Anytime-Competitive Reinforcement Learning with Policy Prior

Jianyi Yang
UC Riverside
Riverside, CA, USA
jyang239@ucr.edu

Pengfei Li
UC Riverside
Riverside, CA, USA
pli081@ucr.edu

Tongxin Li
CUHK Shenzhen
Shenzhen, Guangdong, China
litongxin@cuhk.edu.cn

Adam Wierman
Caltech
Pasadena, CA, USA
adamw@caltech.edu

Shaolei Ren
UC Riverside
Riverside, CA, USA
shaolei@ucr.edu

Abstract

This paper studies the problem of Anytime-Competitive Markov Decision Process (A-CMDP). Existing works on Constrained Markov Decision Processes (CMDPs) aim to optimize the expected reward while constraining the *expected* cost over random dynamics, but the cost in a specific episode can still be unsatisfactorily high. In contrast, the goal of A-CMDP is to optimize the expected reward while guaranteeing a bounded cost in *each* round of *any* episode against a policy prior. We propose a new algorithm, called Anytime-Competitive Reinforcement Learning (ACRL), which provably guarantees the anytime cost constraints. The regret analysis shows the policy asymptotically matches the optimal reward achievable under the anytime competitive constraints. Experiments on the application of carbon-intelligent computing verify the reward performance and cost constraint guarantee of ACRL.

1 Introduction

In mission-critical online decision-making problems such as cloud workload scheduling [49, 20], cooling control in datacenters [63, 16, 46], battery management for Electrical Vehicle (EV) charging [57, 36], and voltage control in smart grids [54, 73], there is always a need to improve the reward performances while meeting the requirements for some important cost metrics. In these mission-critical systems, there always exist some policy priors that meet the critical cost requirements, but they may not perform well in terms of the rewards. In the application of cooling control, for example, some rule-based heuristics [19, 46] have been programmed into the real systems for a long time and have verified performance in maintaining a safe temperature range, but they may not achieve a high energy efficiency. In this paper, we design a Reinforcement Learning (RL) algorithm with the goal of optimizing the reward performance under the guarantee of cost constraints against a policy prior for any round in any episode.

Constrained RL algorithms have been designed to solve various Constrained MDPs (CMDPs) with reward objective and cost constraints. Among them, some are designed to guarantee the expected cost constraints [61, 62], some can guarantee the cost constraints with a high probability (w.h.p.) [13], and others guarantee a bounded violation of the cost constraints [26, 29, 21, 22, 1]. In addition, conservative RL algorithms [27, 68, 31, 65] consider constraints that require the performance of a learning agent is no worse than a known baseline policy in expectation or with a high probability. [27] points out that it is impossible to guarantee the constraints for any round with probability

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Methods	Unknown dynamic	Expected constraints or constriants w.h.p.		Any-episode constraints	Anytime-competitive constraints
		With violation	No violation		
Learning-augmentation	✘	N/A	N/A	[38, 51, 17]	✘
Constrained RL	✔	[29, 21, 22, 1, 67, 26]	[62, 61]	✘	✘
Conservative RL	✔	N/A	[68, 27, 31, 65]	✘	✘
ACRL (this work)	✔	N/A	✔	✔	✔

Table 1: Comparison between ACRL and most related works.

one while such guarantees are desirable. In real mission-critical applications, the cost constraints are often required to be satisfied at each round in any episode even in the worst case, but such anytime constraints are not guaranteed by the existing constrained/conservative RL policies. Recently, learning-augmented online control algorithms [37, 38, 69, 51, 17, 35] have been developed to exploit machine learning predictions with the worst-case control performance guarantee. Nonetheless, the learning-augmented control algorithms require the full knowledge of the dynamic models, which limits their applications in many systems with unknown random dynamic models. A summary of most relevant works is given in Table 1.

To fill in this technical blank, we model the mission-critical decision-making problem as a new Markov Decision Process (MDP) which is called the Anytime-Competitive MDP (A-CMDP). In A-CMDP, the environment feeds back a reward and a cost corresponding to the selected action at each round. The next state is updated based on a random dynamic model which is a function of the current action and state and is not known to the agent. The distribution of the dynamic model is also unknown to the agent and needs to be learned. Importantly, at each round h in any episode, the policy of A-CMDP must guarantee that the cumulative cost J_h is upper bounded by a scaled cumulative cost of the policy prior $\pi^\dagger$ plus a relaxation, i.e. $J_h \leq (1+\lambda)J_h^\dagger + hb$ with $\lambda, b > 0$, which is called an *anytime* competitive constraint or *anytime* competitiveness. The term "competitive" or "competitiveness" is used to signify the performance comparison with a policy prior. Under the anytime cost competitiveness for all rounds, the RL agent explores the policy to optimize the expected reward.

The anytime competitiveness guarantee is more strict than the constraints in typical constrained or conservative MDPs, which presents new challenges for the RL algorithm design. First of all, the anytime competitive constraints are required to be satisfied for any episode, even for the early episodes when the collected sequence samples are not enough. Also, to guarantee the constraints for each round, we need to design a safe action set for each round to ensure that feasible actions exist to meet the constraints in subsequent rounds. Last but not least, without knowing the full transition model, the agent has no access to the action sets defined by the anytime competitive constraints. Thus, in comparison to the control settings with known transition models [38], ensuring the anytime competitiveness for MDPs is more challenging.

Contributions. In this paper, we design algorithms to solve the novel problem of A-CMDP. The contributions are summarized as follows. First, we propose an Anytime-Competitive Decision-making (ACD) algorithm to provably guarantee the anytime competitive constraints for each episode. The key design in ACD is a projection to a safe action set in each round. The safe action set is updated at each round according to a designed rule to gain as much flexibility as possible to optimize the reward. Then, we develop a new model-based RL algorithm (ACRL) to learn the optimal ML model used in ACD. The proposed model-based RL can effectively improve the reward performance based on the new dynamic defined by ACD. Last but not least, we give rigorous analysis on the reward regret of ACRL compared with the optimal-unconstrained policy. The analysis shows that the learned policy performs as well as the optimal ACD policy and there exists a fundamental trade-off between the optimization of the average reward and the satisfaction of the anytime competitive constraints.

2 Related Work

Constrained RL. Compared with the existing literature on constrained RL [68, 1, 67, 10, 13, 2, 21, 29, 26, 22, 61], our study has important differences. Concretely, the existing constrained RL works consider an *average* constraint with or without constraint violation. In addition, existing conservative RL works [68] consider an *average* constraint compared with a policy prior. However, the constraints can be violated especially at early exploration episodes. In sharp contrast, our anytime competitive constraint ensures a strict constraint for any round in each episode, which has not been studied in the existing literature as shown in Table 1. In fact, with the same policy prior, our anytime competitive policy can also meet the average constraint without violation in conservative/constrained RL [68, 25]. [56] considers MDPs that satisfy safety constraint with probability one and proposes an approach with a high empirical performance. However, there is no theoretical guarantee for constraint satisfaction. Comparably, our method satisfies the constraint with theoretical guarantee, which is essential to deploy AI for mission-critical applications.

Our study is also relevant to safe RL. Some studies on safe RL [10, 2, 13] focus on constraining that the system state or action at each time h cannot fall into certain pre-determined restricted regions (often with a high probability), which is orthogonal to our anytime competitiveness requirement that constrains the cumulative cost at each round of an episode. Our study is related to RL with safety constraints [10, 39], but is highlighted by the strict constraint guarantee for each round in each episode. In a study on safe RL [10], the number of safety violation events is constrained almost surely by a budget given in advance, but the safety violation value can still be unbounded. By contrast, our work strictly guarantees the anytime competitive constraints by designing the safety action sets. In a recent study [3], the safety requirement is formulated as the single-round cost constraints. Differently, we consider cumulative cost constraints which have direct motivation from mission-critical applications.

Learning-augmented online decision-making. Learning-based policies can usually achieve good average performance but suffer from unbounded worst-case performance. To meet the requirements for the worst-case performance of learning-based policies, learning-augmented algorithms are developed for online control/optimization problems [38, 51, 17, 35, 34]. To guarantee the performance for each problem instance, learning-augmented algorithm can perform an online switch between ML policy and prior [51], combine the ML policy and prior with an adaptive parameter [38], or project the ML actions into safe action sets relying on the prior actions [35]. Compared with learning-augmented algorithms, we consider more general online settings without knowing the exact dynamic model. Also, our problem can guarantee the cost performance for each round in any episode compared with a policy prior, which has not been studied by existing learning-augmented algorithms.

3 Problem Formulation

3.1 Anytime-Competitive MDP

In this section, we introduce the setting of a novel MDP problem called Anytime-Competitive Markov Decision Process (A-CMDP), denoted as $\mathcal{M}(\mathcal{X}, \mathcal{A}, \mathcal{F}, g, H, r, c, \pi, \pi^\dagger)$. In A-CMDP, each episode has H rounds. The state at each round is denoted as $x_h \in \mathcal{X}, h \in [H]$. At each round of an episode, the agent selects an action a_h from an action set $\mathcal{A}$. The environment generates a reward $r_h(x_h, a_h)$ and a cost $c_h(x_h, a_h)$ with $r_h \in \mathcal{R}$ and $c_h \in \mathcal{C}$. We model the dynamics as $x_{h+1} = f_h(x_h, a_h)$ where $f_h \in \mathcal{F}$ is a random transition function drawn from an unknown distribution $g(f_h)$ with the density $g \in \mathcal{G}$. The agent has no access to the random function f_h but can observe the state x_h at each round h. Note that we model the dynamics in a function style for ease of presentation, and this dynamic model can be translated into the transition probability in standard MDP models [59, 5] as $\mathbb{P}(x_{h+1} \mid x_h, a_h) = \sum_{f_h} \mathbb{1}(f_h(x_h, a_h) = x_{h+1}) g(f_h)$. A policy π is a function which gives the action a_h for each round $h \in [H]$. Let $V_h^\pi(x) = \mathbb{E}\left[\sum_{i=h}^H r_i(x_i, a_i)) \mid x_h = x\right]$ denote the expected value of the total reward from round h by policy π. One objective of A-CMDP is to maximize the expected total reward starting from the first round which is denoted as $\mathbb{E}_{x_1}\left[V_1^\pi(x_1)\right] = \mathbb{E}\left[\sum_{h=1}^H r_h(x_h, a_h))\right]$.

Besides optimizing the expected total reward as in existing MDPs, A-CMDP also guarantees the anytime competitive cost constraints compared with a policy prior $\pi^\dagger$. The policy prior can be a policy that has verified cost performance in real systems or a heuristic policy with strong empirically-guaranteed cost performance, for which concrete examples will be given in the next section. Denote $y_h = (f_h, c_h, r_h)$, and $y_{1:H} = \{y_h\}_{h=1}^H \in \mathcal{Y} = \mathcal{F} \times \mathcal{R} \times \mathcal{C}$ is a sampled sequence of the models in an A-CMDP. Let $J_h^\pi(y_{1:H}) = \sum_{i=1}^h c_i(x_i, a_i)$ be the cost up to round $h \in [H]$ with states $x_i, i \in [h]$ and actions $a_i, i \in [h]$ of a policy π. Also, let $J_h^{\pi^\dagger}(y_{1:H}) = \sum_{i=1}^h c_i(x_i^\dagger, a_i^\dagger)$ be the cost of the prior with states $x_i^\dagger, i \in [h]$ and actions $a_i^\dagger, i \in [h]$ of the prior $\pi^\dagger$. The anytime competitive constraints are defined as below.

Definition 3.1 (Anytime competitive constraints)**.** If a policy π satisfies $(\lambda, b)-$anytime competitiveness, the cost of π never exceeds the cost of the policy prior $\pi^\dagger$ relaxed by parameters $\lambda \geq 0$ and $b \geq 0$, i.e. for any round h in any model sequence $y_{1:H} \in \mathcal{Y}$, it holds that $J_h^\pi(y_{1:H}) \leq (1+\lambda) J_h^{\pi^\dagger}(y_{1:H}) + hb$.

Now, we can formally express the objective of A-CMDP with Π being the policy space as

$$\max_{\pi \in \Pi} \mathbb{E}_{x_1}\left[V_1^\pi(x_1)\right], \quad s.t. \;\; J_h^\pi(y_{1:H}) \leq (1+\lambda) J_h^{\pi^\dagger}(y_{1:H}) + hb, \;\; \forall h \in [H], \forall y_{1:H} \in \mathcal{Y}. \tag{1}$$

Let $\Pi_{\lambda,b}$ be the collection of policies that satisfy the anytime competitive constraints in (1). We design an anytime-competitive RL algorithm that explores the policy space $\Pi_{\lambda,b}$ in K episodes to optimize the expected reward $\mathbb{E}_{x_1}\left[V_1^\pi(x_1)\right]$. Note that different from constrained/conservative MDPs [26, 29, 61, 68, 1, 67], the anytime competitive constraints in (1) must be satisfied for any round in any sampled episode $y_{1:H} \in \mathcal{Y}$ given relaxed parameters $\lambda, b \geq 0$. To evaluate the performance of the learned policy $\pi^k \in \Pi_{\lambda,b}, k \in [K]$ and the impact of the anytime competitive constraints, we consider the regret performance metric defined as

$$\text{Regret}(K) = \sum_{k=1}^{K} \mathbb{E}_{x_1}\left[V_1^{\pi^*}(x_1) - V_1^{\pi^k}(x_1)\right], \text{with } \pi^k \in \Pi_{\lambda,b} \tag{2}$$

where $\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{x_1}\left[V_1^\pi(x_1)\right]$ is an optimal policy without considering the anytime competitive constraints. When λ or b becomes larger, the constraints get less strict and the algorithm has more flexibility to minimize the regret in (2). Thus, the regret analysis will show the trade-off between optimizing the expected reward and satisfying the anytime cost competitiveness.

In this paper, we make additional assumptions on the cost functions, transition functions, and the prior policy which are important for the anytime-competitive algorithm design and analysis.

Assumption 3.2. All the cost functions in the space $\mathcal{C}$ have a minimum value $\epsilon \geq 0$, i.e. $\forall (x, a), \forall h \in [H], c_h(x, a) \geq \epsilon \geq 0$, and are L_c-Lipschitz continuous with respect to action a_h and the state x_h. All the transition functions in the space $\mathcal{F}$ are L_f-Lipschitz continuous with respect to action a_h and the state x_h. The parameters ϵ, L_c and L_f are known to the agent.

The Lipschitz continuity of cost and transition functions can also be found in other works on model-based MDP [45, 5, 28]. The Lispchitz assumptions actually apply to many continuous mission-critical systems like cooling systems [16], power systems [54, 15] and carbon-aware datacenters [49]. In these systems, the agents have no access to concrete cost and transition functions, but they can evaluate the Lipschitz constants of cost and dynamic functions based on the prior knowledge of the systems. The minimum cost value can be as low as zero, but the knowledge of a positive minimum cost ϵ can be utilized to improve the reward performance which will be discussed in Section 5.1.

Definition 3.3 (Telescoping policy)**.** A policy π satisfies the telescoping property if the policy is applied from round h_1 to h_2 with initialized states x_{h_1} and x'_{h_1}, it holds for the corresponding states x_{h_2} and x'_{h_2} at round h_2 that

$$\|x_{h_2} - x'_{h_2}\| \leq p(h_2 - h_1)\|x_{h_1} - x'_{h_1}\|, \tag{3}$$

where $p(h)$ is called a perturbation function with h and $p(0) = 1$.

Assumption 3.4. The prior policy $\pi^\dagger$ satisfies the telescoping property with some perturbation function p. Furthermore, $\pi^\dagger$ is Lipschitz continuous.

The telescoping property in Definition 3.3 indicates that with an initial state perturbation at a fixed round, the maximum divergence of the states afterwards is bounded. Thus, the perturbation function p measures the sensitivity of the state perturbation with respect to a policy prior. The telescoping property is satisfied for many policy priors [60, 43]. It is also assumed for perturbation analysis in model predictive control [42].

Note that in A-CMDP, the constraints are required to be satisfied for any round in any sequence, which is much more stringent than constraint satisfaction in expectation or with a high probability. In fact, the any-time constraints cannot be theoretically guaranteed without further knowledge on the system [8, 56]. This paper firstly shows that Assumption 3.2 and Assumption 3.4, which are reasonable for many mission-critical applications [16, 54, 15, 49], are enough to guarantee the anytime competitive constraints, thus advancing the deployment of RL in mission-critical applications.

3.2 Motivating Examples

The anytime competitiveness has direct motivations from many mission-critical control systems. We present two examples in this section and defer other examples to the appendix.

Safe cooling control in data centers. In mission-critical infrastructures like data centers, the agent needs to make decisions on cooling equipment management to maintain a temperature range and achieve a high energy efficiency. Over many years, rule-based policies have been used in cooling systems and have verified cooling performance in maintaining a suitable temperature for computing [46]. Recently, RL algorithms are developed for cooling control in data centers to optimize the energy efficiency [63, 16, 46]. The safety concerns of RL policies, however, hinder their deployment in real systems. In data centers, an unreliable cooling policy can overheat devices and denial critical services, causing a huge loss [19, 46]. The safety risk is especially high at the early exploration stage of RL in the real environment. Therefore, it is crucial to guarantee the constraints on cooling performance at anytime in any episode for safety. With the reliable rule-based policies as control priors, A-CMDP can accurately model the critical parts of the cooling control problem, opening a path towards learning reliable cooling policies for data centers.

Workload scheduling in carbon-intelligent computing. The world is witnessing a growing demand for computing power due to new computing applications. The large carbon footprint of computing has become a problem that cannot be ignored [49, 64, 52, 23]. Studies find that the amount of carbon emission per kilowatt-hour on electricity grid varies with time and locations due to the various types of electricity generation [33, 32, 9]. Exploiting the time-varying property of carbon efficiency, recent studies are developing workload scheduling policies (e.g. delay some temporally flexible workloads) to optimize the total carbon efficiency [49]. However, an unreliable workload scheduling policy in data centers can cause a large computing latency, resulting in an unsatisfactory Quality of Service (QoS). Thus, to achieve a high carbon efficiency while guaranteeing a low computing latency, we need to solve an A-CMDP which leverages RL to improve the carbon efficiency while guaranteeing the QoS constraints compared with a policy prior targeting at computing latency [20, 30, 12, 72, 71]. This also resembles the practice of carbon-intelligent computing adopted by Google [49].

4 Methods

In this section, we first propose an algorithm to guarantee the anytime competitive constraints for any episode, and then give an RL algorithm to achieve a high expected reward under the guarantee of the anytime competitive constraints.

4.1 Guarantee the Anytime Constraints

It is challenging to guarantee the anytime competitive constraints in (1) for an RL policy in any episode due to the following. First of all, in MDPs, the agent can only observe the *real* states $\{x_h\}_{h=1}^H$ corresponding to the truly-selected actions $\{a_h\}_{h=1}^H$. The agent does not select the actions $a_h^\dagger$ of the prior, so the states of the prior $x_h^\dagger$ are *virtual* states that are not observed. Thus, the agent cannot evaluate the prior cost $J_h^{\pi^\dagger}$ which is in the anytime competitive constraint at each round h. Also, the action at each round h has an impact on the costs in the future rounds $i, i > h$ based on the random transition models $f_i, i \geq h$. Thus, besides satisfying the constraints in the current round, we need to have a good planning for the future rounds to avoid any possible constraint violations even without the exact knowledge of transition and/or cost models. Additionally, the RL policy may be arbitrarily bad in the environment and can give high costs (especially when very limited training data is available), making constraint satisfaction even harder.

Despite the challenges, we design safe action sets $\{\mathcal{A}_h, h \in [H]\}$ to guarantee the anytime competitive constraints: if action a_h is strictly selected from $\mathcal{A}_h$ for each round h, the anytime competitive constraints for all rounds are guaranteed. As discussed above, the anytime competitive constraints cannot be evaluated at any time since the policy prior's state and cost information is not available. Thus, we propose to convert the original anytime competitive constraints into constraints that only depend on the known parameters and the action differences between the real policy and the policy prior. We give the design of the safe action sets based on the next proposition. For the ease of presentation, we denote $c_i = c_i(x_i, a_i)$ as the real cost and $c_i^\dagger = c_i(x_i^\dagger, \pi(x_i^\dagger))$ as the cost of the policy prior at round i.

Proposition 4.1. *Suppose that Assumption 3.2 and 3.4 are satisfied. At round h with costs $\{c_i\}_{i=1}^{h-1}$ observed, the anytime competitive constraints $J_{h'}^{\pi} \leq (1+\lambda)J_{h'}^{\pi^\dagger} + h'b$ for rounds $h' = h, \cdots, H$ are satisfied if for all subsequent rounds $h' = h, \cdots, H$,*

$$\sum_{j=h}^{h'} \Gamma_{j,j} \|a_j - \pi^\dagger(x_j)\| \leq G_{h,h'}, \ \forall h' = h, \cdots, H, \tag{4}$$

where $\Gamma_{j,n} = \sum_{i=n}^{H} q_{j,i}, (j \in [H], \forall n \geq j)$, with $q_{j,i} = L_c \mathbb{1}(j = i) + L_c(1 + L_{\pi^\dagger})L_f p(i - 1 - j)\mathbb{1}(j < i), (\forall j \in [H], i \geq j)$, relying on known parameters, and $G_{h,h'}$ is called the allowed deviation which is expressed as

$$G_{h,h'} = \sum_{i=1}^{h-1} \Big((1+\lambda)\hat{c}_i^\dagger - c_i - \Gamma_{i,h} d_i \Big) + (h' - h + 1)(\lambda\epsilon + b), \tag{5}$$

where $\hat{c}_i^\dagger = \max\Big\{\epsilon, c_i - \sum_{j=1}^{i} q_{j,i} d_j \Big\}, (\forall i \in [H])$, is the lower bound of of $c_i^\dagger$, and $d_j = \|a_j - \pi^\dagger(x_j)\|, \forall j \in [H]$ is the action difference at round j.

At each round $h \in [H]$, Proposition 4.1 provides a sufficient condition for satisfying all the anytime competitive constraints from round h to round H given in (1). The meanings of the parameters in Proposition 4.1 are explained as follows. The weight $q_{j,i}$ measures the impact of action deviation at round j on the cost difference $|c_i - c_i^\dagger|$ at round $i \geq j$, and the weight $\Gamma_{j,n}$ indicates the total impact of the action deviation at round j on the sum of the cost differences from rounds n to round H. Based on the definition of $q_{j,i}$, we get $\hat{c}_i^\dagger$ as a lower bound of the prior cost $c_i^\dagger$. With these bounds, we can calculate the maximum allowed total action deviation compared with the prior actions $\pi^\dagger(x_j)$ from round $j = h$ to h' as $G_{h,h'}$

By applying Proposition 4.1 at initialization, we can guarantee the anytime competitive constraints for all rounds $h' \in [H]$ if we ensure that for all rounds $h' \in [H]$, $\sum_{j=1}^{h'} \Gamma_{j,j} \|a_j - \pi^\dagger(x_j)\| \leq G_{1,h'} = h'(\lambda\epsilon + b)$. This sufficient condition is a long-term constraint relying on the relaxation parameters λ and b. Although we can guarantee the anytime competitive constraints by the sufficient condition obtained at initialization, we apply Proposition 4.1 at all the subsequent rounds with the cost feedback information to get larger action sets and more flexibility to optimize the average reward. In this way, we can update the allowed deviation according to the next corollary.

Corollary 4.2. *At round 1, we initialize the allowed deviation as $D_1 = \lambda\epsilon + b$. At round $h, h > 1$, the allowed deviation is updated as*

$$D_h = \max\{D_{h-1} + \lambda\epsilon + b - \Gamma_{h-1,h-1} d_{h-1}, \ R_{h-1} + \lambda\epsilon + b\} \tag{6}$$

where $R_{h-1} = \sum_{i=1}^{h-1} \Big((1+\lambda)\hat{c}_i^\dagger - c_i - \Gamma_{i,h} d_i \Big)$ with notations defined in Proposition 4.1. The $(\lambda, b)-$anytime competitive constraints in Definition 3.1 are satisfied if it holds at each round h that $\Gamma_{h,h}\|a_h - \pi^\dagger(x_h)\| \leq D_h$.

Corollary 4.2 gives a direct way to calculate the allowed action deviation at each round. In the update rule (6) of the allowed deviation, the first term of the maximum operation is based on the deviation calculation at round $h - 1$ while the second term is obtained by applying Proposition 4.1 for round h.

We can find that the conditions to satisfy the anytime competitive constraints can be controlled by parameters λ and b. With larger λ and b, the anytime competitive constraints are relaxed and the conditions in Corollary 4.2 get less stringent. Also, the conditions in Corollary 4.2 rely on the minimum cost value ϵ and other system parameters including Lipschitz constants $L_c, L_f, L_{\pi^\dagger}$ and telescoping parameters p through $\Gamma_{h,h}$. Since $\Gamma_{h,h}$ increases with the Lipschitz and telescoping parameters, even if the estimated Lipschitz constants and the telescoping parameters are higher than the actual values or the estimated minimum cost is lower than the actual value, the obtained condition by Corollary 4.2 is sufficient to guarantee the anytime competitive constraints, although it is more stringent than the condition calculated by the actual parameters.

By Corollary 4.2, we can define the safe action set at each round h as

$$\mathcal{A}_h(D_h) = \left\{ a \mid \Gamma_{h,h}\|a - \pi^\dagger(x_h)\| \leq D_h \right\}. \tag{7}$$

Algorithm 1 Anytime-Competitive Decision-making (ACD)

```
Initialization: Initialize an allowed deviation: $D_1 = \lambda\epsilon + b$.
for $h = 1, \cdots, H$ do
    Obtain the output of the ML policy $\tilde{\pi}$ as $\tilde{a}_h$.
    Select the action $a_t$ by projecting $\tilde{a}_h$ into the safe action set $\mathcal{A}_h(D_h)$ in (7).
    Update the allowed deviation $D_{h+1}$ by (6).
end for
```

With the safe action set design in (7), we propose a projection-based algorithm called ACD in Algorithm 1. We first initialize an allowed deviation as $D_1 = \lambda\epsilon + b$. When the output $\tilde{a}_h$ of the ML model is obtained at each round h, it is projected into a safe action set $\mathcal{A}_h(D_h)$ depending on the allowed deviation D_h, i.e. $a_h = P_{\mathcal{A}_h(D_h)}(\tilde{a}_h) = \arg\min_{a\in\mathcal{A}_h(D_h)} \|a - \tilde{a}_h\|$. The projection can be efficiently solved by many existing methods on constrained policy learning [67, 11, 4, 41, 24]. The allowed deviation is then updated based on Corollary 4.2. Intuitively, if the actions are closer to the prior actions before h, i.e. the action deviations $\{d_i\}_{i=1}^{h-1}$ get smaller, then R_{h-1} becomes larger and D_h becomes larger, leaving more flexibility to deviate from $a_i^\dagger, i \geq h$ in subsequent rounds.

4.2 Anytime-Competitive RL

The anytime competitive constraints have been satisfied by Algorithm 1, but it remains to design an RL algorithm to optimize the average reward under the anytime competitive cost constraints, which is given in this section.

The anytime-competitive decision-making algorithm in Algorithm 1 defines a new MDP, with an additional set of allowed deviations $\mathcal{D}$ to the A-MDP defined in Section 3.1, denoted as $\tilde{\mathcal{M}}(\mathcal{X}, \mathcal{D}, \mathcal{A}, \mathcal{F}, g, H, r, c, \tilde{\pi}, \pi^\dagger)$. In the new MDP, we define an augmented state s_h which include the original state x_h, the allowed deviation $D_h \in \mathcal{D}$, and history information $\{c_i\}_{i=1}^{h-1}$ and $\{d_i\}_{i=1}^{h-1}$. The transition of x_h is defined by f_h in Section 3.1 and needs to be learned while the transition of D_h is defined in (6) and is known to the agent. The ML policy $\tilde{\pi}$ gives an output $\tilde{a}_h$ and the selected action is the projected action $a_h = P_{\mathcal{A}_h(D_h)}(\tilde{a}_h)$. Then the environment generates a reward $r_h(x_h, P_{\mathcal{A}_h(D_h)}(\tilde{a}_h))$ and a cost $c_h(x_h, P_{\mathcal{A}_h(D_h)}(\tilde{a}_h))$. Thus, the value function corresponding to the ML policy $\tilde{\pi}$ can be expressed as $\tilde{V}_h^{\tilde{\pi}}(s_h) = \mathbb{E}\left[\sum_{i=h}^H r_i(x_i, P_{\mathcal{A}_h(D_h)}(\tilde{a}_h))\right]$ with $\tilde{a}_h$ being the output of the ML policy $\tilde{\pi}$. For notation convenience, we sometimes write the actions of π^* and $\pi^\dagger$ as $\pi^*(s)$ and $\pi^\dagger(s)$ even though they only reply on the original state x in s.

To solve the MDP, we propose a model-based RL algorithm called ACRL in Algorithm 2. Different from the existing model-based RL algorithms [48, 6, 70], ACRL utilizes the dynamic model of A-CMDP and ACD (Algorithm 1) to optimize the average reward. Given a transition distribution g at episode k, we perform value iteration to update $\tilde{Q}$ functions for $h = 1, \cdots, H$.

$$\begin{aligned} &\tilde{Q}_h^k(s_h, \tilde{a}_h) = r_h(x_h, a_h) + \mathbb{E}_g\left[\tilde{V}_{h+1}^k(s_{h+1}) \mid s_h, a_h\right], \; \tilde{V}_h^k(s_h) = \max_{a\in\mathcal{A}} \tilde{Q}_h^k(s_h, a), \\ &\mathbb{E}_g\left[\tilde{V}_{h+1}^k(s_{h+1}) \mid s_h, a_h\right] = \sum_{f\in\mathcal{F}} \tilde{V}_{h+1}^k(s_{h+1}) g(f), \end{aligned} \tag{8}$$

where $a_h = P_{\mathcal{A}_h(D_h)}(\tilde{a}_h)$, $\tilde{Q}_{H+1,k}(s, a) = 0, \tilde{V}_{H+1,k}(s) = 0$. The transition model g is estimated as

$$\hat{g}^k = \arg\min_{g\in\mathcal{G}} \sum_{i=1}^{k-1}\sum_{h=1}^{H} \left(\mathbb{E}_g\left[\tilde{V}_{h+1}^i(s_{h+1}) \mid s_h, a_h\right] - \tilde{V}_{h+1}^i(s_{h+1})\right)^2. \tag{9}$$

Based on the transition estimation, we can calculate the confidence set of the transition model as

$$\mathcal{G}_k = \left\{ g \in \mathcal{G} \,\middle|\, \sum_{i=1}^{k-1}\sum_{h=1}^{H} \left(\mathbb{E}_g\left[\tilde{V}_{h+1}^i(s_{h+1}) \mid s_h, a_h\right] - \mathbb{E}_{\hat{g}^k}\left[\tilde{V}_{h+1}^i(s_{h+1}) \mid s_h, a_h\right]\right)^2 \leq \beta_k \right\}, \tag{10}$$

where $\beta_k > 0$ is a confidence parameter.

Algorithm 2 Anytime-Competitive Reinforcement Learning (ACRL)

```
Initialization: Transition model set $\mathcal{G}_1 = \{\hat{g}^1\}$.
for each episode $k = 1, \cdots, K$ do
    Observe the initial state $s_1^k$.
    Select $g^k = \arg\max_{g \in \mathcal{G}^k} \mathbb{E}_g\left[V_1(s_1^k)\right]$.
    Perform value iteration with $g^k$ in Eqn. (8) and update $\tilde{Q}$ functions $\tilde{Q}_1^k \cdots, \tilde{Q}_H^k$.
    for each round $h = 1, \cdots, H$ do
        Run ACD (Algorithm 1) by ML policy $\tilde{\pi}^k(s_h) = \arg\max_{a \in \mathcal{A}} \tilde{Q}_h^k(s_h, a)$
        Observe state $s_{h+1}^k$ and store values $\tilde{V}_{h+1}^k(s_{h+1}^k)$.
    end for
    Update transition model $\hat{g}^{k+1}$ using (9) and calculate confidence set $\mathcal{G}_{k+1}$.
end for
```

With a learned ML policy $\tilde{\pi}^k$ at each episode k, the policy used for action selection is the ACD policy π^k. Given the optimal ML policy $\tilde{\pi}^* = \arg\max_{\tilde{\pi} \in \tilde{\Pi}} \tilde{V}_1^{\tilde{\pi}}(s_1)$ with $\tilde{\Pi}$ being the ML policy space, the optimal ACD policy is denoted as π°. For state s_h at round h, π^k and π° select actions as

$$\pi^k(s_h) = P_{\mathcal{A}_h(D_h)}(\tilde{\pi}^k(s_h)), \; \pi^\circ(s_h) = P_{\mathcal{A}_h(D_h)}(\tilde{\pi}^*(s_h)). \tag{11}$$

In the definition of A-CMDP, the dimension of the augmented state s_h increases with the length of the horizon H, which cloud cause a scalability issue for implementation. The scalability issues also exit in other RL works with history-dependent states [58, 14]. In practice, tractable methods can be designed through feature aggregation [58] or PODMP [66].

5 Performance Analysis

In this section, we analyze the reward regret of ACRL to show the impacts of anytime cost constraints on the average reward.

5.1 Regret due to Constraint Guarantee

Intuitively, due to the anytime competitive constraints in Eqn. (1), there always exists an unavoidable reward gap between an ACD policy and the optimal-unconstrained policy π^*. In this section, to quantify this unavoidable gap, we bound the regret of the optimal ACD policy π°, highlighting the impact of anytime competitive cost constraints on the average reward performance.

Theorem 5.1. *Assume that the optimal-unconstrained policy π^* has a value function $Q_h^{\pi^*}(x, a)$ which is $L_{Q,h}$-Lipschitz continuous with respect to the action a for all x. The regret between the optimal ACD policy π° that satisfies $(\lambda, b)-$anytime competitiveness and the optimal-unconstrained policy π^* is bounded as*

$$\mathbb{E}_{x_1}\left[V_1^{\pi^*}(x_1) - V_1^{\pi^\circ}(x_1)\right] \leq \mathbb{E}_{y_{1:H}}\left\{\sum_{h=1}^{H} L_{Q,h}\left[\eta - \frac{1}{\Gamma_{h,h}}(\lambda\epsilon + b + \Delta G_h)\right]^+\right\}, \tag{12}$$

where $\eta = \sup_{x \in \mathcal{X}} \|\pi^*(x) - \pi^\dagger(x))\|$ is the maximum action discrepancy between the policy prior $\pi^\dagger$ and optimal-unconstrained policy π^*; $\Gamma_{h,h}$ is defined in Proposition 4.1; $\Delta G_h = [R_{h-1}]^+$ is the gain of the allowed deviation by applying Proposition 4.1 at round h.

The regret bound stated in Theorem 5.1 is intrinsic and inevitable, due to the committed assurance of satisfying the anytime competitive constraints. Such a bound cannot be improved via policy learning, i.e., converge to 0 when the number of episodes $K \to \infty$. This is because to satisfy the $(\lambda, b)-$anytime competitiveness, the feasible policy set $\Pi_{\lambda,b}$ defined under (1) is a subset of the original policy set Π, and the derived regret is an upper bound of $\max_{\pi \in \Pi} \mathbb{E}_{x_1}\left[V_1^\pi(x_1)\right] - \max_{\pi \in \Pi_{\lambda,b}} \mathbb{E}_{x_1}\left[V_1^\pi(x_1)\right]$. Moreover, the regret bound relies on the action discrepancy η. This is because if the optimal-unconstrained policy π^* is more different from the prior $\pi^\dagger$, its actions are altered to a larger extent to guarantee the constraints, resulting in a larger degradation of the reward performance. More importantly, the regret bound indicates the trade-off between the reward optimization and anytime competitive constraint satisfaction governed by the parameters λ and b. When λ or b becomes larger, we can get a smaller regret because the anytime competitive constraints in (1) are relaxed to have more flexibility to optimize the average reward. In the extreme cases when λ or b is large enough, all the policies in Π can satisfy the anytime competitive constraints, so we can get zero regret.

Moreover, the regret bound shows that the update of allowed deviation by applying Proposition 4.1 based on the cost feedback at each round will benefit the reward optimization. By the definition of R_{h-1} in Corollary 4.2, if the real actions deviate more from the prior actions before h, the gain ΔG_i for $i \geq h$ can be smaller, so the actions must be closer to the prior actions in the subsequent rounds, potentially causing a larger regret. Thus, it is important to have a good planing of the action differences $\{d_i\}_{i=1}^{H}$ to get larger allowed action deviations for reward optimization. Exploiting the representation power of machine learning, `ACRL` can learn a good planning of the action differences, and the `ACD` policy π° corresponding to the optimal ML policy $\tilde{\pi}^*$ can achieve the optimal planing of the action differences.

Last but not least, Theorem 5.1 shows the effects of the systems parameters in Assumption 3.2 and Assumption 3.4 on the regret through $\Gamma_{h,h}$ defined in Proposition 4.1 and the minimum cost ϵ. Observing that $\Gamma_{h,h}$ increases with the systems parameters including the Lipschitz parameters $L_f, L_c, L_{\pi^\dagger}$ and telescoping parameters p, a higher estimation of the Lipschitz parameters and telescoping parameters can cause a higher regret. Also, a lower estimation of the minimum cost value can cause a higher regret. Therefore, although knowing the upper bound of the Lipschitz parameters and telescoping parameters and the lower bound of the minimum cost value is enough to guarantee the anytime competitive cost constraints by Proposition 4.1, a lower reward regret can be obtained with a more accurate estimation of these system parameters.

5.2 Regret of `ACRL`

To quantify the regret defined in Eqn. (2), it remains to bound the reward gap between the `ACD` policy π^k and the optimal `ACD` policy π°. In this section, we show that π^k by `ACRL` approaches the optimal one π° as episode $K \to \infty$ by bounding the pseudo regret

$$\mathrm{PReg}(K) = \mathbb{E}_{x_1}\left[\sum_{k=1}^{K}\left(V_1^{\pi^\circ}(s_1) - V_1^{\pi^k}(s_1)\right)\right]. \tag{13}$$

Theorem 5.2. *Assume that the value function is bounded by $\bar{V}$. Denote a set of function as*

$$\mathcal{Q} = \{q \mid \exists g \in \mathcal{G}, \forall (s,a,v) \in \mathcal{S}\times\mathcal{A}\times\mathcal{V}, q(s,a,v) = \mathbb{E}_{f\sim g}\left[v(s') \mid s,a\right]\}. \tag{14}$$

If $\beta_k = 2(\bar{V}H)^2 \log\left(\frac{2\mathcal{N}(\mathcal{Q},\alpha,\|\cdot\|_\infty)}{\delta}\right) + C\bar{V}H$ with $\alpha = 1/(KH\log(KH/\delta))$, C being a constant, and $\mathcal{N}(\mathcal{Q},\alpha,\|\cdot\|_\infty)$ being the covering number of $\mathcal{Q}$, with probability at least $1-\delta$, the pseudo regret of Algorithm 2 is bounded as

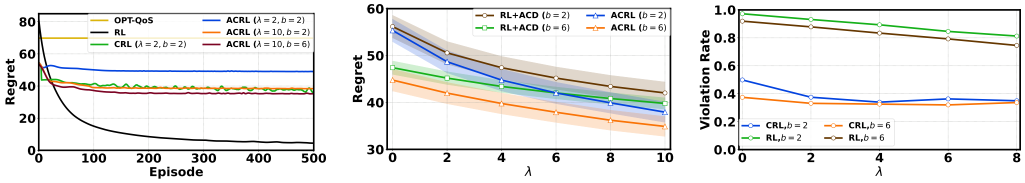
$$\mathrm{PReg}(K) \leq 1 + d_{\mathcal{Q}} H\bar{V} + 4\sqrt{d_{\mathcal{Q}}\beta_K KH} + H\sqrt{2KH\log(1/\delta)}, \tag{15}$$

where $d_{\mathcal{Q}} = \dim_E(\mathcal{Q}, \frac{1}{KH})$ is the Eluder dimension of $\mathcal{Q}$ defined in [50].

Theorem 5.2 bounds the pseudo regret for each episode k. The confidence parameter β_k to balance the exploration and exploitation is chosen to get the pseudo regret bound as shown in Theorem 5.2. A higher β_k is chosen to encourage the exploration if the covering number of the function space $\mathcal{Q}$, the episode length, or the maximum value becomes larger. Also, the pseudo regret relies on the size of the function space $\mathcal{Q}$ through $d_{\mathcal{Q}}$ and β_K. With smaller λ or b, less actions satisfy Corollary 4.2 given a state, and so a smaller state-action space $\mathcal{S}\times\mathcal{A}$ is obtained, which results in a smaller size of the function space $\mathcal{Q}$ and thus a smaller regret.

To get more insights, we also present the overall regret bound when the transition model g can be represented by a linear kernel as in [6, 70], i.e. $g(f) = \langle\phi(f),\theta\rangle$ with dimension of θ as d_θ, the reward regret in Eqn.2 is bounded as

$$\mathrm{Regret}(K) \leq K\mathbb{E}_{y_{1:H}}\left\{\sum_{h=1}^{H} L_{Q,h}\left[\eta - \frac{1}{\Gamma_{h,h}}(\lambda\epsilon + b + \Delta G_h)\right]^+\right\} + \tilde{O}(\sqrt{H^3\bar{V}^2K\log(1/\delta)}), \tag{16}$$



(a) Regret per episode (b) Regret w.r.t. λ (c) Violation rate w.r.t. λ

Figure 1: Regret and cost violation rate of different algorithms. Shadows in Figure 2(b) show the range of the regret.

where $L_{Q,h}$, η, $\Gamma_{h,h}$, and ΔG_h are all defined in Theorem 5.1. The overall regret bound is obtained because under the assumption of linear transition kernel, we have $\beta_K = O((\bar{V}H)^2 \log(\frac{1}{\delta}\mathcal{N}(\mathcal{Q}, \alpha, \|\cdot\|_\infty))) = \tilde{O}((\bar{V}H)^2(d_\theta + \log(1/\delta)))$ [6], and the Eluder dimension is $d_{\mathcal{Q}} = \tilde{O}(d_\theta)$ [50]. Thus the pseudo regret is $\mathrm{PReg}(K) = \tilde{O}(\sqrt{H^3\bar{V}^2 K \log(1/\delta)})$ which is sublinear in terms of K. With the sublinear pseudo regret $\mathrm{PReg}(K)$, the ACD policy π^k performs as asymptotically well as the optimal ACD policy π° when $K \to \infty$. Combining with the regret of the optimal ACD policy in Theorem 5.1, we can bound the overall regret of ACRL. Since in the definition of regret, ACD policy is compared with the optimal-unconstrained policy π^*, the regret bound also includes an unavoidable linear term due to the commitment to satisfy the anytime competitive constraints. The linear term indicates the trade-off between the reward optimization and the anytime competitive constraint satisfaction.

6 Empirical Results

We experiment with the application of resource management for carbon-aware computing [49] to empirically show the benefits of ACRL. The aim of the problem is to jointly optimize carbon efficiency and revenue while guaranteeing the constraints on the quality-of-service (QoS). In this problem, there exists a policy prior $\pi^\dagger$ which directly optimizes QoS based on estimated models. In our experiment, we apply ACRL to optimize the expected reward and guarantee that the real QoS cost is no worse than that of the policy prior. The concrete settings can be found in Appendix A.

Figure 1(a) gives the regret changing the in first 500 episodes. Figure 1(b) shows the regret with different λ and b, demonstrating the trade-off between reward optimization and the satisfaction of anytime competitive constraints. Figure 1(c) shows the probability of the violation of the anytime competitive constraints by RL and constrained RL. ACRL and ML models with ACD have no violation of anytime competitive constraints. More analysis about the results are provided in Appendix A due to space limitations.

7 Concluding Remarks

This paper considers a novel MDP setting called A-CMDP where the goal is to optimize the average reward while guaranteeing the anytime competitive constraints which require the cost of a learned policy never exceed that of a policy prior $\pi^\dagger$ for any round h in any episode. To guarantee the anytime competitive constraints, we design ACD, which projects the output of an ML policy into a safe action set at each round. Then, we formulate the decision process of ACD as a new MDP and propose a model-based RL algorithm ACRL to optimize the average reward under the anytime competitive constraints. Our performance analysis shows the tradeoff between the reward optimization and the satisfaction of the anytime competitive constraints.

Future directions. Our results are based on the assumptions on the Lipschitz continuity of the cost, dynamic functions, and the policy prior, as well as the telescoping properties of the policy prior, which are also supposed or verified in other literature [5, 28, 60, 40]. In addition, to guarantee the anytime competitive constraints, the agent is assumed to have access to the Lipschitz constants, the minimum cost value, and the perturbation function. However, since the anytime competitive constraints are much stricter than the expected constraints or the constraints with a high probability, there is no way to guarantee them without any knowledge of the key properties of a mission-critical system. Our work presents the first policy design to solve A-CMDP, but it would be interesting to design anytime-competitive policies with milder assumptions in the future.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments. Jianyi Yang, Pengfei Li and Shaolei Ren were supported in part by the U.S. NSF under the grant CNS–1910208. Tongxin Li was partially supported by the NSFC grant No. 72301234, the Guangdong Key Lab of Mathematical Foundations for Artificial Intelligence, and the start-up funding UDF01002773 of CUHK-Shenzhen. Adam Wierman was supported in part by the U.S. NSF under grants CNS–2146814, CPS–2136197, CNS–2106403, NGSDI–2105648.

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

[2] Sanae Amani, Christos Thrampoulidis, and Lin Yang. Safe reinforcement learning with linear function approximation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 243–253. PMLR, 18–24 Jul 2021.

[3] Sanae Amani, Christos Thrampoulidis, and Lin Yang. Safe reinforcement learning with linear function approximation. In *International Conference on Machine Learning*, pages 243–253. PMLR, 2021.

[4] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.

[5] Kavosh Asadi, Dipendra Misra, and Michael Littman. Lipschitz continuity in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 264–273. PMLR, 2018.

[6] Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning*, pages 463–474. PMLR, 2020.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[8] Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

[9] Duncan S Callaway, Meredith Fowlie, and Gavin McCormick. Location, location, location: The variable value of renewable energy and demand-side efficiency resources. *Journal of the Association of Environmental and Resource Economists*, 5(1):39–75, 2018.

[10] Agustin Castellano, Hancheng Min, Juan Bazerque, and Enrique Mallada. Reinforcement learning with almost sure constraints. In *Learning for Dynamics and Control*, 2022.

[11] Bingqing Chen, Priya L Donti, Kyri Baker, J Zico Kolter, and Mario Bergés. Enforcing policy feasibility constraints through differentiable projection for energy optimization. In *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, pages 199–210, 2021.

[12] Shuang Chen, Christina Delimitrou, and José F Martínez. Parties: Qos-aware resource partitioning for multiple interactive services. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 107–120, 2019.

[13] Weiqin Chen, Dharmashankar Subramanian, and Santiago Paternain. Policy gradients for probabilistic constrained reinforcement learning, 2022.

[14] Xiaoyu Chen, Xiangming Zhu, Yufeng Zheng, Pushi Zhang, Li Zhao, Wenxue Cheng, Peng Cheng, Yongqiang Xiong, Tao Qin, Jianyu Chen, et al. An adaptive deep rl method for non-stationary environments with piecewise stable context. *Advances in Neural Information Processing Systems*, 35:35449–35461, 2022.

[15] Xin Chen, Guannan Qu, Yujie Tang, Steven Low, and Na Li. Reinforcement learning for decision-making and control in power systems: Tutorial, review, and vision. *arXiv*, 2021.

[16] Yuri Chervonyi, Praneet Dutta, Piotr Trochim, Octavian Voicu, Cosmin Paduraru, Crystal Qian, Emre Karagozler, Jared Quincy Davis, Richard Chippendale, Gautam Bajaj, et al. Semi-analytical industrial cooling system model for reinforcement learning. *arXiv preprint arXiv:2207.13131*, 2022.

[17] Nicolas Christianson, Junxuan Shen, and Adam Wierman. Optimal robustness-consistency tradeoffs for learning-augmented metrical task systems. In *AI STATS*, 2023.

[18] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167, 2017.

[19] Google DeepMind. Safety-first ai for autonomous data centre cooling and industrial control. https://www.deepmind.com/blog/safety-first-ai-for-autonomous-data-centre-cooling-and-industrial-control, 2018.

[20] Christina Delimitrou and Christos Kozyrakis. Paragon: Qos-aware scheduling for heterogeneous datacenters. *ACM SIGPLAN Notices*, 48(4):77–88, 2013.

[21] Dongsheng Ding, Xiaohan Wei, Zhuoran Yang, Zhaoran Wang, and Mihailo Jovanovic. Provably efficient safe exploration via primal-dual policy optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 3304–3312. PMLR, 2021.

[22] Dongsheng Ding, Kaiqing Zhang, Tamer Basar, and Mihailo Jovanovic. Natural policy gradient primal-dual method for constrained markov decision processes. *Advances in Neural Information Processing Systems*, 33:8378–8390, 2020.

[23] Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A Smith, Nicole DeCario, and Will Buchanan. Measuring the carbon intensity of ai in cloud instances. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1877–1894, 2022.

[24] Priya L Donti, David Rolnick, and J Zico Kolter. Dc3: A learning method for optimization with hard constraints. *arXiv preprint arXiv:2104.12225*, 2021.

[25] Yihan Du, Siwei Wang, and Longbo Huang. A one-size-fits-all solution to conservative bandit problems. In *AAAI*, 2021.

[26] Yonathan Efroni, Shie Mannor, and Matteo Pirotta. Exploration-exploitation in constrained mdps. *arXiv preprint arXiv:2003.02189*, 2020.

[27] Evrard Garcelon, Mohammad Ghavamzadeh, Alessandro Lazaric, and Matteo Pirotta. Conservative exploration in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1431–1441. PMLR, 2020.

[28] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pages 2170–2179. PMLR, 2019.

[29] Arnob Ghosh, Xingyu Zhou, and Ness Shroff. Provably efficient model-free constrained rl with linear function approximation. *arXiv preprint arXiv:2206.11889*, 2022.

[30] Íñigo Goiri, Kien Le, Md E Haque, Ryan Beauchea, Thu D Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. Greenslot: scheduling energy consumption in green datacenters. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11, 2011.

[31] Sumeet Katariya, Branislav Kveton, Zheng Wen, and Vamsi K Potluru. Conservative exploration using interleaving. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 954–963. PMLR, 2019.

[32] Imran Khan. Temporal carbon intensity analysis: renewable versus fossil fuel dominated electricity systems. *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, 41(3):309–323, 2019.

[33] Mo Li, Timothy M Smith, Yi Yang, and Elizabeth J Wilson. Marginal emission factors considering renewables: A case study of the us midcontinent independent system operator (miso) system. *Environmental science & technology*, 51(19):11215–11223, 2017.

[34] Pengfei Li, Jianyi Yang, and Shaolei Ren. Expert-calibrated learning for online optimization with switching costs. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(2), Jun 2022.

[35] Pengfei Li, Jianyi Yang, and Shaolei Ren. Robustified learning for online optimization with memory costs. *INDOCOM*, 2023.

[36] Tongxin Li, Yue Chen, Bo Sun, Adam Wierman, and Steven H Low. Information aggregation for constrained online control. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5(2):1–35, 2021.

[37] Tongxin Li, Ruixiao Yang, Guannan Qu, Yiheng Lin, Adam Wierman, and Steven H Low. Certifying black-box policies with stability for nonlinear control. *IEEE Open Journal of Control Systems*, 2:49–62, 2023.

[38] Tongxin Li, Ruixiao Yang, Guannan Qu, Guanya Shi, Chenkai Yu, Adam Wierman, and Steven Low. Robustness and consistency in linear quadratic control with untrusted predictions. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(1), feb 2022.

[39] Yingying Li, Subhro Das, Jeff Shamma, and Na Li. Safe adaptive learning-based control for constrained linear quadratic regulators with regret guarantees. *arXiv preprint arXiv:2111.00411*, 2021.

[40] Yingying Li, James A Preiss, Na Li, Yiheng Lin, Adam Wierman, and Jeff Shamma. Online switching control with stability and regret guarantees. *arXiv preprint arXiv:2301.08445*, 2023.

[41] Enming Liang, Minghua Chen, and Steven H. Low. Low complexity homeomorphic projection to ensure neural-network solution feasibility for optimization over (non-)convex set. In *ICML*, 2023.

[42] Yiheng Lin, Yang Hu, Guannan Qu, Tongxin Li, and Adam Wierman. Bounded-regret mpc via perturbation analysis: Prediction error, constraints, and nonlinearity. *arXiv preprint arXiv:2210.12312*, 2022.

[43] Yiheng Lin, Yang Hu, Guanya Shi, Haoyuan Sun, Guannan Qu, and Adam Wierman. Perturbation-based regret analysis of predictive control in linear time varying systems. *Advances in Neural Information Processing Systems*, 34:5174–5185, 2021.

[44] Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. *arXiv preprint arXiv:2211.02001*, 2022.

[45] Fan-Ming Luo, Tian Xu, Hang Lai, Xiong-Hui Chen, Weinan Zhang, and Yang Yu. A survey on model-based reinforcement learning. *arXiv preprint arXiv:2206.09328*, 2022.

[46] Jerry Luo, Cosmin Paduraru, Octavian Voicu, Yuri Chervonyi, Scott Munns, Jerry Li, Crystal Qian, Praneet Dutta, Jared Quincy Davis, Ningjia Wu, et al. Controlling commercial cooling systems using reinforcement learning. *arXiv preprint arXiv:2211.07357*, 2022.

[47] California Independent System Operator. Calfornia renewable datasets. `https://www.caiso.com/Pages/default.aspx`, 2023.

[48] Ian Osband and Benjamin Van Roy. Model-based reinforcement learning and the eluder dimension. *Advances in Neural Information Processing Systems*, 27, 2014.

[49] Ana Radovanović, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, et al. Carbon-aware computing for datacenters. *IEEE Transactions on Power Systems*, 38(2):1270–1280, 2022.

[50] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[51] Daan Rutten, Nico Christianson, Debankur Mukherjee, and Adam Wierman. Online optimization with untrusted predictions. *arXiv preprint arXiv:2202.03519*, 2022.

[52] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

[53] Mohammad Shahrad, Cristian Klein, Liang Zheng, Mung Chiang, Erik Elmroth, and David Wentzlaff. Incentivizing self-capping to increase cloud utilization. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 52–65, 2017.

[54] Yuanyuan Shi, Guannan Qu, Steven Low, Anima Anandkumar, and Adam Wierman. Stability constrained reinforcement learning for real-time voltage control. In *2022 American Control Conference (ACC)*, pages 2715–2721. IEEE, 2022.

[55] Youngbin Song, Minjun Park, Minhwan Seo, and Sang Woo Kim. Improved soc estimation of lithium-ion batteries with novel soc-ocv curve estimation method using equivalent circuit model. In *2019 4th International Conference on Smart and Sustainable Technologies (SpliTech)*, pages 1–6. IEEE, 2019.

[56] Aivar Sootla, Alexander I Cowen-Rivers, Taher Jafferjee, Ziyan Wang, David H Mguni, Jun Wang, and Haitham Ammar. Sauté rl: Almost surely safe reinforcement learning using state augmentation. In *International Conference on Machine Learning*, pages 20423–20443. PMLR, 2022.

[57] Bo Sun, Ali Zeynali, Tongxin Li, Mohammad Hajiesmaili, Adam Wierman, and Danny HK Tsang. Competitive algorithms for the online multiple knapsack problem with application to electric vehicle charging. *ACM on Measurement and Analysis of Computing Systems (POMACS)*, 4(3), 2021.

[58] Guy Tennenholtz, Nadav Merlis, Lior Shani, Martin Mladenov, and Craig Boutilier. Reinforcement learning with history-dependent dynamic contexts. *ICML*, 2023.

[59] Garrett Thomas. Markov decision processes. 2007.

[60] Hiroyasu Tsukamoto, Soon-Jo Chung, and Jean-Jaques E Slotine. Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview. *Annual Reviews in Control*, 52:135–169, 2021.

[61] Sharan Vaswani, Lin Yang, and Csaba Szepesvári. Near-optimal sample complexity bounds for constrained mdps. *Advances in Neural Information Processing Systems*, 35:3110–3122, 2022.

[62] Honghao Wei, Xin Liu, and Lei Ying. Triple-q: A model-free algorithm for constrained reinforcement learning with sublinear regret and zero constraint violation. In *International Conference on Artificial Intelligence and Statistics*, pages 3274–3307. PMLR, 2022.

[63] William Wong, Praneet Dutta, Octavian Voicu, Yuri Chervonyi, Cosmin Paduraru, and Jerry Luo. Optimizing industrial hvac systems with hierarchical reinforcement learning. *arXiv preprint arXiv:2209.08112*, 2022.

[64] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems*, 4:795–813, 2022.

[65] Yifan Wu, Roshan Shariff, Tor Lattimore, and Csaba Szepesvári. Conservative bandits. In *International Conference on Machine Learning*, pages 1254–1262. PMLR, 2016.

[66] Yi Xiong, Ningyuan Chen, Xuefeng Gao, and Xiang Zhou. Sublinear regret for learning pomdps. *Production and Operations Management*, 31(9):3491–3504, 2022.

[67] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. Projection-based constrained policy optimization. In *International Conference on Learning Representations*, 2020.

[68] Yunchang Yang, Tianhao Wu, Han Zhong, Evrard Garcelon, Matteo Pirotta, Alessandro Lazaric, Liwei Wang, and Simon Shaolei Du. A reduction-based framework for conservative bandits and reinforcement learning. In *International Conference on Learning Representations*, 2022.

[69] Runyu Zhang, Yingying Li, and Na Li. On the regret analysis of online lqr control with predictions. In *2021 American Control Conference (ACC)*, pages 697–703. IEEE, 2021.

[70] Dongruo Zhou, Jiafan He, and Quanquan Gu. Provably efficient reinforcement learning for discounted mdps with feature mapping. In *International Conference on Machine Learning*, pages 12793–12802. PMLR, 2021.

[71] Xingyu Zhou, Ness Shroff, and Adam Wierman. Asymptotically optimal load balancing in large-scale heterogeneous systems with multiple dispatchers. *ACM SIGMETRICS Performance Evaluation Review*, 48(3):57–58, 2021.

[72] Xingyu Zhou, Jian Tan, and Ness Shroff. Flexible load balancing with multi-dimensional state-space collapse: Throughput and heavy-traffic delay optimality. *ACM SIGMETRICS Performance Evaluation Review*, 46(3):10–11, 2019.

[73] Xinyang Zhou, Masoud Farivar, Zhiyuan Liu, Lijun Chen, and Steven H Low. Reverse and forward engineering of local voltage control in distribution networks. *IEEE Transactions on Automatic Control*, 66(3):1116–1128, 2020.

Anytime-Competitive Reinforcement Learning with Policy Prior – Supplementary Material

We provide the empirical results and proofs of our theorems in the appendix.

A Empirical Results - Carbon-Aware Resource Management

A.1 Problem Formulation

We consider the sustainable workload scheduling problem in datacenters to jointly optimize the carbon efficiency and the revenue while guaranteeing the quality-of-service (QoS). In this problem, the average carbon efficiency and revenue can be optimized while QoS must always be ensured at each step. The agent needs to decide the computing resource a_h measured by energy (kWh) for each round h. The state x_h is the remaining demand for each round h and is updated as

$$x_h = f(x_{h-1}, \mu_h, a_h) = [V_x(x_{h-1}) + \mu_h - V_a(a_h)]^+, \tag{17}$$

where V_x is a random function of x_{h-1} measuring the randomly decayed remaining demands (e.g., due to workload dropping), μ_h is the arrival demand at round h, and V_a is a random function in terms of a_h and outputs the amount of processed workload. With the random functions V_x and V_a, the remaining workload x_h at round h is drawn from $\mathbb{P}(x_h \mid x_{h-1}, \mu_h, a_h)$. Here, we focus on flexibly deferrable workloads (e.g., model training and batch data processing) [49].

The energy efficiency reward is modeled by a penalty for the carbon footprint at each round. Let C_h be the amount of renewable at round h, the energy efficiency reward is expressed as $\text{efficiency}_h = -([a_h - C_h]^+)^2$. The revenue function is modeled as a general power-law function [53] as $\text{revenue}_h = C_r V_a^{\alpha}(a_h)$ with $\alpha \in (0, 1)$. In datacenters, we also need to consider a switching cost $\gamma_2 \|a_h - a_{h-1}\|^2$ at each round h to avoid switching on/off servers frequently. Thus, the reward in this problem is formulated as

$$\text{reward}_h = \text{efficiency}_h + \gamma_1 \cdot \text{revenue}_h - \gamma_2 \cdot \|a_h - a_{h-1}\|^2. \tag{18}$$

Besides the reward, QoS is also crucial for deferrable workloads in datacenters. In this work, we model QoS as a cost function of the remaining demand as follows:

$$cost_{\text{QoS},h} = x_h^\top Q_1 x_h + Q_2^\top x_h + Q_3, \tag{19}$$

where Q_1, Q_2 and Q_3 are constants.

As shown in Eqn. (20), given a baseline $\pi^\dagger$ that has been verified to achieve a satisfactory QoS, our goal is to optimize the expected reward and guarantee the QoS for any time in any sequence, i.e.

$$\begin{aligned} &\max_{\pi \in \Pi} \quad \mathbb{E}\left[\sum_{h=1}^{H} \text{reward}_h\right], \\ &s.t. \quad \sum_{h=1}^{h'} cost_{\text{QoS},h}(\pi) \leq (1+\lambda) \sum_{h=1}^{h'} cost_{\text{QoS},h}(\pi^\dagger) + h'b, \quad \forall h' \in [H], \end{aligned} \tag{20}$$

which is consistent with the definition of anytime competitive constraints in Definition 3.1.

A.2 Baselines

In the experiments, we consider different baselines as below.

• QoS Optimization (`OPT-QoS`): This baseline policy prior directly optimizes QoS in (19) based on estimated models $\hat{V}_x$ and $\hat{V}_a$. Without taking efficiency or revenue into consideration, `OPT-QoS` essentially always schedules as many computing resources as possible to lower the QoS cost based on the estimated arrival demand.

• Reinforcement Learning (`RL`): This is a model-based reinforcement learning algorithm to optimize the expected reward $\mathbb{E}\left[\sum_{h=1}^{H} \text{reward}_h\right]$ without considering any QoS constraints.

• Constrained Reinforcement Learning (CRL): This is a constrained reinforcement learning to optimize the reward with the expected QoS cost constraint as shown below:

$$\max_{\pi\in\Pi} \quad \mathbb{E}\left[\sum_{h=1}^{H} \text{reward}_h\right], \quad s.t. \quad \mathbb{E}\left[\sum_{h=1}^{H} cost_{\text{QoS},h}(\pi) - (1+\lambda)\sum_{h=1}^{H} cost_{\text{QoS},h}(\pi^{\dagger})\right] \leq B. \quad (21)$$

• Random RL policy with ACD (Random +ACD): This algorithm selects actions by ACD in Algorithm 1 with a random ML model $\tilde{\pi}$ as the input of ACD.

• Trained RL policy with ACD (RL +ACD): This algorithm selects actions by ACD in Algorithm 1 with the RL policy trained to optimize the expected reward without accounting for QoS.

• Anytime-Competitive Reinforcement Learning (ACRL): This is the proposed Algorithm 2 which optimizes the expected reward while guaranteeing the anytime competitive QoS cost constraints in (20). In each inference, ACD in Algorithm 1 is used to select actions.

A.3 Experiment Settings

In the experiments, we evaluate the performances with the following experiment settings.

System parameters. We evaluate the regret and the cost constraints for different choices of parameters. The results are given for different anytime competitive constraint parameters including λ chosen from $[0, 10]$ and b chosen from $\{2, 6\}$. With smaller λ and b, we have more stringent constraints, and vice versa. In the experiments, we choose $\alpha = 0.5$ for the revenue function to simulate a typical effect of the scheduled resource on the revenue. To scale different rewards into the same magnitude, we choose the weight for the revenue as $\gamma_1 = 4$, and the weight for the switching cost as $\gamma_2 = 1$. For the QoS cost function, we choose $Q_1 = Q_2 = Q_3 = 1$, so we have the minimum QoS cost as $\epsilon = Q_3 = 1$. The transition model f are from a function space defined by random functions V_x and V_a. To create the environment for RL, $V_x(x_h)$ is drawn from a uniform distribution with range $[0.9 \cdot x_h, x_h]$, and $V_a(a_h)$ is drawn from a normal distribution with $0.8 \cdot a_h$ as the center.

Data. For experiments, we create an environment based on a renewable dataset and a demand dataset. The renewable dataset is a public dataset from California Independent System Operator [47] which contains the hourly renewable generation in 2019. The renewable sequences from multiple sources (solar, wind, water) are summed together and scaled to be the values of $\{C_h\}_{h=1}^{H}$ in the problem formulation. In addition, we use the Azure Cloud Dataset [18] as the demand dataset which includes hourly CPU utilization in the same year of 2019. We choose the sequences of the first three months and augment them to 4000 episodes for policy exploration, and we hold out the sequences of the last two months for testing.

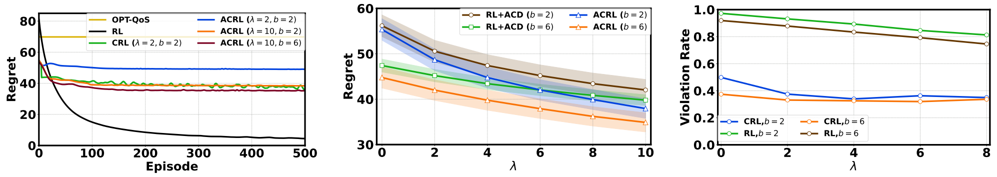
Learning settings. To ensure fair comparisons, we choose the same neural network architecture as the policy network for different methods. The policy neural network has two hidden layers and each hidden layer has 40 neurons. For training, the policy network parameters are initialized by Gaussian distribution. The reinforcement learning has total $K = 4000$ episodes. We update the neural network every 50 episodes with a weight update rate of 10^{-3}. We apply Adam optimizer to update the weights of neural networks.

A.4 Results

We show the empirical results for both regret and QoS cost and discuss the insights from these results.

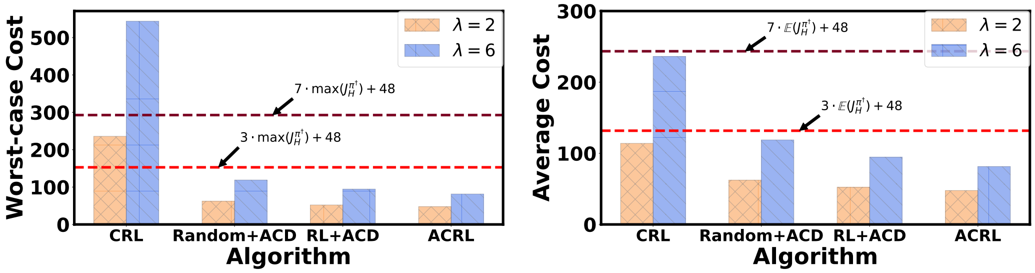
Regret evaluation. The reward regrets as defined in Eqn. (2) are given in Figure 2. To evaluate the regret, we use the RL policy after the exploration for total 4000 episodes as the optimal RL policy π^*. The results are given for different anytime competitiveness parameters λ and b.

Figure 2(a) shows the varying regret of different algorithms for the first 500 episodes. Without including reward as an objective, the policy prior OPT-QoS is an algorithm that is not updated over time and always gives the highest regret. Without the QoS cost constraints, RL approaches the optimal RL policy that can give the best regret as time goes on. The constrained RL (CRL) and anytime-competitive RL (ACRL) are guaranteed to satisfy the expected constraint in (21) and the anytime competitive constraints in (20), respectively, so their reward regrets are higher than RL. Also, we can find that with larger λ and/or b, ACRL can achieve lower regret after about 200 episodes. This



(a) Regret per episode (b) Regret w.r.t. λ (c) Violation rate w.r.t. λ

Figure 2: Regret and cost violation rate of different algorithms. Figure 2(a) gives the regret changing with episodes. Figure 2(b) shows the regret with different λ and b after exploration for all the 4000 episodes. Shadows in Figure 2(b) show the range of regret. Figure 2(c) shows the probability of the violation of the anytime competitive constraints. Figure 2(c) shows the probability of the violation of the anytime competitive constraints.



(a) Worst-case cost (b) Average cost

Figure 3: QoS costs of different algorithms. Figure 3(a) and Figure 3(b) give the worst-case costs and the average cost for different algorithms under $b = 2$, $\lambda = 2$ and $\lambda = 6$, respectively. For `OPT-QoS` $\pi^\dagger$, the worst-case cost $\max(J_H^{\pi^\dagger})$ and average cost $\mathbb{E}(J_H^{\pi^\dagger})$ are 34.95 and 27.92, respectively.

is because the larger λ or b gives less stringent anytime competitive constraints and leaves more flexibility to reduce the regret as shown in Theorem 5.1. Moreover, we can observe that `ACRL` with smaller λ and b (e.g. $\lambda = 2, b = 2$ in the figure) converges faster to the optimal policy under the anytime competitive constraints. The reason is that the constraints with smaller λ and b provide a smaller policy space to explore, resulting in a smaller Eluder dimension $d_{\mathcal{Q}}$ shown in Theorem 5.2.

In Figure 2(b), we give the optimal regret of the algorithms after enough exploration. `ACRL` and `RL` `+ACD` are different in terms of the RL policy $\tilde{\pi}$ used in Algorithm 1. `Random` `+ACD` uses randomly initialized RL policy as $\tilde{\pi}$ and has a regret as large as from 61 to 64.51, exceeding the limit of the regret axis. The optimal regrets of different algorithms decrease with the parameters λ and b, which is consistent with the findings in Theorem 5.1. Importantly, we can find that under the same λ and b, `ACRL` can always improve the regret of `RL` `+ACD`. This is because `RL` explores the original environment defined in Section 3.1 while `ACRL` explores the environment of the new MDP defined at the beginning of Section 4.2, which is the true environment created by `ACD`. This highlights the advantage of `ACRL` in terms of reducing the reward regret by learning with the awareness of the anytime constraints in the true environment.

Cost evaluation. In Figure 2(c), we show the violation probability of the anytime competitive constraints for RL and CRL. Thanks to the theoretical guarantee of the anytime competitive constraints, the algorithms based on `ACD` (`ACRL`, RL+`ACD`) all have zero violation probability, so they are not shown in the figure. Since RL only optimizes the expected reward, the probability of cost constraint violation becomes higher when the competitive constraint becomes more stringent (smaller λ or b). Although CRL considers the expected cost constraint and achieves a lower violation rate, the violation rate is still not zero since there is no theoretical guarantee of the anytime competitiveness. The violation rate of anytime competitive constraints is not allowed for mission-critical applications, showing the need of an RL algorithm with anytime competitiveness guarantee like `ACRL`.

We evaluate the QoS costs in Figure 3 to verify that the constraints are satisfied.

Figure 3(a) gives the worst-case cost of different algorithms for all the sequences in the testing dataset. The RL algorithm without considering the cost objective achieves the worst-case cost of 5015.62, exceeding the range of the cost axis to a large extent. The dotted horizon lines show the maximum cost bound required by the anytime competitive constraint in Definition 3.1. Clearly, we can find that given any RL policy (even a randomly initialized RL policy) as an input, ACD can guarantee the anytime competitive constraints even for the worst-case, but CRL fails to guarantee the anytime competitive constraints. In the experiments, we verify that all ACD algorithms have zero violation of anytime competitive constraints no matter what ML policy is used, but CRL has a violation rate of 27.5% for $\lambda = 2$ and a violation rate of 58.6% for $\lambda = 6$.

Figure 3(b) shows the average cost of different algorithms for all the sequences in the testing dataset. The RL algorithm achieves an average QoS cost of 2070.14, exceeding the range of the cost axis to a large extent. The dotted lines give the maximum average QoS cost bound required by the expected constraint in (21). Since CRL is designed to optimize the regret subject to the average QoS constraints in (21), it has no violation in terms of the average QoS cost. The algorithms that use ACD guarantees the stricter anytime competitive constraints than the average constraint, so they can also guarantee the average QoS constraint.

B Empirical Results - Sustainable AI Inference

B.1 Problem Formulation

AI tasks are widely deployed on edge datacenters. The renewables are utilized in edge datacenters to reduce the carbon emissions from AI inference. The renewable sources are known for their time-varying and unstable nature. Multiple AI models are often available for a given AI inference service [7]. This provides a flexible balance between accuracy and energy consumption. Thus, the agent needs to decide the model size at each round to optimize the inference performance with a constrained amount of carbon emission. [49, 52].

Specifically, the edge datacenter utilizes a battery to store the renewables and unused energy. The state x_h is the battery state of charge (SoC) at each round h. Given an action a_h (the energy consumed by selected models) and renewable e_h at round h, the battery SoC is updated as

$$x_h = f(x_{h-1}, e_h, a_h) = [x_{h-1} + V_e(e_h) - V_a(a_h)]^+, \tag{22}$$

where V_e and V_a are random functions with the randomness coming from the charging and discharging rates of different batteries under different temperatures [55]. Since recycled batteries may be used in edge datacenters for sustainability, the charging and discharging functions $V_e(\cdot)$ and $V_a(\cdot)$ are generally unknown given a new problem instance. If at some round h, the consumption is larger than the sum of the renewable replenishment and remaining battery energy, i.e. $x_{h-1} + V_e(e_h) < V_a(a_h)$, the fossil energy is used and a cost that penalizes the carbon emission is formulated as

$$cost_{\text{carbon},h} = Q_1 \left\| [V_a(a_h) - x_{h-1} - V_e(e_h)]^+ \right\|^2, \tag{23}$$

where Q_1 is a constant.

The reward includes the demand satisfaction revenue and inference performance. With more energy, more demand is satisfied, so we directly penalize the demand that is not served at each round as $\text{reward}_{d,h} = -C_d \cdot ([\mu_h - a_h]^+)^2$ given an AI inference workload μ_h. The inference performance is dependent on the action and is modeled by the log utility to capture the diminishing return, i.e. $\text{reward}_{i,h} = \log(1 + C_i * a_h)$. The total reward for round h is represented as

$$\text{reward}_h = \text{reward}_{d,h} + \gamma_2 \cdot \text{reward}_{i,h} - \gamma_1 \cdot \|a_h - a_{h-1}\|^2 - \gamma_3 \cdot cost_{\text{carbon},h}. \tag{24}$$

Same as the previous problems, given a policy prior $\pi^\dagger$ that balances the minimization of carbon emission and the demand satisfaction, the goal is

$$\begin{aligned} &\max_{\pi \in \Pi} \quad \mathbb{E}\left[\sum_{h=1}^{H} \text{reward}_h\right], \\ &s.t. \quad \sum_{h=1}^{h'} cost_{\text{carbon},h}(\pi) \le (1+\lambda) \sum_{h=1}^{h'} cost_{\text{carbon},h}(\pi^\dagger) + h'b, \quad \forall h' \in [H], \end{aligned} \tag{25}$$

B.2 Settings and Baselines

In the experiments, we consider different baselines as below.

• Carbon Bound (`Carbon-B`): This baseline policy prior makes decisions as below. When the estimated SoC $x_{h-1}+\hat{V}_e(x_{h-1},e_h)+Q_c$ based on an estimated charging function $\hat{V}_e$ and a slackness Q_c is equal to or higher than the demand μ_h. The action is set as μ_h to meet the demand. Otherwise, the action is selected to bound the estimated carbon by a positive value Q_c based on an estimated discharging function, i.e. solving $Q_1\left\|[\hat{V}_a(a_h)-x_{h-1}-\hat{V}_e(e_h)]^+\right\|^2=Q_c$.

• Reinforcement Learning (`RL`): This is a model-based reinforcement learning algorithm to optimize the expected reward $\mathbb{E}\left[\sum_{h=1}^{H}\text{reward}_h\right]$ without considering any cost constraints.

• Constrained Reinforcement Learning (`CRL`): This is a constrained reinforcement learning to optimize the reward with the expected carbon cost constraint as shown below:

$$\max_{\pi\in\Pi}\quad\mathbb{E}\left[\sum_{h=1}^{H}\text{reward}_h\right],\quad s.t.\quad\mathbb{E}\left[\sum_{h=1}^{H}cost_{\text{carbon},h}(\pi)-(1+\lambda)\sum_{h=1}^{H}cost_{\text{carbon},h}(\pi^\dagger)\right]\leq B.\tag{26}$$

• Anytime-Competitive Reinforcement Learning (`ACRL`): This is the proposed Algorithm 2 which optimizes the expected reward while guaranteeing the anytime competitive carbon cost constraints in (25). In each inference, `ACD` in Algorithm 1 is used to select actions.

We evaluate the methods in the following experiment environments.

System parameters. We evaluate the performance for different choices of parameters. The results are given for different anytime competitive constraint parameters including λ chosen from $\{5,7\}$ and b chosen as 6 which controls how stringent the cost constraints are. In the experiments, we convert demand satisfaction reward, the inference performance, the carbon costs and the switching costs into monetary values through parameters $\gamma_1=0.5$, $\gamma_2=2$, and $\gamma_3=0.1$. We choose $Q_1=1$ in the carbon cost function. To create the environment for RL, $V_a(a_h)$ is drawn from a uniform distribution with range $[0.7\cdot a_h,a_h]$, and $V_e(e_h)$ is drawn from a uniform distribution with range $[0.8\cdot e_h,e_h]$.

Data. The inference demand μ_t comes from the GPU power usage of a large language model [44]. We still use California Independent System Operator [47] dataset to simulate the renewable replenishment for edge data center. We choose the sequences of the first three months and augment them to 2160 episodes for policy training, and we hold out 1440 sequences for testing.

Learning settings. The neural network architecture is designed as below. The policy neural network has two hidden layers and each hidden layer has 50 neurons. For training, the policy network parameters are initialized by Gaussian distribution. The reinforcement learning has total $K=2160$ episodes. We update the neural network every 50 episodes with a weight update rate of 10^{-4}. We apply Adam optimizer to update the weights of neural networks.

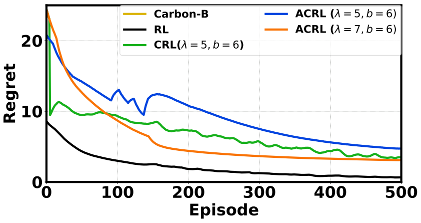


Figure 4: Regrets of different algorithms. The regret of `Carbon-B` is 51.417 and is out of the range of y axis.

B.3 Regret Evaluation

We give the dynamic reward regret for different algorithms within first 500 episodes in Figure 4. We can find that the regrets of all algorithms decrease as time goes. RL, which directly optimizes the optimal reward without accounting the cost constraint satisfaction, can achieve the lowest reward regret. Both CRL and `ACRL` consider the competitive cost constraints and achieve a larger regret than RL as is indicated by Theorem 5.2. Among them, CRL guarantees the competitive cost constraints in expectation, so it achieves a lower regret than `ACRL` which guarantees the anytime competitive cost constraint with the same parameters $\lambda=5$ and $b=6$. However, `ACRL` can theoretically guarantee the anytime competitive constraint, which makes it more suitable for the sustainable AI inference where strict requirements for carbon emission exist.

Moreover, when λ increases to 7, we can find that the regret of ACRL is reduced since the anytime competitive constraint is less stringent and there is more flexibility to optimize the expected reward, which demonstrates the trade-off between reward optimization and competitiveness satisfaction given by Theorem 5.1.

C Proof of Theorems in Section 4

C.1 Proof of Proposition 4.1

Proposition 4.1 *Suppose that Assumption 3.2 and 3.4 are satisfied. At round h with costs $\{c_i\}_{i=1}^{h-1}$ observed, the anytime competitive constraints $J^{\pi}_{h'} \le (1+\lambda)J^{\pi^\dagger}_{h'} + h'b$ for rounds $h' = h, \cdots, H$ are satisfied if for all subsequent rounds $h' = h, \cdots, H$,*

$$\sum_{j=h}^{h'} \Gamma_{j,j}\|a_j - \pi^\dagger(x_j)\| \le G_{h,h'}, \; \forall h' = h, \cdots, H,$$

where $\Gamma_{j,n} = \sum_{i=n}^{H} q_{j,i}, (j \in [H], \forall n \ge j)$, with $q_{j,i} = L_c \mathbb{1}(j=i) + L_c(1+L_{\pi^\dagger})L_f p(i-1-j)\mathbb{1}(j<i), (\forall j \in [H], i \ge j)$, relying on known parameters, and $G_{h,h'}$ is called the allowed deviation which is expressed as

$$G_{h,h'} = \sum_{i=1}^{h-1}\Big((1+\lambda)\hat{c}_i^\dagger - c_i - \Gamma_{i,h} d_i\Big) + (h'-h+1)(\lambda\epsilon + b),$$

where $\hat{c}_i^\dagger = \max\Big\{\epsilon, c_i - \sum_{j=1}^{i} q_{j,i} d_j\Big\}, (\forall i \in [H])$, is the lower bound of of $c_i^\dagger$, and $d_j = \|a_j - \pi^\dagger(x_j)\|, \forall j \in [H]$ is the action difference at round j. □

Proof. First, we bound the state perturbation. As is shown in Figure 5, denote $x_h^{\dagger(i)}$ is the state by applying the policy prior $\pi^{\dagger(i)}$ from round i, so the state difference at round h is expressed as

$$\begin{aligned}
\|x_h - x_h^\dagger\| &= \|\sum_{i=1}^{h-1}(x_h^{\dagger(i+1)} - x_h^{\dagger(i)})\| \\
&\le \sum_{i=1}^{h-1}\|(x_h^{\dagger(i+1)} - x_h^{\dagger(i)})\| \\
&\le \sum_{i=1}^{h-1} p(h-1-i)\|(x_{i+1} - x_{i+1}^{\dagger(i)})\| \\
&\le L_f \sum_{i=1}^{h-1} p(h-1-i)\|a_i - \pi^\dagger(x_i)\|,
\end{aligned} \tag{27}$$

where the first inequality holds by triangle inequality, and the second inequality holds by the telescoping property of $\pi^\dagger$.

Then, we bound the gap between the expert cost and true cost at round h as

$$\begin{aligned}
&|c_h(x_h, a_h) - c_h(x_h^\dagger, a_h^\dagger)| \\
=&c_h(x_h, a_h) - c_h(x_h, \pi^\dagger(x_h)) + c_h(x_h, \pi^\dagger(x_h)) - c_h(x_h^\dagger, \pi^\dagger(x_h^\dagger)) \\
\le&L_c\|a_h - \pi^\dagger(x_h)\| + L_c(1+L_{\pi^\dagger})\|x_h - x_h^\dagger\| \\
\le&L_c\|a_h - \pi^\dagger(x_h)\| + L_c(1+L_{\pi^\dagger})L_f \sum_{j=1}^{h-1} p(h-1-j)\|a_j - \pi^\dagger(x_j)\| \\
=&\sum_{j=1}^{h} q_{j,h}\|a_j - \pi^\dagger(x_j)\|,
\end{aligned} \tag{28}$$

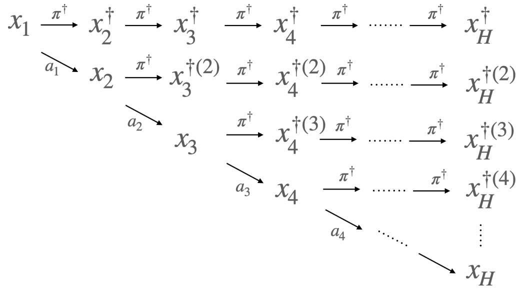


Figure 5: Illustration of state perturbation

where the first inequality holds by the Lipschitz continuity of cost functions c_h and policy $\pi^\dagger$, the second inequality holds by (27), and $q_{j,h} = L_c\mathbb{1}(j=h) + L_c(1+L_{\pi^\dagger})L_f p(h-1-j)\mathbb{1}(j<h)$.

Recall that the anytime competitive constraint for any round $h' \in [H]$ is $\sum_{i=1}^{h'} c_i(x_i, \pi(x_i)) \leq (1+\lambda)\sum_{i=1}^{h'} c_i(x_i^\dagger, \pi^\dagger(x_i^\dagger)) + h'b$ which is equivalent to $\sum_{i=1}^{h'}\left(c_i(x_i,\pi(x_i)) - c_i(x_i^\dagger, \pi^\dagger(x_i^\dagger))\right) \leq \lambda\sum_{i=1}^{h'} c_i(x_i^\dagger, \pi^\dagger(x_i^\dagger)) + h'b$. Based on the cost difference bound in (28) and cost assumption in 3.2, we can get a sufficient condition of the anytime competitive constraint as

$$\sum_{i=1}^{h'}\sum_{j=1}^{i} q_{j,h}\|a_j - \pi^\dagger(x_j)\| \leq h'(\lambda\epsilon + b). \tag{29}$$

Since $\sum_{i=1}^{h'}\sum_{j=1}^{i} q_{j,h}\|a_j - \pi^\dagger(x_j)\| \leq \sum_{j=1}^{h'}\Gamma_{j,j}\|a_j - \pi^\dagger(x_j)\|$, this proves Proposition 4.1 for round $h=1$.

With the cost feedback $\{c_i\}_{i=1}^{h-1}$ collected at round h, we can get a lower bound of the prior cost. Based on the cost difference bound in (28) and cost assumption in 3.2 for $i = 1, \cdots, h-1$ as

$$c_i(x_i^\dagger, \pi^\dagger(x_i^\dagger)) \geq \hat{c}_i^\dagger = \max\left\{\epsilon, c_i(x_i, a_i) - \sum_{j=1}^{i} q_{j,i} d_j,\right\} \tag{30}$$

where $d_j = \|a_j - \pi^\dagger(x_j)\|, \forall j \in [H]$. Thus, at round h, with true cost feedback from the first round to the $(h-1)$-th round, we get the sufficient condition for the anytime competitive constraint of any $h' \geq h$ as

$$\sum_{i=1}^{h-1} c_i(x_i,a_i) + \sum_{i=h}^{h'}\sum_{j=1}^{i} q_{i,j}\|a_j - \pi^\dagger(x_j)\| \leq (1+\lambda)\sum_{i=1}^{h-1}\max\left\{\epsilon, c_i - \sum_{j=1}^{i} q_{j,i} d_j\right\} + (h'-h+1)(\lambda\epsilon+b). \tag{31}$$

Recognizing that

$$\begin{aligned}\sum_{i=h}^{h'}\sum_{j=1}^{i} q_{i,j}\|a_j - \pi^\dagger(x_j)\| &= \sum_{j=1}^{h-1}\sum_{i=h}^{h'} q_{j,i}\|a_j - \pi^\dagger(x_j)\| + \sum_{j=h}^{h'}\sum_{i=j}^{h'} q_{j,i}\|a_j - \pi^\dagger(x_j)\| \\ &\leq \sum_{j=1}^{h-1}\Gamma_{j,h}\|a_j - \pi^\dagger(x_j)\| + \sum_{j=h}^{h'}\Gamma_{j,j}\|a_j - \pi^\dagger(x_j)\|\end{aligned} \tag{32}$$

Thus, at round h, a sufficient condition for the anytime competitive constraint at round h' can be calculated as

$$\sum_{j=h}^{h'}\Gamma_{j,j}\|a_j - \pi^\dagger(x_j)\| \leq \sum_{i=1}^{h-1}\left((1+\lambda)\hat{c}_i^\dagger - c_i - \Gamma_{i,h}d_i\right) + (h'-h+1)(\lambda\epsilon+b), \tag{33}$$

which proves Proposition 4.1. □

C.2 Proof of Corollary 4.2

Corollary 4.2. *At round 1, we initialize the allowed deviation as $D_1 = \lambda\epsilon + b$. At round $h, h > 1$, the allowed deviation is updated as*

$$D_h = \max\{D_{h-1} + \lambda\epsilon + b - \Gamma_{h-1,h-1}d_{h-1},\ R_{h-1} + \lambda\epsilon + b\}$$

where $R_{h-1} = \sum_{i=1}^{h-1}\left((1+\lambda)\hat{c}_i^\dagger - c_i - \Gamma_{i,h}d_i\right)$ with notations defined in Proposition 4.1. The $(\lambda, b)-$anytime competitiveness in Definition 3.1 are satisfied if it holds at each round h that $\Gamma_{h,h}\|a_h - \pi^\dagger(x_h)\| \le D_h$.

Proof. We prove by induction. At the first round, by Proposition 4.1, the sufficient condition for the anytime competitive constraint at round $h' \ge 1$ is

$$\sum_{j=1}^{h'}\Gamma_{j,j}\|a_j - \pi^\dagger(x_j)\| \le D_1 + (h'-1)(\lambda\epsilon + b),\ \forall h' = 1, \cdots, H. \tag{34}$$

Thus, with $D_1 = \lambda\epsilon + b$, $\Gamma_{1,1}\|a_1 - \pi^\dagger(x_1)\| \le D_1$ satisfies (34) for $h' = 1$.

Assume that at round $h-1$, the sufficient condition for anytime competitive constraint at round $h' \ge h-1$ is

$$\sum_{j=h-1}^{h'}\Gamma_{j,j}\|a_j - \pi^\dagger(x_j)\| \le D_{h-1} + (h'-h+1)(\lambda\epsilon + b),\ \forall h' = h-1, \cdots, H. \tag{35}$$

Thus, at round h, a sufficient condition for anytime competitive constraint at round $h' \ge h$ is

$$\sum_{j=h}^{h'}\Gamma_{j,j}\|a_j - \pi^\dagger(x_j)\| \le D_{h-1} + (h'-h+1)(\lambda\epsilon + b) - \Gamma_{h-1,h-1}d_{h-1},\ \forall h' = h, \cdots, H. \tag{36}$$

Also, by applying Proposition 4.1 for round h, we get another sufficient condition for anytime competitive constraint at round $h' \ge h$ as

$$\sum_{j=h}^{h'}\Gamma_{j,j}\|a_j - \pi^\dagger(x_j)\| \le R_{h-1} + (h'-h+1)(\lambda\epsilon + b),\ \forall h' = h, \cdots, H. \tag{37}$$

Choose the maximum bound in the two sufficient conditions, we get the sufficient condition for anytime competitive constraint at round $h' \ge h$ as

$$\sum_{j=h}^{h'}\Gamma_{j,j}\|a_j - \pi^\dagger(x_j)\| \le D_h + (h'-h)(\lambda\epsilon + b),\ \forall h' = h, \cdots, H. \tag{38}$$

Therefore, for any $h \in [H]$, the sufficient condition for anytime constraint at round $h' \ge h$ is (38). By choosing a_h such that $\Gamma_{h,h}\|a_h - \pi^\dagger(x_h)\| \le D_h$ at each round, (38) is satisfied for round $h' = h \in [H]$ and so the anytime competitive constraints are satisfied for all rounds. □

D Proofs in Section 5

D.1 Proof of Theorem 5.1

Theorem 5.1. *Assume that the optimal-unconstrained policy π^* has a value function $Q_h^{\pi^*}(x, a)$ which is $L_{Q,h}$-Lipschitz continuous with respect to the action a for all x. The regret between the optimal ACD policy π° that satisfies $(\lambda, b)-$anytime competitiveness and the optimal-unconstrained policy π^* is bounded as*

$$\mathbb{E}_{x_1}\left[V_1^{\pi^*}(x_1) - V_1^{\pi^\circ}(x_1)\right] \le \mathbb{E}_{y_{1:H}}\left\{\sum_{h=1}^{H}L_{Q,h}\left[\eta - \frac{1}{\Gamma_{h,h}}(\lambda\epsilon + b + \Delta G_h)\right]^+\right\}, \tag{39}$$

where $\eta = \sup_{x\in\mathcal{X}} \|\pi^*(x) - \pi^+(x))\|$ is the maximum action discrepancy between the policy prior $\pi^\dagger$ and optimal-unconstrained policy π^*; $\Gamma_{h,h}$ is defined in Proposition 4.1; $\Delta G_h = [R_{h-1}]^+$ is the gain of the allowed deviation by applying Proposition 4.1 at round h.

We first define a projected policy based on the optimal-unconstrained policy $\pi^*(s) = \pi^*(x)$ and augmented state s as

$$\pi_h^\perp(s) = \arg\min_{a\in\mathcal{A}_h(D_h)} \|a - \pi_h^*(s)\|. \tag{40}$$

In the next Lemma, we decompose the concerned regret based on the newly-defined policy $\pi_h^\perp$.

Lemma D.1. *If $Q_h^{\pi^*}$ is $L_{Q,h}-$ Lipschitz continuous with respect to the action, then the regret between π° and π^* can be bounded as*

$$\mathbb{E}_{x_1}\left[V_1^{\pi^\circ}(s_1) - V_1^{\pi^*}(s_1)\right] \leq \mathbb{E}_{y_{1:H}}\left[\sum_{h=1}^{H} L_{Q,h}\|\pi_h^\perp(s_h^\perp) - \pi_h^*(s_h^\perp)\|\right]. \tag{41}$$

Proof. Let $\xi_h(s) = Q_h^{\pi^*}(x, \pi_h^\perp(s)) - Q_h^{\pi^*}(x, \pi_h^*(s))$. For any round h and any augmented state s obtained by ACD policy, we can bound the value difference as

$$\begin{aligned}
V_h^{\pi^\circ}(s) - V_h^{\pi^*}(s) &= \tilde{Q}_h^{\tilde{\pi}^*}(s, \tilde{\pi}^*(s)) - Q_h^{\pi^*}(s, \pi^*(s)) \\
&\leq \tilde{Q}_h^{\tilde{\pi}^*}(s, \pi^\perp(s)) - Q_h^{\pi^*}(s, \pi^*(s)) \\
&= Q_h^{\pi^\circ}(s, \pi^\perp(s)) - Q_h^{\pi^*}(s, \pi^\perp(s)) + Q_h^{\pi^*}(s, \pi^\perp(s)) - Q_h^{\pi^*}(s, \pi^*(s)) \\
&= \mathbb{E}_{s_{h+1}}\left[V_{h+1}^{\pi^\circ}(s_{h+1}) - V_{h+1}^{\pi^*}(s_{h+1}) \mid s, \pi^\perp(s)\right] + \xi_h(s),
\end{aligned} \tag{42}$$

where the first equality holds since π° is the ACD policy based on ML policy $\tilde{\pi}^*(s)$, the inequality holds since $\tilde{\pi}^*$ optimizes $\tilde{Q}_h^{\tilde{\pi}^*}$, the third equality holds since π° is the optimal ACD policy with $\tilde{\pi}^*$ as the ML model, and the last equality holds by the definition of Q_h^π.

Iteratively applying (42), we get

$$\begin{aligned}
&V_1^{\pi^\circ}(s_1) - V_1^{\pi^*}(s_1) \\
\leq &\left(\prod_{h=1}^{H} \mathbb{E}_{s_{h+1}|s_h,\pi^\perp(s_h)}\right)\left[V_{H+1}^{\pi^\circ}(s_{H+1}) - V_{H+1}^{\pi^*}(s_{H+1})\right] + \sum_{h=1}^{H}\left(\prod_{i=1}^{h-1} \mathbb{E}_{s_{i+1}|s_i,\pi^\perp(s_i)}\right)[\eta_h(s_h)] \\
=&\mathbb{E}_{y_{1:H}}\left[\sum_{h=1}^{H} \eta_h(s_h^\perp)\right] \leq \mathbb{E}_{y_{1:H}}\left[\sum_{h=1}^{H} L_{Q,h}\|\pi_h^\perp(s_h^\perp) - \pi_h^*(s_h^\perp)\|\right],
\end{aligned} \tag{43}$$

where $s_h^\perp$ is the state generated by policy $\pi^\perp$, the last equality holds since $V_{H+1}^\pi = 0$, and the last inequality holds by the Lipschitz continuity of $Q_h^{\pi^*}$. □

Lemma D.2. *Given any s_h generated by policy $\pi^\perp$, the action difference between $\pi^\perp$ and π^* is bounded as*

$$\|\pi_h^\perp(s_h) - \pi_h^*(s_h)\| \leq \left[\eta - \frac{1}{\Gamma_{h,h}}\left(\lambda\epsilon + b + \Delta G_h\right)\right]^+, \tag{44}$$

where $\Delta G_h = [R_{h-1}]^+ \geq 0$.

Proof. Since $\pi_h^\perp$ is the projection of π_h^* into the action norm ball $\mathcal{A}_h(D_h) = \left\{a \mid \Gamma_{h,h}\|a - \pi^\dagger(x_h)\| \leq D_h\right\}$, we have

$$\begin{aligned}
\|\pi_h^\perp(s_h) - \pi_h^*(s_h)\| &= \left[\|\pi^\dagger(x_h) - \pi_h^*(x_h)\| - \frac{D_h}{\Gamma_{h,h}}\right]^+ \\
&\leq \left[\eta - \frac{D_h}{\Gamma_{h,h}}\right]^+,
\end{aligned} \tag{45}$$

where the last inequality holds by the definition of η.

Since $D_h \geq \Gamma_{h,h} d_h$ for any $h \in [H]$, we have $D_h \geq \lambda\epsilon + b + [R_{h-1}]^+ = \lambda\epsilon + b + \Delta G$. Thus completed the proof. □

Proof of Theorem 5.1

Proof. By Lemma D.1 and Lemma D.2, we have

$$\begin{aligned}\mathbb{E}_{x_1}\left[V_1^{\pi^*}(x_1) - V_1^{\pi^\circ}(x_1)\right] &= \mathbb{E}_{x_1}\left[V_1^{\pi^*}(s_1) - V_1^{\pi^\circ}(s_1)\right] \\ &\le \mathbb{E}_{y_{1:H}}\left[\sum_{h=1}^{H} L_{Q,h}\|\pi_h^\perp(s_h^\perp) - \pi_h^*(s_h^\perp)\|\right] \\ &\le \mathbb{E}_{y_{1:H}}\left[\sum_{h=1}^{H} L_{Q,h}\left[\eta - \frac{1}{\Gamma_{h,h}}\left(\lambda\epsilon + b + \Delta G_h\right)\right]^+\right],\end{aligned} \tag{46}$$

where $\Delta G_h = [R_{h-1}]^+$. Thus completes the proof. □

D.2 Proof of Theorem 5.2

Theorem 5.2. *Assume that the value function is bounded by $\bar{V}$. Denote a set of function as*

$$\mathcal{Q} = \{q \mid \exists g \in \mathcal{G}, \forall (s,a,v) \in \mathcal{S}\times\mathcal{A}\times\mathcal{V}, q(s,a,v) = \mathbb{E}_{f\sim g}\left[v(s') \mid s,a\right]\}. \tag{47}$$

If $\beta_k = 2(\bar{V}H)^2 \log\left(\frac{2\mathcal{N}(\mathcal{Q},\alpha,\|\cdot\|_\infty)}{\delta}\right) + C\bar{V}H$ with $\alpha = 1/(KH\log(KH/\delta))$, C being a constant, and $\mathcal{N}(\mathcal{Q},\alpha,\|\cdot\|_\infty)$ being the covering number of $\mathcal{Q}$, with probability at least $1-\delta$, the pseudo regret of Algorithm 2 is bounded as

$$\mathrm{PReg}(K) \le 1 + d_{\mathcal{Q}}H\bar{V} + 4\sqrt{d_{\mathcal{Q}}\beta_K KH} + H\sqrt{2KH\log(1/\delta)}, \tag{48}$$

where $d_{\mathcal{Q}} = \dim_E(\mathcal{Q}, \frac{1}{KH})$ is the Eluder dimension of $\mathcal{Q}$ defined in [50].

Lemma D.3. *Assume that $g \in \mathcal{G}_k$, the difference between the reward of the optimal `ACD` policy π° and the reward of policy π^k is bounded as*

$$V_1^{\pi^\circ}(s_1) - V_1^{\pi^k}(s_1) \le \sup_{\tilde{g}\in\mathcal{G}_t} \sum_{h=1}^{H-1} \mathbb{E}_{\tilde{g}-g}\left[\tilde{V}_{h+1}^k(s_{h+1}^k) \mid s_h^k, a_h^k\right] + \sum_{h=1}^{H-1}\xi_{h+1,k}, \tag{49}$$

where $\xi_{h+1,t} = \mathbb{E}_g\left[\tilde{V}_{h+1}^k(s_{h+1}) - V_{h+1}^{\pi^k}(s_{h+1})\right] - \left[\tilde{V}_{h+1}^k(s_{h+1}) - V_{h+1}^{\pi^k}(s_{h+1})\right]$.

Proof. Since the true transition model $g \in \mathcal{G}_k$, we have $V_1^{\pi^\circ}(s_1^k) \le \tilde{V}_1^k(s_1^k)$, and so

$$V_1^{\pi^\circ}(s_1^k) - V_1^{\pi^k}(s_1^k) \le \tilde{V}_1^k(s_1^k) - V_1^{\pi^k}(s_1^k). \tag{50}$$

At round h, we have

$$\begin{aligned}&\tilde{V}_h^k(s_h^k) - V_h^{\pi^k}(s_h^k) \\ =&\left(r(s_h^k,a_h^k) + \mathbb{E}_{g^k}\left[\tilde{V}_{h+1}^k(s_{h+1}) \mid s_h^k, a_h^k\right]\right) - \left(r(s_h^k,a_h^k) + \mathbb{E}_g\left[V_{h+1}^{\pi^k}(s_{h+1}) \mid s_h^k, a_h^k\right]\right) \\ =&\sum_{f\in\mathcal{F}} \tilde{V}_{h+1}^k(f(x_h,a_h),D_{h+1})g^k(f) - \sum_{f\in\mathcal{F}} V_{h+1}^{\pi^k}(f(x_h,a_h),D_{h+1})g(f) \\ =&\sum_{f\in\mathcal{F}} \tilde{V}_{h+1}^k(f(x_h,a_h),D_{h+1})\left(g^k(f) - g(f)\right) \\ &+ \sum_{f\in\mathcal{F}}\left(\tilde{V}_{h+1}^k(f(x_h,a_h),D_{h+1}) - V_{h+1}^{\pi^k}(f(x_h,a_h),D_{h+1})\right)g(f) \\ =&\mathbb{E}_{g^k-g}\left[\tilde{V}_{h+1}^k(s_{h+1}^k) \mid s_h^k, a_h^k\right] + \mathbb{E}_g\left[\tilde{V}_{h+1}^k(s_{h+1}) - V_{h+1}^{\pi^k}(s_{h+1}) \mid s_h^k, a_h^k\right]\end{aligned} \tag{51}$$

Let $\xi_{h+1,t} = \mathbb{E}_g\left[\tilde{V}_{h+1}^k(s_{h+1}) - V_{h+1}^{\pi^k}(s_{h+1})\right] - \left[\tilde{V}_{h+1}^k(s_{h+1}) - V_{h+1}^{\pi^k}(s_{h+1})\right]$. By the fact that $V_{H+1} = 0$ and summing $\left[\tilde{V}_h^k(s_h^k) - V_h^{\pi^k}(s_h^k)\right] - \left[\tilde{V}_{h+1}^k(s_{h+1}) - V_{h+1}^{\pi^k}(s_{h+1})\right] =$

$\mathbb{E}_{g^k-g}\left[\tilde{V}_{h+1}^k(s_{h+1}^k) \mid s_h^k, a_h^k\right] + \xi_{h+1,t}$ from $h=1$ to H, we have

$$\begin{aligned}
&\tilde{V}_1^k(s_1^k) - V_1^{\pi^k}(s_1^k) \\
&= \sum_{h=1}^{H-1} \mathbb{E}_{g^k-g}\left[\tilde{V}_{h+1}^k(s_{h+1}^k) \mid s_h^k, a_h^k\right] + \sum_{h=1}^{H-1} \xi_{h+1,k} \\
&\leq \sup_{\tilde{g}\in\mathcal{G}_t} \sum_{h=1}^{H-1} \mathbb{E}_{\tilde{g}-g}\left[\tilde{V}_{h+1}^k(s_{h+1}^k) \mid s_h^k, a_h^k\right] + \sum_{h=1}^{H-1} \xi_{h+1,k}
\end{aligned} \tag{52}$$

□

Lemma D.4 ([6]). *Let* $\beta_k = 2H^2 \log\left(\frac{2\mathcal{N}(\mathcal{Q},\alpha,\|\cdot\|_\infty)}{\delta}\right) + 2H(kH-1)\alpha\left\{2 + \sqrt{\log(\frac{4kH(kH-1)}{\delta})}\right\}$ *with* $\alpha > 0$. *Then with probability* $1-\delta, \delta \in (0,1)$, *we have* $g \in \mathcal{G}_k$ *for any* $k \geq 1$.

Lemma D.5 ([50]). *Let* $d_{\mathcal{Q}} = \dim_E(\mathcal{Q}, \frac{1}{KH})$ *be the Eluder dimension of the function set* $\mathcal{Q}$ *defined in* (14). *When* $g \in \cap_{k\in[K]}\mathcal{G}_k$, *the cumulative value estimation error is bounded as*

$$\sup_{\tilde{g}\in\mathcal{G}_t} \sum_{h=1}^{H-1} \mathbb{E}_{\tilde{g}-g}\left[\tilde{V}_{h+1}^k(s_{h+1}^k) \mid s_h^k, a_h^k\right] \leq 1 + Hd_{\mathcal{Q}} + 4\sqrt{d_{\mathcal{Q}}\beta_K KH}. \tag{53}$$

Proof of Theorem 5.2

Proof. By choosing $\alpha = 1/(KH\log(KH/\delta))$ and the assumption that V is upper bounded by $\bar{V}$, the β_k in Lemma D.4 is expressed as $\beta_k = 2(\bar{V}H)^2 \log\left(\frac{2\mathcal{N}(\mathcal{Q},\alpha,\|\cdot\|_\infty)}{\delta}\right) + C\bar{V}H$.

Since $\xi_{2,1}, \cdots, \xi_{H,1}, \cdots, \xi_{2,K}, \cdots, \xi_{H,K}$ is a martingale sequence, with probability at $1-\delta$, the sampling error is bounded as

$$\sum_{t=1}^{T}\sum_{h=1}^{H} \xi_{h=1}^{H} \leq T\sqrt{2TH\log(1/\delta)}. \tag{54}$$

By Lemma D.5 and union bound, we have with probability with $1-\delta$,($\delta \in (0,1)$),

$$V_1^{\pi^\circ}(s_1) - V_1^{\pi^k}(s_1) \leq 1 + Hd_{\mathcal{Q}}\bar{V} + 4\sqrt{d_{\mathcal{Q}}\beta_K KH} + T\sqrt{2TH\log(2/\delta)}. \tag{55}$$

□