Jiang and Powell 2015a) as well as structural features of the optimal policy (Kunnumkal and Topaloglu 2008). Additional examples include factored MDPs (Osband and Van Roy 2014) and weakly coupled systems (Killian et al. 2021, El Shar and Jiang 2023, Brown and Smith 2025, Nadarajah and Cire 2025). Our contribution to this line of work lies in identifying a new form of MDP structure, which then informs the design of a new algorithmic approach.

3. Fast–Slow MDPs

In this section, we introduce the base model, the original MDP to be solved, and formally introduce the notion of a fast–slow MDP. We then provide a hierarchical reformulation of the base model using fixed-horizon policies and show the equivalence (in optimal value) between the two models. Because of the considerable amount of notation in this paper, we provide a table of notation in Table 1 for reference.

3.1. Base Model

Consider a discrete-time MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{W}, f, r, \gamma \rangle$, where $\mathcal{S}$ is the finite state space; $\mathcal{A}$ is the finite action space; $\mathcal{W}$ is the space of possible realizations of an exogenous, independent and identically distributed (i.i.d.) noise process $\{w_t\}$ defined on a discrete probability space $(\Omega, \mathcal{F}, \mathbb{P})$; $f : \mathcal{S} \times \mathcal{A} \times \mathcal{W} \to \mathcal{S}$ is the transition function; $r : \mathcal{S} \times \mathcal{A} \to [0, r_{\max}]$ is the bounded reward function; and $\gamma \in [0, 1)$ is the discount factor for future rewards (Puterman 2014). The objective is

$$U^*(s) = \max_{\{\nu_t\}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, \nu_t(s_t)) \,\Big|\, s_0 = s\right], \qquad (1)$$

where states transition according to $s_{t+1} = f(s_t, a_t, w_{t+1})$ and we optimize over sequences of policies $\nu_t : \mathcal{S} \to \mathcal{A}$, which are deterministic mappings from states to actions. The expectation is taken over exogenous noise process $\{w_t\}_{t=1}^{\infty}$. We assume throughout that $\mathcal{S}$, $\mathcal{A}$, $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{S} \times \mathcal{A}$, and $\mathcal{X} \times \mathcal{Y}$ are equipped with the Euclidean metric,[2] which is naturally the case for many applications.

Assumption 1 (Separability and the Fast–Slow Property). *Suppose the following hold:*

i. *The state space $\mathcal{S}$ is separable and can be written as $\mathcal{S} = \mathcal{X} \times \mathcal{Y}$. We call $\mathcal{X}$ the slow state space and $\mathcal{Y}$ the fast state space.*

ii. *Let $s_t = (x_t, y_t) \in \mathcal{S}$, where $x_t \in \mathcal{X}$ is the slow state and $y_t \in \mathcal{Y}$ the fast state, $a_t \in \mathcal{A}$, and $w_{t+1} \in \mathcal{W}$. The transition dynamics $s_{t+1} = f(s_t, a_t, w_{t+1}) \in \mathcal{S}$ can be written with the notation:*

$$x_{t+1} = f_{\mathcal{X}}(x_t, y_t, a_t, w_{t+1}) \in \mathcal{X} \quad \text{and}$$
$$y_{t+1} = f_{\mathcal{Y}}(x_t, y_t, a_t, w_{t+1}) \in \mathcal{Y},$$

for some $f_{\mathcal{X}} : \mathcal{S} \times \mathcal{A} \times \mathcal{W} \to \mathcal{X}$ and $f_{\mathcal{Y}} : \mathcal{S} \times \mathcal{A} \times \mathcal{W} \to \mathcal{Y}$.

iii. *For any state $(x, y) \in \mathcal{S}$, action $a \in \mathcal{A}$, and exogenous noise $w \in \mathcal{W}$, suppose the one-step transitions of x and y satisfy*

$$\|y - f_{\mathcal{Y}}(x, y, a, w)\|_2 \le d_{\mathcal{Y}} \quad \text{and} \quad \|x - f_{\mathcal{X}}(x, y, a, w)\|_2 \le \alpha d_{\mathcal{Y}},$$

for some $d_{\mathcal{Y}} < \infty$ and $\alpha \in [0, 1]$.

Table 1. Frequently Used Notation in the Fast–Slow MDP Framework

Symbol	Description
$\mathcal{S}$	State space with generic element $s \in \mathcal{S}$
$\mathcal{X}$	Slow state space with generic element $x \in \mathcal{X}$
$\mathcal{Y}$	Fast state space with generic element $y \in \mathcal{Y}$
$\mathcal{A}$	Action space with generic element $a \in \mathcal{A}$
$\mathcal{W}$	Exogenous noise space with generic element $w \in \mathcal{W}$
f	Transition function: $f : \mathcal{S} \times \mathcal{A} \times \mathcal{W} \to \mathcal{S}$
$f_{\mathcal{X}}$	Slow state transition: $f_{\mathcal{X}} : \mathcal{S} \times \mathcal{A} \times \mathcal{W} \to \mathcal{X}$
$f_{\mathcal{Y}}$	Fast state transition: $f_{\mathcal{Y}} : \mathcal{S} \times \mathcal{A} \times \mathcal{W} \to \mathcal{Y}$
r	Reward function: $r : \mathcal{S} \times \mathcal{A} \to [0, r_{\max}]$
γ	Discount factor
α	Constant in the fast–slow property controlling changes in slow state
$d_{\mathcal{Y}}$	Bound on the one-step change in the fast state
T	Number of periods in the lower level problem
U^*	Optimal value function of the base MDP
ν^*	The optimal policy for the base MDP
L_r, L_f, L_U	Lipschitz constants for r, f, and U^*
$(\mu, \boldsymbol{\pi})$	Generic T-period policy with upper level policy μ and lower level policy $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_{T-1})$
$R(s_0, \mu(s_0), \boldsymbol{\pi})$	T-period reward of hierarchical reformulation associated with T-period policy $(\mu, \boldsymbol{\pi})$
$(\mu^*, \boldsymbol{\pi}^*)$	The optimal policy for the hierarchical reformulation; has same value as ν^*
$\tilde{R}(s_0, a, J_1)$	Approximation to T-period reward using lower level value function approximation J_1
J_t^*, V^k	Lower and upper level value function approximation used in FSVI
$(\tilde{\mu}^k, \tilde{\boldsymbol{\pi}}^*)$	Output policy of FSVI after k iterations
U^i	Value function approximation used in base VI
ν^k	Output policy of base VI after k iterations
H	Bellman operator for the base MDP
$\tilde{H}$	Bellman operator for lower level frozen-state problem (discount factor γ)
$F_{J_1, \boldsymbol{\pi}}$	Bellman operator for upper level problem given lower level J_1 and $\boldsymbol{\pi}$ (discount factor γ^T)

Remark 1. Note that one particularly instructive example is the case of exogenous slow states, where $x_{t+1} = f_{\mathcal{X}}(x_t, w_{t+1})$. Here, the transition does not depend on the action a_t, nor does it depend on the fast state y_t. Such a model is common in practice: examples of exogenous slow states include prices, weather conditions, and other environmental variables that are not influenced by the decision maker's actions or the states of the primary system. See, for example, Yu and Mannor (2009), who study a related model called the arbitrarily modulated MDP.

Assumption 2 (Lipschitz Properties). *Suppose that the reward function r, transition function f, and optimal value function U^* are Lipschitz with respect to $\|\cdot\|_2$:*

$$|r(s,a) - r(s',a')| \le L_r \|(s,a) - (s',a')\|_2, \tag{2}$$

$$\|f(s,a,w) - f(s',a',w)\|_2 \le L_f \|(s,a) - (s',a')\|_2, \tag{3}$$

$$\|U^*(s) - U^*(s')\|_2 \le L_U \|s - s'\|_2, \tag{4}$$

for some Lipschitz constants L_r, L_f, and L_U. Lipschitz assumptions are common in the literature; see, for example, Ok et al. (2018), Domingues et al. (2021), and Sinclair et al. (2020, 2022). In Online Appendix F, *we give bounds on L_U in terms of L_r and L_f. Whereas we could use those results directly and omit the assumption on L_U, we opt to include (4) to increase the clarity of our results.*

Definition 1 (Fast–Slow MDP). An MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{W}, f, r, \gamma \rangle$ is called a $(\alpha, d_{\mathcal{Y}})$–fast–slow MDP if Assumptions 1 and 2 are satisfied.

Given any state (x, y), noise w, and policy ν, we use the notation $f^\nu(x,y,w) = f(x,y,\nu(x,y),w)$, $f_{\mathcal{X}}^\nu(x,y,w) = f_{\mathcal{X}}(x,y,\nu(x,y),w)$, $f_{\mathcal{Y}}^\nu(x,y,w) = f_{\mathcal{Y}}(x,y,\nu(x,y),w)$, and $r(x, y, \nu) = r(x,y,\nu(x,y))$ throughout the paper. The value of a stationary policy[3] ν at state (x,y) is the expected cumulative reward starting from state (x,y) following policy ν, that is,

$$U^\nu(x,y) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, y_t, \nu) \,\middle|\, (x_0, y_0) = (x,y)\right]$$
$$= r(x,y,\nu) + \gamma\,\mathbb{E}[U^\nu(x', y')],$$

where $(x', y') = f^\nu(x,y,w)$ and $(x_{t+1}, y_{t+1}) = f^\nu(x_t, y_t, w_t)$ for all t. The optimal value function at state $U^*(x,y)$, as defined in (1), satisfies the Bellman equation, that is,

$$U^*(x,y) = \max_a r(x,y,a) + \gamma\,\mathbb{E}[U^*(x', y')]. \tag{5}$$

Denote by H the Bellman operator of the base model; for any state (x,y) and value function U,

$$(HU)(x,y) = \max_a r(x,y,a) + \gamma\,\mathbb{E}[U(f(x,y,a,w))]. \tag{6}$$

A policy that is greedy with respect to the optimal value function, that is,

$$\nu^*(x,y) = \arg\max_a r(x,y,a) + \gamma\, \mathbb{E}[U^*(x',y')].$$

is an optimal policy, and the optimal value U^* and the value of the optimal policy U^{ν^*} are the same.

3.2. Hierarchical Reformulation Using Fixed-Horizon Policies

In this section, we derive an exact hierarchical reformulation with the original timescale broken up into groups of T periods each. The reformulation holds for a general MDP $\langle \mathcal{S},\mathcal{A},\mathcal{W},f,r,\gamma\rangle$, but the concepts that we introduce in this section serve as the basis for developing our frozen-state computational approach for fast–slow MDPs.

Denote $(\mu,\boldsymbol{\pi})$ as a T-horizon policy, which is a sequence of T policies $(\mu,\pi_1,\ldots,\pi_{T-1})$, $\mu:\mathcal{S}\to\mathcal{A}$, $\pi_t:\mathcal{S}\to\mathcal{A}$, and $\boldsymbol{\pi}=(\pi_1,\ldots,\pi_{T-1})$. Following $(\mu,\boldsymbol{\pi})$ means that we take the first action according to μ and then next $T-1$ actions according to $\boldsymbol{\pi}$. Given any state s_0, the T-period reward function (of the base model) associated with $(\mu,\boldsymbol{\pi})$ is written as

$$R(s_0,\mu(s_0),\boldsymbol{\pi}) = r(s_0,\mu) + \sum_{t=1}^{T-1}\gamma^t r(s_t,\pi_t), \qquad (7)$$

where $s_1 = f^{\mu}(s_0,w_1)$ and $s_{t+1} = f^{\pi_t}(s_t,w_{t+1})$ for $t>0$.

A T-periodic policy $(\mu,\boldsymbol{\pi})$ refers to the infinite sequence that repeatedly applies the T-horizon policy $(\mu,\boldsymbol{\pi})$, that is, $(\mu,\boldsymbol{\pi},\mu,\boldsymbol{\pi},\ldots)$. Note that, despite it not being a stationary policy, the T-periodic policy $(\mu,\boldsymbol{\pi})$ can be implemented in the infinite horizon problem defined in (1). The value of the T-periodic policy $(\mu,\boldsymbol{\pi})$ at state s_0 is

$$\begin{aligned}\bar{U}^{\mu}(s_0,\boldsymbol{\pi}) &= \mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^{kT}R(s_k,\mu(s_k),\boldsymbol{\pi})\middle| s_0=s\right]\\ &= \mathbb{E}[R(s_0,\mu(s_0),\boldsymbol{\pi}) + \gamma^T\,\bar{U}^{\mu}(s_T,\boldsymbol{\pi})],\end{aligned}$$

where, again, $s_1 = f^{\mu}(s_0,w_1)$ and $s_{t+1}=f^{\pi_t}(s_t,w_{t+1})$ for $t>0$ within each cycle of T periods. Figure 1 compares stationary policy ν and a T-periodic policy $(\mu,\boldsymbol{\pi})$ for the case of $T=4$. In the figure, we also illustrate how rewards can be written in an aggregated fashion over the T periods using (7).

The optimal value function satisfies the following Bellman equation:

$$\bar{U}^*(s_0) = \max_{(\mu,\boldsymbol{\pi})}\mathbb{E}[R(s_0,\mu(s_0),\boldsymbol{\pi}) + \gamma^T\bar{U}^*(s_T)], \qquad (8)$$

where the action now involves selecting the $\boldsymbol{\pi}$ as well. Denote $(\mu^*,\boldsymbol{\pi}^*)$ an optimal T-periodic policy, which solves (8). In Proposition 1, we prove that the base model (5) and the hierarchical reformulation (8) are equivalent in a certain sense.

Proposition 1. *Given an MDP $\langle \mathcal{S},\mathcal{A},\mathcal{W},f,r,\gamma\rangle$, the following hold:*

i. *The optimal value of the base model (5) is equal to the optimal value of the hierarchical reformulation (8), that is, $U^*=\bar{U}^*$.*

ii. *An optimal stationary policy ν^* with respect to the base model (5) is also an optimal policy for the hierarchical reformulation (8), that is, $\bar{U}^*=\bar{U}^{\nu^*}$.*

Proof. See Online Appendix D.2. □

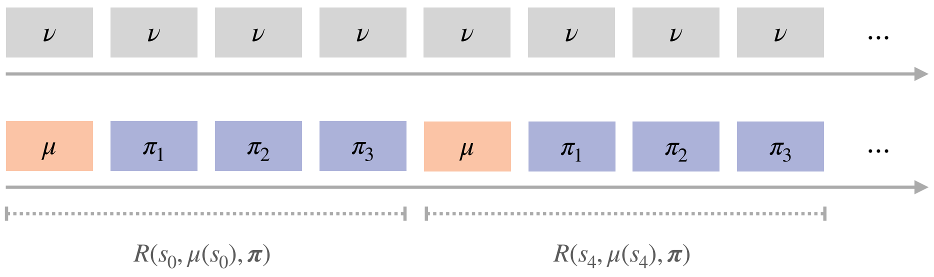
Part (i) of Proposition 1 is most relevant to our situation in the sense that the optimal T-periodic policy $(\mu^*,\boldsymbol{\pi}^*)$ is no better than the stationary optimal policy ν^*. Therefore, solving the hierarchical reformulation (8) allows us to achieve the same value as the ν^*, the optimal policy to the original base model (5).

Note that, at this point, we have simply reformulated the problem, but (8) is no easier to solve than (5). Despite the more favorable discount factor γ^T in (8), its action space is now effectively the space of T-horizon policies rather than a single action a. In the next section, we propose an approximation that allows us to fix a lower level policy $\boldsymbol{\pi}$ and only optimize μ. This allows us to enjoy the γ^T discount factor, maintaining the same action space.

4. The Frozen-State Approximation

We propose a frozen-state approximation, in which we make two simplifications to the T-period finite-horizon problem with terminal value U^* that is embedded in

Figure 1. (Color online) Illustration of a Stationary Policy μ (Upper Timeline) and a T-Periodic Policy $(\mu,\boldsymbol{\pi})$ (Lower Timeline) for $T=4$



Note. The periods covered by the T-period reward associated with $(\mu,\boldsymbol{\pi})$ are visualized with brackets in the lower timeline.

each T-period time step of (8), termed the lower level problem. First, motivated by the slow transitions of x given in Assumption 1, we freeze slow states for all T periods of the lower level problem, and second, we decouple the problem from the main MDP by solving an approximation with zero terminal value instead of U^*.

The first simplification reduces the computation needed to solve the finite-horizon MDP. The second simplification, because of the decoupling from the main problem, allows us to precompute an approximation to $\boldsymbol{\pi}^*$, which we denote $\tilde{\boldsymbol{\pi}}^*$. By then fixing $\tilde{\boldsymbol{\pi}}^*$, we are able to construct an auxiliary problem that proceeds at a timescale that is a factor of T slower than the MDP of the base model (equivalently, the discount factor becomes γ^T instead of γ) yet optimizing over the same action space. This naturally leads to algorithms with computational benefits (see Section 5). The number of periods T to freeze the state is a parameter to the approach. See Figure 2 for a high-level illustration; we provide a detailed description of the approach in the next few sections.

Remark 2. It is important to note that the freezing of states only occurs within the algorithm as a step toward more efficient computation of policies. Our resulting policies are then implemented in the underlying base model MDP, which proceeds naturally according to its true dynamics. Our theoretical and empirical results always attempt to answer the question: how well does an approximate policy, which is computed by pretending certain states are frozen, perform in the true model?

4.1. The Lower Level MDP (Frozen Slow States)

We view the problem from period 1 to period T as the lower level of the frozen-state approximation.[4] To form the lower level problem of the frozen-state approximation, we consider this $T-1$ period problem in isolation:

$$J_1^{\tilde{\boldsymbol{\pi}}}(x,y) = \mathbb{E}\left[\sum_{t=1}^{T-1} \gamma^{t-1} r(x_1, y_t, \tilde{\pi}_t) \,\middle|\, (x_1,y_1) = (x,y)\right] \quad \text{and}$$

$$\max_{\tilde{\boldsymbol{\pi}}} J_1^{\tilde{\boldsymbol{\pi}}}(x,y), \tag{9}$$

where $x_{t+1} = x_t = x$ remains frozen, $y_{t+1} = f_y^{\tilde{\pi}_t}(x, y_t, w_{t+1})$, and $\tilde{\boldsymbol{\pi}} = (\tilde{\pi}_1, \dots, \tilde{\pi}_{T-1})$. Problem (9) can be solved using backward dynamic programming: accordingly, let the terminal $J_T^* \equiv 0$ and for $t = 1, 2, \dots, T-1$, let

$$J_t^*(x,y) = \max_a r(x,y,a) + \gamma\,\mathbb{E}[J_{t+1}^*(x,y')], \tag{10}$$

where $y' = f_y(x,y,a,w)$. We also have the standard recursion for the performance of a policy:

$$J_t^{\tilde{\boldsymbol{\pi}}}(x,y) = r(x,y,\tilde{\pi}_t(x,y)) + \gamma\,\mathbb{E}[J_t^{\tilde{\boldsymbol{\pi}}}(x, f_y^{\tilde{\pi}_t}(x,y,w_{t+1}))], \tag{11}$$

with $J_T^{\tilde{\boldsymbol{\pi}}} \equiv 0$. We denote by $\tilde{H}$ the Bellman operator of the lower level problem, which is on the same timescale as the base model (hence, the discount factor is γ) and looks similar to the Bellman operator H defined in (6), but the transition of the slow state x is frozen. For any state (x,y) and lower level value function J_{t+1},[5] define

$$(\tilde{H}J_{t+1})(x,y) = \max_a r(x,y,a) + \gamma\,\mathbb{E}[J_{t+1}(x, f_y(x,y,a,w))]. \tag{12}$$

Note that (12) can be viewed as an approximation to (6). Analogously, let $\tilde{H}^{\tilde{\boldsymbol{\pi}}}$ be the Bellman operator associated with (11).

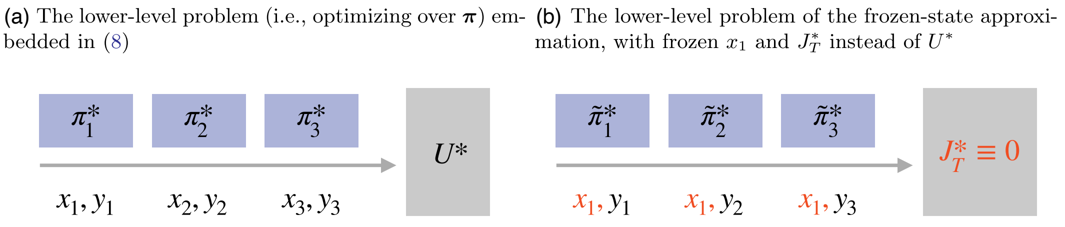
Also, let $\tilde{\boldsymbol{\pi}}^* = (\tilde{\pi}_1^*, \dots, \tilde{\pi}_{T-1}^*)$ be the finite-horizon policy that is greedy with respect to J_t^*:

$$\tilde{\pi}_t^*(x,y) = \arg\max_a r(x,y,a) + \gamma\,\mathbb{E}[J_{t+1}^*(x,y')].$$

Note that using $J_T^* \equiv 0$ is aligned with the fact that the lower level Problem (9) contains $T-1$ periods. It, thus, follows that $J_1^{\tilde{\boldsymbol{\pi}}^*} = J_1^*$, meaning that we can use J_1^* to represent the value of the lower level problem. We see in the next section that the upper level problem depends on an approximation of $J_1^{\tilde{\boldsymbol{\pi}}^*}$, which is the value of the lower level policy in $T-1$ periods (and zero terminal value after that).

Remark 3 (Nonzero Terminal Value). What about when one wants to solve the lower level problem with some $J_T^* \neq 0$? In some applications, it may be natural or desirable to assign a nonzero terminal value at time T. This allows for a heuristic continuation value based on the algorithm designer's domain knowledge. In this case, the lower level policy $\tilde{\boldsymbol{\pi}}^*$ can still be computed using finite-horizon dynamic programming, starting from the nonzero J_T^* as the terminal condition

Figure 2. (Color online) A Comparison of the Lower Level Problem of the Hierarchical Reformulation vs. the Lower Level Problem of the Frozen-State Approximation


(a) The lower-level problem (i.e., optimizing over $\boldsymbol{\pi}$) embedded in (8)
π_1^*
π_2^*
π_3^*
U^*
x_1, y_1
x_2, y_2
x_3, y_3
(b) The lower-level problem of the frozen-state approximation, with frozen x_1 and J_T^* instead of U^*
$\tilde{\pi}_1^*$
$\tilde{\pi}_2^*$
$\tilde{\pi}_3^*$
$J_T^* \equiv 0$
x_1, y_1
x_1, y_2
x_1, y_3

in Recursion (10). However, for consistency of the upper level problem, it is important to evaluate the performance of the resulting policy $\tilde{\boldsymbol{\pi}}^*$ using the original objective (9) (without the terminal value). In other words, when $J_T^* \neq 0$, $J_1^{\tilde{\boldsymbol{\pi}}^*}$ is no longer guaranteed to be the same as J_1^*. Therefore, to use a nonzero terminal value, we need to perform an additional policy evaluation step for $\tilde{\boldsymbol{\pi}}^*$ to obtain $J_1^{\tilde{\boldsymbol{\pi}}^*}$. This is a relatively simple one-time calculation that can be performed concurrently with the computation of $\tilde{\boldsymbol{\pi}}^*$. We assume throughout the paper that $J_T^* \equiv 0$ for simplicity.

It may not immediately be clear why freezing slow states is desired. There are two main computational benefits to solving (10) instead of an analogous version of (10) without freezing x:

- In algorithms such as value iteration (Puterman 2014), each update requires computing expectations over successor states, and therefore, the number of successor states impacts the number of operations for each step of value iteration. When x is frozen, the number of successor states is much smaller because we only have successor fast states (y'): in other words, we only need to compute $\mathbb{E}[J_{t+1}^*(x, y')]$ instead of $\mathbb{E}[J_{t+1}^*(x', y')]$.[6]
- Second, (10) can effectively be viewed as $|\mathcal{X}|$ independent MDPs, one for each $x \in \mathcal{X}$, allowing for the possibility of computing the policy with additional parallelism.

4.2. The Upper Level MDP (True State Dynamics)

Let us now consider the upper level problem of the frozen-state approximation, which is an infinite horizon problem with groups of T periods aggregated. Denote the stationary upper level policy by $\mu : \mathcal{S} \to \mathcal{A}$, which is the policy that we are attempting to optimize in the upper level problem. The upper level problem takes two inputs related to the lower level problem: (i) J_1, an approximation of the optimal lower level value J_1^*, and (ii) $\boldsymbol{\pi}$, a lower level finite-horizon policy. Fixing these inputs, the value at state $s_0 = (x_0, y_0)$ by executing policy μ is

$$V^\mu(s_0, J_1, \boldsymbol{\pi}) = \mathbb{E}[\tilde{R}(s_0, \mu(s_0), J_1) + \gamma^T V^\mu(s_T(\mu, \boldsymbol{\pi}), J_1, \boldsymbol{\pi})],$$

where $s_T(\mu, \boldsymbol{\pi})$ is the state reached according to the true system dynamics by following $(\mu, \boldsymbol{\pi})$, starting from s_0 and

$$\tilde{R}(s_0, a, J_1) = r(s_0, a) + \gamma J_1(f(s_0, a, w)) \tag{13}$$

is a one-step approximation to the T-period reward function R defined in (7). Figure 3 helps to visualize the upper level MDP.

The optimal value (for this approximation) at state s_0 can be written as

$$V^*(s_0, J_1, \boldsymbol{\pi}) = \max_a \mathbb{E}[\tilde{R}(s_0, a, J_1) + \gamma^T V^*(s_T(a, \boldsymbol{\pi}), J_1, \boldsymbol{\pi})], \tag{14}$$

where $s_T(a, \boldsymbol{\pi})$ is the state reached according to the true system dynamics by first taking action a and then following $\boldsymbol{\pi}$, starting from s_0.

Throughout the paper, we use the notations $V^\mu(J_1, \boldsymbol{\pi}) : \mathcal{S} \to \mathbb{R}$ and $V^*(J_1, \boldsymbol{\pi}) : \mathcal{S} \to \mathbb{R}$ to refer to the value function obtained when the MDP is evaluated or solved for a fixed J_1 and $\boldsymbol{\pi}$, respectively. We also define the Bellman operator associated with (14):

$$(F_{J_1, \boldsymbol{\pi}} V)(s_0) = \max_a \mathbb{E}[\tilde{R}(s_0, a, J_1) + \gamma^T V(s_T(a, \boldsymbol{\pi}))], \tag{15}$$

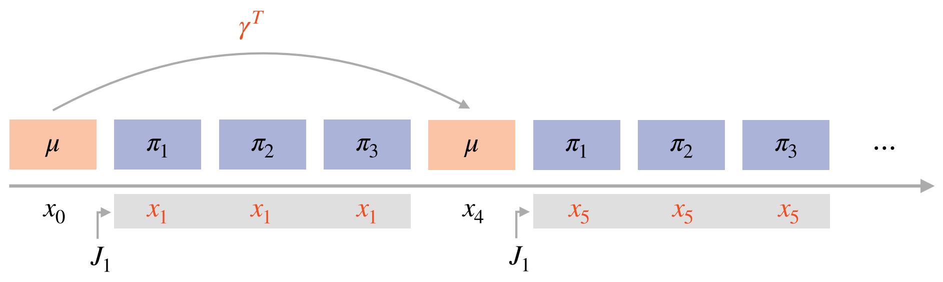
which becomes useful later.

Recall that the optimal lower level policy (for the frozen-state model) is denoted $\tilde{\boldsymbol{\pi}}^*$, and its optimal value is J_1^*. Let $\tilde{\mu}^*$ be the optimal upper level policy corresponding to these inputs, that is, the policy that is greedy with respect to $V^*(s_0, J_1^*, \tilde{\boldsymbol{\pi}}^*)$. Thus, $(\tilde{\mu}^*, \tilde{\boldsymbol{\pi}}^*)$ is the resulting T-periodic policy from the frozen-state hierarchical approximation; we refer to it as the T-periodic frozen-state policy.

5. Frozen-State Value Iteration

In this section, we introduce our new approach: the FSVI algorithm. We start by mentioning that the naive approach to solving the base model MDP (5) is to directly apply standard VI (see, e.g., Bertsekas and Tsitsiklis 1996). For completeness, we provide the full description of standard VI in Algorithm 1.

Figure 3. (Color online) Illustration of the Upper Level Problem



Notes. Notably, the discount factor is γ^T, and the reward function, from the point of view of μ, depends on the lower level value function J_1. This value function is computed by freezing states as visualized by the gray box.

Our new approach, FSVI, is given in Algorithm 2. The main ideas of the algorithm are

i. Solve the lower level MDP with frozen states to obtain a policy $\tilde{\boldsymbol{\pi}}^*$ and its value J_1^*. Because the lower level problem is a finite horizon MDP, it can be solved exactly using $T-1$ steps of VI. An alternative way of viewing the lower level problem is that it is $|\mathcal{X}|$ independent MDPs, which we can potentially solve in parallel. Note that Algorithm 2 emphasizes this by looping over the slow state $x \in \mathcal{X}$ in line 1.

ii. Apply VI to the upper level problem starting with some initial value function V^0, using J_1^* to approximate the T-horizon reward and $\tilde{\boldsymbol{\pi}}^*$ for T-step transitions. Note that this is an infinite-horizon MDP that operates at a slower timescale and enjoys a more favorable discount factor of γ^T.

We denote the resulting value function approximation after k iterations of value iteration as V^k from which we obtain a policy $\tilde{\mu}^k$. The T-periodic policy output by FSVI is formed by combining $\tilde{\mu}^k$ with the optimal finite-horizon policy $\tilde{\boldsymbol{\pi}}^*$ from the lower level MDP: $(\tilde{\mu}^k, \tilde{\boldsymbol{\pi}}^*)$.

Algorithm 1 (Exact VI for the Base Model (Base VI))

```
Input: Initial values $U^0$, number of iterations $k$.
Output: Approximation to the optimal policy $\nu^k$.
1 for $k' = 1,2,\ldots,k$ do
2     for $s$ in the state space $\mathcal{S}$ do
3         $U^{k'}(s) = \max_a r(s,a) + \gamma\,\mathbb{E}[U^{k'-1}(f(s,a,w))]$.
4     end
5 end
6 for $s$ in the state space $\mathcal{S}$ do
7     $\nu^k(s) = \arg\max_a r(s,a) + \gamma\,\mathbb{E}[U^k(f(s,a,w))]$.
8 end
```

Algorithm 2 (FSVI)

```
Input: Initial values $J_T^* \equiv 0$ and $V^0$, number of iterations $k$.
Output: Approximation of the $T$-periodic frozen-state policy $(\tilde{\mu}^k, \tilde{\boldsymbol{\pi}}^*)$ and $J_1^*$.
1 for each slow state $x \in \mathcal{X}$ do
2     for $t = T-1, T-2, \ldots, 1$ do
3         for each fast state $y \in \mathcal{Y}$ do
4             $J_t^*(x,y) = \max_a r(x,y,a) + \gamma\,\mathbb{E}[J_{t+1}^*(x, f_{\mathcal{Y}}(x,y,a,w))]$.
5             $\tilde{\pi}_t^*(x,y) = \arg\max_a r(x,y,a) + \gamma\,\mathbb{E}[J_{t+1}^*(x, f_{\mathcal{Y}}(x,y,a,w))]$.
6         end
7     end
8 end
9 for $k' = 1,2,\ldots,k$ do
10    for $s_0 = (x_0,y_0)$ in the state space $\mathcal{X}\times\mathcal{Y}$ do
11        $V^{k'}(x_0,y_0,J_1^*,\tilde{\boldsymbol{\pi}}^*) = \max_a \mathbb{E}[\tilde{R}(s_0,a,J_1^*) + \gamma^T V^{i-1}(x_T,y_T,J_1^*,\tilde{\boldsymbol{\pi}}^*)]$.
12    end
13 end
14 for $s_0 = (x_0,y_0)$ in the state space $\mathcal{X}\times\mathcal{Y}$ do
15    $\tilde{\mu}^k(x_0,y_0) = \arg\max_a \mathbb{E}[\tilde{R}(s_0,a,J_1^*) + \gamma^T V^k(x_T,y_T,J_1^*,\tilde{\boldsymbol{\pi}}^*)]$.
16 end.
```

5.1. Natural Application Domains for FSVI

Whereas the FSVI algorithm is general and can be applied to any MDP, we do not necessarily expect it to perform well in any arbitrary setting. Its approximation relies heavily on a separation of timescales and on the idea that some form of structured behavior can be captured by freezing slow variables over short horizons. That said, there are two broad classes of problems in which we expect FSVI to work particularly well:

i. Problems with implicit hierarchical structure. Many real-world applications do not exhibit explicit hierarchy in their MDP formulations but nevertheless contain a natural two-level behavioral structure. We illustrate this type of problem in Section 9 through our grid world with spatial tasks domain: the agent first accepts a task (each task comes with a starting and ending location) to undertake and then executes a sequence of navigation actions to complete it. This problem was motivated by operations domains such as warehouse robotics (Yang et al. 2020) and on-demand spatial services from the driver's perspective (Chung et al. 2018, Jiang et al. 2020, Ong et al. 2021, Ashkrof et al. 2024), in which high-level task acceptance decisions are followed by fast-paced and relatively more myopic execution.

ii. Problems with natural cyclical behavior. In some domains, optimal or near-optimal behavior follows a cyclic structure. A classic example is inventory control, in which the policy repeatedly builds up and depletes inventory. We show results on inventory control with fixed costs in Section 9. Related problems, such as energy or commodity trading (Carmona and Ludkovski 2010, Löhndorf and Minner 2010, Kim and Powell 2011, Jiang and Powell 2015b) or inventory repositioning (He et al. 2020, Benjaafar et al. 2022), may also exhibit similar cyclical behavior. Another domain we consider in Section 9 is a dynamic pricing problem with reference effects, in which cyclic pricing strategies are shown to perform well under various assumptions (Wang 2016, Chen et al. 2017). In all of the above settings, freezing the slow state over a short horizon can capture the cyclical dynamics of the optimal policy.

A key advantage of FSVI is that it can discover both forms of structure automatically through its two-level decomposition. Therefore, the user of the algorithm need not encode any concrete hierarchical or cyclical behavior and only needs to set the parameter T, which can often be selected using domain expertise. In Section 7, we also show how our theoretical analyses may guide the selection of T.

5.2. Computational Cost of FSVI

Let us now discuss the computational cost of FSVI. Here, we take a dynamic programming perspective,[7] in which the model is known and expectations can be computed. It is well-known that each iteration of standard VI, which provides a contraction factor of γ, has time complexity $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$ (Littman et al. 1995). The upper level of FSVI, on the other hand, enjoys an improved contraction factor γ^T with the same per-iteration running time of $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$ given that we pay a one-time fixed cost of solving the lower level. The quantity $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$ consists of a factor of $|\mathcal{S}||\mathcal{A}|$ because of the number of state–action pairs at which to compute the Bellman update and another factor of $|\mathcal{S}|$ because of the number of successor states. Because freezing slow states restricts the successor states to $\mathcal{Y}$, each iteration of the lower level VI (first part of Algorithm 2) has a running time $\mathcal{O}(|\mathcal{X}||\mathcal{Y}|^2|\mathcal{A}|)$. An additional $\mathcal{O}(|\mathcal{S}|^2 T)$ is required to compute the T-step transition probabilities of following $\tilde{\boldsymbol{\pi}}^*$, to be used in the upper level VI steps, resulting in a one-time fixed cost of $\mathcal{O}(|\mathcal{X}||\mathcal{Y}|^2|\mathcal{A}|T + |\mathcal{S}|^2 T)$. Particularly when $|\mathcal{X}|$ is large, this can be a reasonable fixed cost to pay in order to get the much improved discount factor of γ^T going forward. The equations below give a direct comparison between the computational cost of VI versus FSVI:

$$\text{Cost}_{\text{VI}}(k) = \mathcal{O}(k\,|\mathcal{S}|^2\,|\mathcal{A}|), \qquad \text{(VI after } k \text{ iterations)}$$

$$\text{Cost}_{\text{FSVI}}(k) = \mathcal{O}(\underbrace{|\mathcal{X}|\,|\mathcal{Y}|^2\,|\mathcal{A}|\,T + |\mathcal{S}|^2\,T}_{\substack{\text{one-time lower-level}\\ \text{pre-processing cost}}} + \underbrace{k\,|\mathcal{S}|^2\,|\mathcal{A}|}_{\substack{\text{upper-level}\\ \text{iterations}}}). \qquad \text{(FSVI after } k \text{ iterations)}$$

6. Theoretical Analysis

In this section, we develop the theory to understand the regret of FSVI. First, in Section 6.1, we derive the reward approximation error between the hierarchical reformulation and the frozen-state approximation. This result becomes needed in later sections. In Section 6.2, we define the notion of regret that we use in the paper. Then, in Section 6.3, we give an important lemma that illustrates the trade-offs between the various sources of error and how they contribute to the overall regret. Finally in Section 6.4, we give the main result of the section, and this characterizes the regret of FSVI for a particular T and k.

6.1. Reward Approximation Error

Recall that $(\mu^*, \boldsymbol{\pi}^*)$ is an optimal T-periodic policy of the base model's hierarchical reformulation (8). Suppose $\boldsymbol{\pi}^*$ is available. Then, the Bellman equation of the base model reformulation is

$$\begin{aligned} U^*(x_0,y_0) &= \bar{U}^*(x_0,y_0) \\ &= \max_a \mathbb{E}[R(x_0,y_0,a,\boldsymbol{\pi}^*) + \gamma^T \bar{U}^*(x_T,y_T)] \\ &= \max_a \mathbb{E}\Bigg[r(x_0,y_0,a) + \sum_{t=1}^{T-1}\gamma^t r(x_t,y_t,\pi_t^*) \\ &\qquad + \gamma^T U^*(x_T,y_T)\Bigg] \\ &= \max_a \mathbb{E}[r(x_0,y_0,a) + \gamma\,(H^{T-1}U^*)(x_1,y_1)], \end{aligned} \quad (16)$$

where the notation H^k is shorthand for k applications of the operator H, that is, $H^kU = H(H^{k-1}U)$ and $H^1U = HU$. Therefore, the expected T-horizon reward can be written as

$$\begin{aligned} \mathbb{E}[R(x_0,y_0,a,\boldsymbol{\pi}^*)] &= \mathbb{E}[r(x_0,y_0,a) + \gamma\,(H^{T-1}U^*)(x_1,y_1) \\ &\qquad - \gamma^T U^*(x_T,y_T)]. \end{aligned} \quad (17)$$

Given the optimal value J_1^* of the lower level (10), the T-horizon reward of the upper level (14) can be written as

$$\begin{aligned} \mathbb{E}[\tilde{R}(x_0,y_0,a,J_1^*)] &= r(x_0,y_0,a) + \gamma\,\mathbb{E}[J_1^*(x_1,y_1)] \\ &= r(x_0,y_0,a) + \gamma(\tilde{H}^{T-1}J_T^*)(x_1,y_1), \\ &= r(x_0,y_0,a) + \gamma(\tilde{H}^{T-1}\mathbf{0})(x_1,y_1), \end{aligned} \quad (18)$$

where $\mathbf{0}$ is the all-zero value function. The difference between (17) and (18) can be interpreted as follows: in the former, we follow a lower level policy that is aware of a terminal value U^* (but exclude that value when defining the T-horizon reward), whereas in the latter, we follow a lower level policy that sees zero terminal reward at the end of the $T-1$ periods.

The first step to understanding the performance of the frozen-state policy is to analyze the reward approximation $\mathbb{E}[\tilde{R}(s_0,a,J_1^*)]$ compared with the true reward $\mathbb{E}[R(s_0,a,\boldsymbol{\pi}^*)]$. Proposition 2 shows how the difference between two reward functions is dependent on the number of frozen periods T along with the problem parameters.

Proposition 2 (Reward Approximation Error). *Let $\langle\mathcal{S},\mathcal{A},\mathcal{W},f,r,\gamma\rangle$ be a $(\alpha,d_{\mathcal{Y}})$–fast–slow MDP satisfying Assumption 2. Let $\boldsymbol{\pi}^*$ be the optimal lower level policy for the base model reformulation (8) and J_1^* be the optimal (first stage) value of the lower level problem in the frozen-state* *approximation* (10). *For any state $s_0 = (x_0, y_0)$ and action a, the approximation error between the T-horizon reward of hierarchical reformulation and the frozen-state approximation, that is, the discrepancy between* (17) *and* (18), *can be bounded as*

$$|\mathbb{E}[R(s_0,a,\boldsymbol{\pi}^*)] - \mathbb{E}[\tilde{R}(s_0,a,J_1^*)]| \le \underbrace{\alpha d_{\mathcal{Y}}\left(L_r\sum_{i=1}^{T-2}\gamma^i\sum_{j=0}^{i-1}L_f^j\right)}_{\text{freeze error}} + \underbrace{\gamma^{T-1}L_U\left[\alpha d_{\mathcal{Y}}\sum_{j=0}^{T-2}L_f^j + \gamma d_{\mathcal{Y}}(\alpha+2)(T-1)\right]}_{\text{end-of-horizon error}}, \tag{19}$$

Proof. The detailed proof is in Online Appendix E.2. □

For more convenient notation, we define $\epsilon_r(\gamma, \alpha, d_{\mathcal{Y}}, L_r, L_f, T)$ to be the right-hand side of (19):

$$\epsilon_r(\gamma,\alpha,d_{\mathcal{Y}},\mathbf{L},T) = \alpha d_{\mathcal{Y}}\left(L_r\sum_{i=1}^{T-2}\gamma^i\sum_{j=0}^{i-1}L_f^j\right) + \gamma^{T-1}L_U\left[\alpha d_{\mathcal{Y}}\sum_{j=0}^{T-2}L_f^j + \gamma d_{\mathcal{Y}}(\alpha+2)(T-1)\right],$$

where $\mathbf{L} = (L_r, L_f, L_U)$ emphasizes the dependence on the various Lipschitz constants. In subsequent sections, we use ϵ_r as an ingredient in analyzing the regret of various frozen-state policies.

6.2. Defining Regret

We start with definitions of the regret of both stationary and T-periodic policies.

Definition 2 (Regret). Consider a fast–slow MDP with initial state s_0 and optimal policy ν^*. The regret of a stationary policy ν is defined as

$$\mathcal{R}(s_0,\nu) = U^{\nu^*}(s_0) - U^{\nu}(s_0) \quad \text{and} \quad \mathcal{R}(\nu) = \max_{s_0} \mathcal{R}(s_0,\nu).$$

The regret of the T-periodic policy $(\mu, \boldsymbol{\pi})$ is defined as

$$\mathcal{R}(s_0,\mu,\boldsymbol{\pi}) = U^{\nu^*}(s_0) - \bar{U}^{\mu}(s_0,\boldsymbol{\pi}) = \bar{U}^*(s_0) - \bar{U}^{\mu}(s_0,\boldsymbol{\pi}) \quad \text{and}$$
$$\mathcal{R}(\mu,\boldsymbol{\pi}) = \max_{s_0} \mathcal{R}(s_0,\mu,\boldsymbol{\pi}).$$

The second equality in the definition of $\mathcal{R}(s_0, \mu, \boldsymbol{\pi})$ uses the value equivalence between the base model and its hierarchical reformulation (Proposition 1).

Remark 4. As a follow-up comment to Remark 2, notice that $V^*(s_0, J_1^*, \tilde{\boldsymbol{\pi}}^*)$ does not directly enter the regret definition as $V^*(s_0, J_1^*, \tilde{\boldsymbol{\pi}}^*)$ is just the optimal value of the frozen-state approximation, not the value of its implied greedy policy $\tilde{\mu}^*$ when evaluated in the base model. However, the regret, of course, depends on $V^*(s_0, J_1^*, \tilde{\boldsymbol{\pi}}^*)$ indirectly because $\tilde{\mu}^*$ depends on $V^*(s_0, J_1^*, \tilde{\boldsymbol{\pi}}^*)$.

6.3. A General Lemma

In this section, we first prove a general lemma that is used to analyze FSVI.

Lemma 1. *Suppose we have an approximation $(\boldsymbol{\pi}, J_1)$ to the lower level solution $(\boldsymbol{\pi}^*, U^*)$. Further, suppose we have an approximation V to the upper level solution $V^*(J_1, \boldsymbol{\pi})$. Consider a T-periodic policy $(\mu, \boldsymbol{\pi})$, where*

$$\mu(s_0) = \arg\max_{a\in\mathcal{A}} \mathbb{E}[\tilde{R}(s_0,a,J_1) + \gamma^T V(s_T(a,\boldsymbol{\pi}))]. \tag{20}$$

Then, the regret of $(\mu, \boldsymbol{\pi})$ can be bounded as follows:

$$\mathcal{R}(\mu,\boldsymbol{\pi}) \le \left(\frac{2\gamma^T}{(1-\gamma^T)^2} + \frac{2}{1-\gamma^T}\right)\epsilon_r(\boldsymbol{\pi}^*,J_1) + \left(\frac{2\gamma^{2T}}{(1-\gamma^T)^2} + \frac{2\gamma^T}{1-\gamma^T}\right)L_U\, d(\alpha,d_{\mathcal{Y}},T) + \frac{2\gamma^T}{1-\gamma^T}\|V^*(J_1,\boldsymbol{\pi}) - V\|_\infty,$$

where $\epsilon_r(\boldsymbol{\pi}^*, J_1) = \max_{s,a}|\mathbb{E}[R(s,a,\boldsymbol{\pi}^*)] - \mathbb{E}[\tilde{R}(s,a,J_1)]|$ and $d(\alpha, d_{\mathcal{Y}}, T) = 2d_{\mathcal{Y}}(\alpha+1)(T-1)$.

Proof. See Online Appendix E.3. □

The result of Lemma 1 can be interpreted as the regret being bounded by

reward error(& end-of-horizon effect)
+ upper-level freeze error + V-approximation error,

which directly corresponds to the three terms in the bound. The reward error is due to freezing the slow state in the lower level along with using zero terminal value; the upper level freeze error propagates the lower level freeze error to the upper level's infinite horizon, and the V-approximation error is due to not solving the upper level problem exactly.

6.4. Regret of FSVI

An instance of FSVI is associated with two primary quantities: k, the number of VI iterations, and T, the number of periods in which the slow state is frozen in the frozen-state approximation. The next theorem gives a bound on the regret of the policy obtained for a particular k and T.

Theorem 1. *Let $(\tilde{\mu}^k, \tilde{\boldsymbol{\pi}}^*)$ be the resulting T-periodic policy after running FSVI for k iterations. The regret incurred when running $(\tilde{\mu}^k, \tilde{\boldsymbol{\pi}}^*)$ in the base model satisfies*

$$\mathcal{R}(\tilde{\mu}^k,\tilde{\boldsymbol{\pi}}^*) \le \left(\frac{2\gamma^T}{(1-\gamma^T)^2} + \frac{2}{1-\gamma^T}\right)\epsilon_r(\gamma,\alpha,d_{\mathcal{Y}},\mathbf{L},T) + \left(\frac{2\gamma^{2T}}{(1-\gamma^T)^2} + \frac{2\gamma^T}{1-\gamma^T}\right)L_U\, d(\alpha,d_{\mathcal{Y}},T) + \frac{2r_{\max}\gamma^{(k+1)T}}{(1-\gamma)(1-\gamma^T)},$$

where the last term, which depends on k, accounts for the error because of value iteration.

Proof. See Online Appendix E.4. □

7. Interpreting the Regret Analysis

It is common for theoretical analyses in reinforcement learning to depend on Lipschitz constants of the underlying MDP (see, e.g., Pirotta et al. 2015, Asadi et al. 2018, Gelada et al. 2019, Sinclair et al. 2022, Maran et al. 2024). Because these Lipschitz constants are typically conservative approximations of the true MDP dynamics, our regret bounds may not always align closely with the empirical regret observed in real-world settings. This is consistent with much of the literature in which bounds often serve more as qualitative guides than precise predictors.

Regardless, such regret analysis is valuable as it can help us understand and clarify the trade-offs involved when selecting various parameters, primarily T, the horizon of the frozen lower level problem. This is particularly true in our setting, in which we are balancing several sources of error with computational improvements. In this section, we numerically compute the regret bounds for various values of T in order to obtain insights.

7.1. Illustrating Reward Error

First, we examine the reward approximation error analyzed in Proposition 2. Depending on how the freeze error and the end-of-horizon error interact, we see different behaviors of the overall reward approximation error as illustrated in Figure 4. For both examples, the end-of-horizon error increases at first but then decreases toward zero as T becomes large. In the example with $L_f < 1$ (left), the freeze error converges as T increases; however, if $L_f > 1$, the freeze error grows without bound. This leads to two distinct patterns in the overall reward approximation error: (i) when the freeze error is well-controlled, the overall reward approximation error generally follows the unimodal shape of the end-of-horizon error, and (ii) when the freeze error grows without bound, the reward approximation error follows a unimodal shape at first but eventually grows again.

7.2. Regret of Base VI

We pause here briefly to state Proposition 3, a well-known property that gives the regret of exact VI on the base model. We use this result as a comparison to the regret of FSVI in the next section.

Proposition 3. *Let v^k be the result of running Algorithm 1 on the base model (5). Then,*

$$\mathcal{R}(v^k) = \|U^{v_k} - U^*\|_\infty \le \frac{2r_{\max}\gamma^{k+1}}{(1-\gamma)^2},$$

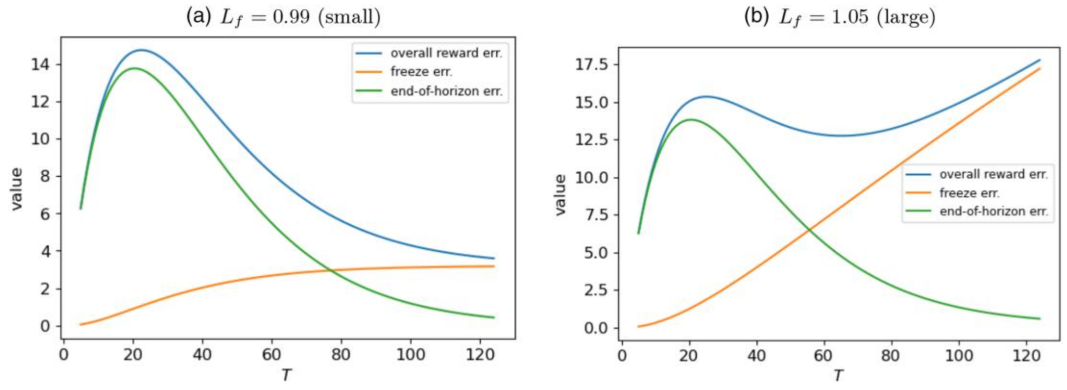
and we recall that $r_{\max}$ is an upper bound on the reward.

Proof. See Online Appendix E.5. □

7.3. Regret vs. Computational Cost for FSVI and VI: Selecting *T*

Next, we show how Proposition 3 (the regret of base VI) and Theorem 1 (the regret of FSVI) can be combined to provide intuition about whether to use base VI or FSVI (and, if so, which T and k). We consider the trade-off between regret and computational cost, using the definition of cost given in Section 5.2. For FSVI, the concrete problem facing a practitioner is as follows: given a computational cost budget B, what combination of (T, k) should be used? We make two observations: first, for a fixed T, regret decreases with k (see Theorem 1), and second, for a fixed T, cost increases with k (see Section 5.2). Therefore, for a fixed T, one should run FSVI for $k_{\max}(T, B)$ iterations, where $k_{\max}(T, B)$ is the largest k such that the computational cost of running FSVI with T is less than B. It, thus, follows that, to select T, one should simply examine the regret for different values of T at $k = k_{\max}(T, B)$ and select the value of T with the lowest regret.

Figure 4. (Color online) Reward Approximation Error Comparison for Two Values of L_f

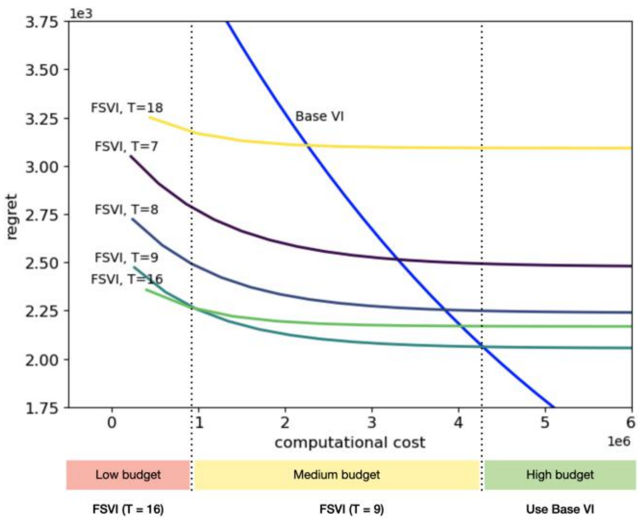


The above procedure can be converted into a simple visualization. In Figure 5, we plot regret versus computational cost. This is done for both FSVI for different values of T and also for standard value iteration (base VI). Each colored curve traces the regret–cost trade-off generated by varying k for a distinct T (though k is not shown). Imposing the computational budget corresponds to drawing the vertical line $x = B$, and when this line intersects a curve, one can read off the best attainable regret given budget B for the algorithm represented by that curve. By scanning the intersection points and selecting the one with the lowest regret, one can identify the best algorithm to use whether it is FSVI with a particular T or simply base VI. Hence, the plot allows us to visually solve the constrained optimization problem of minimizing regret under a given computational budget. This exercise leads us to interesting takeaways illustrated in Figure 5:

i. If the available computational budget is low, one should use FSVI with a large value of T (in this case, $T = 16$). Despite introducing more error because of freezing, this allows us to get the quickest decrease in regret initially because of the lower discount factor, but as we can see from the plot, the decrease does not sustain as we increase computation.

ii. If the computational budget is high, one should simply use base VI as it eventually achieves zero regret. This is natural and intuitive because FSVI introduces unavoidable error that does not vanish as the number of value iteration steps increases.

Figure 5. (Color online) Regret vs. Computational Cost for Base VI and FSVI as the Number of Value Iteration Steps Increases



Note. In this example, we see three regimes depending on the computational budget: in the low-budget regime, the lowest regret algorithm is FSVI with $T = 16$; in the medium-budget regime, FSVI with $T = 9$ achieves lowest regret; and in the high-budget regime, not surprisingly, base VI achieves lowest regret.

iii. With a medium computational budget, FSVI with a moderate value of $T = 9$ leads to the best regret guarantee. This regime is where we expect many real-world problems to fall.

Algorithm 3 (FVI (Munos and Szepesvári 2008))

```
Input: State distribution $\rho$, number of iterations $k$, number of state samples $N$, number of next state samples $M$, and a regression procedure Regress.
Output: Approximate value function $U^k$ and policy $\nu^k$.
1  Initialize $U^0$ arbitrarily.
2  for $k' = 1, 2, \ldots, k$ do
3      Sample a data set $\mathcal{D} = \{s^{(i)}\}_{i=1}^N \sim \rho$
4      for $s^{(i)} \in \mathcal{D}$ do
5          $u^{(i)} = \max_a r(s^{(i)}, a) + \gamma \frac{1}{M} \sum_{j=1}^{M} U^{k'-1}(s^{(i,a,j)})$, where $s^{(i,a,j)}$ is sampled from the generative model, conditional on state $s^{(i)}$ and action $a$.
6      end
7      $U^{k'} = \texttt{Regress}(\{(s^{(i)}, u^{(i)})\}_{i=1}^N)$
8  end
9  for $s \in \mathcal{S}$ do
10     $\nu^k(s) = \arg\max_a r(s,a) + \gamma \frac{1}{M} \sum_{j=1}^{M} U^k(s^{(a,j)})$, where $s^{(a,j)}$ is sampled from the generative model, conditional on state $s$ and action $a$.
11 end.
```

Algorithm 4 (FSFVI)

```
Input: State distribution $\rho$, number of iterations $k$, number of state samples $N$, number of next state samples $M_l$ (lower level), $M_u$ (upper level), and a regression procedure Regress.
Output: Approximation of the $T$-periodic frozen-state policy $(\tilde{\mu}^k, \tilde{\boldsymbol{\pi}}^*)$ and $\tilde{J}_1$.
1  Initialize $\tilde{J}_T \equiv 0$.
2  for $t = T-1, T-2, \ldots, 1$ do
3      Sample a data set $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N \sim \rho$
4      for each $(x^{(i)}, y^{(i)}) \in \mathcal{D}$ do
5          $J_t^{(i)} = \max_{a \in \mathcal{A}} r(x^{(i)}, y^{(i)}, a) + \gamma \frac{1}{M_l} \sum_{j=1}^{M_l} \tilde{J}_{t+1}(x^{(i)}, y^{(i,a,j)})$, where $y^{(i,a,j)}$ is sampled from the generative model, conditional on state $(x^{(i)}, y^{(i)})$ and action $a$.
6      end
7      $\tilde{J}_t = \texttt{Regress}(\{((x^{(i)}, y^{(i)}), J_t^{(i)})\}_{i=1}^N)$
8  end
9  Let $\tilde{\boldsymbol{\pi}}^* = (\pi_1^*, \ldots \pi_{T-1}^*)$, with $\pi_t^*(s) = \arg\max_a r(s,a) + \gamma \frac{1}{M_l} \sum_{j=1}^{M_l} \tilde{J}_{t+1}(s^{(a,j)})$, where $s^{(a,j)}$ is sampled from the generative model, conditional on state $s$ and action $a$.
```

```
10 Initialize $\hat{V}^0$ arbitrarily.
11 for $k' = 1, 2, \ldots, k$ do
12     Sample a data set $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N \sim \rho$
13     for each $(x^{(i)}, y^{(i)}) \in \mathcal{D}$ do
14         $V^{(k')} = \max_{a \in \mathcal{A}} \frac{1}{M_u} \sum_{j=1}^{M_u} [\tilde{R}^{(i,a,j)} + \gamma^T V^{k'-1}(x^{(i,a,j)}, y^{(i,a,j)})]$
           where $x_T^{(i,a,j)}$, $y_T^{(i,a,j)}$, and $\tilde{R}^{(i,a,j)}$ are sampled
           using a $T$-step transition conditioned on
           $x^{(i)}, a, \tilde{\boldsymbol{\pi}}^*$ and $\tilde{J}_1$.
15     end
16     $\tilde{V}^{k'} = \texttt{Regress}(\{((x^{(i)}, y^{(i)}), V^{(i)})\}_{i=1}^N)$
17 end
18 for $s_0 = (x_0, y_0) \in \mathcal{S}$ do
19     $\tilde{\mu}^k(x_0, y_0) = \arg\max_{a \in \mathcal{A}} \frac{1}{M_u} \sum_{j=1}^{M_u} [\tilde{R}^{(i,a,j)} + \gamma^T \tilde{V}^k(x_T^{(i,a,j)}, y_T^{(i,a,j)})]$,
       where $x_T^{(i,a,j)}$, $y_T^{(i,a,j)}$, and $\tilde{R}^{(i,a,j)}$ are sampled from
       a $T$-step transition as above.
20 end.
```

8. Frozen-State Fitted Value Iteration Under a Generative Model

So far, we have operated in the dynamic programming setting, in which the model (rewards and transitions) is assumed to be known. We now move on to a more practical and realistic setting in which we assume access to a generative model or simulator of the environment (also referred to as a restart distribution), an assumption frequently used in the RL literature (Kakade and Langford 2002, Kearns et al. 2002, Munos and Szepesvári 2008, Agarwal et al. 2020, Li et al. 2020). This setting is widely studied and captures many real-world use cases in which it is possible to simulate environment transitions (Bellemare et al. 2012, Todorov et al. 2012, Brockman et al. 2016, Tunyasuvunakool et al. 2020) without having full access to closed-form model dynamics. This is an important setting that occupies a middle ground between dynamic programming and traditional model-free RL.

FVI is a simple method that approximates the value iteration backup step using regression (Munos and Szepesvári 2008), which we keep as an abstract subroutine using the notation $\texttt{Regress}(\{(s^{(i)}, y^{(i)})\}_{i=1}^N)$, which refers to the problem of regressing $\{y^{(i)}\}$ onto $\{s^{(i)}\}$. The `Regress` procedure is meant to encode all of the information needed to perform the regression: basis functions, approximation architecture (e.g., linear model or neural network), optimization procedure (e.g., stochastic gradient descent), etc. In Algorithm 3, we show the abstract version of FVI, and in Algorithm 4, we give the frozen-state extension, FSFVI, which can be viewed as a practical extension of FSVI for the RL setting.

Additionally, in Online Appendix C, we present an alternative version of FVI that is less practical but more amenable to theoretical analysis, leading to a regret bound that we provide in Online Theorem C.1. These results are motivated by the approximate value iteration method first presented in Tsitsiklis and Van Roy (1996). Given that both the method and analysis are technical and notation-heavy, we relegated the discussion to the Online Appendix.

9. Numerical Experiments

For our numerical experiments, we operate in the simulator-based RL setting described above, in which a generative model or simulator is available but the MDP is not explicitly known. Note that both FVI and FSFVI naturally apply in this setting as expectations are approximated via sample averages using either M, M_l, or M_u next state samples. In addition, we introduce empirical counterparts to base VI (Algorithm 1) and FSVI (Online Algorithm 7), which we refer to as base empirical-VI (base E-VI) and empirical-FSVI (E-FSVI), following the naming convention of Haskell et al. (2016). These algorithms are identical to their exact counterparts except that all expectations are replaced with sample averages. For completeness, we provide full specifications in Online Appendix A.

In Section 5.2, we analyze the computational cost of FSVI under the assumption of a known model with exact expectation computations. In the RL setting, however, this definition requires adjustment. As a natural analogue, we use the number of value function evaluations as a proxy for computational effort. This makes sense in the RL context because value-based methods, whether value iteration, Q-iteration, or their fitted variants, all rely on repeatedly evaluating a value function (or Q-function). These evaluations typically dominate the overall computation time, especially when using function approximation. Thus, counting value function evaluations provides a measure of computational cost that aligns closely with RL practice, being closely related to the exact results from Section 5.2.

We compare this with a range of algorithms. For purely tabular methods, we consider

- E-FSVI with $T \in \{3, 6, 12\}$: Empirical-FSVI as described in Online Algorithm 6. For each domain, we run E-FSVI for each T value in order to illustrate the trade-offs.
- Base E-VI: Empirical-VI as described in Online Algorithm 5, applied to the base, nonhierarchical problem. We also use base E-VI to compute an approximation of the optimal value.
- Base E-QI: Empirical Q-iteration is the analogue of E-VI using a Q-function representation.
- Slow-Agn Base E-VI: Slow-agnostic base E-VI is a simple baseline that pretends the slow state does not exist and assumes the value function only depends on the fast state.[8]

With function approximation, we test

- {Lin, Deep}-FSFVI: Our proposed frozen-state fitted value iteration as introduced in Algorithm 4, using either a linear function approximator (Lin-FSFVI) or a three-layer neural network function approximator (Deep-FSVI).
- {Lin, Deep}-FVI: Fitted value iteration (Munos and Szepesvári 2008) using either linear function approximator (Lin-FVI) or a three-layer neural network function approximator (Deep-FVI). This algorithm is also known as least square Monte Carlo, popularized by Longstaff and Schwartz (2001) and frequently used in the operations literature (see, e.g., Nadarajah et al. 2017).
- {Lin, Deep}-FQI: Fitted Q-iteration (Ernst et al. 2005) using either linear function approximator (Lin-FQI) or a three-layer neural network function approximator (Deep-FQI).

For Lin-FSFVI and Lin-FVI, we use the raw state feature vectors as input to a linear function approximator without applying any feature transformation or basis functions. For Deep-FSFVI, FVI, we input the raw state vector into a fully connected feedforward neural network with two hidden layers, each comprising four units and rectified linear unit activations, followed by a final linear output layer. For Lin-FQI and Deep-FQI, the input is formed by concatenating the raw state vector with a one-hot encoding of the discrete action (a standard practice in RL for handling discrete action spaces). The architecture of the Deep-FQI network is otherwise identical to that of Deep-FVI. All models are trained using the mean squared error loss and optimized with the Adam optimizer (Kingma and Ba 2014). A detailed open-source implementation is also provided with this paper.

We test these algorithms across three domains, summarized in Table 2. Each domain uses a high discount factor of $\gamma = 0.995$, corresponding to an effective planning horizon of approximately $1/(1-\gamma) = 200$ steps. We describe each experiment below; in each section, we use self-contained notation. Detailed experimental settings are given in Online Appendix B.

9.1. Inventory Control with Fixed Order Costs

We consider an infinite horizon inventory control problem (Porteus 1990) with fixed order costs, lost sales, and limited storage (Zhao et al. 2007). The state of the system is $s_t = (y_t, d_t)$, where $y_t \in \{0, 1, \ldots, y_{\max}\}$ (the fast state) is the inventory level and $d_t \in \mathcal{D}$ is the current demand (the slow state), which evolves exogenously over time. At the beginning of each period t, the decision maker observes the current state s_t and chooses an order quantity $a_t \in \mathcal{A}_{\text{order}} = \{0, a_1, \ldots, a_{\max}\}$. The cost incurred includes a fixed cost K if $a_t > 0$, a per-unit ordering cost c, and a holding cost h per unit of end-of-period inventory. Unsatisfied demand is lost. The transition equation is

$$y_{t+1} = \min\{y_t + a_t - \min\{y_t, d_{t+1}\}, y_{\max}\}, \quad d_{t+1} \sim P_d(\cdot|d_t),$$

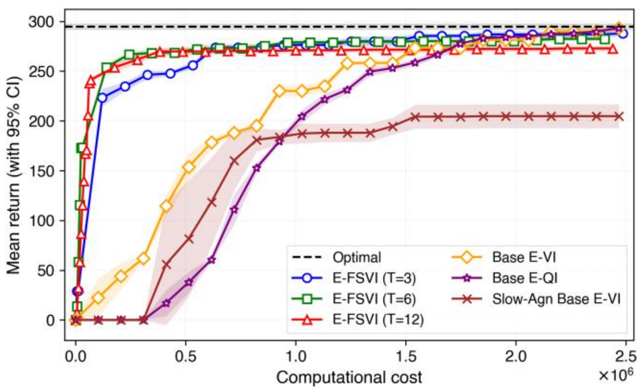
where d_{t+1} is drawn from a distribution P_d conditional on d_t. The (expected) reward in each period is given by

$$r(s_t, a_t) = p\,\mathbb{E}(\min\{y_t, d_{t+1}\}) - c\,a_t - K\mathbf{1}_{\{a_t > 0\}},$$

which represents the revenue, variable order cost, and fixed order cost. The objective is to maximize infinite horizon discounted reward with $\gamma = 0.995$. In our experiment, we set $y_{\max} = 50$, $\mathcal{A}_{\text{order}} = \{0, 5, \ldots, 50\}$, and $d_{t+1} = \text{clip}_{[0,50]}(d_t + \epsilon_{t+1})$, and $\{\epsilon_t\}$ are i.i.d. random variables that take value 0 with probability 0.8 and ± 1 with probability 0.1 each.[9] We use $M = M_u = 50$ for all methods, and $M_l = 1$ for E-FSVI because the lower level problem is deterministic.

The results, shown in Figure 6, demonstrate that frozen-state methods substantially outperform standard approaches in terms of computational efficiency.

Figure 6. (Color online) Mean Return (with 95% Confidence Intervals) vs. Computational Cost for the Inventory Problem Across Five Independent Replications



Notes. For the E-FSVI variants, we plot the performance of the policy after each lower level value iteration and three times per upper level value iteration. For the other methods, we plot three times per value iteration. To compute the optimal value, we ran base E-VI for 50 iterations with $M = 100$ samples per backup.

Table 2. Summary of Experimental Domains

Domain	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{S}\times\mathcal{A}\rvert$	Algorithms tested
Inventory control	561	11	6,171	Base E-VI, E-FSVI, Base E-QI, Slow-Agn Base E-VI
Spatial task grid world	4,356	32	139,392	Base E-VI, E-FSVI, Base E-QI, Slow-Agn Base E-VI
Dynamic pricing	15,150	21	318,150	Lin-FSFVI, Lin-FVI, Lin-FQI, Deep-FSFVI, Deep-FVI, Deep-FQI

Specifically, E-FSVI with moderate values of T, such as $T = 6$ and $T = 12$, achieves near-optimal return with significantly fewer value function evaluations compared with base E-VI and E-QI. This supports the intuition that freezing the slow-moving demand state d_t over short planning horizons can preserve essential structure, simplifying the lower level planning problem.

Among the E-FSVI variants, we find that $T = 6$ offers the best overall performance when considering both reward and computational cost. Larger values of T, such as $T = 12$, initially converge faster but eventually suffer from increased approximation error because of longer freezing horizons. Conversely, if we look at the earlier part of the curves, E-FSVI with $T = 3$ seems to benefit less from the improved discount factor.

Whereas base E-VI and E-QI eventually reach optimal performance, they require significantly more computation. In contrast, the slow-agnostic baseline performs poorly, confirming that ignoring the slow state leads to suboptimal policies. These findings reinforce both our theoretical claims from Section 6 and the trade-offs we observe in Figure 5 from Section 7.

9.2. Grid World with Spatial Tasks

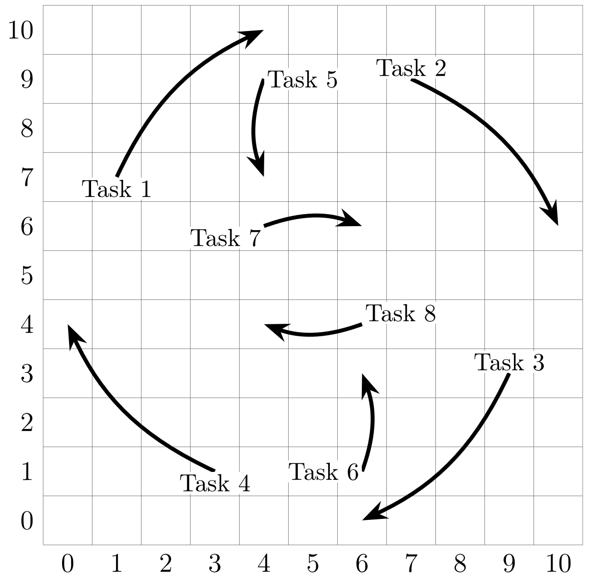
The main idea behind our second experiment domain is illustrated in Figure 7 and is straightforward: an RL agent must navigate the grid world and complete tasks. There are unlimited quantities of each task, which are labeled using arrows in the figure. The agent first must accept a task, after which the agent begins working on that task. To earn the completion reward, the agent must pick up an object from the start location of the task (beginning of the arrow) and bring it to the end location of the task. The agent also earns a small reward for picking up the object. The state of the agent is $s_t = (x_t, y_t, i_t, o_t, w_t)$, where (x_t, y_t) are the coordinates of the agent, $i_t \in \{0, 1, \ldots, 8\}$ is the identity of the agent's current task (where $i_t = 0$ means no current accepted task), $o_t \in \{0, 1\}$ is a binary state indicating whether the agent has picked up the object associated with the current task, and finally $w_t \in \{0, 1\}$ is a reward signal indicating whether the high rewards are in the outer ring ($w_t = 0$) or the inner ring ($w_t = 1$). The agent's action is $a_t = (i'_t, d_t)$, where $i'_t \in \{1, 2, \ldots, 8\}$ is the identity of the task that the agent wishes to accept and d_t is a cardinal direction to move. The rest of the environment works as follows:

- If $i_t = 0$ (no current task), then $i_t = i'_t$. Otherwise, i'_t takes no effect.
- If the agent reaches the start location of the current task i_t, then o_t transitions from zero to one.
- If the agent reaches the end location of the current task i_t and $i_t = 1$, then the agent receives a reward $r(i_t, w_t)$.

The agent wishes to maximize the infinite-horizon discounted reward with $\gamma = 0.995$. Again, we use $M = M_u = 50$ for all methods, and $M_l = 1$ for E-FSVI. We omit a detailed formulation of the MDP for succinctness but give the full reward specification in Online Appendix B.2.

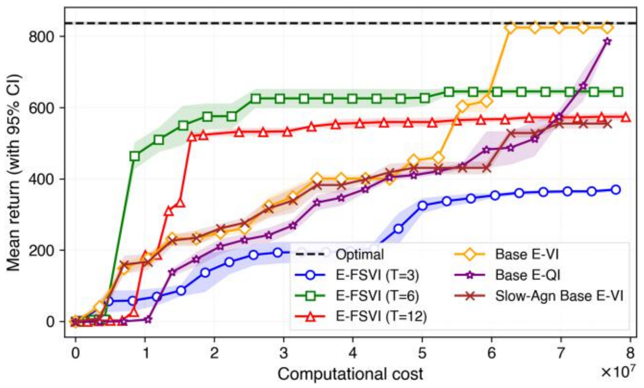
The results are shown in Figure 8. In contrast to the inventory control problem, the performance differences between E-FSVI algorithms with different values of T are far more pronounced in this domain: the smallest T value of $T = 3$ improves gradually and consistently but is never quite able to achieve good

Figure 7. Spatial Tasks on an 11×11 Grid World



Notes. Start and end locations of each task are represented by the arrows. In order to receive a reward, the agent must pick up an object from the start location of the task and move it to the end location.

Figure 8. (Color online) Mean Return (with 95% Confidence Intervals) vs. Computational Cost for the Spatial Grid World Problem Across Five Independent Replications



Notes. For the E-FSVI variants, we plot the performance of the policy after each lower level value iteration and two times per upper level value iteration. For the other methods, we plot two times per value iteration. To compute the optimal value, we ran base E-VI for 50 iterations with $M = 100$ samples per backup.

performance. This effect is likely because of the nature of the tasks and the need to coordinate navigation across multiple steps. When T is too small, the frozen lower level policy is unable to fully capture meaningful navigation plans, leading to suboptimal behavior at the upper level. Interestingly, we observe that an intermediate value such as $T = 6$ provides a strong compromise: it allows the lower level policy to encode useful subtask behavior, retaining enough flexibility for the upper level to guide high-level decisions. We also notice that base VI and QI slowly improve and eventually reach optimality but require nearly three times the computation to reach 75% optimality that E-FSVI with $T = 6$ is able to achieve very quickly.

This experiment underscores that, in environments with complex dynamics, the choice of T can be quite consequential.

9.3. Dynamic Pricing with Reference Effects and Stochastic Consumer Behavior

We consider a dynamic pricing problem in which a seller interacts with a market of consumers who exhibit reference effects: that is, their purchase behavior depends not only on the current price but also on a history of past prices. Reference prices represent consumers' internal price expectations for the product based on historical prices. When the firm sets prices below (above) the consumer's reference price, there is a positive (negative) impact on demand. This is motivated by real-world e-commerce and retail environments in which consumers are sensitive to perceived fairness or price changes over time. The model we consider is a slight extension of the one in Chen et al. (2017).

The state variable is $s_t = (p_t^{\text{ref}}, \alpha_t, \eta_t^+, \eta_t^-)$, where p_t^{ref} is the reference price, α_t is the market's memory factor, and (η_t^+, η_t^-) are the marginal reference price effect for gains and losses. The action is the price in the current period $p_t \in \mathcal{P}_t$, which induces a change in the reference price $p_{t+1}^{\text{ref}} = \alpha_t p_t^{\text{ref}} + (1 - \alpha_t) p_t$. We model the demand as

$$D_t(p_t, p_t^{\text{ref}}) = b_t - a_t p_t + \eta_t^+ \max\{p_t^{\text{ref}} - p_t, 0\} + \eta_t^- \min\{p_t^{\text{ref}} - p_t, 0\}.$$

Consumer behavior is jointly described by $(\alpha_t, \eta_t^+, \eta_t^-)$, which evolve exogenously according to

$$(\alpha_{t+1}, \eta_{t+1}^+, \eta_{t+1}^-) = \text{clip}_{[(0.4,3,8),(0.9,7,12)]}((\alpha_t, \eta_t^+, \eta_t^-) + \epsilon_{t+1}),$$

where clip is applied componentwise and $\epsilon_{t+1} \in \mathbb{R}^3$ and each component is i.i.d., taking value 0 with 0.9 probability, ± 0.1 with 0.05 probability each for the α_t dimension, and ± 1 with 0.05 probability each for the η_t^+, η_t^- dimensions. The reward function is $r(s_t, p_t) = p_t D_t(p_t, p_t^{\text{ref}})$.

This problem is discretized as follows. The possible values of the action p_t are $\{0, 0.5, 1.0, \ldots, 10.0\}$, and the possible values of the reference price p_t^{ref} are $\{0, 0.1, 0.2, \ldots, 10.0\}$. For $\alpha_t, \eta_t^+, \eta_t^-$, the possible values are $\{0.4, 0.5, 0.6, 0.7\}$, $\{3, 4, 5, 6, 7\}$, and $\{8, 9, 10, 11, 12\}$.

As shown in Table 2, this is the largest problem out of the three. We, therefore, use this domain to test the fitted variants of our algorithms. We set N, the number of state samples, to be 10% of the size of the state space. As before, we use $M = M_u = 50$ for all methods, and $M_l = 1$ for E-FSVI.

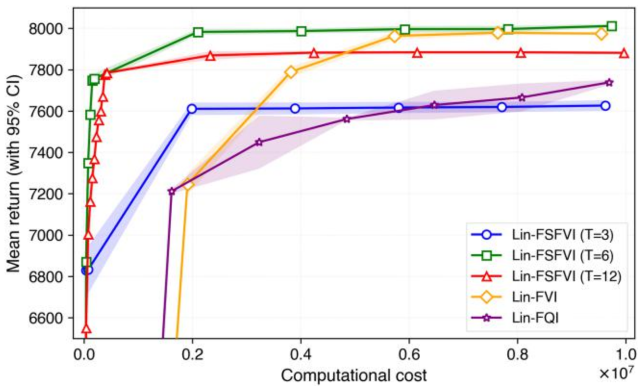
Figures 9 and 10 compare the performance of frozen-state and baseline methods under two different architectures: linear function approximation (Figure 9) and neural network function approximation (Figure 10). We point out several takeaways:

i. Frozen-state methods improve upon baselines in both settings, but the gains are larger in the setting using neural network function approximation. This is largely because of the relatively poor performance of Deep-FVI and Deep-FQI, which appear to suffer from higher variance and instability during training. We hypothesize that this is due to the difficulty of fitting high-capacity models under long-horizon dynamics, in which bootstrapping can amplify noise. In contrast, the structured nature of Deep-FSFVI, specifically the reduced effective discount factor and frozen slow states, likely contributes to improved stability and more reliable learning.

ii. Again, we see that that the $T = 3$ version of Deep-FSFVI performed poorly under both architectures, but $T = 6$ and $T = 12$ were both adequate.

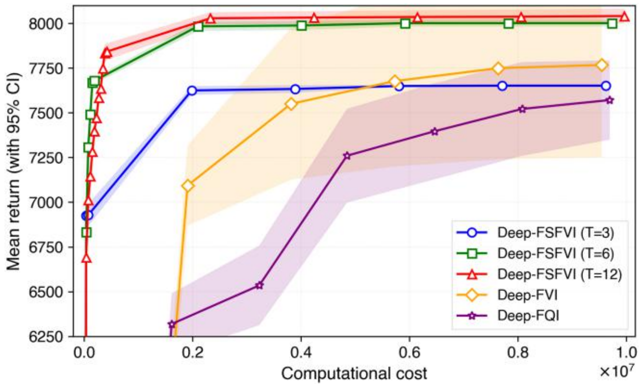
Overall, the results suggest that frozen-state methods are broadly effective and, surprisingly, seem to enjoy some additional robustness even in the face of

Figure 9. (Color online) Mean Return (with 95% Confidence Intervals) vs. Computational Cost for the Dynamic Pricing Problem Using Linear Architectures Across Five Independent Replications



Notes. For the FSFVI variants, we plot the performance of the policy after each lower level value iteration and once per upper level value iteration. For the other methods, we plot once per value iteration.

Figure 10. (Color online) Mean Return (with 95% Confidence Intervals) vs. Computational Cost for the Dynamic Pricing Problem Using Linear Network Architectures Across Five Independent Replications



Notes. For the FSFVI variants, we plot the performance of the policy after each lower level value iteration and once per upper level value iteration. For the other methods, we plot once per value iteration.

neural network architectures that are typically difficult to fit.

10. Conclusion

In this paper, we study a new class of MDPs with a type of structure called fast–slow structure, motivated by practical applications in which some components of the state evolve slowly over time. Based on this structure, we propose a set of new algorithms based on the idea of freezing the slow state for several periods at a time to ease the computational burden of value iteration algorithms. We analyze the regret of the resulting policy using a novel analysis of various Bellman operators. Empirically, on three example applications, we show that our new frozen-state methods converge significantly faster to good policies than standard methods, and notably, ignoring the slow state leads to low-performing policies. Therefore, our method can be viewed as a viable compromise between solving the full MDP (often computationally intractable) and completely ignoring states during the modeling process (computationally easy but potentially highly suboptimal in the true model).

Acknowledgments

The authors are grateful to the department editor, associate editor, and three anonymous reviewers for their multiple rounds of careful reading and constructive feedback. Their comments helped us catch important issues and substantially improve the paper, including strengthening the organization, sharpening the intuition and exposition, and clarifying key points throughout.

Endnotes

[1] See Online Appendix B.1 for details on how we simulate this behavior in our experiments.

[2] However, as long as the relevant spaces are metric spaces, the framework continues to hold. We choose Euclidean metrics as they are natural for our applications.

[3] It is well-known that there exists an optimal policy to (1) that is both stationary and deterministic. See Puterman (2014).

[4] This corresponds to the periods relevant to $\boldsymbol{\pi}$ from $(\mu, \boldsymbol{\pi})$ in the hierarchical reformulation (8), whose structure the frozen-state approximation mimics.

[5] We include time indexing on the value function to emphasize that this Bellman operator is used in a finite-horizon (i.e., nonstationary) setting, but the definition of $\tilde{H}$ itself does not depend on t.

[6] Even in the case that the expectation is approximated via sampling, the former requires sampling from a lower dimensional successor state distribution.

[7] In our numerical results of Section 9, we take a different, more practical perspective, in which we work with a generative model or simulator of the environment.

[8] To implement this method, we convert the full state to a partial state by deleting the slow state, and when needing to interact with the simulator, we convert from a partial state back to a full state by injecting a random value for the slow state. See Online Appendix B.1 for further details.

[9] We define $\text{clip}_{[a,b]}(x) = \max\{\min\{x, b\}, a\}$.

References

Agarwal A, Kakade S, Yang LF (2020) Model-based reinforcement learning with a generative model is minimax optimal. Abernethy J, Agarwal S, eds. *Proc. 33rd Conf. Learning Theory*, vol. 125 (PMLR, New York), 67–83.

Asadi K, Misra D, Littman M (2018) Lipschitz continuity in model-based reinforcement learning. Dy J, Krause A, eds. *Proc. 35th Internat. Conf. Machine Learn.*, vol. 80 (PMLR, New York), 264–273.

Ashkrof P, Homem de Almeida Correia G, Cats O, Van Arem B (2024) On the relocation behavior of ride-sourcing drivers. *Transportation Lett.* 16(4):330–337.

Barto AG, Mahadevan S (2003) Recent advances in hierarchical reinforcement learning. *Discrete Event Dynam. Systems* 13(1–2): 41–77.

Bellemare M, Veness J, Bowling M (2012) Investigating contingency awareness using Atari 2600 games. Hoffmann J, Selman B, eds. *Proc. 26th AAAI Conf. Artificial Intelligence*, vol. 26 (AAAI Press, Palo Alto, CA), 864–871.

Benjaafar S, Jiang D, Li X, Li X (2022) Dynamic inventory repositioning in on-demand rental networks. *Management Sci.* 68(11): 7861–7878.

Bertsekas DP, Tsitsiklis JN (1996) *Neuro-Dynamic Programming* (Athena Scientific, Belmont, MA).

Bhatnagar S, Panigrahi JR (2006) Actor-critic algorithms for hierarchical Markov decision processes. *Automatica* 42(4):637–644.

Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2016) OpenAI Gym. Preprint, submitted June 5, https://doi.org/10.48550/arXiv.1606.01540.

Brown DB, Smith JE (2025) Unit commitment without commitment: A dynamic programming approach for managing an integrated energy system under uncertainty. *Oper. Res.* 73(4):1744–1766.

Carmona R, Ludkovski M (2010) Valuation of energy storage: An optimal switching approach. *Quant. Finance* 10(4):359–374.

Chang HS, Fard PJ, Marcus SI, Shayman M (2003) Multitime scale Markov decision processes. *IEEE Trans. Automatic Control* 48(6): 976–987.

Chen X, Hu P, Hu Z (2017) Efficient algorithms for the dynamic pricing problem with reference price effect. *Management Sci.* 63(12):4389–4408.

Chung H, Freund D, Shmoys DB (2018) Bike angels: An analysis of Citi Bike's incentive program. *Proc. First ACM SIGCAS Conf. Comput. Sustainable Soc.*, 1–9.

Ciosek K, Silver D (2015) Value iteration with options and state aggregation. Preprint, submitted January 16, http://arxiv.org/abs/1501.03959.

Dietterich TG (2000) Hierarchical reinforcement learning with the MAXQ value function decomposition. *J. Artificial Intelligence Res.* 13:227–303.

Digney BL (1998) Learning hierarchical control structures for multiple tasks and changing environments. Pfeifer R, Blumberg B, Meyer J-A, Wilson SW, eds. *Proc. Fifth Internat. Conf. Simulation Adaptive Behav. Animals Animats*, vol. 5 (MIT PRESS, Cambridge, MA), 321–330.

Domingues OD, Ménard P, Pirotta M, Kaufmann E, Valko M (2021) Kernel-based reinforcement learning: A finite-time analysis. Meila M, Zhang T, eds. *Proc. 38th Internat. Conf. Machine Learn.*, vol. 139 (PMLR, New York), 2783–2792.

El Shar IE, Jiang D (2023) Weakly coupled deep q-networks. Oh A, Naumann T, Globerson A, Saenko K. Hardt M, Levine S, eds. *Adv. Neural Inform. Processing Systems*, vol. 36 (Curran Associates, Inc., Red Hook, NY), 43931–43950.

Ernst D, Geurts P, Wehenkel L (2005) Tree-based batch mode reinforcement learning. *J. Machine Learn. Res.* 6:503–556.

Gelada C, Kumar S, Buckman J, Nachum O, Bellemare MG (2019) DeepMDP: Learning continuous latent space models for representation learning. *Proc. 36th Internat. Conf. Machine Learn. (ICML 2019)*, vol. 97 (PMLR, New York), 2170–2179.

Haskell WB, Jain R, Kalathil D (2016) Empirical dynamic programming. *Math. Oper. Res.* 41(2):402–429.

He L, Hu Z, Zhang M (2020) Robust repositioning for vehicle sharing. *Manufacturing Service Oper. Management* 22(2):241–256.

Jacobson M, Shimkin N, Shwartz (1999) A piecewise stationary Markov decision processes, II: Constant gain, https://adam.net.technion.ac.il/files/2017/01/2TimesScale.pdf.

Jiang DR, Powell WB (2015a) An approximate dynamic programming algorithm for monotone value functions. *Oper. Res.* 63(6): 1489–1511.

Jiang DR, Powell WB (2015b) Optimal hour-ahead bidding in the real-time electricity market with battery storage using approximate dynamic programming. *INFORMS J. Comput.* 27(3):525–543.

Jiang DR, Al-Kanj L, Powell WB (2020) Optimistic Monte Carlo tree search with sampled information relaxation dual bounds. *Oper. Res.* 68(6):1678–1697.

Jiang N, Kulesza A, Singh S, Lewis R (2015) The dependence of effective planning horizon on model accuracy. Bordini RH, Elkind E, Weiss G, Yolum PK, eds. *Proc.14th Internat. Conf. Autonomous Agents Multiagent Systems* ((IFAAMAS, Richland, SC), 1181–1189.

Jonsson A, Barto AG (2005) A causal approach to hierarchical decomposition of factored MDPs. De Raedt L, Wrobel S, eds. *Proc. 22nd Internat. Conf. Machine Learn. (ICML 2005)* (ACM, New York), 401–408.

Kakade S, Langford J (2002) Approximately optimal approximate reinforcement learning. Sammut C, Hoffmann A, eds. *Proc. 19th Internat. Conf. Machine Learn*. (Morgan Kaufmann, San Francisco, CA), 267–274.

Kearns M, Mansour Y, Ng AY (2002) A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learn.* 49:193–208.

Killian JA, Biswas A, Shah S, Tambe M (2021) Q-learning Lagrange policies for multi-action restless bandits. Zhu F, Ooi BC, Miao C, eds. *Proc. 27th ACM SIGKDD Conf. Knowledge Discovery Data Mining* (ACM, New York), 871–881.

Kim JH, Powell WB (2011) Optimal energy commitments with storage and intermittent supply. *Oper. Res.* 59(6):1347–1360.

Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. Preprint, submitted December 22, https://arxiv.org/abs/1412.6980.

Kunnumkal S, Topaloglu H (2008) Using stochastic approximation methods to compute optimal base-stock levels in inventory control problems. *Oper. Res.* 56(3):646–664.

Li G, Wei Y, Chi Y, Gu Y, Chen Y (2020) Breaking the sample size barrier in model-based reinforcement learning with a generative model. Larochelle H, Ranzato MA, Hadsell R, Balcan M-F, Lin H-T, eds. *Proc. 34th Conf. Neural Inform. Processing Systems*, vol. 33 (Curran Associates Inc., Red Hook, NY), 12861–12872.

Littman ML, Dean TL, Kaelbling LP (1995) On the complexity of solving Markov decision problems. Besnard P, Hanks S, eds. *Proc. 11th Conf. Uncertainty Artificial Intelligence* (Morgan Kaufmann, San Francisco, CA) 394–402.

Löhndorf N, Minner S (2010) Optimal day-ahead trading and storage of renewable energies—An approximate dynamic programming approach. *Energy Systems* 1:61–77.

Longstaff FA, Schwartz ES (2001) Valuing American options by simulation: A simple least-squares approach. *Rev. Financial Stud.* 14(1):113–147.

Mannor S, Menache I, Hoze A, Klein U (2004) Dynamic abstraction in reinforcement learning via clustering. Greiner R, Schuurmans D, eds. *Proc. 21st Internat. Conf. Machine Learn.* (ACM, New York).

Maran D, Metelli AM, Papini M, Restell M (2024) No-regret reinforcement learning in smooth MDPs. Preprint, submitted February 6, https://arxiv.org/abs/2402.03792.

McGovern A, Barto AG (2001) Automatic discovery of subgoals in reinforcement learning using diverse density. Brodley CE, Danyluk AP, eds. *Proc. Eighth Internat. Conf. Machine Learn*. (Morgan Kaufmann, San Francisco, CA), 361–368.

Menache I, Mannor S, Shimkin N (2002) Q-cut-dynamic discovery of sub-goals in reinforcement learning. *Proc. 13th Eur. Conf. Machine Learn*., Lecture Notes in Computer Science, vol. 2430 (Springer, Berlin), 295–306.

Munos R, Szepesvári C (2008) Finite-time bounds for fitted value iteration. *J. Machine Learn. Res.* 9(27):815–857.

Nadarajah S, Cire AA (2025) Self-adapting network relaxations for weakly coupled Markov decision processes. *Management Sci.* 71(2):1779–1802.

Nadarajah S, Margot F, Secomandi N (2017) Comparison of least squares Monte Carlo methods with applications to energy real options. *Eur. J. Oper. Res.* 256(1):196–204.

Nascimento JM, Powell WB (2009) An optimal approximate dynamic programming algorithm for the lagged asset acquisition problem. *Math. Oper. Res.* 34(1):210–237.

Ok J, Proutiere A, Tranos D (2018) Exploration in structured reinforcement learning. Bengio S, et al. eds. *Adv. Neural Inform. Processing Systems*, vol. 31 (Curran Associates, Inc. City: Red Hook, NY), 5057–5067.

Ong HY, Freund D, Crapis D (2021) Driver positioning and incentive budgeting with an escrow mechanism for ride-sharing platforms. *INFORMS J. Appl. Anal.* 51(5):373–390.

Osband I, Van Roy B (2014) Near-optimal reinforcement learning in factored MDPs. Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ, eds. *Proc. 28th Conf. Adv. Neural Inform. Processing Systems*, vol. 27 (Curran Associates Inc., Red Hook, NY), 604–612.

Panigrahi JR, Bhatnagar S (2004) Hierarchical decision making in semiconductor fabs using multi-time scale Markov decision processes. *43rd IEEE Conf. Decision Control*, vol. 4 (IEEE), 4387–4392.

Papadaki KP, Powell WB (2002) Exploiting structure in adaptive dynamic programming algorithms for a stochastic batch service problem. *Eur. J. Oper. Res.* 142(1):108–127.

Parr RE, Russell S (1998) *Hierarchical Control and Learning for Markov Decision Processes* (University of California, Berkeley, CA).

Pereira MV, Pinto LM (1991) Multi-stage stochastic optimization applied to energy planning. *Math. Programming* 52:359–375.

Philpott AB, Guan Z (2008) On the convergence of stochastic dual dynamic programming and related methods. *Oper. Res. Lett.* 36(4):450–455.

Pirotta M, Restelli M, Bascetta L (2015) Policy gradient in Lipschitz Markov decision processes. *Machine Learn.* 100(2–3):255–283.

Porteus EL (1990) Stochastic inventory theory. Heyman DP, Sobel MJ, eds. *Handbooks in Operations Research and Management Science*, vol. 2 (Elsevier, Amsterdam), 605–652.

Precup D (2000) Temporal abstraction in reinforcement learning. Unpublished PhD thesis, University of Massachusetts, Boston.

Puterman ML (2014) *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (John Wiley & Sons, Hoboken, NJ).

Şimşek Ö, Barto AG (2004) Using relative novelty to identify useful temporal abstractions in reinforcement learning. *Proc. 21st Internat. Conf. Machine Learn.*, 95.

Şimşek Ö, Wolfe AP, Barto AG (2005) Identifying useful subgoals in reinforcement learning by local graph partitioning. *Proc. 22nd Internat. Conf. Machine Learn.*, 816–823.

Sinclair S, Banerjee S, Yu CL (2022) Adaptive discretization in online reinforcement learning. *Oper. Res.* 71(5):1636–1652.

Sinclair S, Wang T, Jain G, Banerjee S, Yu C (2020) Adaptive discretization for model-based reinforcement learning. *Adv. Neural Inform. Processing Systems: Proc. 34th Conf. Neural Inform. Processing Systems*, vol. 33 (Curran Associates, Inc., Red Hook, NY), 3858–3871.

Song R, Xu K (2022) Temporal concatenation for Markov decision processes. *Probab. Engrg. Inform. Sci.* 36(4):999–1026.

Stolle M, Precup D (2002) Learning options in reinforcement learning. *Proc. Fifth Internat. Sympos. Abstraction Reformulation Approximation* (Springer, Berlin), 212–223.

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

Tunyasuvunakool S, Muldal A, Doron Y, Liu S, BohezS, Merel J, Erez T, Lillicrap T, Heess N, Tassa Y (2020) dm_control: Software and tasks for continuous control. *Software Impacts* 6:100022.

Todorov E, Erez T, Tassa Y (2012) Mujoco: A physics engine for model-based control. *IEEE/RSJ Internat. Conf. Intelligent Robots Systems* (IEEE, Piscataway, NJ), 5026–5033.

Tsitsiklis JN, Van Roy B (1996) Feature-based methods for large scale dynamic programming. *Machine Learn.* 22(1–3):59–94.

Wang Z (2016) Intertemporal price discrimination via reference price effects. *Oper. Res.* 64(2):290–296.

Wang S, Bi S, Zhang Y-JA (2018) Demand response management for profit maximizing energy loads in real-time electricity market. *IEEE Trans. Power Systems* 33(6):6387–6396.

Wang Y, Poloczek M, Jiang DR (2023) Dynamic subgoal-based exploration via Bayesian optimization. *Trans. Machine Learn. Res.* (September 28), https://openreview.net/forum?id=ThJl4d5JRg.

Wongthatsanekorn W, Realff MJ, Ammons JC (2010) Multi-time scale Markov decision process approach to strategic network growth of reverse supply chains. *Omega* 38(1–2):20–32.

Yang B, Li W, Wang J, Yang J, Wang T, Liu X (2020) A novel path planning algorithm for warehouse robots based on a two-dimensional grid model. *IEEE Access* 8:80347–80357.

Yu JY, Mannor S (2009) Arbitrarily modulated Markov decision processes. *Proc. 48h IEEE Conf. Decision Control* (IEEE, Piscataway, NJ), 2946–2953.

Zhao X, Fan F, Liu X, Xie J (2007) Storage-space capacitated inventory system with (r, q) policies. *Oper. Res.* 55(5):854–865.

Zhu C, Zhou J, Wu W, Mo L (2006) Hydropower portfolios management via Markov decision process. *32nd Annual Conf. IEEE Indust. Electronics* (IEEE), 2883–2888.