Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Adaptive Discretization in Online Reinforcement Learning

Sean R. Sinclair, Siddhartha Banerjee, Christina Lee Yu

To cite this article:
Sean R. Sinclair, Siddhartha Banerjee, Christina Lee Yu (2022) Adaptive Discretization in Online Reinforcement Learning. Operations Research

Published online in Articles in Advance 11 Nov 2022

. https://doi.org/10.1287/opre.2022.2396

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

OPERATIONS RESEARCH
Articles in Advance, pp. 1–17
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Methods

Adaptive Discretization in Online Reinforcement Learning

Sean R. Sinclair,[a,*] **Siddhartha Banerjee,**[a] **Christina Lee Yu**[a]

[a] Operations Research and Information Engineering, Cornell University, Ithaca, New York 14853
*Corresponding author
Contact: srs429@cornell.edu, https://orcid.org/0000-0002-7011-8253 (SS); sbanerjee@cornell.edu, https://orcid.org/0000-0002-8954-4578 (SB); cleeyu@cornell.edu, https://orcid.org/0000-0002-2165-5220 (CY)

Received: May 31, 2021
Revised: March 3, 2022
Accepted: September 29, 2022
Published Online in Articles in Advance: November 11, 2022

Area of Review: Machine Learning and Data Science

https://doi.org/10.1287/opre.2022.2396

Abstract. Discretization-based approaches to solving online reinforcement learning problems are studied extensively on applications such as resource allocation and cache management. The two major questions in designing discretization-based algorithms are how to create the discretization and when to refine it. There are several experimental results investigating heuristic approaches to these questions but little theoretical treatment. In this paper, we provide a unified theoretical analysis of model-free and model-based, tree-based adaptive hierarchical partitioning methods for online reinforcement learning. We show how our algorithms take advantage of inherent problem structure by providing guarantees that scale with respect to the "zooming" instead of the ambient dimension, an instance-dependent quantity measuring the benignness of the optimal Q_h^* function. Many applications in computing systems and operations research require algorithms that compete on three facets: low sample complexity, mild storage requirements, and low computational burden for policy evaluation and training. Our algorithms are easily adapted to operating constraints, and our theory provides explicit bounds across each of the three facets.

Funding: This work is supported by funding from the National Science Foundation [Grants ECCS-1847393, DMS-1839346, CCF-1948256, and CNS-1955997] and the Army Research Laboratory [Grant W911NF-17-1-0094].
Supplemental Material: The online appendix is available at https://doi.org/10.1287/opre.2022.2396.

Keywords: reinforcement learning • metric spaces • adaptive discretization • online learning

1. Introduction

Reinforcement learning (RL) is a popular approach for sequential decision making and is successfully applied to games (Silver et al. 2016) and systems applications (Alizadeh et al. 2010). In these models, a principal interacts with a system that has stochastic transitions and rewards. The principal aims to control the system either online through exploring available actions using real-time feedback or off-line by exploiting known properties of the system and an existing data set.

These sequential decision-making problems are considered across multiple communities. As data has become more readily available and computing power improves, the new zeitgeist for these fields is developing data-driven decision algorithms: algorithms that adapt to the structure of information, constraints, and objectives in any given domain. This paradigm highlights the importance of taking advantage of data collected and the inherent structure and geometry of the problem to help algorithms scale to complex domains.

With the successes of neural networks as a universal function approximator, RL has received a lot of interest in the design of algorithms for large-scale systems using parametric models (Jiang et al. 2017, Mozur 2017). Whereas these results highlight the power of RL in learning complex control policies, they are infeasible for many applications arising in operations research and computing systems (Hubbs et al. 2020). As an example, the AlphaGo Zero algorithm that mastered Chess and Go from scratch was trained over 72 hours using four tensor processing units and 64 graphics processing units (Silver et al. 2016). The limiting factor in using these large-scale parametric algorithms is implementing regression oracles or gradient steps on computing hardware and the large storage burden in maintaining the models. These issues don't typically arise in game-based or robotics applications. However, they are key algorithmic ingredients that are ignored in theoretical treatment analyzing storage and time complexity. Moreover, these models require strict parametric assumptions, suffer under model misspecification, and do not adapt to the underlying geometry of the problem.

In contrast, RL has received interest in designing small-scale and efficient controllers for problems arising

in operations research (OR) and computing systems, including memory systems (Alizadeh et al. 2010) and resource allocation in cloud-based computing (Ipek et al. 2008). Their engineering approaches use discretizations at various levels of coarseness to learn estimates in a data-efficient manner. Common to these examples are computation and storage limitations on the devices used for the controller, requiring algorithms to compete on three major facets: efficient learning, low computation, and low storage requirements—a trifecta for RL in OR.

Motivated by this paradigm, we consider nonparametric discretization (or quantization) techniques that map complex problems to discrete ones. These algorithms are based on simple primitives that are easy to implement on hardware, can leverage existing hardware quantization techniques, and have been tested heuristically in practice (Uther and Veloso 1998; Pyeatt et al. 2001; Lolos et al. 2017; Araújo et al. 2020a, b). A key challenge in this approach is picking a discretization to manage the trade-off between the discretization error and the errors accumulated from solving the discrete problem. Moreover, if the discretization is fixed a priori, then the algorithm cannot adapt to the underlying structure in the problem, and so adaptive discretizations are necessary for instance-specific gains. We develop theoretical foundations for an adaptive discretization of the space, in which the discretization is only refined on an as needed basis using collected data. We answer the two important aspects of designing adaptive discretization algorithms: how to create the discretization and when to refine it by exploiting the metric structure induced by specific problem instances.

1.1. Our Contributions

We provide a unified analysis of AdaMB and AdaQL, model-based and model-free algorithms that discretize the state action space in a data-driven way so as to minimize regret. This extends the well-studied adaptive discretization techniques seen in contextual multiarmed bandits to dynamic environments (Slivkins 2014, Kleinberg et al. 2019). Moreover, it illustrates that adaptive discretization can be viewed as an all-purpose tool for improving fixed discretization algorithms to better take advantage of problem structure.

These algorithms require that the state and action spaces are embedded in compact metric spaces, and the problem primitives are Lipschitz continuous with respect to this metric. This encompasses discrete and continuous state action spaces with mild assumptions on the transition kernel and rewards. Our algorithms only requires access to the metric unlike prior nonparametric algorithms that require access to simulation oracles or impose additional assumptions on the action space to be computationally efficient. In fact, the assumption that $|\mathcal{A}| < \infty$ is quite common in theoretical treatment for simplicity. However, this ignores the technical and computational hurdles required. Our algorithms avoid this issue through the efficient discretization of the action space.

We show that AdaMB and AdaQL achieve near optimal dependence of the regret on the zooming dimension of the metric space, an instance-dependent quantity that measures the intrinsic complexity and geometry of the problem by scaling with the dimension of the level sets of the optimal $Q_h^\star$ function instead of the ambient dimension. Our main result is summarized in the following informal theorem.

Informal Theorem 1. *For an H-step MDP played over K episodes, our algorithms achieve regret*

$$\text{Regret}(K) \lesssim \begin{cases} \text{AdaQL} : & H^{5/2}K^{\frac{z_{max}+1}{z_{max}+2}} \\ \text{AdaMB} : & H^{3/2}K^{\frac{z_{max}+d_S-1}{z_{max}+d_S}} \quad d_S > 2 \\ \text{AdaMB} : & H^{3/2}K^{\frac{z_{max}+1}{z_{max}+2}} \quad d_S \le 2 \end{cases}$$

where d_S is the covering of the state space and $z_{max} = \max_h z_h$ is the worst case over the step h zooming dimensions z_h.

Our bounds are uniformly better in terms of dependence on K and H than the best existing bounds for nonparametric RL (see Table 1). Our bounds exhibit explicit dependence on the zooming instead of the ambient dimension, leading to exponential improvements in sample complexity because the zooming dimension is trivially upper bounded by the ambient dimension. In addition, AdaQL matches the lower bound up to polylogarithmic factors, whereas AdaMB suffers from additional d_S terms when $d_S > 2$. In general, the lower bound shows that this exponential scaling is necessary in nonparametric settings (a fundamental trade-off of using nonparametric algorithms), but note that the exponential scaling is with respect to the zooming instead of ambient dimensions. We also show that AdaMB matches the bounds of AdaQL under additional assumptions of the transition distribution.

In addition to having lower regret, AdaMB and AdaQL are also simple and practical to implement with low query complexity and storage compared with other techniques (see Table 1). To the best of our knowledge, our algorithms are the first to have provably sublinear regret with improved time and storage complexity in the setting of continuous state and action MDPs. We complement our theory with synthetic experiments comparing model-free and model-based algorithms using both fixed and adaptive discretizations. We picked experiments of varying complexity in low-dimensional spaces, including those with provably smaller zooming dimensions, to help highlight the adaptive discretizations matching the level sets of the underlying $Q_h^\star$ value. Through our experiments, we measure the three aspects of the trifecta for RL in OR, comparing the performance of these algorithms in terms of regret and

Table 1. Comparison of Our Bounds with Several State-of-the-Art Bounds for Nonparametric RL

Algorithm	Type	Regret	Time	Space
AdaMB ($d_{\mathcal{S}} > 2$)	MB	$H^{3/2}K^{\frac{z_{max}+d_{\mathcal{S}}-1}{z_{max}+d_{\mathcal{S}}}}$	$HK^{\frac{d+2d_{\mathcal{S}}}{d+d_{\mathcal{S}}}}$	HK
AdaMB ($d_{\mathcal{S}} \leq 2$)	MB	$H^{3/2}K^{\frac{z_{max}+1}{z_{max}+2}}$	$HK^{\frac{d+d_{\mathcal{S}}+2}{d+d_{\mathcal{S}}}}$	$HK^{\frac{d+d_{\mathcal{S}}}{d+2}}$
AdaQL	MF	$H^{5/2}K^{\frac{z_{max}+1}{z_{max}+2}}$	$HK\log_d(K)$	$HK^{\frac{d}{d+2}}$
Kernel UCBVI (Domingues et al. 2020)	MB	$H^3\ K^{\frac{2d}{2d+1}}$	HAK^2	HK
Net-Based Q-Learning (Song and Sun 2019)	MF	$H^{5/2}K^{\frac{d+1}{d+2}}$	HK^2	HK
Net-Based UCBVI	MB	$H^{3/2}K^{\frac{2d+1}{2d+2}}$	H^2K^2	HK
Lower Bound	—	$H\ K^{\frac{z_{max}+1}{z_{max}+2}}$	—	—

Notes. d is the covering dimension of the state-action space, $d_{\mathcal{S}}$ is the covering dimension of the state space, H is the horizon of the MDP, and K is the total number of episodes. Under "Type" we denote whether the algorithm is model-based (MB) or model-free (MF). As implementing Kernel UCBVI (Domingues et al. 2020) is unclear under general action spaces, we specialize the time complexity under a finite set of actions of size A, but more details are included in their paper. See Online Appendix H for a discussion on Net-Based UCBVI.

time and space complexity. Our experiments show that, with a fixed discretization, model-based algorithms outperform model-free ones and suffer from worse storage and computational complexity. However, when using an adaptive partition of the space, model-based and model-free algorithms perform similarly.

1.2. Motivating Examples

Reinforcement learning has enjoyed remarkable success in recent years in large-scale game playing and robotics. These results, however, mask the high underlying costs in terms of computational resources, energy costs, training time, and hyperparameter tuning that their demonstrations require (Mnih et al. 2013, 2016; Silver et al. 2016, 2017). On the other hand, RL is applied heuristically in the following problems.

1.2.1. Memory Management. Many computing systems have two sources of memory: on-chip memory, which is fast but limited, and off-chip memory, which has low bandwidth and suffers from high latency. Designing memory controllers for these systems requires a scheduling policy to adapt to changes in workload and memory reference streams, ensuring consistency in the memory and controlling for long-term consequences of scheduling decisions (Alizadeh et al. 2010, 2013; Chinchali et al. 2018).

1.2.2. Online Resource Allocation. Cloud-based clusters for high-performance computing must decide how to allocate computing resources to different users or tasks with highly variable demand. Controllers for these algorithms make decisions online to manage the tradeoffs between computation cost, server costs, and delay in job completions (Ipek et al. 2008, Nishtala et al. 2013, Lykouris and Vassilvitskii 2018, Tessler et al. 2022).

Common to these examples are computation and storage limitations on the devices used for the controller.

1. Limited memory: As any RL algorithm requires memory to store estimates of relevant quantities, algorithms for computing systems must manage their storage requirements, so frequently needed estimates are stored in on-chip memory.

2. Power consumption: Many applications require low power consumption for executing RL policies on general computing platforms.

3. Latency requirements: Problems for computing systems (e.g., memory management) have strict latency quality of service requirements that limits reinforcement learning algorithms to execute their policy quickly.

A common technique in these domains is cerebellar model articulation controllers (CMACs) or other quantization and hashing based methods, which are used in optimizing controllers for dynamic RAM access (Ipek et al. 2008, Nishtala et al. 2013, Lykouris and Vassilvitskii 2018). The CMAC technique uses a random discretization of the space at various levels of coarseness combined with hashing. The other approaches are hierarchical decision trees, in which researchers investigate splitting heuristics for refining the adaptive partition by testing their empirical performance (Uther and Veloso 1998, Pyeatt and Howe 2001, Lolos et al. 2017). These quantization-based algorithms address the computation and storage limitations by allowing algorithm implementations to exploit existing hashing and caching techniques for memory management because the algorithms are based on simple look-up tables. Our algorithms are motivated by these approaches, taking a first step toward designing theoretically efficient reinforcement learning algorithms for continuous spaces.

1.3. Related Work

There is an extensive and growing literature on reinforcement learning; we highlight the work that is closest to ours, but for more extensive references, see Sutton and Barto (2018), Agarwal et al. (2019), Puterman (1994), and Powell (2022) for RL and Bubeck and Cesa-Bianchi (2012) and Slivkins (2019) for bandits.

1.3.1. Tabular RL. There is a long line of research on the regret for RL in tabular settings. In particular, the first asymptotically tight regret bound for tabular model-based algorithms with nonstationary dynamics is established to be $O(H^{3/2}\sqrt{SAK})$, where S and A are the size of the state and action spaces, respectively (Azar et al. 2013). These bounds were matched (in terms of K) using an "asynchronous value-iteration" (or one-step planning) approach (Azar et al. 2017, Efroni et al. 2019), which is simpler to implement (Ipek et al. 2008, Nishtala et al. 2013, Lykouris and Vassilvitskii 2018, Tessler et al. 2022). This regret bound was also matched (in terms of K) for model-free algorithms (Jin et al. 2018). More recently, the analysis was extended to develop instance-dependent bounds as a function of the variance or shape of the underlying $Q_h^\star$ function (Simchowitz and Jamieson 2019, Zanette and Brunskill 2019). Our work extends this latter approach to continuous spaces using adaptive discretization to obtain instance-specific guarantees scaling with the zooming dimension instead of the ambient dimension of the space.

1.3.2. Parametric Algorithms. For RL in continuous spaces, several recent works focus on the use of linear function approximation (Osband and Van Roy 2014; Du et al. 2019; Wang et al. 2019, 2020; Zanette et al. 2019; Jin et al. 2020; Chen and Zhang 2021). These works assume that the principal has a feature extractor under which the process is well-approximated by a linear model. In practice, these algorithms require an initial "feature engineering" process to learn features under which the problem is linear, and the guarantees then hinge upon a perfect construction of features. If the requirements are violated, it is shown that the theoretical guarantees degrade poorly (Du et al. 2020). Other work extends this approach to problems with bounded eluder dimension and other notions of dimension of parametric problems (Russo and Van Roy 2013, Wang et al. 2020).

1.3.3. Nonparametric Algorithms. In contrast, nonparametric algorithms only require mild local assumptions on the underlying process, most commonly that the Q-function is Lipschitz continuous with respect to a given metric. For example, Yang et al. (2019) and Shah and Xie (2018) consider nearest neighbor methods for deterministic, infinite horizon discounted settings. Others assume access to a generative model (Kakade et al. 2003, Henaff 2019, Shah et al. 2020).

The works closest to ours concern online algorithms for finite horizon problems with continuous state action spaces (see also Table 1). In model-free settings, tabular algorithms are adapted to continuous state-action spaces via fixed discretization (i.e., ϵ-nets) (Song and Sun 2019). In model-based settings, researchers tackle continuous spaces using kernel methods based on either a fixed discretization of the space (Lakshmanan et al. 2015) or with smooth kernel functions (Domingues et al. 2020). Whereas the latter learns a data-driven representation of the space, it requires solving a complex optimization problem over actions at each step and, hence, is efficient mainly for finite action sets (more discussion on this is in Section 4). Finally, adaptive discretization is successfully implemented and analyzed in model-free and model-based settings (Sinclair et al. 2019, Cao and Krishnamurthy 2020, Sinclair et al. 2020). This work serves as a follow-up providing a unified analysis between the two approaches, improved performance guarantees for AdaMB, instance-dependent guarantees scaling with - the zooming instead of the ambient dimension, and additional numerical simulations.

1.3.4. Discretization-Based Approaches. Discretization-based approaches to reinforcement learning are explored heuristically in different settings. One line of work investigates adaptive basis functions, in which the parameters of the functional model (e.g., neural network) are learned online, simultaneously adapting the basis functions (Menache et al. 2005, Keller et al. 2006, Whiteson and Stone 2006). Similar techniques are done with soft state aggregation (Singh et al. 1995). Most similar to our algorithm, though, are tree-based partitioning rules, which store a hierarchical partition of the state and action space that is refined over time (Uther and Veloso 1998, Pyeatt and Howe 2001, Lolos et al. 2017). These were tested heuristically with various splitting rules (e.g., Gini index, etc.), and instead, our algorithm splits based off the metric and statistical uncertainty in the estimates. Researchers also extend our adaptive discretization techniques to using a single partition in infinite horizon time-discounted settings, and the algorithm is benchmarked on various control tasks in Open AI showing comparative performance between discretization and deep learning techniques (Araújo et al. 2020a, b).

1.4. Outline of Paper

Section 2 introduces the model, nonparametric assumptions, and the zooming dimension. Our algorithms, AdaQL and AdaMB, are described in Section 3 with the regret bound and proof sketch given in Sections 4 and 5, respectively. Proof details are included in Online Appendices A–F with miscellaneous technical results in Online Appendix I. The experimental results are in Online Appendix J.

2. Preliminaries

2.1. MDP and Policies

We consider the online episodic reinforcement learning setting, in which an agent is interacting with an underlying finite-horizon Markov decision process (MDP) over K sequential episodes, denoted $[K] = \{1, \ldots, K\}$(Puterman 1994).

Definition 1. An episodic MDP is given by a five-tuple $(\mathcal{S}, \mathcal{A}, H, T, R)$, where the horizon H is the number of steps indexed $[H] = \{1, 2, \ldots, H\}$ in each episode and $(\mathcal{S}, \mathcal{A})$ denotes the set of states and actions in each step. State transitions are governed by a collection of transition kernels $T = \{T_h(\cdot|x,a)\}_{h\in[H],\, x\in\mathcal{S},\, a\in\mathcal{A}}$, where $T_h(\cdot|x,a) \in \Delta(\mathcal{S})$ gives the distribution over states in $\mathcal{S}$ if action a is taken in state x at step h. The instantaneous rewards are bounded in $[0,1]$, and their distributions are specified by a collection of parameterized distributions $R = \{R_h\}_{h\in[H]}$, $R_h : \mathcal{S} \times \mathcal{A} \to \Delta([0,1])$. We let $r_h(x,a) = \mathbb{E}_{r\sim R_h(x,a)}[r]$ denote the mean reward.

The agent interacts with the MDP by selecting a policy, and a policy π is a sequence of distributions $\pi = \{\pi_h | h \in [H]\}$, where each $\pi_h : \mathcal{S} \to \Delta(\mathcal{A})$ is a mapping from a given state $x \in \mathcal{S}$ to a distribution over actions in $\mathcal{A}$.

2.2. Value Function and Bellman Equations

For any policy π, let A_h^π denote the (potentially random) action taken in step h under policy π, that is, $A_h^\pi = \pi_h(X_h^k)$.

Definition 2. We define the policy value function at step h under policy π to be the expected sum of future rewards under policy π starting from $X_h = x$ in step h until the end of the episode, which we denote $V_h^\pi : \mathcal{S} \to \mathbb{R}$. Formally,

$$V_h^\pi(x) := \mathbb{E}\left[\sum_{h'=h}^{H} R_{h'} \,\middle|\, X_h = x\right] \quad \text{for} \quad R_{h'} \sim R_{h'}(X_{h'}, A_{h'}^\pi). \tag{1}$$

We define the state-action value function (or Q-function) $Q_h^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ at step h as the sum of the expected rewards received after taking action $A_h = a$ at step h from state $X_h = x$ and then following policy π in all subsequent steps of the episode. Formally,

$$Q_h^\pi(x,a) := r_h(x,a) + \mathbb{E}\left[\sum_{h'=h+1}^{H} R_{h'} \,\middle|\, X_{h+1} \sim T_h(\cdot|x,a)\right] \quad \text{for} \quad R_{h'} \sim R_{h'}(X_{h'}, A_{h'}^\pi). \tag{2}$$

Under suitable assumptions on $\mathcal{S} \times \mathcal{A}$ there exists a deterministic optimal policy $\pi^\star$ that gives the optimal value $V_h^\star(x) = \sup_\pi V_h^\pi(x)$ for all $x \in \mathcal{S}$ and $h \in [H]$ (Puterman 1994). For ease of notation, we denote $Q^\star = Q^{\pi^\star}$ and $V^\star = V^{\pi^\star}$. Recall the Bellman equations (Puterman 1994) which state that

$$\begin{aligned} V_h^\pi(x) &= Q_h^\pi(x, \pi_h(x)) && \forall\, x \in \mathcal{S} \\ Q_h^\pi(x,a) &= r_h(x,a) + \mathbb{E}[V_{h+1}^\pi(X_{h+1}) | X_h = x, A_h = a] && \forall\, (x,a) \in \mathcal{S} \times \mathcal{A} \\ V_{H+1}^\pi(x) &= 0 && \forall\, x \in \mathcal{S}. \end{aligned} \tag{3}$$

For the optimal policy $\pi^\star$, it additionally holds that $V_h^\star(x) = \max_{a\in\mathcal{A}} Q_h^\star(x,a)$.

2.2.1. Online Interaction Structure.

We consider an agent interacting with the MDP in the online setting. At the beginning of each episode k, the agent fixes a policy π^k for the entire episode and is given an initial (arbitrary) state $X_1^k \in \mathcal{S}$. In each step $h \in [H]$, the agent receives the state X_h^k, picks an action $A_h^k = \pi_h^k(X_h^k)$, receives reward $R_h^k \sim R_h(X_h^k, A_h^k)$, and transitions to a random state $X_{h+1}^k \sim T_h(\cdot|X_h^k, A_h^k)$ sampled from the transition distribution. This continues until the final transition to state X_{H+1}^k, at which point the agent chooses policy π^{k+1} for the next episode after incorporating observed data (including the rewards and transitions), and the process is repeated. The goal is to maximize the total expected reward $\sum_{k=1}^K V_1^{\pi^k}(X_1^k)$. We benchmark the agent on the regret: the additive loss over all episodes the agent experiences using the agent's policy instead of the optimal one. As the policies cannot be anticipatory, we introduce $\mathcal{F}_k = \sigma((X_h^{k'}, A_h^{k'}, R_h^{k'})_{h\in[H],k'<k,}, X_1^k)$ to denote the information available to the decision maker at the start of episode k.

Definition 3. The REGRET for an algorithm that deploys a sequence of $\mathcal{F}_k$-measurable policies $\{\pi^k\}_{k\in[K]}$ given a sequence of initial states $\{X_1^k\}_{k\in[K]}$ is defined as

$$\text{REGRET(K)} = \sum_{k=1}^{K} \left(\mathrm{V}_1^\star(\mathrm{X}_1^k) - \mathrm{V}_1^{\pi^k}(\mathrm{X}_1^k)\right). \tag{4}$$

Our goal is to develop algorithms that have regret REGRET(K) growing sublinearly in K and low per-step storage and computational requirements.

2.3. Metric Space and Lipschitz Assumptions

In contrast to parametric algorithms, our algorithms require flexible assumptions on the underlying process. At a high level, we require the algorithm to have access to a metric on the state action space under which the underlying $Q_h^\star$ function (or rewards and dynamics) are Lipschitz continuous. This is well-motivated in problems in continuous domains (in which the metric can be taken to be any ℓ_p metric on the Euclidean space). For large discrete spaces, it requires an embedding of the space in a metric space that encodes meaningful relationships between the discrete states and actions (Ipek et al. 2008). Trivially, any problem can be embedded in a metric space in which the metric is taken to be the difference of $Q_h^\star$ values, but this requires knowing the optimal values for determining the adaptive partition. Recent work investigates the options of selecting a metric in terms of its induced topological structure on the space (Le Lan et al. 2021).

We assume the state space $\mathcal{S}$ and the action space $\mathcal{A}$ are each separable compact metric spaces with metrics $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{A}}$. We assume that the transition kernels $\{T_h(\cdot|x,a)\}_{x,a\in\mathcal{S}\times\mathcal{A}}$ are Borel measures with respect to the topology induced by $\mathcal{D}_{\mathcal{S}}$ on $\mathcal{S}$. These metrics impose a metric structure $\mathcal{D}$ on $\mathcal{S}\times\mathcal{A}$ via the product metric or any subadditive metric such that $\mathcal{D}((x,a),(x',a')) \le \mathcal{D}_{\mathcal{S}}(x,x') + \mathcal{D}_{\mathcal{A}}(a,a')$.

The covering dimension of a compact metric space $\mathcal{X}$ is defined as $d_{\mathcal{X}} = \min\{d > 0 : N_r(\mathcal{X}) \le cr^{-d} \ \ \forall r > 0\}$ with $N_r(\mathcal{X})$ as the r-packing number of the set $\mathcal{X}$. This metric structure on $\mathcal{S}\times\mathcal{A}$ ensures that the covering dimension of $\mathcal{S}\times\mathcal{A}$ is at most $d = d_{\mathcal{S}} + d_{\mathcal{A}}$, where $d_{\mathcal{S}}$ and $d_{\mathcal{A}}$ are the covering dimensions of $\mathcal{S}$ and $\mathcal{A}$, respectively. For notational brevity we omit with respect to which metric the packing numbers are computed as it should be clear from the context.

We assume without loss of generality that $\mathcal{S}\times\mathcal{A}$ has diameter one, and we denote the diameter of $Y \subset \mathcal{S}$ as $\mathcal{D}(Y) = \sup_{a\in\mathcal{A},(x,y)\in Y^2} \mathcal{D}((x,a),(y,a))$ and overload notation and use $\text{diam}(B) = \max\{\mathcal{D}((x,a),(y,b))|(x,a),(y,b) \in B\}$ to be the diameter of a region $B \subset \mathcal{S}\times\mathcal{A}$. For more information on metrics and covering dimension, see Slivkins (2014), Kleinberg et al. (2019), Sinclair et al. (2019), and Royden and Fitzpatrick (1988) for a summary.

To motivate the discretization approach, we also assume Lipschitz structure on the system. This can come in two forms, which we call *model-free Lipschitz* and *model-based Lipschitz* (required for AdaQL and AdaMB, respectively). We start with model-free Lipschitz, which assumes the underlying $Q_h^\star$ and $V_h^\star$ functions are Lipschitz continuous.

Assumption 1 (Model-Free Lipschitz). *The optimal $Q_h^\star$ and $V_h^\star$ are Lipschitz continuous with respect to $\mathcal{D}$ and $\mathcal{D}_{\mathcal{S}}$, that is, for every $x,x',a,a',h \in \mathcal{S}^2\times\mathcal{A}^2\times[H]$,*

$$|Q_h^\star(x,a) - Q_h^\star(x',a')| \le L_V \mathcal{D}((x,a),(x',a'))$$
$$|V_h^\star(x) - V_h^\star(x')| \le L_V \mathcal{D}_{\mathcal{S}}(x,x').$$

The next assumption, model-based Lipschitz, puts Lipschitz assumptions on the underlying rewards and dynamics of the system.

Assumption 2 (Model-Based Lipschitz). *The average reward function $r_h(x,a)$ is Lipschitz continuous with respect to $\mathcal{D}$, and the transition kernels $T_h(\cdot|x,a)$ are Lipschitz continuous in the 1-Wasserstein metric d_W with respect to $\mathcal{D}$,that is, for every $x,x',a,a',h \in \mathcal{S}^2\times\mathcal{A}^2\times[H]$,*

$$|r_h(x,a) - r_h(x',a')| \le L_r \mathcal{D}((x,a),(x',a'))$$
$$d_W(T_h(\cdot|x,a), T_h(\cdot|x',a')) \le L_T \mathcal{D}((x,a),(x',a')).$$

It's important to note that Assumption 2 implies Assumption 1.

Lemma 1. *Suppose that Assumption 2 holds. Then, Assumption 1 holds with $L_V = \sum_{h=0}^{H} L_r L_T^h$.*

The next assumption is similar to previous literature for algorithms in general metric spaces (Slivkins 2014, Kleinberg et al. 2019, Sinclair et al. 2019). This assumes access to the similarity metrics. Learning (or picking) the metric is important in practice but beyond the scope of this paper (Wanigasekara and Yu 2019, Le Lan et al. 2021).

Assumption 3. *The agent has oracle access to the similarity metrics via several queries that are used by the algorithm.*

In particular, AdaMB and AdaQL require access to several covering and packing oracles that are used throughout the algorithm. For more details on the assumptions required and implementing the algorithm in practice, see Online Appendix H.

2.4. Zooming Dimension

Our theoretical guarantees scale with respect to an instance-dependent zooming dimension of $\mathcal{S}\times\mathcal{A}$ instead of the ambient dimension. This serves as an analog to the zooming dimension originally appearing in instance-dependent bounds in the bandit literature (Slivkins 2014) extended to dynamic settings and a continuous analog to instance-dependent guarantees developed for RL in the tabular setting (Simchowitz and Jamieson 2019). Analyzing the zooming dimension for reinforcement learning problems is much more technical than in the simpler bandit setting because of having to account for the dynamics of the problem. We start by introducing the concept of a GAP, a quantity measuring the suboptimality of a given state action pair $(x,a) \in \mathcal{S}\times\mathcal{A}$.

Definition 4. For any $(x,a) \in \mathcal{S}\times\mathcal{A}$ and $h \in [H]$, the stage-dependent suboptimality gap is $\text{GAP}_h(x,a) = V_h^\star(x) - Q_h^\star(x,a)$.

One can interpret $\text{GAP}_h(x,a)$ as a measure of regret the algorithm experiences upon taking action a in state x in step h instead of the optimal action. This definition simplifies to the same definition of GAP developed in the contextual bandit literature when the transition distribution of the problem is independent of the given state and action.

In bandits, existing results show that adaptive discretization algorithms only discretize a subset of the entire state and action set, defined as the set of points whose gap is small. Whereas we later see in Section 5 that the same does not extend to reinforcement learning, we are still able to bound the regret of the algorithm based on the size of a set of near optimal points.

Definition 5. We define the near-optimal set of $\mathcal{S}\times\mathcal{A}$ for a given value r as

$$Z_h^r = \{(x,a) \in \mathcal{S}\times\mathcal{A}|\ \text{GAP}_h(x,a) \le C_L(H+1)r\},$$

where C_L is an absolute constant depending on the Lipschitz constants of the problem.

Clearly, we have that $Z_h^r \subset \mathcal{S} \times \mathcal{A}$. However, for many problem instances, Z_h^r can be a (much) lower dimensional manifold. Finally, we define the step h zooming dimension as follows.

Definition 6. The step h zooming dimension with constant c_h is z_h such that

$$z_h = \inf\{d > 0 : N_r(Z_h^r) \le c_h r^{-d} \quad \forall r > 0\}.$$

We also denote $z_{max} = \max_{h\in[H]} z_h$ to be the worst case zooming dimension across all of the steps.

To give some intuition behind the zooming dimension, first notice that, whereas the covering dimension is focused on covering the entire metric space, the zooming dimension focuses instead on covering a near-optimal subset of it. This serves as a way to quantify the benignness of a problem instance. Whereas it is trivially no larger than the zooming dimension, in many settings, it can be significantly smaller.

Lemma 2. *The following examples show improved scaling of the zooming dimension over the ambient dimension:*

– Linear $Q_h^\star$: Suppose that $Q_h^\star(x,a) = \theta^\top(x,a)$ for some vector $\theta \in \mathbb{R}^{d_\mathcal{S}+d_\mathcal{A}}$ with $\mathcal{S} \subset \mathbb{R}^{d_\mathcal{S}}$ and $\mathcal{A} \subset \mathbb{R}^{d_\mathcal{A}}$ under any ℓ_p norm. Then, $z_h \le d_\mathcal{S} + d_\mathcal{A} - \|\theta_\mathcal{A}\|_0$.

– Low-dimensional optimality: Suppose that there exists a set $Y \subset \mathcal{A}$ that contains all optimal or near-optimal actions for every state. Then, $z_h \le d_\mathcal{S} + d_Y$.

– Strongly concave: Suppose that the metric space is $\mathcal{S} = [0,1]^{d_\mathcal{S}}$ and $\mathcal{A} = [0,1]^{d_\mathcal{A}}$ under any ℓ_p metric. If $Q_h^\star(x,a)$ is C^2 smooth, and for all $x \in \mathcal{S}$, we have that $Q_h^\star(x,\cdot)$ has a unique maxima and is strongly concave in a neighborhood around the maxima, then $z_h \le d_\mathcal{S} + \frac{d_\mathcal{A}}{2}$.

Analyzing the zooming dimension in reinforcement learning is more complicated than in bandit settings as you have to show properties of the $Q_h^\star$ function, which is coupled by the dynamics of the system. In the experiments in Online Appendix J, we highlight problem instances with improved bounds on the zooming dimension.

Note that all of the examples presented in Lemma 2 have $z_h \ge d_\mathcal{S}$. This is as the zooming dimension does not take into account the distribution over states that are visited by the optimal policy. As such, scaling with respect to $d_\mathcal{S}$ is inevitable because the set $\{(x, \pi_h^\star(x)) : x \in \mathcal{S}\}$ is contained in Z_h^r for any $r > 0$. This results in the following lower bound on the zooming dimension.

Lemma 3. *For any h, we have that $z_h \ge d_\mathcal{S} - 1$.*

Even in the simpler contextual multiarmed bandit model, the zooming dimension necessarily scales with the dimension of the context space regardless of the support or mass over the context space the context distribution places. Whereas, analytically, we cannot show gains in the state space dimension, we see empirically in Online Appendix J that the algorithms only cover the state space in regions the optimal policy visits, but it is unclear how to include this intuition formally in the definition. Revisiting new notions of "instance-specific" complexity is an interesting direction for future work in both tabular and continuous RL.

Algorithm 1 (Adaptive Discretization for Online Reinforcement Learning (AdaMB, AdaQL))

procedure Adaptive Discretization for Online RL($\mathcal{S}$, $\mathcal{A}, \mathcal{D}, H, K, \delta$)
  Initialize partitions $\mathcal{P}_h^0 = \mathcal{S} \times \mathcal{A}$ for $h \in [H]$, estimates $\overline{\mathbf{Q}}_h^0(\cdot) = \overline{\mathbf{V}}_h^k(\cdot) = H - h + 1$
  for each episode $k \leftarrow 1, \dots K$ **do**
    Receive starting state X_1^k
    for each step $h \leftarrow 1, \dots, H$ **do**
      Observe X_h^k and determine $\text{RELEVANT}_h^k(X_h^k) = \{B \in \mathcal{P}_h^{k-1} | X_h^k \in B\}$
      Greedy selection rule: pick $B_h^k = \arg\max_{B \in \text{RELEVANT}_h^k(X_h^k)} \overline{\mathbf{Q}}_h^{k-1}(B)$
      Play action $A_h^k = \tilde{a}(B_h^k)$ associated with ball B_h^k; receive R_h^k and transition to X_{h+1}^k
      Update Estimates($X_h^k, A_h^k, X_{h+1}^k, R_h^k, B_h^k$) via AdaMB or AdaQL
      if $\text{Conf}_h^k(B_h^k) \le \text{diam}(B_h^k)$ **then** Refine Partition (B_h^k)
procedure Refine Partition(B, h, k)
  Construct $\mathcal{P}(B) = \{B_1, \dots, B_m\}$ as the children of B in the hierarchical partition
  Update $\mathcal{P}_h^k = \mathcal{P}_h^{k-1} \cup \mathcal{P}(B) \setminus B$
  For each B_i, initialize estimates from parent ball
procedure Update Estimates (AdaMB) ($X_h^k, A_h^k, X_{h+1}^k, R_h^k, B_h^k$)
  for each $h \leftarrow 1, \dots H$ and $B \in \mathcal{P}_h^k$ **do**: Update $\overline{\mathbf{Q}}_h^k(B)$ and $\overline{\mathbf{V}}_h^k(\cdot)$ via Equations (6) and (8)
procedure Update Estimates (AdaQL) ($X_h^k, A_h^k, X_{h+1}^k, R_h^k, B_h^k$)
  Update $\overline{\mathbf{Q}}_h^k(B)$ via Equation (9).

3. Algorithm

Reinforcement learning algorithms come in two primary forms: policy- or value-based learning. Policy-based learning focuses on directly iterating on the policy used in episode k π^k through optimizing over a set of candidate policies (Bhandari and Russo 2021). Our algorithms use value-based learning, which focuses on constructing estimates $\overline{\mathbf{Q}}_h^k$ for $Q_h^\star$. The algorithms then play the policy π_h^k that is greedy with respect to the estimates, that is,

$$\pi_h^k(X_h^k) = \arg\max_{a \in \mathcal{A}} \overline{\mathbf{Q}}_h^{k-1}(X_h^k, a).$$

The motivation behind this approach is that the optimal policy plays $\pi_h^\star(X_h^k) = \arg\max_{a\in\mathcal{A}} Q_h^\star(X_h^k, a)$. The hope is

that, when $\overline{\mathbf{Q}}_h$ is uniformly close to $Q_h^\star$, then the value of the policy used is similar to that of the optimal policy. What distinguishes between different value-based algorithms is the method used to construct the estimates for $Q_h^\star$. We provide both model-based and model-free variations of our algorithm. Note that, for both algorithms, we set $\overline{\mathbf{V}}_h^k(x) = \max_{a\in\mathcal{A}} \overline{\mathbf{Q}}_h^k(x,a)$.

1. Model-based (AdaMB) estimates the MDP parameters directly (r_h and T_h) and plugs in the estimates to the Bellman Equation (3) to set

$$\overline{\mathbf{Q}}_h^k(x,a) \approx \overline{\mathbf{r}}_h^k(x,a) + \mathbb{E}_{Y\sim\overline{\mathbf{T}}_h^k(x,a)}[\overline{\mathbf{V}}_{h+1}^k(Y)],$$

where $\overline{\mathbf{r}}$ and $\overline{\mathbf{T}}$ denote explicit empirical estimates for the reward and transition kernel.

2. Model-free (AdaQL) foregoes estimating the MDP parameters directly and instead does one-step updates in the Bellman Equation (3) to obtain $\overline{\mathbf{Q}}_h^k(x,a) \approx (1-\alpha_t)\overline{\mathbf{Q}}_h^{k-1}(x,a) + \alpha_t(R_h^k + \overline{\mathbf{V}}_{h+1}^k(X_{h+1}^k))$. By decomposing the recursive relationship, you get

$$\overline{\mathbf{Q}}_h^k(x,a) \approx \sum_{i=1}^{t} \alpha_t^i (R_h^{k_i} + \overline{\mathbf{V}}_{h+1}^{k_i}(X_{h+1}^{k_i})),$$

where t is the number of times (x, a) has been visited and $k_1,\ldots,k_t$ denote the episodes it was visited before, α_t is the learning rate, and $\alpha_t^i = \alpha_i\prod_{j=i+1}^t(1-\alpha_j)$. Note that this instead stores implicit estimates of the average reward (weighted by the learning rate) and the transition kernel. The key difference is that the estimate of the next step is not updated based on the current episode k as $\overline{\mathbf{V}}_{h+1}^k(\cdot)$, but instead is updated based on the episode it was visited in the past by $\overline{\mathbf{V}}_{h+1}^{k_i}(\cdot)$. As a result, the learning rates are chosen to impose recency bias for the estimates (Jin et al. 2018). If $\overline{\mathbf{V}}_{h+1}^k(\cdot)$ was used, the algorithm would need to store all of the data and recompute these quantities, leading to regret improvements only in logarithmic terms. We see later that this approach leads to substantial time and space complexity improvements.

3.1. Algorithm Description

We now present our model-based and model-free reinforcement learning algorithms with adaptive partitioning, which we refer to as AdaMB and AdaQL, respectively. Our algorithms proceed in the following four steps:

1. (Adaptive partition) The algorithms maintain an adaptive hierarchical partition of $\mathcal{S}\times\mathcal{A}$ for each step h, which is used to represent the (potentially continuous) state and action space. The algorithms also maintain estimates $\overline{\mathbf{Q}}_h$ and $\overline{\mathbf{V}}_h$ for $Q_h^\star$ and $V_h^\star$ over each region that are constructed based on collected data.

2. (Selection rule) Upon visiting a state X_h^k in step h episode k, the algorithms pick a region containing X_h^k that maximizes the estimated future reward $\overline{\mathbf{Q}}_h^{k-1}$. They then play any action contained in that region.

3. (Update estimates) After collecting data $(X_h^k, A_h^k, R_h^k, X_{h+1}^k)$ with the observed reward and transitions, the algorithms update the estimates $\overline{\mathbf{Q}}_h^k$ and $\overline{\mathbf{V}}_h^k$, which are used for the next episode.

4. (Splitting rule) After observing data and updating the estimates, the algorithm determines whether to subpartition the current region. This is a key algorithmic step differentiating adaptive discretization algorithms from their uniform discretization counterparts. Our splitting rule arises by splitting a region once the confidence in its estimate is smaller than its diameter, forming a bias–variance trade-off.

For a discussion on implementation details, see Online Appendix H. We now describe the four steps of the algorithms in more detail.

3.2. Adaptive Partition

We first start by introducing the adaptive hierarchical partitions that are used by the algorithm to efficiently discretize the state and action space for learning and computation (Munos 2014, Slivkins 2014, Kleinberg et al. 2019). For each step $h\in[H]$, AdaMB and AdaQL maintain a partition of the space $\mathcal{S}\times\mathcal{A}$ into a collection of "balls," which is refined over episodes $k\in[K]$. We use the term "ball" loosely here as, in general, the regions form a nondisjoint partition of $\mathcal{S}\times\mathcal{A}$. However, in our diagrams, we use the ℓ_∞ metric on $\mathbb{R}^2$ under which the metric-balls indeed form a discrete partition of the space. In particular, we assume that the algorithms are given access to a hierarchical partition of $\mathcal{S}\times\mathcal{A}$ of the following form.

Definition 7. A hierarchical partition of $\mathcal{S}\times\mathcal{A}$ is a collection of disjoint regions $\mathcal{P}_\ell$ for levels $\ell = 0,\ldots,K$ such that

– Each region $B\in\mathcal{P}_\ell$ is of the form $\mathcal{S}(B)\times\mathcal{A}(B)$, where $\mathcal{S}(B)\subset\mathcal{S}$ and $\mathcal{A}(B)\subset\mathcal{A}$.

– $\mathcal{P}_0 = \{\mathcal{S}\times\mathcal{A}\}$.

– For every ℓ, we have that $\mathcal{S}\times\mathcal{A} = \cup_{B\in\mathcal{P}_\ell} B$.

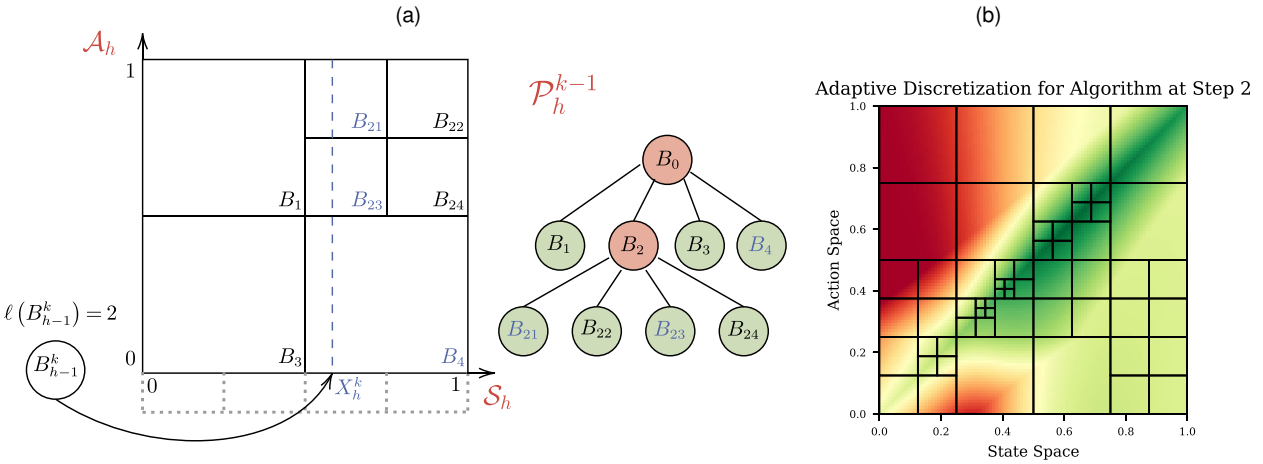
– For every ℓ and $B\in\mathcal{P}_\ell$, we have $\mathrm{diam}(B)\le 2^{-\ell}$.

– For any two regions $B_1, B_2\in\mathcal{P}_\ell$, their centers are at a distance at least $2^{-\ell}$ from each other.

– For each ℓ and $B\in\mathcal{P}_\ell$, there exists a unique $A\in\mathcal{P}_{\ell-1}$ (referred to as the parent of B) such that $B\subsetneq A$.

Whereas one can construct a hierarchical partition for any compact metric space, a canonical example to keep in mind is $\mathcal{S} = [0,1]^{d_\mathcal{S}}, \mathcal{A} = [0,1]^{d_\mathcal{A}}$ with the infinity norm $\mathcal{D}((x,a),(x',a')) = ||(x,a)-(x',a')||_\infty$. Here, one can simply take $\mathcal{P}_\ell$ to be the dyadic partition of $\mathcal{S}\times\mathcal{A}$ into regions of diameter $2^{-\ell}$. Moreover, each hierarchical partition admits a tree, whereby the root corresponds to

Figure 1. (Color online) Partitioning Scheme for $\mathcal{S} \times \mathcal{A} = [0,1]^2$



Notes. (a) Our scheme. Partition $\mathcal{P}_h^{k-1}$ is depicted with corresponding tree (showing active balls as leaves, and inactive parents as nodes with children). The algorithm plays ball B_{h-1}^k in step $h-1$, leading to new state X_h^k. Because $(B_{h-1}^k) = 2$, in AdaMB, we store transition estimates $\overline{\mathbf{T}}_{h-1}^k(\cdot|B_{h-1}^k)$ for all subsets of $\mathcal{S}$ of diameter 2^{-2} denoted as $\mathcal{S}_2$ (depicted via dotted lines). The set of relevant balls $\text{RELEVANT}_h^k(X_h^k) = \{B_4, B_{21}, B_{23}\}$ are highlighted. $\mathcal{S}(\mathcal{P}_h^{k-1})$ here would be $\{[0,\frac{1}{2}],[\frac{1}{2},\frac{3}{4}],[\frac{3}{4},1]\}$. (b) The partition $\mathcal{P}_2^K$ from one of our synthetic experiments. The scale denotes the true $Q_2^\star$ values with the diagonal corresponding to higher values. Note that the partition is more refined in areas that have higher $Q_2^\star$.

the node representing $\mathcal{S} \times \mathcal{A}$, and there is a node for each region $B \in \mathcal{P}_\ell$ with an edge to its unique parent, which is the region $A \in \mathcal{P}_{\ell-1}$ such that $B \subsetneq A$.

Our algorithm maintains an adaptive partition $\mathcal{P}_h^k$ over the hierarchical partition for each step h and k, which can be thought of as a subtree of the original hierarchical partition. Originally, we set $\mathcal{P}_h^0 = \{\mathcal{S} \times \mathcal{A}\}$ for every h. Over time, the partition is refined by adding new nodes to the subtree originating from the hierarchical partition. The leaf nodes represent the active balls, and inactive parent balls of $B \in \mathcal{P}_h^k$ corresponding to $\{B' \in \mathcal{P}_h^k | B' \supset B\}$; moreover, $\ell(B)$ is the depth of B in the tree (with the root at level zero). See Figure 1 for an example partition and tree generated by the algorithm. We let $\text{CENTER}(B) = (\mathfrak{x}(B), \mathfrak{a}(B))$ be the center of B.

AdaMB requires the induced state partition of the adaptive partition when maintaining estimates of the value function and a hierarchical partition over the state space for maintaining estimates of the transition kernel. We use the following notation:

$$\mathcal{S}(\mathcal{P}_h^k) := \bigcup_{B \in \mathcal{P}_h^k \text{ s.t. } \nexists B' \in \mathcal{P}_h^k, \mathcal{S}(B') \subset \mathcal{S}(B)} \mathcal{S}(B) \quad \text{and}$$

$$\mathcal{S}(\mathcal{P}_\ell) = \{\mathcal{S}(B) | B \in \mathcal{P}_\ell\} \tag{5}$$

to denote the partition over the state space induced by the current state-action partition $\mathcal{P}_h^k$ and the hierarchical partition over the state space at level ℓ. For example, in Figure 1(a), we have a representation of $\mathcal{S}(\mathcal{P}_h^{k-1})$ and $\mathcal{S}_2$.

3.3. Selection Rule

Upon visiting state X_h^k in step h of episode k, the algorithms find all of the relevant balls defined via $\text{RELEVANT}_h^k(X_h^k) = \{\text{active } B \in \mathcal{P}_h^{k-1} | (X_h^k, a) \in B \text{ for some } a \in \mathcal{A}\}$. The algorithm then selects the selected ball:

$$B_h^k = \underset{B \in \text{RELEVANT}_h^k(X_h^k)}{\arg\max} \overline{\mathbf{Q}}_h^{k-1}(B).$$

Once the selected ball B_h^k is chosen, the algorithm plays either action $\tilde{a}(B_h^k)$ or any distribution over actions a such that $(X_h^k, a) \in B_h^k$.

3.4. Maintaining Estimates

At a high level, for each active ball $B \in \mathcal{P}_h^k$, our algorithms maintain the following statistics:

– $n_h^k(B)$: the number of times the ball B or its ancestors in the tree have been selected up to and including episode k.

– $\overline{\mathbf{Q}}_h^k(B)$: an estimate for $Q_h^\star(x,a)$ for points $(x,a) \in B$.

In addition, AdaMB also maintains estimates of the rewards and transitions via

– $\overline{\mathbf{r}}_h^k(B)$: the empirical reward earned from playing actions in B and its ancestors.

– $\overline{\mathbf{T}}_h^k(\cdot|B)$: the empirical fractions of transitions from playing actions in B and its ancestors to sets in a $2^{-\ell(B)}$-coarse partition of $\mathcal{S}$ formed by taking the projection of the hierarchical partition to the state space (which we denote as $\mathcal{S}(\mathcal{P}_{\ell(B)})$).

3.4.1. Update Counts. After playing action A_h^k in state X_h^k at step h, the algorithm transitions to a new state X_{h+1}^k and observes a reward R_h^k. Each algorithm then updates the number of samples collected from the selected ball B_h^k via $n_h^k(B_h^k) = n_h^{k-1}(B_h^k) + 1$. In addition, AdaMB updates

the estimates for the average reward and transition as follows:

– Update average reward: $\overline{\mathbf{r}}_h^k(B_h^k) = \frac{n_h^{k-1}(B_h^k)\overline{\mathbf{r}}_h^{k-1}(B_h^k)+R_h^k}{n_h^k(B_h^k)}$.

– Update $\overline{\mathbf{T}}_h^k(\cdot|B_h^k)$ as follows: For each set A in a $2^{-\ell(B_h^k)}$-coarse partition of $\mathcal{S}$ denoted by $\mathcal{S}(\mathcal{P}_{\ell(B_h^k)})$, we set

$$\overline{\mathbf{T}}_h^k(A|B_h^k) = \frac{n_h^{k-1}(B_h^k)\overline{\mathbf{T}}_h^{k-1}(A|B_h^k) + \mathbb{1}_{\{X_{h+1}^k \in A\}}}{n_h^k(B_h^k)}.$$

This is maintaining an empirical estimate of the transition kernel for a ball B_h^k at a level of granularity proportional to its diameter. We use this to ensure that the error of the estimates for a ball B is proportional to its diameter, but this also serves as an efficient compression of the data collected so that the algorithm isn't required to store all collected samples. Other model-based algorithms forego this step, suffering from added computational and storage burden by maintaining the full empirical transition kernel (Shah and Xie 2018, Domingues et al. 2020).

With this in place, we now describe how AdaMB and AdaQL use these statistics in order to generate estimates for $\overline{\mathbf{Q}}_h^k(B)$ over the partition.

3.4.2. Compute Estimates—AdaMB. We start by defining confidence terms according to

$$\text{RUCB}_h^k(B) = \sqrt{\frac{2\log(2HK^2/\delta)}{n_h^k(B)}}$$

$$\text{TUCB}_h^k(B) = \begin{cases} L_V\left(4\sqrt{\frac{\log(2HK^2/\delta)}{n_h^k(B)}} + c\left(n_h^k(B)\right)^{-1/d_{\mathcal{S}}}\right) & \text{if } d_{\mathcal{S}} > 2 \\ L_V\left(4\sqrt{\frac{\log(2HK^2/\delta)}{n_h^k(B)}} + c\frac{\log(K)}{\sqrt{n_h^k(B)}}\right) & \text{if } d_{\mathcal{S}} \le 2 \end{cases}$$

$$\text{BIAS}(B) = 4L_r\text{diam}(B) + L_V(5L_T + 4)\text{diam}(B).$$

Here, c is an absolute constant, and the difference in definitions of $\text{TUCB}_h^k(\cdot)$ comes from the dimension dependence in Wasserstein concentration. The first term corresponds to the uncertainty in the rewards, the second to the uncertainty in the dynamics of the environment, and the third the biases in the ball resulting from aggregation. With these in place, we compute the estimate $\overline{\mathbf{Q}}_h^k(B)$ according to

$$\overline{\mathbf{Q}}_h^k(B) := \begin{cases} \overline{\mathbf{r}}_h^k(B) + \text{RUCB}_h^k(B) + \text{BIAS}(B) & \text{if } h = H \\ \overline{\mathbf{r}}_h^k(B) + \text{RUCB}_h^k(B) + \mathbb{E}_{A\sim\overline{\mathbf{T}}_h^k(\cdot|B)}[\overline{\mathbf{V}}_{h+1}^k(A)] \\ \quad + \text{TUCB}_h^k(B) + \text{BIAS}(B) & \text{if } h < H. \end{cases} \quad (6)$$

The value function estimates are computed in a two-stage process. We first define $\tilde{\mathbf{V}}_h^k(A)$ over the balls $A \in \mathcal{S}(\mathcal{P}_h^k)$ according to

$$\tilde{\mathbf{V}}_h^k(A) := \min\left\{\tilde{\mathbf{V}}_h^k(A), \max_{B\in\mathcal{P}_h^k:\mathcal{S}(B)\supseteq A} \overline{\mathbf{Q}}_h^k(B)\right\}. \quad (7)$$

We define $\overline{\mathbf{V}}_h^k(x)$ to extrapolate the estimate across all points in the state space in a Lipschitz manner. For each point $x \in \mathcal{S}$, we define

$$\overline{\mathbf{V}}_h^k(x) = \min_{A'\in\mathcal{S}(\mathcal{P}_h^k)}\left(\tilde{\mathbf{V}}_h^k(A') + L_V\mathcal{D}_{\mathcal{S}}(x, \tilde{x}(A'))\right). \quad (8)$$

The Lipschitz property is used to show concentration of expectations over $\overline{\mathbf{V}}_h^k(\cdot)$ taken with respect to transition kernel estimates. As the support of $\overline{\mathbf{T}}_h^k(\cdot|B)$ is only over sets in $\mathcal{S}(\mathcal{P}_{\ell(B)})$, we overload notation to let $\overline{\mathbf{V}}_h^k(A) = \overline{\mathbf{V}}_h^k(\tilde{x}(A))$. We equivalently overload notation so that $x \sim \overline{\mathbf{T}}_h^k(\cdot|B)$ refers to sampling over the centers associated to balls in $\mathcal{S}(\mathcal{P}_{\ell(B)})$.

This corresponds to a value iteration step, in which we replace the true rewards and transitions in the Bellman equations (Equation (3)) with their estimates. We compute full updates instead of one-step updates as in Efroni et al. (2019) for ease of presentation, but see Sinclair et al. (2020) for discussion on the one-step update procedure.

3.4.3. Compute Estimates—AdaQL. Next, we discuss the update rule $\overline{\mathbf{Q}}_h^k$ for AdaQL. Fix an episode k, step h, and ball $B \in \mathcal{P}_h^k$. Let $t = n_h^k(B)$ be the number of times B or its ancestors have been selected by the algorithm at step h in episodes up to the current episode k. The confidence terms are defined according to

$$\text{RUCB}_h^k(B) = 2\sqrt{\frac{H\log(2HK^2/\delta)}{n_h^k(B)}}$$

$$\text{TUCB}_h^k(B) = 2\sqrt{\frac{H^3\log(2HK^2/\delta)}{n_h^k(B)}}$$

$$\text{BIAS}(B) = 2L_V\text{diam}(B).$$

The first term corresponds to the uncertainty in the rewards, the second to the uncertainty in the dynamics of the environment, and the third the biases in the ball resulting from aggregation.

Upon visiting state X_h^k taking action A_h^k with selected ball $B_h^k = B$, we update the estimate for the selected ball $\overline{\mathbf{Q}}_h^k(B)$ (leaving all other estimates unchanged) via

$$\overline{\mathbf{Q}}_h^k(B) = (1-\alpha_t)\overline{\mathbf{Q}}_h^{k-1}(B) + \alpha_t(R_h^k + \text{RUCB}_h^k(B) + \overline{\mathbf{V}}_{h+1}^{k-1}(X_{h+1}^k) + \text{TUCB}_h^k(B) + \text{BIAS}(B)), \quad (9)$$

where R_h^k is the observed reward, X_{h+1}^k is the state to which the agent transitions, $\alpha_t = \frac{H+1}{H+t}$ is the learning rate, and

$$\overline{\mathbf{V}}_{h+1}^{k-1}(x) = \min\left(H, \max_{B \in \text{RELEVANT}_{h+1}^{k-1}(x)} \overline{\mathbf{Q}}_{h+1}^{k-1}(B)\right) \tag{10}$$

is our estimate of the expected future reward for being in a given state. Note that, with this recursive update, we can equivalently write $\overline{\mathbf{Q}}_h^k(B)$ by unraveling the recursion. First, denote

$$\alpha_t^i = \alpha_i \prod_{j=i+1}^{t} (1-\alpha_j). \tag{11}$$

Lemma 4 (Recursive Relationship for AdaQL). *For any $h,k \in [H] \times [K]$ and ball $B \in \mathcal{P}_h^k$, let $t = n_h^k(B)$ be the number of times that B or its ancestors were encountered during the algorithm before episode k. Further, suppose that B and its ancestors were encountered at step h of episodes $k_1 < k_2 < \cdots < k_t \le k$. By the update rule of Q, we have that*

$$\overline{\mathbf{Q}}_h^k(B) = \mathbb{1}_{[t=0]}H + \sum_{i=1}^{t} \alpha_t^i \Big(R_h^{k_i} + \text{RUCB}_h^{k_i}(B_h^{k_i}) + \overline{\mathbf{V}}_{h+1}^{k_i}(X_{h+1}^{k_i}) + \text{TUCB}_h^{k_i}(B_h^{k_i}) + \text{BIAS}(B_h^{k_i})\Big).$$

Proof of Lemma 4. We show the claim by induction on $t = n_h^k(B)$.

First suppose that $t = 0$, that is, that the ball B has not been encountered before by the algorithm. Then, initially $\overline{\mathbf{Q}}_h^0(B) = H = \mathbb{1}_{[t=0]}H$.

Now, for the step case, we notice that $\overline{\mathbf{Q}}_h^k(B)$ was last updated at episode k_t. k_t is either the most recent episode when ball B was encountered or the most recent episode when its parent was encountered if B was activated and not yet played. In either case, by the update rule (Equation (9)), we have

$$\begin{aligned}
\overline{\mathbf{Q}}_h^k(B) &= (1-\alpha_t)\overline{\mathbf{Q}}_h^{k_t-1}(B) + \alpha_t\Big(R_h^{k_t} + \text{RUCB}_h^{k_t}(B_h^{k_t}) \\
&\quad + \overline{\mathbf{V}}_{h+1}^{k_t}(X_{h+1}^{k_t}) + \text{TUCB}_h^{k_t}(B_h^{k_t}) + \text{BIAS}(B_h^{k_t})\Big) \\
&= (1-\alpha_t)\alpha_{t-1}^0 H + (1-\alpha_t)\sum_{i=1}^{t-1}\alpha_t^i\Big(R_h^{k_i} + \text{RUCB}_h^{k_i}(B_h^{k_i}) \\
&\quad + \overline{\mathbf{V}}_{h+1}^{k_i}(X_{h+1}^{k_i}) + \text{TUCB}_h^{k_i}(B_h^{k_i}) + \text{BIAS}(B_h^{k_i})\Big) \\
&\quad + \alpha_t\Big(R_h^{k_t} + \overline{\mathbf{V}}_{h+1}^{k_t}(X_{h+1}^{k_t}) + \text{TUCB}_h^{k_t}(B_h^{k_t}) + \text{BIAS}(B_h^{k_t})\Big) \\
&\quad \text{by the induction hypothesis} \\
&= \mathbb{1}_{[t=0]}H + \sum_{i=1}^{t}\alpha_t^i\Big(R_h^{k_i} + \text{RUCB}_h^{k_i}(B_h^{k_i}) + \overline{\mathbf{V}}_{h+1}^{k_i}(x_{h+1}^{k_i}) \\
&\quad + \text{TUCB}_h^{k_i}(B_h^{k_i}) + \text{BIAS}(B_h^{k_i})\Big)
\end{aligned}$$

by definition of α_t^i. □

3.5. Splitting Rule

To refine the partition over episodes, we split a ball when the confidence in its estimate is smaller than its bias. Formally, because of the Lipschitz property on the Q function from the assumption, we know that the bias in the estimates is proportional to the diameter of the ball. In episode k, step h, we split the selected ball B_h^k if

$$\text{CONF}_h^k(B_h^k) \le \text{diam}(B_h^k). \tag{12}$$

Here, $\text{CONF}_h^k(B_h^k) = \tilde{C}/n_h^k(B)^\alpha$ is the dominating term of $\text{RUCB}_h^k(B)$ and $\text{TUCB}_h^k(B)$ for some polylogarithmic constant $\tilde{C}$. In particular, we take

$$\text{CONF}_h^k(B) := \begin{cases} 4\sqrt{\dfrac{H^3\log(2HK^2/\delta)}{n_h^k(B)}}\ \text{ADAQL} \quad : \alpha = \dfrac{1}{2} \\ 4L_V c\sqrt{\dfrac{\log(2HK^2/\delta)}{n_h^k(B)}}\ \text{ADAMB} \quad d_S \le 2 : \alpha = \dfrac{1}{2} \\ \dfrac{4L_V c\sqrt{\log(2HK^2/\delta)}}{n_h^k(B)^{1/d_S}}\ \text{ADAMB} \quad d_S > 2 : \alpha = \dfrac{1}{d_S}. \end{cases}$$

This splitting rule differs from previous analysis of adaptive discretization for RL (Sinclair et al. 2019, 2020; Cao and Krishnamurthy 2020) and is key for achieving the proper instance-dependent guarantees. By taking the splitting threshold to depend explicitly on the dominating term in the confidence bounds, we are able to upper bound the bias of the ball (proportional to the diameter) by these confidence terms as we later see in the regret analysis.

In episode k step h, if we need to split $B^{par} = B_h^k$, then we add the child nodes in the hierarchical partition immediately under B_h^k to form $\mathcal{P}_h^k$. Each child ball then inherits all estimates and counts from its parent ball in the adaptive partition with the exception of the estimate of the transition distribution in ADAMB .

Recall that $\overline{\mathbf{T}}_h^k(\cdot|B^{par})$ defines a distribution over $\mathcal{S}(\mathcal{P}_{\ell(B^{par})})$, whereas $\overline{\mathbf{T}}_h^k(\cdot|B)$ for children B of B^{par} should define a distribution over $\mathcal{S}(\mathcal{P}_{\ell(B)})$. As a result, for ADAMB, we need to additionally update the transitional kernel estimates to map to a distribution over $\mathcal{S}(\mathcal{P}_{\ell(B)})$ by splitting the mass equally over subregions according to

$$\overline{\mathbf{T}}_h^k(A|B) = 2^{-d_S}\overline{\mathbf{T}}_h^k(A^{par}|B^{par}),$$

where B^{par} is the parent of B and A^{par} is the parent of A, that is, the unique element in $\mathcal{S}(\mathcal{P}_{\ell(B^{par})})$ such that $A \subset A^{par}$. Each element is weighted by 2^{-d_S} to split the mass evenly as each element of $\mathcal{S}(\mathcal{P}_{\ell(B^{par})})$ has 2^{d_S} children in $\mathcal{S}(\mathcal{P}_{\ell(B)})$.

4. Main Results

In this section, we outline the main results for our paper. We provide guarantees for AdaMB and AdaQL on all three aspects of the trifecta for RL in OR, including regret bounds, storage requirements, and computational complexity for the algorithms. We refer the readers to Table 1 for a summary of the main results.

4.1. Regret Minimization Guarantees

We start by providing instance-dependent bounds on the regret for AdaMB and AdaQL. These bounds explicitly depend on the zooming dimension as outlined in Section 2.4 measuring the complexity of a problem instance instead of the ambient dimension.

Theorem 1. *Let z_h be the step h zooming dimension and d_S be the covering dimension of the state space. Then, the regret of* AdaMB *and* AdaQL *for any sequence of starting states $\{X_1^k\}_{k=1}^K$ is upper bounded with probability at least $1-3\delta$ by*

$$\text{Regret}(K) \lesssim \begin{cases} H^{3/2}\sqrt{K} + L\sum_{h=1}^{H} K^{\frac{z_h+d_S-1}{d+d_S}} & \text{AdaMB} : d_S > 2 \\ H^{3/2}\sqrt{K} + L\sum_{h=1}^{H} K^{\frac{z_h+1}{z_h+2}} & \text{AdaMB} : d_S \le 2 \\ LH^{3/2}\sum_{h=1}^{H} K^{\frac{z_h+1}{z_h+2}} & \text{AdaQL} \end{cases}$$

where $L = 1 + L_r + L_V + L_V L_T$ and $\lesssim$ omits poly-logarithmic factors of $\frac{1}{\delta}, H, K,$ d and any universal constants.

4.1.1. Comparison Between Model-Free and Model-Based Methods.

As we see from Theorem 1, the bounds for AdaMB have better dependence on the number of steps H. This is expected as current analysis for model-free and model-based algorithms under tabular settings shows that model-based algorithms achieve better dependence on the horizon. However, under the Lipschitz assumptions, the constant L scales with H, so the true dependence is masked (see Lemma 1). When we compare the dependence on the number of episodes K, we see that the dependence is worse for AdaMB, primarily because of the additional factor of d_S, the covering dimension of the state space. This discrepancy arises as model-based algorithms maintain an estimate of the transition kernel, whose worst case statistical complexity for Wasserstein concentration depends on d_S when $d_S > 2$. In Online Appendix G we give improved bounds for AdaMB under additional assumptions on the transition distribution that match the performance of AdaMB.

4.1.2. Metric-Specific Guarantees.

Our guarantees scale with respect to the zooming instead of the ambient dimension of the metric space. As the zooming dimension can be much smaller than the ambient dimension (see Lemma 2), our adaptive discretization algorithms are able to achieve exponentially better regret guarantees than other nonparametric algorithms. Moreover, in the final regret bound in Online Appendix E, we provide more fine-tuned metric-dependent guarantees.

4.1.3. Comparison with Other Nonparametric Methods.

Current state-of-the-art model-based algorithms achieve regret scaling such as $H^3K^{2d/(2d+1)}$ (Domingues et al. 2020). We achieve better scaling with respect to both H and K, and our algorithm has lower time and space complexity. However, we require additional oracle assumptions on the metric space to be able to construct packings and coverings efficiently, whereas Kernel-UCBVI uses the data and the metric itself. Better dependence on H and K is achieved by using recent work on concentration for the Wasserstein metric and by showing zooming dimension guarantees. These guarantees allow us to construct tighter confidence intervals that are independent of H, obviating the need to construct a covering of H-uniformly bounded Lipschitz functions as in prior work (see Online Appendix B).

In addition, Kernel-UCBVI uses a fixed bandwidth parameter in the kernel interpolation. We instead keep an adaptive partition of the space, helping our algorithm maintain a smaller and more efficient discretization and adapting to the zooming dimension of the space instead of the ambient dimension.

4.1.4. Policy-Identification Guarantees.

Using similar arguments from Jin et al. (2018), it is straightforward to show sample complexity guarantees on learning a policy of a desired quality in the probably approximately correct guarantee framework for learning RL policies (Watkins 1989).

4.2. Lower Bounds

Existing work for the contextual bandit literature shows that the worst case regret scales exponentially with respect to the zooming dimension (Slivkins 2014). This construction can be modified directly to obtain a lower bound for the RL setting as follows.

Theorem 2. *Let $(\mathcal{S}, \mathcal{D}_S)$ and $(A, \mathcal{D}_A)$ be arbitrary metric spaces and $\mathcal{D}$ be the product metric. Fix an arbitrary time horizon K and number of steps H. There exists a distribution $\mathcal{I}$ over problem instances on $(\mathcal{S}\times\mathcal{A}, \mathcal{D})$ such that, for any algorithm,*

$$\mathbb{E}_{\mathcal{I}}[\text{Regret}(K)] \ge \Omega\left(\sum_{h=1}^{H} K^{\frac{z_h+1}{z_h+2}}/\log(K)\right)$$

This builds on the lower bounding technique from (Jaksch et al. 2010, Slivkins 2014, Kleinberg et al. 2019) by using a needle-in-the-haystack example. The haystack

consists of several actions with an expected payoff of $\frac{1}{2}$ and the needle an action whose expected payoff is slightly higher. In fact, the construction from Slivkins (2014) can be used directly by developing a problem instance of length H that is a sequence of H contextual bandit problems.

4.2.1. Comparison with Lower Bounds. Comparing our regret bounds to the lower bound, we see that AdaQL and AdaMB (for $d_{\mathcal{S}} \leq 2$) match the lower bound with respect to the instance dependent zooming dimension z_h. However, AdaMB when $d_{\mathcal{S}} > 2$ has the additionally factors of $d_{\mathcal{S}}$ as a result of maintaining the explicit estimate of the transition kernel.

4.2.2. Exponential Scaling on Episodes. The regret is always upper bounded by HK. In order for Theorem 1 to give a nontrivial regret guarantee, then the number of episodes needs to be on the order of $H^{z_{max}}$. This exponential dependence is a fundamental factor in all estimation problems with nonparametric statistics. Our work focuses on improving on the exponent by showing exponential sample complexity gains from scaling with the zooming dimension instead of the ambient dimension. Moreover, the algorithms require no prior knowledge of the zooming dimension in order to achieve these regret bounds. Theoretically speaking, the value of these results is limited when the dimension of the space starts to grow. However, experimental results show the value of nonparametric algorithms in practice (Ipek et al. 2008; Araújo et al. 2020a, b).

4.3. Space and Time Complexity Guarantees

Next, we consider the storage and time complexity of both of our algorithms. In particular, we are able to show the following.

Theorem 3. *The storage and time complexity for* AdaMB *and* AdaQL *can be upper bounded via*

$$\textsc{Space}(K) \lesssim \begin{cases} HK & \text{AdaMB} \ : d_{\mathcal{S}} > 2 \\ HK^{\frac{d+d_{\mathcal{S}}}{d+d_{\mathcal{S}}+2}} & \text{AdaMB} \ : d_{\mathcal{S}} \leq 2 \\ HK^{\frac{d}{d+2}} & \text{AdaQL} \end{cases}$$

$$\textsc{Time}(K) \lesssim \begin{cases} HK^{\frac{d+2d_{\mathcal{S}}}{d+d_{\mathcal{S}}}} & \text{AdaMB} \ : d_{\mathcal{S}} > 2 \\ HK^{\frac{d+d_{\mathcal{S}}+2}{d+d_{\mathcal{S}}}} & \text{AdaMB} \ : d_{\mathcal{S}} \leq 2 \\ HK \log_d(K) & \text{AdaQL.} \end{cases}$$

4.3.1. Comparison Between Model-Free and Model-Based Methods. As we can see, both the storage and time complexity for AdaQL are uniformly better than that of AdaMB. This should not come as a surprise as, even in the tabular setting, model-free algorithms have better storage and computational requirements than model-based ones as they forego maintaining and using explicit estimates of the transition kernel.

4.3.2. Comparison with Other Nonparametric Methods. As seen in Table 1, our bounds are uniformly better for both storage and time complexity than other nonparametric algorithms. These gains are primarily a result of using the discretization to maintain an efficient compression of the data and statistical accuracy and utilizing quantizing techniques to speed up the algorithms.

4.3.3. Monotone Increasing Runtime and Storage Complexity. The runtime and storage complexity guarantees presented are monotonically increasing with respect to the number of episodes K. However, to get sublinear minimax regret in a continuous setting for nonparametric Lipschitz models, the model complexity must grow over episodes. In practice, one runs our adaptive discretization algorithms until running out of space, and our experiments show that the algorithms use resources (storage and computation) much better than a uniform discretization.

4.3.4. Comparison with Lower Bounds. To the best of our knowledge, there are no existing results showing storage or computational lower bounds for an RL algorithm in continuous spaces.

5. Proof Sketch

We start with giving the proof sketch of Theorem 1 before going into the proof of Theorem 3 in Section 5.4. The high-level proof of Theorem 1 is divided into three sections. First, we show concentration, clean events, and optimism under which our estimates constitute upper bounds on their relevant quantities. Afterward, we show a regret decomposition with clipping, which upper bounds the difference between the estimated value and the value accumulated by the algorithm as a function of the confidence terms. This uses the clipping operator first introduced in Simchowitz and Jamieson (2019) for obtaining instance-dependent regret guarantees in tabular settings. Finally, we use an argument to bound the sum of confidence terms, which is used for the final regret bound. We include a brief discussion here on each of the three parts, but the full details are included in the online appendix.

5.1. Concentration, Clean Events, and Optimism (Online Appendices B and C)

AdaMB explicitly maintains estimates $\overline{\mathbf{r}}_h^k(B)$ and $\overline{\mathbf{T}}_h^k(\cdot|B)$ for the unknown rewards and transitions of the underlying MDP. Similarly, AdaQL implicitly maintains estimates for the rewards and transitions; the rewards are taken via $\sum_t \alpha_t^i R_h^{k_i}$ and the transitions are taken using

old estimates of $\overline{\mathbf{V}}_h^{k_i}$ instead of $\overline{\mathbf{V}}_h^k$. In order to ensure that the one-step value iteration update in Equations (6) and (9) concentrates, we need to verify that these estimates provide good approximations to their true quantities. In particular, we show that

AdaMB:

$$\begin{cases} |\overline{r}_h^k(B) - r_h(x,a)| & \lesssim \text{RUCB}_h^k(B) + L_r \text{diam}(B) \\ d_W(\overline{T}_h^k(\cdot|B), T_h(\cdot|x,a)) & \lesssim \text{TUCB}_h^k(B) + L_V L_T \text{diam}(B), \end{cases}$$

AdaQL:

$$\begin{cases} \left| \sum_{i=1}^t \alpha_t^i (R_h^{k_i} - r_h(x,a)) \right| & \lesssim \text{RUCB}_h^k(B) \\ \left| \sum_{i=1}^t \alpha_t^i (V_{h+1}^\star(X_{h+1}^{k_i}) - \mathbb{E}_{Y \sim T_h(\cdot|X_h^{k_i}, A_h^{k_i})}[V_{h+1}^\star(Y)] \right| & \lesssim \text{TUCB}_h^k(B). \end{cases}$$

However, recall that our estimates for $\overline{\mathbf{Q}}_h^k(B)$ are constructed via

$$\overline{\mathbf{Q}}_h^k(B) = \tilde{r}_h^k(B) + \text{RUCB}_h^k(B) + \tilde{T}_h^k(\overline{\mathbf{V}}_{h+1}^{\tilde{k}}(B)) + \text{TUCB}_h^k(B) + \text{BIAS}(B),$$

where $\tilde{r}$ and $\tilde{T}$ vary for the two different algorithms. As such, the concentration results leads to upper and lower bounds on $\overline{\mathbf{Q}}_h^k(B)$ via the bonus terms of the following form.

Informal Theorem 2. *For any $(h,k) \in [H] \times [K]$, $B \in \mathcal{P}_h^k$ and $(x,a) \in B$, we have that*

$$0 \le \overline{\mathbf{Q}}_h^k(B) - Q_h^\star(x,a) \lesssim \text{CONF}_h^k(B) + \text{BIAS}(B) + f_{h+1}^k,$$

where f_{h+1}^k is an algorithm-dependent term depending on the estimates at step $h+1$.

5.2. Upper Bound via Clipping (Online Appendix D)

We use an argument based on the clipping operator first introduced in Simchowitz and Jamieson (2019) for obtaining instance-dependent regret guarantees for tabular reinforcement learning. We define $\text{CLIP}[\mu|\nu] = \mu \mathbb{1}_{[\mu \ge \nu]}$. The value of this function is zero until $\mu \ge \nu$, and afterward, it takes on the value of μ. We use this operator to bound the regret at step h in episode k in two terms. The first part corresponds to the clipped bonus terms and the bias of the estimate using concentration and Lipschitz properties of the estimation procedure. The second term is f_{h+1}^k, the algorithm-dependent quantity measuring the downstream effects of errors at the next time step.

In particular, consider the ball B_h^k selected by the algorithm in step h of episode k. Letting (X_h^k, A_h^k) be the state–action pair played at that time step, we note that

$$\begin{aligned} \text{GAP}_h(B_h^k) &= \min_{(x,a) \in B_h^k} \text{GAP}_h(x,a) \le \text{GAP}_h(X_h^k, A_h^k) \\ &= V_h^\star(X_h^k) - Q_h^\star(X_h^k, A_h^k) \\ &\le \overline{\mathbf{Q}}_h^{k-1}(B_h^k) - Q_h^\star(X_h^k, A_h^k) \le \text{CONF}_h^{k-1}(B_h^k) \\ &\quad + \text{BIAS}(B_h^k) + f_{h+1}^{k-1}. \end{aligned}$$

Via some simple algebraic manipulations, we are able to show that this gives

$$\begin{aligned} &\overline{\mathbf{Q}}_h^{k-1}(B_h^k) - Q_h^\star(X_h^k, A_h^k) \\ &\le \text{CLIP}\left[\text{CONF}_h^{k-1}(B_h^k) + \text{BIAS}(B_h^k) \middle| \frac{\text{GAP}_h(B_h^k)}{H+1} \right] + \left(1 + \frac{1}{H}\right) f_{h+1}^{k-1}. \end{aligned} \tag{13}$$

This expression can be thought of as bounding the one-step regret of the algorithm by a term scaling with respect to the confidence in the estimates and a second term scaling with the downstream misestimation errors.

5.3. Regret Bound via the Splitting Rule (Online Appendix E)

Finally, we use the previous equation to develop a final regret bound for the algorithm. In particular, consider the quantity $\Delta_h^k = \overline{\mathbf{V}}_h^{k-1}(X_h^k) - V_h^{\pi^k}(X_h^k)$. Via optimism and the greedy selection rule, we have that

$$\begin{aligned} \text{REGRET}(K) &= \sum_{k=1}^K V_1^\star(X_1^k) - V_1^{\pi^k}(X_1^k) \\ &\le \sum_{k=1}^K \overline{\mathbf{V}}_1^{k-1}(X_1^k) - V_1^{\pi^k}(X_1^k) = \sum_{k=1}^K \Delta_1^k. \end{aligned}$$

We use the bound on $\overline{\mathbf{Q}}_h^{k-1}(B_h^k) - Q_h^\star(X_h^k, A_h^k)$ and the definition of f_{h+1}^{k-1} to show that, for each algorithm,

$$\begin{aligned} \sum_{k=1}^K \Delta_h^k &\lesssim \sum_{k=1}^K \text{CLIP}\left[\text{CONF}_h^{k-1}(B_h^k) + \text{BIAS}(B_h^k) \middle| \frac{\text{GAP}_h(B_h^k)}{H+1} \right] \\ &\quad + \left(1 + \frac{1}{H}\right) \sum_{k=1}^K \xi_{h+1}^k + \sum_{k=1}^K \Delta_{h+1}^k, \end{aligned}$$

where ξ_{h+1}^k is an algorithm-dependent martingale difference sequence. Using this and recursing backward, we have that

$$\begin{aligned} &\text{REGRET}(K) \\ &\lesssim \sum_{h=1}^H \sum_{k=1}^K \text{CLIP}\left[\text{CONF}_h^{k-1}(B_h^K) + \text{BIAS}(B_h^k) \middle| \frac{\text{GAP}_h(B_h^k)}{H+1} \right] \\ &+ \text{lower order terms.} \end{aligned}$$

By properties of the splitting rule for B_h^k, $\text{diam}(B_h^k) \le \text{CONF}_h^{k-1}(B_h^k) \le 4\text{diam}(B_h^k)$. Moreover, the $\text{BIAS}(B_h^k)$ is of

the form $C_L \text{diam}(B_h^k)$ for some Lipschitz-dependent constant C_L. Thus, we get that the term inside of the clipping operator can be upper bounded by (C_L+1) $\text{CONF}_h^{k-1}(B_h^k) \le 4(C_L+1)\text{diam}(B_h^k)$.

By definition of the clip operator, we only need to consider when $4(C_L+1)\text{diam}(B_h^k) \ge \frac{\text{GAP}_h(B_h^k)}{H+1}$. However, this implies that $\text{GAP}_h(B_h^k) \le 4(H+1)(C_L+1)\text{diam}(B_h^k)$. Letting (x_c, a_c) denote the center of B_h^k, we can show using the Lipschitz assumption that

$$\begin{aligned}\text{GAP}_h(x_c,a_c) &\le \text{GAP}_h(B_h^k) + 2L_V \text{diam}(B_h^k)\\ &\le (4(H+1)(C_L+1)+2L_V)\text{diam}(B_h^k)\\ &\le C_L(H+1)\text{diam}(B_h^k),\end{aligned}$$

and so (x_c, a_c) lies in the set Z_h^r for $r = \text{diam}(B_h^k)$ by redefining the constant C_L. The final regret bound follows by replacing the clipping operator with the indicator that the center of the ball lies in the near-optimal set and considering the scaling of the confidence terms.

This regret derivation for our adaptive discretization algorithm serves in contrast to typical guarantees seen for "zooming algorithms" in contextual bandits (Slivkins 2014). In particular, in contextual bandits, we can show that a ball B is sampled proportional to its diameter for action elimination (essentially eliminating the second term in Equation (13)). However, in RL, we have to account for downstream uncertainty, requiring a more nuanced analysis using the clipping argument for the final regret bound.

5.4. Bound on Time and Storage Complexity (Online Appendix F)

We use properties of the splitting rule in order to generate bounds on the size of the partition that are needed for the time and space complexity guarantees. In particular, we formulate these quantities as a linear program (LP) in which the objective function is to maximize a sum of terms associated with a valid adaptive partition (represented as a tree) constructed by the algorithm. The constraints follow from conditions on the number of samples required before a ball is split into subsequent child balls. To derive an upper bound on the value of the LP, we find a tight dual feasible solution. This argument can be more broadly useful and modified for problems with additional structures by including additional constraints into the LP. Start by defining a quantity $n_+(\ell(B))$ as an upper bound of the number of times a ball at level $\ell(B)$ is sampled before it is split (which we call the splitting thresholds). Based on the splitting rule, we have that $n_+(\ell(B)) \approx \text{diam}(B)^{-1/\alpha}$, where α is determined by the dominating term of the bonus terms (see Section 3.4). However, noting that $\text{diam}(B) \approx 2^{-\ell(B)}$ we get that $n_+(\ell(B)) \approx 2^{\gamma\ell(B)}$.

Lemma 5. *Consider any partition $\mathcal{P}_h^k$ for any $k \in [K], h \in [H]$ induced with splitting thresholds $n_+(\ell(B)) = \phi 2^{\gamma\ell(B)}$ and consider any "penalty" vector $\{a_\ell\}_{\ell\in\mathbb{N}_0}$ that satisfies $a_{\ell+1} \ge a_\ell \ge 0$ and $2a_{\ell+1}/a_\ell \le n_+(\ell)/n_+(\ell-1)$ for all $\ell \in \mathbb{N}_0$. Define $\ell^\star = \inf\{\ell | 2^{d(\ell-1)} n_+(\ell-1) \ge k\}$. Then,*

$$\sum_{\ell=0}^{\infty} \sum_{B\in\mathcal{P}_h^k : \ell(B)=\ell} a_\ell \le 2^{d\ell^\star} a_{\ell^\star}$$

One immediate corollary of Lemma 5 is a bound on the size of the partition $\mathcal{P}_h^k$ for any h and k by taking $a_\ell = 1$ for every ℓ. In particular, one can show the following.

Corollary 1. *For any h and k, we have that $|\mathcal{P}_h^k| \le 4^d \left(\frac{k}{\phi}\right)^{\frac{d}{d+\gamma}}$ and $\ell^\star \le \frac{1}{d+\gamma}\log_2(k/\phi) + 2$.*

These results are used to bound the storage and time complexity of the algorithms by noting the dominating complexity terms in the algorithm description, writing the total accumulation by formulating them as an LP and applying the result from Lemma 5.

6. Conclusion

In this paper, we present a unified analysis of model-based and model-free reinforcement learning algorithms using adaptive discretization. In worst case instances, we show regret bounds for our algorithms with exponential improvements over other online nonparametric RL algorithms (i.e., the underlying model is Lipschitz continuous with a known metric of the space). This is partially a result of our instance-dependent regret bounds, exhibiting how the discretization and regret scales with respect to the zooming dimension of the problem instead of the ambient dimension. We provided simulations comparing model-based and model-free methods using adaptive and fixed discretizations of the space on several canonical control problems. Our experiments show that adaptive partitioning empirically performs better than fixed discretizations in terms of both faster convergence and lower memory.

One future direction for this work is analyzing the discrepancy between model-based and model-free methods in continuous settings as model-based algorithms so far have suboptimal dependence on the dimension of the state space. Whereas in Online Appendix G, we give specific instances in which the regret of AdaMB matches AdaQL, in general, the regret has additional dependence on d_S because of uniform Wasserstein concentration on the state space. Moreover, we are interested in deriving the "optimal" space and time complexity for an algorithm in continuous settings. We also believe that new hardware techniques can help improve the complexities of implementing these adaptive discretization algorithms in practice.

Acknowledgments

Part of this work was done when Sean Sinclair and Christina Yu were visiting the Simons Institute for the Theory of Computing for the semester on the Theory of Reinforcement Learning.

References

Agarwal A, Jiang N, Kakade SM (2019) Reinforcement learning: Theory and algorithms. Technical report, University of Washington, Seattle.

Alizadeh M, Yang S, Sharif M, Katti S, McKeown N, Prabhakar B, Shenker S (2013) pFabric: Minimal near-optimal datacenter transport. *Comput. Comm. Rev.* 43(4):435–446.

Alizadeh M, Greenberg A, Maltz D, Padhye J, Patel P, Prabhakar B, Sengupta S, Sridharan M (2010) DCTCP: Efficient packet transport for the commoditized data center (ACM SIGCOMM). https://www.microsoft.com/en-us/research/publication/dctcp-efficient-packet-transport-for-the-commoditized-data-center/.

Araújo JP, Figueiredo MA, Botto MA (2020a) Control with adaptive Q-learning. Preprint, submitted November 3, https://arxiv.org/abs/2011.02141.

Araújo JP, Figueiredo M, Botto MA (2020b) Single-partition adaptive Q-learning. Preprint, submitted July 14, https://arxiv.org/abs/2007.06741.

Azar MG, Munos R, Kappen HJ (2013) Minimax PAC bounds on the sample complexity of reinforcement learning with a generative model. *Machine Learn.* 91(3):325–349.

Azar MG, Osband I, Munos R (2017) Minimax regret bounds for reinforcement learning. *Proc. 34th Internat. Conf. Machine Learn.*, vol. 70, 263–272.

Bhandari J, Russo D (2021) On the linear convergence of policy gradient methods for finite MDPs. Banerjee A, Fukumizu K, eds. *Proc. 24th Internat. Conf. Artificial Intelligence Statist.*, vol. 130 (PMLR), 2386–2394.

Bubeck S, Cesa-Bianchi N (2012) *Regret Analysis of Stochastic and Nonstochastic Multi-Armed Bandit Problems*, Foundations and Trends in Machine Learning, vol. 5 (now Publishers Inc., Norwell, MA).

Cao T, Krishnamurthy A (2020) Provably adaptive reinforcement learning in metric spaces. *Adv. Neural Inform. Processing Systems* 33:9736–9744.

Chen S, Zhang B (2021) Estimating and improving dynamic treatment regimes with a time-varying instrumental variable. Preprint, submitted April 15, https://arxiv.org/abs/2104.07822.

Chinchali S, Hu P, Chu T, Sharma M, Bansal M, Misra R, Pavone M, Katti S (2018) Cellular network traffic scheduling with deep reinforcement learning. *32nd AAAI Conf. Artificial Intelligence*.

Domingues OD, Ménard P, Pirotta M, Kaufmann E, Valko M (2020) Regret bounds for kernel-based reinforcement learning. Preprint, submitted April 12, https://arxiv.org/abs/2004.05599.

Du SS, Kakade SM, Wang R, Yang LF (2020) Is a good representation sufficient for sample efficient reinforcement learning? *Internat. Conf. Learn. Representations*.

Du SS, Luo Y, Wang R, Zhang H (2019) Provably efficient Q-Learning with function approximation via distribution shift error checking oracle. *Adv. Neural Inform. Processing Systems* 33:8058–8068.

Efroni Y, Merlis N, Ghavamzadeh M, Mannor S (2019) Tight regret bounds for model-based reinforcement learning with greedy policies. *Adv. Neural Inform. Processing Systems* 33:12203–12213.

Henaff M (2019) Explicit explore-exploit algorithms in continuous state spaces. *Adv. Neural Inform. Processing Systems* 32:9372–9382.

Hubbs CD, Perez HD, Sarwar O, Sahinidis NV, Grossmann IE, Wassick JM (2020) OR-gym: A reinforcement learning library for operations research problems. Preprint, submitted August 14, https://arxiv.org/abs/2008.06319.

Ipek E, Mutlu O, Martínez JF, Caruana R (2008) Self-optimizing memory controllers: A reinforcement learning approach. *SIGARCH Comput. Architecture. News* 36(3):39–50.

Jaksch T, Ortner R, Auer P (2010) Near-optimal regret bounds for reinforcement learning. *J. Machine Learn. Res.* 11:1563–1600.

Jiang N, Krishnamurthy A, Agarwal A, Langford J, Schapire RE (2017) Contextual decision processes with low Bellman rank are PAC-learnable. *Internat. Conf. Machine Learn.* (PMLR), 1704–1713.

Jin C, Allen-Zhu Z, Bubeck S, Jordan MI (2018) Is Q-learning provably efficient? Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R, ed. *Advances in Neural Information Processing Systems*, vol. 31 (Curran Associates, Inc., Red Hook, NY), 4863–4873.

Jin C, Yang Z, Wang Z, Jordan MI (2020) Provably efficient reinforcement learning with linear function approximation. *Conf. Learn. Theory* (PMLR), 2137–2143.

Kakade S, Kearns MJ, Langford J (2003) Exploration in metric state spaces. *Proc. 20th Internat. Conf. Machine Learn.*, 306–312.

Keller PW, Mannor S, Precup D (2006) Automatic basis function construction for approximate dynamic programming and reinforcement learning. *Proc. 23rd Internat. Conf. Machine Learn.*, 449–456.

Kleinberg R, Slivkins A, Upfal E (2019) Bandits and experts in metric spaces. *J. ACM* 66(4):1–77.

Lakshmanan K, Ortner R, Ryabko D (2015) Improved regret bounds for undiscounted continuous reinforcement learning. *Internat. Conf. Machine Learn.*, 524–532.

Le Lan C, Bellemare MG, Castro PS (2021) Metrics and continuity in reinforcement learning. *Proc. Conf. AAAI Artificial Intelligence*, vol. 35, 8261–8269.

Lolos K, Konstantinou I, Kantere V, Koziris N (2017) Adaptive state space partitioning of Markov decision processes for elastic resource management. *IEEE 33rd Internat. Conf. Data Engrg.*, 191–194.

Lykouris T, Vassilvitskii S (2018) Competitive caching with machine learned advice. Preprint, submitted February 15, https://arxiv.org/abs/1802.05399.

Menache I, Mannor S, Shimkin N (2005) Basis function adaptation in temporal difference reinforcement learning. *Ann. Oper. Res.* 134(1):215–238.

Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing Atari with deep reinforcement learning. Preprint, submitted December 19, https://arxiv.org/abs/1312.5602.

Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. *Internat. Conf. Machine Learn.*, 1928–1937.

Mozur P (2017) Google's Alphago defeats Chinese Go master in win for A.I. *The New York Times Online* (May 23), https://www.nytimes.com/2017/05/23/business/google-deepmind-alphago-go-champion-defeat.html.

Munos R (2014) From bandits to Monte-Carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, vol. 7 (now Publishers Inc., Norwell, MA), 1–129.

Nishtala R, Fugal H, Grimm S, Kwiatkowski M, Lee H, Li HC, McElroy R, et al. (2013) Scaling memcache at Facebook. *Proc. 10th Sympos. Networked Systems Design Implementation*, vol. 13, 385–398.

Osband I, Van Roy B (2014) Model-based reinforcement learning and the eluder dimension. *Adv. Neural Inform. Processing Systems* 27:1466–1474.

Powell WB (2022) *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions* (Wiley-Blackwell, Hoboken, NJ).

Puterman ML (1994) *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. (John Wiley & Sons, Inc., New York).

Pyeatt LD, Howe AE (2001) Decision tree function approximation in reinforcement learning. *Proc. Third Internat. Sympos. Adaptive Systems: Evolutionary Comput. Probabilistic Graphical Models*, vol. 2, 70–77.

Royden HL, Fitzpatrick P (1988) *Real Analysis*, vol. 32 (Macmillan, New York).

Russo D, Van Roy B (2013) Eluder dimension and the sample complexity of optimistic exploration. *Adv. Neural Inform. Processing Systems* 26:2256–2264.

Shah D, Xie Q (2018) Q-Learning with nearest neighbors. *Adv. Neural Inform. Processing Systems* 31:3111–3121.

Shah D, Song D, Xu Z, Yang Y (2020) Sample efficient reinforcement learning via low-rank matrix estimation. *Adv. Neural Inform. Processing Systems* 33:12092–12103.

Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, et al. (2016) Mastering the game of Go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, et al. (2017) Mastering chess and shogi by self-play with a general reinforcement learning algorithm. Preprint, submitted December 5, https://arxiv.org/abs/1712.01815.

Simchowitz M, Jamieson KG (2019) Non-asymptotic gap-dependent regret bounds for tabular MDPs. *Adv. Neural Inform. Processing Systems* 33:1151–1160.

Sinclair SR, Banerjee S, Yu CL (2019) Adaptive discretization for episodic reinforcement learning in metric spaces. *Proc. ACM Measurement Anal. Comput. Systems*, vol. 3, 1–44.

Sinclair SR, Wang T, Jain G, Banerjee S, Yu C (2020) Adaptive discretization for model-based reinforcement learning. *Adv. Neural Inform. Processing Systems* 34:33.

Singh SP, Jaakkola T, Jordan MI (1995) Reinforcement learning with soft state aggregation. *Adv. Neural Inform. Processing Systems* 7:361–368.

Slivkins A (2014) Contextual bandits with similarity information. *J. Machine Learn. Res.* 15(1):2533–2568.

Slivkins A (2019) Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning*, vol. 12 (now Publishers Inc., Norwell, MA).

Song Z, Sun W (2019) Efficient model-free reinforcement learning in metric spaces. Preprint, submitted May 1, https://arxiv.org/abs/1905.00475.

Sutton RS, Barto AG (2018) *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA).

Tessler C, Shpigelman Y, Dalal G, Mandelbaum A, Haritan Kazakov D, Fuhrer B, Chechik G, Mannor S (2022) Reinforcement learning for datacenter congestion control. *Performance Evaluation Rev.* 49(2):43–46.

Uther WT, Veloso MM (1998) Tree based discretization for continuous state space reinforcement learning. AAAI/IAAI, 769–774.

Wang R, Salakhutdinov R, Yang LF (2020) Provably efficient reinforcement learning with general value function approximation. Preprint, submitted May 21, https://arxiv.org/abs/2005.10804.

Wang Y, Wang R, Du SS, Krishnamurthy A (2019) Optimism in reinforcement learning with generalized linear function approximation. Preprint, submitted December 9, https://arxiv.org/abs/1912.04136.

Wanigasekara N, Yu C (2019) Nonparametric contextual bandits in metric spaces with unknown metric. *Adv. Neural Inform. Processing Systems* 32:14657–14667.

Watkins CJCH (1989) Learning from delayed rewards. PhD thesis, Cambridge University, Cambridge, UK.

Whiteson S, Stone P (2006) Evolutionary function approximation for reinforcement learning. *J. Machine Learn. Res.* 7:877–917.

Yang LF, Ni C, Wang M (2019) Learning to control in metric space with optimal regret. Preprint, submitted May 5, https://arxiv.org/abs/1905.01576.

Zanette A, Brunskill E (2019) Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. Preprint, submitted January 1, https://arxiv.org/abs/1901.00210.

Zanette A, Lazaric A, Kochenderfer MJ, Brunskill E (2019) Limiting extrapolation in linear approximate value iteration. *Adv. Neural Inform. Processing Systems* 32:5616–5625.

Sean R. Sinclair is a graduate student in the school of operations research and information engineering at Cornell, working on developing algorithms for data-driven sequential decision making in societal applications.

Siddhartha Banerjee is an associate professor in the school of operations research and information engineering at Cornell, working on topics at the intersection of data-driven decision making, network algorithms, and market design.

Christina Lee Yu is an assistant professor at Cornell University in the school of operations research and information engineering.