OPT2022: 14th Annual Workshop on Optimization for Machine Learning

Understanding Curriculum Learning in Policy Optimization for Online Combinatorial Optimization

Runlong Zhou VECTORZH@CS.WASHINGTON.EDU
University of Washington, Seattle

Yuandong Tian YUANDONG@FB.COM
Meta AI

Yi Wu JXWUYI@GMAIL.COM
Tsinghua University

Simon S. Du SSDU@CS.WASHINGTON.EDU
University of Washington, Seattle

Abstract

Over the recent years, reinforcement learning (RL) starts to show promising results in tackling combinatorial optimization (CO) problems, in particular when coupled with curriculum learning to facilitate training. Despite emerging empirical evidence, theoretical study on why RL helps is still at its early stage. This paper presents the first systematic study on policy optimization methods for online CO problems. We show that online CO problems can be naturally formulated as latent Markov Decision Processes (LMDPs), and prove convergence bounds on natural policy gradient (NPG) for solving LMDPs. Furthermore, our theory explains the benefit of curriculum learning: it can find a strong sampling policy and reduce the distribution shift, a critical quantity that governs the convergence rate in our theorem. For a canonical online CO problem, Secretary Problem, we formally prove that distribution shift is reduced *exponentially* with curriculum learning *even if the curriculum is randomly generated*. Our theory also shows we can simplify the curriculum learning scheme used in prior work from multi-step to single-step. Lastly, we provide extensive experiments on Secretary Problem and Online Knapsack to verify our findings.

1. Introduction

In recent years, machine learning has shown promising results in solving combinatorial optimization (CO) problems, including traveling salesman problem (TSP, Kool et al. [13]), maximum cut [11] and satisfiability problem [17]. While in the worst case some CO problems are NP-hard, in practice, the probability that we need to solve the worst-case problem instance is low [6].

A significant subclass of CO problems is called online CO problems, which has gained much attention [8–10]. Online CO problems entail a sequential decision-making process, which perfectly matches the nature of reinforcement learning (RL).

This paper concerns using RL to tackle online CO problems. RL is often coupled with specialized techniques including (a particular type of) Curriculum Learning [12], human feedback and correction (Pérez-Dattari et al. [15], Scholten et al. [16]), and policy aggregation (boosting, Brukhim et al. [5]) to accelerate the training speed.

However, it is unclear when and why they improve the performance. We particularly focus on *RL with Curriculum Learning* (Bengio et al. [4], also named "bootstrapping" in Kong et al. [12]): train the agent from an easy task and *gradually* increase the difficulty until the target task.

Main contributions. In this paper, we initiate the formal study on using RL to tackle online CO problems, with a particular emphasis on understanding the specialized techniques developed in this emerging subarea. Our contributions are summarized below.

• **Formalization.** For online CO problems, we want to learn a single policy that enjoys good performance over *a distribution* of problem instances. This motivates us to use Latent Markov Decision Process (LMDP) [14] instead of standard MDP formulation. We give concrete examples, Secretary Problem (SP) and Online Knapsack, to show how LMDP models online CO problems.

• **Provable efficiency of policy optimization.** We analyze the performance of NPG for LMDP. The performance bound is characterized by the number of iterations, the excess risk of policy evaluation, the transfer error, and the relative condition number κ that characterizes the distribution shift between the sampling policy and the optimal policy.

• **Understanding and simplifying Curriculum Learning.** Using our performance guarantee on NPG for LMDP, we study when and why Curriculum Learning is beneficial to RL for online CO problems. Our main finding is that the main effect of Curriculum Learning is to give a stronger sampling policy. Under certain circumstances, Curriculum Learning reduces the relative condition number κ, improving the convergence rate. For the Secretary Problem, we provably show that Curriculum Learning can *exponentially reduce* κ compared with using the naïve sampling policy. Surprisingly, this means *even a random curriculum of SP accelerates the training exponentially*. As a direct implication, we show that the multi-step Curriculum Learning proposed in [12] can be significantly simplified into a single-step scheme. Lastly, to obtain a complete understanding, we study the failure mode of Curriculum Learning. To verify our theories, we conduct extensive experiments on two classical online CO problems (Secretary Problem and Online Knapsack).

2. Problem setup

In this section, we first introduce LMDP and why it naturally formulates online CO problems. The next are necessary components required by the algorithm, Natural Policy Gradient.

2.1. Latent Markov Decision Process

Latent MDP [14] is a collection of MDPs $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_M\}$. All the MDPs share state set $\mathcal{S}$, action set $\mathcal{A}$ and horizon H. Each MDP $\mathcal{M}_m = (\mathcal{S}, \mathcal{A}, H, \nu_m, P_m, r_m)$ has its own initial state distribution $\nu_m \in \Delta(\mathcal{S})$, transition $P_m : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ and reward $r_m : \mathcal{S} \times \mathcal{A} \to [0, 1]$, where $\Delta(\mathcal{S})$ is the probability simplex over $\mathcal{S}$. Let $w_1, w_2, \ldots, w_M$ be the mixing weights of MDPs such that $w_m > 0$ for any m and $\sum_{m=1}^{M} w_m = 1$. At the start of every episode, one MDP $\mathcal{M}_m \in \mathcal{M}$ is randomly chosen with probability w_m.

Due to the time and space complexities of finding the optimal history-dependent policies, we stay in line with Kong et al. [12] and care only about finding the optimal *history-independent* policy. Let $\Pi = \{\pi : \mathcal{S} \to \Delta(\mathcal{A})\}$ denote the class of all the history-independent policies.

Log-linear policy. Let $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ be a feature mapping function where d denotes the dimension of feature space. Assume that $\|\phi(s, a)\|_2 \le B$. A log-linear policy is of the form:

$$\pi_\theta(a|s) = \frac{\exp(\theta^\top \phi(s, a))}{\sum_{a' \in \mathcal{A}} \exp(\theta^\top \phi(s, a'))}, \text{ where } \theta \in \mathbb{R}^d.$$

Entropy regularized value function, Q-function and advantage function. We incorporate *entropy regularization* for completeness because prior works used it to facilitate training. The value function $V_{m,h}^{\pi,\lambda}(s)$ is defined as the sum of future λ-regularized rewards starting from s and executing π for h steps in M_m. The λ-regularized reward is defined as $r_m^{\pi,\lambda}(s,a) := r_m(s,a) + \lambda \ln \frac{1}{\pi(a|s)}$, which depends on the current policy π. Denote $V^{\pi,\lambda} := \sum_{m=1}^{M} w_m \sum_{s_0 \in \mathcal{S}} \nu_m(s_0) V_{m,H}^{\pi,\lambda}(s_0)$. Under regularization, we seek for $\pi_\lambda^\star = \arg\max_{\pi \in \Pi} V^{\pi,\lambda}$. Denote $V^{\star,\lambda} = V^{\pi_\lambda^\star,\lambda}$. For notational ease, we abuse $\pi^\star$ with $\pi_\lambda^\star$. The Q-function $Q_{m,h}^{\pi,\lambda}(s,a)$ and advantage function $A_{m,h}^{\pi,\lambda}(s,a) := Q_{m,h}^{\pi,\lambda}(s,a) - V_{m,h}^{\pi,\lambda}(s)$ are defined in a similar manner.

Modelling of online CO problems. Refer to Sec. A for the formulation of two online CO problems, Secretary Problem and Online Knapsack (decision version), using LMDPs.

2.2. Algorithm components

In this subsection we will introduce some necessary notions used by our main algorithm.

Definition 1 (State(-action) Visitation Distribution) *The state visitation distribution and state-action visitation distribution at step $h \geq 0$ with respect to π in $\mathcal{M}_m$ are defined as*

$$d_{m,h}^{\pi}(s) := \mathbb{P}(s_h = s \mid \mathcal{M}_m, \pi),$$

$$d_{m,h}^{\pi}(s,a) := \mathbb{P}(s_h = s, a_h = a \mid \mathcal{M}_m, \pi).$$

We define a grafted distribution $\widetilde{d}_{m,h}^{\pi}(s,a) := d_{m,h}^{\pi}(s) \circ \mathrm{Unif}_{\mathcal{A}}(a)$ which can be attained by first acting under π for h steps to get states then sample actions from the uniform distribution $\mathrm{Unif}_{\mathcal{A}}$.

Denote $d_{m,h}^{\clubsuit} := d_{m,h}^{\pi_\clubsuit}$ and $d^{\clubsuit}$ as short for $\{d_{m,h}^{\clubsuit}\}_{1 \leq m \leq M, 0 \leq h \leq H-1}$, here $\clubsuit$ can be any symbol.

In the following definitions, let v be the collection of any distribution, which will be instantiated by $d^\star$, d^t, etc. in the remaining sections.

Definition 2 (Compatible Function Approximation Loss) *Let g be the parameter update weight, then NPG is related to finding the minimizer for the following function:*

$$L(g;\theta,v) := \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a \sim v_{m,H-h}} \left[\left(A_{m,h}^{\pi_\theta,\lambda}(s,a) - g^\top \nabla_\theta \ln \pi_\theta(a|s) \right)^2 \right].$$

Definition 3 (Generic Fisher Information Matrix)

$$\Sigma_v^\theta := \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a \sim v_{m,H-h}} \left[\nabla_\theta \ln \pi_\theta(a|s) \left(\nabla_\theta \ln \pi_\theta(a|s) \right)^\top \right].$$

Particularly, denote $F(\theta) = \Sigma_{d^\theta}^\theta$ as the Fisher information matrix induced by π_θ.

3. Learning procedure

Natural Policy Gradient. Starting from θ_0, the algorithm updates the parameter by setting $\theta_{t+1} = \theta_t + \eta g_t$, where η is a predefined constant learning rate, and g_t is the update weight. Denote $\pi_t := \pi_{\theta_t}$, $V^{t,\lambda} := V^{\pi_t,\lambda}$ and $A_{m,h}^{t,\lambda} := A_{m,h}^{\pi_t,\lambda}$ for convenience. NPG satisfies $g_t \in \arg\min_g L(g;\theta_t, d^{\theta_t})$. When we only have samples, we use the approximate version of NPG: $g_t \approx \arg\min_{g \in \mathcal{G}} L(g;\theta_t, d^{\theta_t})$, where $\mathcal{G} = \{x : \|x\|_2 \leq G\}$ for some hyper-parameter G.

We also introduce a variant of NPG: instead of sampling from d^{θ_t} using the current policy π_t, we sample from $\widetilde{d}^{\pi_s}$ using a *fixed* sampling policy π_s. The update rule is $g_t \approx \arg\min_{g \in \mathcal{G}} L(g;\theta_t, \widetilde{d}^{\pi_s})$. This version makes a closed-form analysis for SP possible.

The main algorithm is shown in Alg. 1 (deferred to App. B). It admits two types of training: ① If $\pi_s = \text{None}$, it samples $s, a \sim d^{\theta_t}$; ② If $\pi_s \neq \text{None}$, it samples $s, a \sim \widetilde{d}^{\pi_s}$.

In both cases, we denote d^t as the sampling distribution and Σ_t as the induced Fisher Information Matrix used in step t, i.e. $d^t := d^{\theta_t}, \Sigma_t := F(\theta_t)$ if $\pi_s = \text{None}$; $d^t := \widetilde{d}^{\pi_s}, \Sigma_t := \Sigma^{\theta_t}_{\widetilde{d}^{\pi_s}}$ otherwise. The update rule can be written in a unified way as $g_t \approx \arg\min_{g \in \mathcal{G}} L(g; \theta_t, d^t)$. This is equivalent to solving a constrained quadratic optimization and we can use existing solvers.

Curriculum Learning. We use Curriculum Learning to facilitate training. Alg. 3 (deferred to App. B) is our proposed training framework, which first constructs an easy environment E' and trains a (near-)optimal policy π_s of it. In the target environment E, we either use π_s to sample data while training a new policy from scratch, or simply continue training π_s. To be specific and provide clarity for the results in Sec. 5, we name a few training modes (without regularization) here, and the rest are in Tab. 1.

`curl`, the standard Curriculum Learning, runs Alg. 3 with sampler always to be the current policy; `fix_samp_curl` stands for the fixed sampler Curriculum Learning, running Alg. 3 with π_s as the sampler. `direct` means directly learning in E without curriculum, i.e., running Alg. 1; `naive_samp` also directly learns in E, while using the naïve random policy to sample data.

4. Performance analysis

4.1. Natural Policy Gradient for Latent MDP

Let $g_t^\star \in \arg\min_{g \in \mathcal{G}} L(g; \theta_t, d^t)$ denote the true minimizer. We have the following definitions:

Definition 4 *Define for $0 \le t \le T$:*

- *(Excess risk)* $\epsilon_{\text{stat}} := \max_t \mathbb{E}[L(g_t; \theta_t, d^t) - L(g_t^\star; \theta_t, d^t)]$;
- *(Transfer error)* $\epsilon_{\text{bias}} := \max_t \mathbb{E}[L(g_t^\star; \theta_t, d^\star)]$;
- *(Relative condition number)* $\kappa := \max_t \mathbb{E}\left[\sup_{x \in \mathbb{R}^d} \frac{x^\top \Sigma^{\theta_t}_{d^\star} x}{x^\top \Sigma_t x}\right]$. *Note that term inside the expectation is a random quantity as θ_t is random.*

The expectation is with respect to the randomness in the sequence of weights $g_0, g_1, \ldots, g_T$.

ϵ_{stat} is due to that the minimizer g_t from samples may not minimize the population loss L. ϵ_{bias} quantifies the approximation error due to feature maps. κ characterizes the distribution mismatch between d^t and $d^\star$.

Here we introduce a fitting error which depicts the closeness between $\pi^\star$ and any policy π.

Definition 5 (Fitting Error) *Suppose the update rule is $\theta_{t+1} = \theta_t + \eta g_t$, define*

$$\text{err}_t := \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{(s,a) \sim d^\star_{m,H-h}} \left[A^{t,\lambda}_{m,h}(s,a) - g_t^\top \nabla_\theta \ln \pi_t(a|s) \right].$$

Thm. 6 shows the convergence rate of Alg. 1, and its proof is deferred to Sec. C.1.

Theorem 6 *With Def. 4, 5 and 8, our algorithm enjoys the following performance bound:*

$$\mathbb{E}\left[\min_{0 \le t \le T} V^{\star,\lambda} - V^{t,\lambda}\right] \le \frac{\lambda(1-\eta\lambda)^{T+1}\Phi(\pi_0)}{1-(1-\eta\lambda)^{T+1}} + \eta\frac{B^2G^2}{2} + \sqrt{H\epsilon_{\text{bias}}} + \sqrt{H\kappa\epsilon_{\text{stat}}},$$

where $\Phi(\pi_0)$ is the Lyapunov potential function which is only relevant to the initialization.

As a remark: If we can find a policy whose κ is small with a single curriculum, we do not need the multi-step curriculum learning procedure used in Kong et al. [12].

4.2. Curriculum learning for Secretary Problem

For SP, there exists a threshold policy that is optimal [3]. Suppose the threshold is $p \in (0, 1)$, then the policy is: accept candidate i if and only if $\frac{i}{n} > p$ and $X_i = 1$. For the classical SP where all the $n!$ instances have equal probability to be sampled, the optimal threshold is $1/\mathrm{e}$.

To show that curriculum learning makes the training converge faster, Thm. 6 gives a direct hint: *curriculum learning produces a good sampler leading to much smaller κ than that of a naïve random sampler.* Here we focus on the cases where $samp = \texttt{pi\_s}$ because the sampler is fixed, while when $samp = \texttt{pi\_t}$ it is impossible to analyze a dynamic procedure. We show Thm. 7 to characterize κ in SP. Its full statement and proof is deferred to Sec. C.2.

Theorem 7 *Assume that each candidate is independent of others and the i-th candidate has a probability P_i of being the best so far (Sec. 2.1). Assume the optimal policy is a p-threshold policy and the sampling policy is a q-threshold policy. There exists a policy parameterization such that:*

$$\kappa_{\text{curl}} = \Theta\left(\begin{cases} \prod_{j=\lfloor nq\rfloor+1}^{\lfloor np\rfloor} \frac{1}{1-P_j}, & q \le p, \\ 1, & q > p, \end{cases}\right), \quad \kappa_{\text{naïve}} = \Theta\left(2^{\lfloor np\rfloor} \max\left\{1, \max_{i\ge\lfloor np\rfloor+2} \prod_{j=\lfloor np\rfloor+1}^{i-1} 2(1-P_j)\right\}\right), \quad (1)$$

where κ_{curl} and $\kappa_{\text{naïve}}$ are κ of the sampling policy and the naïve random policy, respectively.

To understand how curriculum learning influences κ, we apply Thm. 7 to three concrete cases. They show that, when the state distribution induced by the optimal policy in the small problem is similar to that in the original large problem, then a single-step curriculum suffices.

The classical case: an exponential improvement. The probability series for the classical SP is $P_i = \frac{1}{i}$. Substituting them into Eq. 1 directly gives:

$$\kappa_{\text{curl}} = \begin{cases} \frac{\lfloor n/\mathrm{e}\rfloor}{\lfloor nq\rfloor}, & q \le \frac{1}{\mathrm{e}}, \\ 1, & q > \frac{1}{\mathrm{e}}, \end{cases} \qquad \kappa_{\text{naïve}} = 2^{n-1}\frac{\lfloor n/\mathrm{e}\rfloor}{n-1}.$$

Except for the corner case where $q < \frac{1}{n}$, we have that $\kappa_{\text{curl}} = O(n)$ while $\kappa_{\text{naïve}} = \Omega(2^n)$. Notice that any distribution with $P_i \le \frac{1}{i}$ leads to an exponential improvement.

A more general case. Consider the case where $P_i \le \frac{1}{2}$ for $i \ge 2$. Eq. 1 now becomes:

$$\kappa_{\text{curl}} \le \begin{cases} 2^{\lfloor np\rfloor-\lfloor nq\rfloor}, & q \le p, \\ 1, & q > p, \end{cases} \qquad \kappa_{\text{naïve}} \ge 2^{\lfloor np\rfloor}.$$

Clearly, $\kappa_{\text{curl}} \le \kappa_{\text{naïve}}$ always holds. When q is close to p, the difference is exponential in $\lfloor nq\rfloor$.

Failure mode of Curriculum Learning. From Eq. 1, $\kappa_{\text{naïve}} \le 2^{n-1}$. Similar as Sec. 3 of Beckmann [3], the optimal threshold p satisfies:

$$\sum_{i=\lfloor np\rfloor+2}^{n} \frac{P_i}{1-P_i} \le 1 < \sum_{i=\lfloor np\rfloor+1}^{n} \frac{P_i}{1-P_i}.$$

So letting $P_n > \frac{1}{2}$ results in $p \in [\frac{n-1}{n}, 1)$. Further, if $q < \frac{n-1}{n}$ and $P_j > 1 - 2^{-\frac{n}{n-\lfloor nq\rfloor-1}}$ for any $\lfloor nq\rfloor + 1 \le j \le n-1$, then from Eq. 1, $\kappa_{\text{curl}} > 2^n > \kappa_{\text{naïve}}$. This means that Curriculum Learning can always be manipulated adversarially. Sometimes there is hardly any reasonable curriculum.

5. Experiments

More formulation details and results are presented in Sec. D. In Curriculum Learning the entire training process splits into two phases. We call the training on curriculum (small scale instances) "warm-up phase" and the training on large scale instances "final phase". We ran more than one experiments for each problem. In one experiment there are more than one training processes to show

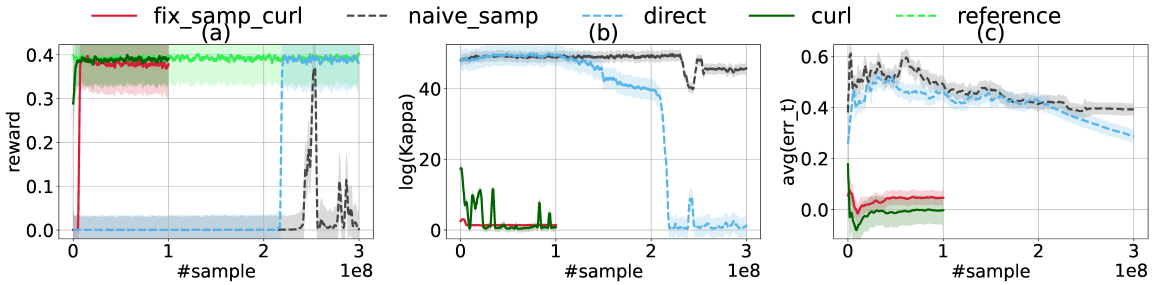

Figure 1: One experiment of SP. The x-axis is the number of trajectories, i.e., number of epsidoes $\times$ horizon $\times$ batch size. **Dashed lines represent only final phase training and solid lines represent Curriculum Learning.** The shadowed area shows the 95% confidence interval for the expectation. The explanation for different modes can be found in Sec. 3. The reference policy is the optimal threshold policy.

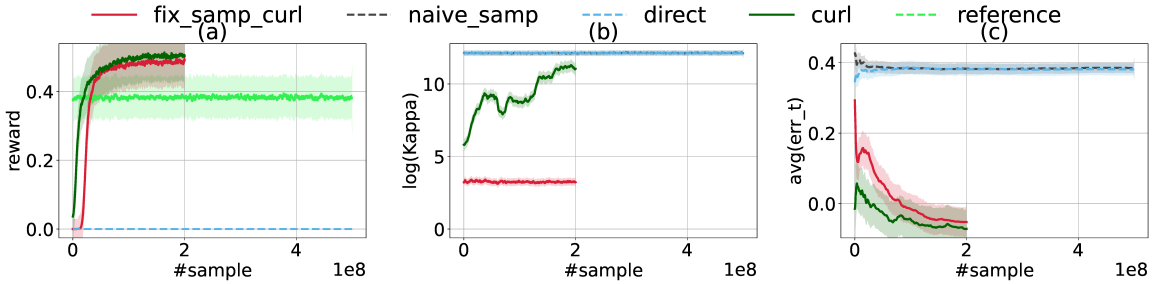

Figure 2: One experiment of OKD. Legend description is the same as that of Fig. 1. The reference policy is the bang-per-buck algorithm for Online Knapsack (Sec. 3.1 of Kong et al. [12].

the effect of different samplers and regularization coefficients. To highlight the effect of Curriculum Learning, we omit the results regarding regularization, and they can be found in supplementary files. All the trainings in the same experiment have *the same distributions* over LMDPs for final phase and warm-up phase (if any), respectively.

Secretary Problem. We show one of the four experiments in Fig. 1. Aside from reward and $\ln \kappa$, we plot the weighted average of err_t according to the proof of Thm. 6: $\text{avg}(\text{err}_t) = \frac{\sum_{i=0}^{t}(1-\eta\lambda)^{t-i}\,\text{err}_i}{\sum_{i'=0}^{t}(1-\eta\lambda)^{t-i'}}$. All the instance distributions are generated from parameterized series $\{P_n\}$ with fixed random seeds, which guarantees reproducibility and comparability. There is no explicit relation between the curriculum and the target environment, so the curriculum can be viewed as *random and independent*. The experiments clearly demonstrate that curriculum learning can boost the performance by a large margin and curriculum learning indeed dramatically reduces κ, even the curriculum is randomly generated.

Online Knapsack (decision version). We show one of the three experiments in Fig. 2. $\ln \kappa$ and $\text{avg}(\text{err}_t)$ are with respect to the reference policy, a bang-per-buck algorithm, which is not the optimal policy. Thus, they are only for reference. The curriculum generation is also parameterized, random and independent of the target environment. The experiments again demonstrate the effectiveness of curriculum learning and curriculum learning indeed dramatically reduces κ.

REFERENCES

[1] Alekh Agarwal, Sham M. Kakade, J. Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *J. Mach. Learn. Res.*, 22: 98:1–98:76, 2021.

[2] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert D. Kleinberg. A knapsack secretary problem with applications. In *APPROX-RANDOM*, 2007.

[3] M.J. Beckmann. Dynamic programming and the secretary problem. *Computers & Mathematics with Applications*, 19(11):25–28, 1990. ISSN 0898-1221. doi: https://doi.org/10.1016/0898-1221(90)90145-A. URL https://www.sciencedirect.com/science/article/pii/089812219090145A.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

[5] Nataly Brukhim, Elad Hazan, and Karan Singh. A boosting approach to reinforcement learning, 2021.

[6] Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4348–4355. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/595. URL https://doi.org/10.24963/ijcai.2021/595. Survey Track.

[7] Semih Cayci, Niao He, and R. Srikant. Linear convergence of entropy-regularized natural policy gradient with linear function approximation, 2021.

[8] Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. 2008.

[9] Martin Grötschel, Sven O Krumke, Jörg Rambau, Thomas Winter, and Uwe T Zimmermann. Combinatorial online optimization in real time. *Online optimization of large scale systems*, pages 679–704, 2001.

[10] Zhiyi Huang. Online combinatorial optimization problems with non-linear objectives. In *Nonlinear Combinatorial Optimization*, pages 179–205. Springer, 2019.

[11] Elias Boutros Khalil, Hanjun Dai, Yuyu Zhang, Bistra N. Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *NIPS*, 2017.

[12] Weiwei Kong, Christopher Liaw, Aranyak Mehta, and D. Sivakumar. A new dog learns old tricks: Rl finds classic optimization algorithms. In *ICLR*, 2019.

[13] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *ICLR*, 2019.

[14] Jeongyeol Kwon, Yonathan Efroni, Constantine Caramanis, and Shie Mannor. Rl for latent mdps: Regret guarantees and a lower bound. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24523–24534. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/cd755a6c6b699f3262bcc2aa46ab507e-Paper.pdf.

[15] Rodrigo Pérez-Dattari, Carlos Celemin, Javier Ruiz del Solar, and Jens Kober. Interactive learning with corrective feedback for policies based on deep neural networks. In *ISER*, 2018.

[16] Jan Scholten, Daan Wout, Carlos Celemin, and Jens Kober. Deep reinforcement learning with feedback-based exploration. *2019 IEEE 58th Conference on Decision and Control (CDC)*, Dec 2019. doi: 10.1109/cdc40024.2019.9029503. URL http://dx.doi.org/10.1109/CDC40024.2019.9029503.

[17] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT solver from single-bit supervision. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJMC_iA5tm.

Appendix A. Motivating Online Combinatorial Optimization problems

Online CO problems are a natural class of problems that admit constructions of small-scale instances, because the hardness of online CO problems can be characterized by the input length, and instances of different scales are similar. This property simplifies the construction of curricula and underscores curriculum learning.

In this section we introduce two motivating online CO problems. We are interested in these problems because they have all been extensively studied and have closed-form, easy-to-implement policies as references. Furthermore, they were studied in Kong et al. [12], the paper that motivates our work. They also have real-world applications, e.g., auction design [2].

A.1. Secretary Problem

In SP, the goal is to maximize the *probability* of choosing the best among n candidates, where n is known. All candidates have *different* scores to quantify their abilities. They arrive sequentially and when the i-th candidate shows up, the decision-maker observes the relative ranking X_i among the first i candidates, which means being the X_ith-best so far. A decision that whether to accept or reject the i-th candidate must be made *immediately* when the candidate comes, and such decisions *cannot be revoked*. Once one candidate is accepted, the game ends immediately.

The ordering of the candidates is unknown. There are in total $n!$ permutations, and an instance of SP is drawn from an unknown distribution over these permutations. In the classical SP, each permutation is sampled with equal probability. The optimal solution for the classical SP is the well-known $1/\mathrm{e}$-*threshold strategy*: reject all the first $\lfloor n/\mathrm{e} \rfloor$ candidates, then accept the first one who is the best so-far. In this paper, we also study some different distributions.

Modeling SP. For SP, each instance is a permutation of length n, and in each round an instance is drawn from an unknown distribution over all permutations. In the i-th step for $i \in [n]$, the state encodes the i-th candidate and relative ranking so far. The transition is deterministic according to the problem definition. A reward of 1 is given if and only if the best candidate is accepted. We model the distribution as follows: suppose for candidate i, he/she is the best so far with probability P_i and is independent of other i'. The classical SP satisfies $P_i = \frac{1}{i}$.

A.2. Online Knapsack (decision version)

In Online Knapsack problems the decision-maker observes n (which is known) items arriving sequentially, each with value v_i and size s_i revealed upon arrival. A decision to either accept or reject the i-th item must be made *immediately* when it arrives, and such decisions *cannot be revoked*. At any time the accepted items should have their total size no larger than a known budget B.

The goal of standard Online Knapsack is to maximize the total value of accepted items. In this paper, we study the decision version, denoted as OKD, whose goal is to maximize the *probability* of total value reaching a known target V.

We assume that all values and sizes are sampled independently from two fixed distributions, namely $v_1, v_2, \ldots, v_n \overset{\text{i.i.d.}}{\sim} F_v$ and $s_1, s_2, \ldots, s_n \overset{\text{i.i.d.}}{\sim} F_s$. In Kong et al. [12] the experiments were carried out with $F_v = F_s = \mathrm{Unif}_{[0,1]}$, and we also study other distributions.

Modeling OKD. For OKD, each instance is a sequence of items with values and sizes drawn from unknown distributions F_v and F_s. In the i-th step for $i \in [n]$, the state encodes the information of i-th item's value and size, the remaining budget, and the remaining target value to fulfill. The

transition is also deterministic according to the problem definition, and a reward of 1 is given if and only if the agent obtains the target value for the first time. $F_v = F_s = \text{Unif}_{[0,1]}$ in Kong et al. [12].

Appendix B. Skipped algorithms

In this section, we present the algorithms skipped in the main text. Alg. 1 is the NPG for LMDP. Alg. 2 is the sampling function. Alg. 3 is the Curriculum Learning framework.

Algorithm 1: `NPG`: Sample-based NPG.

Input: Environment E; learning rate η; episode number T; batch size N; initialization θ_0; sampler π_s; regularization coefficient λ; entropy clip bound U; optimization domain $\mathcal{G}$.
for $t \leftarrow 0, 1, \ldots, T-1$ **do**
  Initialize $\widehat{F}_t \leftarrow 0^{d\times d}, \widehat{\nabla}_t \leftarrow 0^d$.
  for $n \leftarrow 0, 1, \ldots, N-1$ **do**
    for $h \leftarrow 0, 1, \ldots, H-1$ **do**
      if π_s is not None **then**
        $s_h, a_h, \widehat{A}_{H-h}(s_h, a_h) \leftarrow$ `Sample` $(E, \pi_s, \text{True}, \pi_t, h, \lambda, U)$ (see Alg. 2).
        // $s, a \sim \widetilde{d}^{\pi_s}_{m,h}$, estimate $A^{t,\lambda}_{m,H-h}(s,a)$.
      else
        $s_h, a_h, \widehat{A}_{H-h}(s_h, a_h) \leftarrow$ `Sample` $(E, \pi_t, \text{False}, \pi_t, h, \lambda, U)$.
        // $s, a \sim d^{\theta_t}_{m,h}$, estimate $A^{t,\lambda}_{m,H-h}(s,a)$.
      end if
    end for
    Update:

$$\widehat{F}_t \leftarrow \widehat{F}_t + \sum_{h=0}^{H-1} \nabla_\theta \ln \pi_{\theta_t}(a_h|s_h) \left(\nabla_\theta \ln \pi_{\theta_t}(a_h|s_h)\right)^\top,$$

$$\widehat{\nabla}_t \leftarrow \widehat{\nabla}_t + \sum_{h=0}^{H-1} \widehat{A}_{H-h}(s_h, a_h) \nabla_\theta \ln \pi_{\theta_t}(a_h|s_h).$$

  end for
  Call any solver to get $\widehat{g}_t \leftarrow \arg\min_{g\in\mathcal{G}} g^\top \widehat{F}_t g - 2g^\top \widehat{\nabla}_t$.
  Update $\theta_{t+1} \leftarrow \theta_t + \eta \widehat{g}_t$.
end for
Return: θ_T.

Appendix C. Proof of the main results

C.1. Performance of Natural Policy Gradient for LMDP

First we give the skipped definition of the Lyapunov potential function Φ, then prove Thm. 6.

Algorithm 2: `Sample`: Sampler for $s \sim d_{m,h}^{\pi_{\text{samp}}}$ where $m \sim$ Multinomial $(w_1, \ldots, w_M)$, $a \sim \text{Unif}_{\mathcal{A}}$ if $unif =$ True and $a \sim \pi_{\text{samp}}(\cdot|s)$ otherwise, and estimate of $A_{m,H-h}^{t,\lambda}(s,a)$.

Input: Environment E; sampler policy π_{samp}; whether to sample uniform actions after state $unif$; current policy π_t; time step h; regularization coefficient λ; entropy clip bound U.
E.reset().
for $i \leftarrow 0, 1, \ldots, h-1$ **do**
    $s_i \leftarrow E$.get_state().
    Sample action $a_i \sim \pi_{\text{samp}}(\cdot|s_i)$ and E.execute(a_i).
end for
$s_h \leftarrow E$.get_state().
if $unif$ = True **then**
    $a_h \sim \text{Unif}_{\mathcal{A}}$.
else
    $a_h \sim \pi_{\text{samp}}(\cdot|s_h)$.
end if
$(s, a) \leftarrow (s_h, a_h)$.
Get a random number $p \sim \text{Unif}[0, 1]$.
if $p < \frac{1}{2}$ **then**
    Override $a_h \sim \pi_t(\cdot|s_h)$.
    Set importance weight $C \leftarrow -2$.
    $r_h \leftarrow E$.execute(a_h).
    Initialize cumulative reward $R \leftarrow r_h + \lambda \mathcal{H}(\pi_t(\cdot|s_h))$.
else
    $C \leftarrow 2$.
    $r_h \leftarrow E$.execute(a_h).
    $R \leftarrow r_h + \lambda \min\{\ln \frac{1}{\pi_t(a_h|s_h)}, U\}$.
end if
for $i \leftarrow h+1, h+2, \ldots, H-1$ **do**
    $s_i \leftarrow E$.get_state().
    $a_i \sim \pi_t(\cdot|s_i)$ and $r_h \leftarrow E$.execute(a_i).
    $R \leftarrow R + r_i + \lambda \mathcal{H}(\pi_t(\cdot|s_i))$.
end for
Return: $s, a, \widehat{A}_{H-h}^{t,\lambda}(s,a) = CR$.

Definition 8 (Lyapunov Potential Function [7]) *We define the potential function $\Phi : \Pi \to \mathbb{R}$ as follows: for any $\pi \in \Pi$,*

$$\Phi(\pi) = \sum_{m=1}^{M} w_m \sum_{h=0}^{H-1} \mathop{\mathbb{E}}_{(s,a)\sim d_{m,h}^{\star}} \left[\ln \frac{\pi^{\star}(a|s)}{\pi(a|s)} \right].$$

Algorithm 3: Curriculum learning framework.

Input: Environment E; learning rate η; episode number T; batch size N; sampler type $samp \in \{$ `pi_s, pi_t` $\}$; regularization coefficient λ; entropy clip bound U; optimization domain $\mathcal{G}$.
Construct an environment E' with a task easier than E. This environment should have optimal policy similar to that of E.
$\theta_s \leftarrow$ `NPG` $(E', \eta, T, N, 0^d, \text{None}, \lambda, U, \mathcal{G})$ (see Alg. 1).
if $samp =$`pi_s` **then**
    $\theta_T \leftarrow$ `NPG` $(E, \eta, T, N, 0^d, \pi_s, \lambda, U, \mathcal{G})$.
else
    $\theta_T \leftarrow$ `NPG` $(E, \eta, T, N, \theta_s, \text{None}, \lambda, U, \mathcal{G})$.
end if
Return: θ_T.

Theorem 6 (Restatement of Thm. 6) *With Def. 4, 5 and 8, our algorithm enjoys the following performance bound:*

$$\mathbb{E}\left[\min_{0\le t\le T} V^{\star,\lambda} - V^{t,\lambda}\right] \le \frac{\lambda(1-\eta\lambda)^{T+1}\Phi(\pi_0)}{1-(1-\eta\lambda)^{T+1}} + \eta\frac{B^2G^2}{2} + \frac{\sum_{t=0}^{T}(1-\eta\lambda)^{T-t}\,\mathbb{E}[\,\mathrm{err}_t]}{\sum_{t'=0}^{T}(1-\eta\lambda)^{T-t'}}$$
$$\le \frac{\lambda(1-\eta\lambda)^{T+1}\Phi(\pi_0)}{1-(1-\eta\lambda)^{T+1}} + \eta\frac{B^2G^2}{2} + \sqrt{H\epsilon_{\text{bias}}} + \sqrt{H\kappa\epsilon_{\text{stat}}}.$$

Proof Here we make shorthands for the sub-optimality gap and potential function: $\Delta_t := V^{\star,\lambda} - V^{t,\lambda}$ and $\Phi_t := \Phi(\pi_t)$. From Lem. 15 we have

$$\eta\Delta_t \le (1-\eta\lambda)\Phi_t - \Phi_{t+1} + \eta\,\mathrm{err}_t + \eta^2\frac{B^2G^2}{2}.$$

Taking expectation over the update weights, we have

$$\mathbb{E}[\eta\Delta_t] \le (1-\eta\lambda)\,\mathbb{E}[\Phi_t] - \mathbb{E}[\Phi_{t+1}] + \eta\,\mathbb{E}[\,\mathrm{err}_t] + \eta^2\frac{B^2G^2}{2}.$$

Thus,

$$\begin{aligned}
&\mathbb{E}\left[\eta\sum_{t=0}^{T}(1-\eta\lambda)^{T-t}\Delta_t\right] \\
&\le \sum_{t=0}^{T}(1-\eta\lambda)^{T-t+1}\,\mathbb{E}[\Phi_t] - \sum_{t=0}^{T}(1-\eta\lambda)^{T-t}\,\mathbb{E}[\Phi_{t+1}] \\
&\quad + \eta\sum_{t=0}^{T}(1-\eta\lambda)^{T-t}\,\mathbb{E}[\,\mathrm{err}_t] + \eta^2\frac{B^2G^2}{2}\sum_{t=0}^{T}(1-\eta\lambda)^{T-t} \\
&= (1-\eta\lambda)^{T+1}\Phi_0 - \mathbb{E}[\Phi_{T+1}] + \eta\sum_{t=0}^{T}(1-\eta\lambda)^{T-t}\,\mathbb{E}[\,\mathrm{err}_t] + \eta^2\frac{B^2G^2}{2}\sum_{t=0}^{T}(1-\eta\lambda)^{T-t}
\end{aligned}$$

$$\leq (1-\eta\lambda)^{T+1}\Phi_0 + \eta\sum_{t=0}^{T}(1-\eta\lambda)^{T-t}\,\mathbb{E}[\,\mathrm{err}_t] + \eta^2\frac{B^2G^2}{2}\sum_{t=0}^{T}(1-\eta\lambda)^{T-t},$$

where the last step uses the fact that $\Phi(\pi) \geq 0$. This is a weighted average, so by normalizing the coefficients,

$$\begin{aligned}\mathbb{E}\left[\min_{0\leq t\leq T}\Delta_t\right] &\leq \frac{\lambda(1-\eta\lambda)^{T+1}\Phi_0}{1-(1-\eta\lambda)^{T+1}} + \eta\frac{B^2G^2}{2} + \frac{\sum_{t=0}^{T}(1-\eta\lambda)^{T-t}\,\mathbb{E}[\,\mathrm{err}_t]}{\sum_{t'=0}^{T}(1-\eta\lambda)^{T-t'}} \\ &\leq \frac{\lambda(1-\eta\lambda)^{T+1}\Phi_0}{1-(1-\eta\lambda)^{T+1}} + \eta\frac{B^2G^2}{2} + \sqrt{H\epsilon_{\text{bias}}} + \sqrt{H\kappa\epsilon_{\text{stat}}},\end{aligned}$$

where the last step comes from Lem. 16 and Jensen's inequality. This completes the proof. ■

C.2. Curriculum learning and the constant gap for Secretary Problem

Theorem 7 (Formal statement of Thm. 7) *For SP, set $samp = \texttt{pi\_s}$ in Alg. 3. Assume that each candidate is independent from others and the i-th candidate has probability P_i of being the best so far (see formulation in Sec. 2.1 and D.1). Assume the optimal policy is a p-threshold policy and the sampling policy is a q-threshold policy. There exists a policy parameterization and quantities*

$$k_{\text{curl}} = \begin{cases}\prod_{j=\lfloor nq\rfloor+1}^{\lfloor np\rfloor}\frac{1}{1-P_j}, & q\leq p,\\ 1, & q>p,\end{cases} \qquad k_{\text{naïve}} = 2^{\lfloor np\rfloor}\max\left\{1, \max_{i\geq\lfloor np\rfloor+2}\prod_{j=\lfloor np\rfloor+1}^{i-1}2(1-P_j)\right\},$$

such that $k_{\text{curl}} \leq \kappa_{\text{curl}} \leq 2k_{\text{curl}}$ and $k_{\text{naïve}} \leq \kappa_{\text{naïve}} \leq 2k_{\text{naïve}}$. Here κ_{curl} and $\kappa_{\text{naïve}}$ correspond to κ induced by the q-threshold policy and the naïve random policy respectively.

Proof We need to calculate three state-action visitation distributions: that induced by the optimal policy, $d^\star$; that induced by the sampler which is the optimal for the curriculum, $\widetilde{d}^{\text{curl}}$; and that induced by the naïve random sampler, $\widetilde{d}^{\text{naïve}}$. This then boils down to calculating the state(-action) visitation distribution under two types of policies: any threshold policy and the naïve random policy.

For any policy π, denote $d^\pi\left(\frac{i}{n}\right)$ as the probability for the agent acting under π to see states $\frac{i}{n}$ with arbitrary x_i. We do not need to take the terminal state g into consideration, since it stays in a zero-reward loop and contributes 0 to $L(g;\theta,d)$. We use the LMDP distribution described in Sec. 5.

Denote π_p as the p-threshold policy, i.e. accept if and only if $\frac{i}{n} > p$ and $x_i = 1$. Then

$$\begin{aligned}d^{\pi_p}\left(\frac{i}{n}\right) &= \mathbb{P}(\text{reject all previous } i-1 \text{ states}|\pi_p) \\ &= \prod_{j=1}^{i-1}\left(\mathbb{P}\left(\frac{j}{n},1\right)\mathbb{1}\left[\frac{j}{n}\leq p\right] + 1 - \mathbb{P}\left(\frac{j}{n},1\right)\right) \\ &= \prod_{j=\lfloor np\rfloor+1}^{i-1}\left(1-\mathbb{P}\left(\frac{j}{n},1\right)\right) \\ &= \prod_{j=\lfloor np\rfloor+1}^{i-1}(1-P_j).\end{aligned}$$

Denote $\pi_{\text{naïve}}$ as the naïve random policy, i.e., accept with probability $\frac{1}{2}$ regardless of the state. Then

$$d^{\pi_{\text{naïve}}}\left(\frac{i}{n}\right) = \mathbb{P}(\text{reject all previous } i-1 \text{ states}|\pi_{\text{naïve}}) = \frac{1}{2^{i-1}}.$$

For any π, we can see that the state visitation distribution satisfies $d^{\pi}\left(\frac{i}{n}, 1\right) = P_i d^{\pi}\left(\frac{i}{n}\right)$ and $d^{\pi}\left(\frac{i}{n}, 0\right) = (1-P_i)d^{\pi}\left(\frac{i}{n}\right)$.

To show the possible largest difference, we use a parameterization that for each state s, $\phi(s) = \text{One-hot}(s)$. The policy is then

$$\pi_\theta(\text{accept}|s) = \frac{\exp(\theta^\top \phi(s))}{\exp(\theta^\top \phi(s)) + 1}, \quad \pi_\theta(\text{reject}|s) = \frac{1}{\exp(\theta^\top \phi(s)) + 1}.$$

Denote $\pi_\theta(s) = \pi_\theta(\text{accept}|s)$, we have

$$\nabla_\theta \ln \pi_\theta(\text{accept}|s) = (1-\pi_\theta(s))\phi(s), \quad \nabla_\theta \ln \pi_\theta(\text{reject}|s) = -\pi_\theta(s)\phi(s).$$

Now suppose the optimal threshold and the threshold learned through curriculum are p and q, then

$$\begin{aligned}
\Sigma^\theta_{d^\star} &= \sum_{s\in\mathcal{S}} d^{\pi_p}(s)\left(\pi_p(s)(1-\pi_\theta(s))^2 + (1-\pi_p(s))\pi_\theta(s)^2\right)\phi(s)\phi(s)^\top, \\
\Sigma^\theta_{\widetilde{d}^{\text{curl}}} &= \sum_{s\in\mathcal{S}} d^{\pi_q}(s)\left(\frac{1}{2}(1-\pi_\theta(s))^2 + \frac{1}{2}\pi_\theta(s)^2\right)\phi(s)\phi(s)^\top, \\
\Sigma^\theta_{\widetilde{d}^{\text{naïve}}} &= \sum_{s\in\mathcal{S}} d^{\text{naïve}}(s)\left(\frac{1}{2}(1-\pi_\theta(s))^2 + \frac{1}{2}\pi_\theta(s)^2\right)\phi(s)\phi(s)^\top.
\end{aligned}$$

Denote $\kappa_{\clubsuit}(\theta) = \sup_{x\in\mathbb{R}^d} \frac{x^\top \Sigma^\theta_{d^\star} x}{x^\top \Sigma^\theta_{d^{\clubsuit}} x}$. From parameterization we know all $\phi(s)$ are orthogonal. Abusing π_q with π_{curl}, we have

$$\kappa_{\clubsuit}(\theta) = \max_{s\in\mathcal{S}} \frac{d^{\pi_p}(s)\left(\pi^\star(s)(1-\pi_\theta(s))^2 + (1-\pi^\star(s))\pi_\theta(s)^2\right)}{d^{\clubsuit}(s)\left(\frac{1}{2}(1-\pi_\theta(s))^2 + \frac{1}{2}\pi_\theta(s)^2\right)}.$$

We can separately consider each $s \in \mathcal{S}$ because of the orthogonal features. Observe that $\pi_p(s) \in \{0,1\}$, so for $s\in\mathcal{S}$, its corresponding term in $\kappa_{\clubsuit}(\theta)$ is maximized when $\pi_\theta(s) = 1-\pi_p(s)$ and is equal to $2\frac{d^{\pi_p}(s)}{d^{\clubsuit}(s)}$. By definition, $\kappa_{\clubsuit} = \max_{0\le t\le T}\mathbb{E}[\kappa_{\clubsuit}(\theta_t)]$. Since $\theta_0 = 0^d$, we have $\kappa_{\clubsuit} \ge \kappa_{\clubsuit}(0^d)$ where $\pi_\theta(s) = \frac{1}{2}$ and the corresponding term is $\frac{d^{\pi_p}(s)}{d^{\clubsuit}(s)}$. So

$$\max_{s\in\mathcal{S}} \frac{d^{\pi_p}(s)}{d^{\clubsuit}(s)} \le \kappa_{\clubsuit} \le 2\max_{s\in\mathcal{S}} \frac{d^{\pi_p}(s)}{d^{\clubsuit}(s)}.$$

We now have an order-accurate result $k_{\clubsuit} = \max_{s\in\mathcal{S}} \frac{d^{\pi_p}(s)}{d^{\clubsuit}(s)}$ for $\kappa_{\clubsuit}$. Direct computation gives

$$k_{\text{curl}} = \begin{cases} \prod_{j=\lfloor nq\rfloor+1}^{\lfloor np\rfloor} \frac{1}{1-P_j}, & q \le p, \\ 1, & q > p, \end{cases}$$

$$k_{\text{naïve}} = 2^{\lfloor np\rfloor} \max\left\{1, \max_{i\ge\lfloor np\rfloor+2} \prod_{j=\lfloor np\rfloor+1}^{i-1} 2(1-P_j)\right\}.$$

This completes the proof. ■

Appendix D. Full experiments

Here are all the experiments not shown in Sec. 5. All the experiments were run on a server with CPU AMD Ryzen 9 3950X, GPU NVIDIA GeForce 2080 Super and 128G memory. For legend description please refer to the caption of Fig. 1. For experiment data (code, checkpoints, logging data and policy visualization) please refer to the supplementary files.

Policy parameterization. Since in all the experiments there are exactly two actions, we can use $\phi(s) = \phi(s, \text{accept}) - \phi(s, \text{reject})$ instead of $\phi(s, \text{accept})$ and $\phi(s, \text{reject})$. Now the policy is $\pi_\theta(\text{accept}|s) = \frac{\exp(\theta^\top \phi(s))}{\exp(\theta^\top \phi(s))+1}$ and $\pi_\theta(\text{reject}|s) = \frac{1}{\exp(\theta^\top \phi(s))+1}$.

Training schemes. We ran seven experiments in total, three for Secretary Problem and four for Online Knapsack (decision version). The difference between the experiments of the same problem lies in the distribution over instances (i.e., $\{w_m\}$). In the following subsections, we will introduce how we parameterized the distribution in detail. In a single experiment, we ran eight setups, each representing a combination of sampler policies, initialization policies of the final phase, and whether we used regularization. For visual clarity, we did not plot setups with entropy regularization, but the readers can plot it using `plot.py` (comment L55-58 and uncomment L59-62) in the supplementary files. We make a detailed list of the training schemes in Tab. 1.

D.1. Secretary Problem

State and action spaces. States with $X_i > 1$ are the same. To make the problem "scale-invariant", we use $\frac{i}{n}$ to represent i. So the states are $(\frac{i}{n}, x_i = \mathbb{1}[X_i = 1])$. There is an additional terminal state $g = (0, 0)$. For each state, the agent can either accept or reject.

Transition and reward. Any action in g leads back to g. Once the agent accepts the i-th candidate, the state transits into g, and reward is 1 if i is the best in the instance. If the agent rejects, then the state goes to $(\frac{i+1}{n}, x_{i+1})$ if $i < n$ and g if $i = n$. For all other cases, rewards are 0.

Feature mapping. Recall that all states are of the form (f, x) where $f \in [0, 1],\ x \in \{0, 1\}$. We set a degree d_0 and the feature mapping is constructed as the collection of polynomial bases with degree less than d_0 ($d = 2d_0$):

$$\phi(f, x) = (1, f, \ldots, f^{d_0-1}, x, fx, \ldots, f^{d_0-1}x).$$

LMDP distribution. We model the distribution as follows: for each i, we can have $x_i = 1$ with probability P_i and is independent from other i'. By definition, $P_1 = 1$ while other P_i can be arbitrary. The classical SP satisfies $P_i = \frac{1}{i}$. We also experimented on three other distributions (so in total there are four experiments), each with a series of numbers $p_2, p_3, \ldots, p_n \overset{\text{i.i.d.}}{\sim} \text{Unif}_{[0,1]}$ and set $P_i = \frac{1}{i^{2p_i+0.25}}$.

For each experiment, we run eight setups, each with different combinations of sampler policies, initialization policies of the final phase, and the value of regularization coefficient λ. For the warm-up phases we set $n = 10$ and for final phases $n = 100$.

Results. Fig. 3 (with its full view Fig. 4), Fig. 5, Fig. 6, along with Fig. 1 (with seed 2018011309) show four experiments of SP. They shared a learning rate of 0.2, batch size of 100 per step in horizon, final $n = 100$ and warm-up $n = 10$ (if applied curriculum learning). [1]

1. All the four trainings shown in the figures have their counterparts with regularization ($\lambda = 0.01$). Check the supplementary files and use TensorBoard for visualization.

The experiment in Fig. 3 was done in the classical SP environment, i.e., all permutations have probability $\frac{1}{n!}$ to be sampled. Experiments Fig. 1, Fig. 5 and Fig. 6 were done with other distributions (see LMDP distribution of Sec. 5): the only differences are the random seeds, which we fixed and used to generate P_is for reproducibility.

The experiment of classical SP was run until the direct training of $n = 100$ converges, while all other experiments were run to a maximum episode of 30000 (hence sample number of $THb = 30000 \times 100 \times 100 = 3 \times 10^8$).

The optimal policy was derived from dynamic programming.

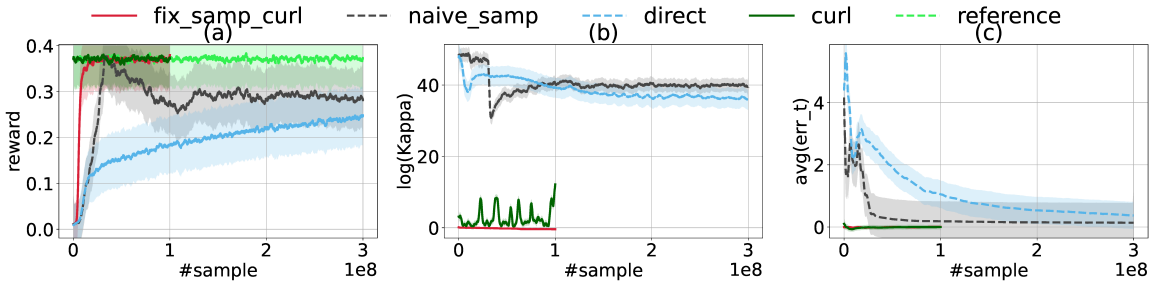

Figure 3: Classical SP, truncated to 3×10^8 samples.

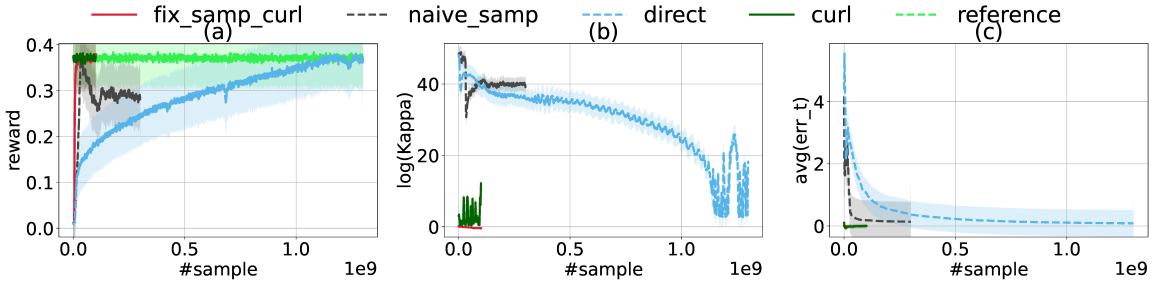

Figure 4: Classical SP, full view.

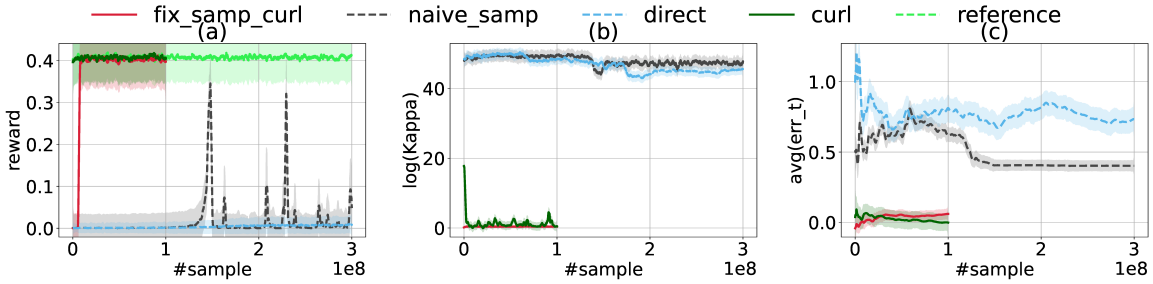

Figure 5: SP, with seed 20000308.

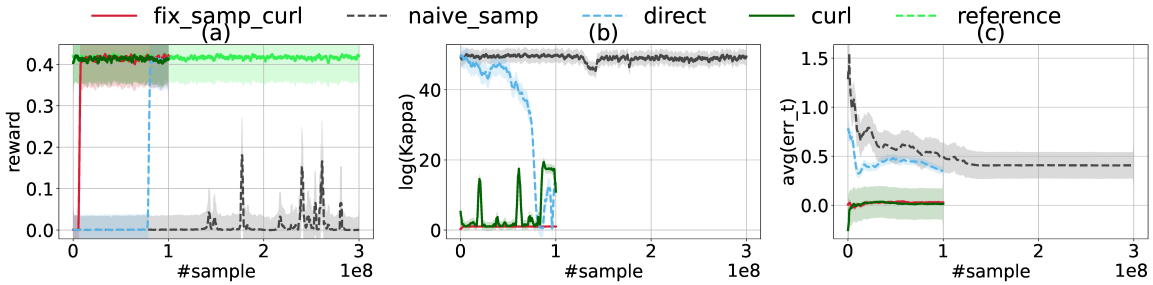

Figure 6: SP, with seed 19283746.

D.2. Online Knapsack (decision version)

State and action spaces. The states are represented as

$$\left(\frac{i}{n}, s_i, v_i, \frac{\sum_{j=1}^{i-1} x_j s_j}{B}, \frac{\sum_{j=1}^{i-1} x_j v_j}{V}\right),$$

where $x_j = \mathbb{1}[\text{item } j \text{ was successfully chosen}]$ for $1 \le j \le i-1$ (in the instance). There is an additional terminal state $g = (0,0,0,0,0)$. For each state (including g for simplicity), the agent can either accept or reject.

Transition and reward. The transition is implied by the definition of the problem. Any action in terminal state g leads back to g. The item is successfully chosen if and only if the agent accepts and the budget is sufficient. A reward of 1 is given only the first time $\sum_{j=1}^{i} x_i v_i \ge V$, and then the state goes to g. For all other cases, reward is 0.

Feature mapping. Suppose the state is (f, s, v, r, q). We set a degree d_0 and the feature mapping is constructed as the collection of polynomial bases with degree less than d_0 ($d = d_0^5$): $\phi(f,s,v,r,q) = (f^{i_f} s^{i_s} v^{i_v} r^{i_r} q^{i_q})_{i_f,i_s,i_v,i_r,i_q}$ where $i_\clubsuit \in \{0,1,\dots,d_0-1\}$.

LMDP distribution. In Sec. A.2 the values and sizes are sampled from F_v ans F_s. If F_v or F_s is not $\text{Unif}_{[0,1]}$, we model the distribution as: first set a granularity $gran$ and take $gran$ numbers $p_1, p_2, \dots, p_{gran} \overset{\text{i.i.d.}}{\sim} \text{Unif}_{[0,1]}$. p_i represents the (unnormalized) probability that $x \in (\frac{i-1}{gran}, \frac{i}{gran})$. To sample, we take $i \sim \text{Multinomial}(p_1, p_2, \dots, p_{gran})$ and return $x \sim \frac{i-1+\text{Unif}_{[0,1]}}{gran}$.

For each experiment, we ran four setups, each with different combinations of sampler policies and initialization policies of the final phase. For the warm-up phases $n = 10$ and for final phases we set $n = 100$ in all experiments, while B and V vary. In one experiment it satisfies that $\frac{B}{n}$ are close for warm-up and final, and $\frac{V}{B}$ increases from warm-up to final.

Results. Fig. 7, Fig. 8, along with Fig. 2 (with $F_v = F_s = \text{Unif}_{[0,1]}$) show three experiments of OKD. They shared a learning rate of 0.1, batch size of 100 per step in horizon, final $n = 100$ and warm-up $n = 10$ (if applied curriculum learning).

Experiments Fig. 7 and Fig. 8 were done with other value and size distributions (see LMDP distribution of Sec. 5): the only differences are the random seeds, which we fixed and used to generate F_v and F_s for reproducibility.

All experiments were run to a maximum episode of 50000 (hence sample number of $THb = 50000 \times 100 \times 100 = 5 \times 10^8$).

The reference policy is a bang-per-buck algorithm (Sec. 3.1 of Kong et al. [12]): given a threshold r, accept i-th item if $\frac{v_i}{s_i} \geq r$. We searched for the optimal r with respect to Online Knapsack because we found that in general the reward is unimodal to r and contains no "plain area", so we can easily apply ternary search (the reward of OKD contains "plain area").

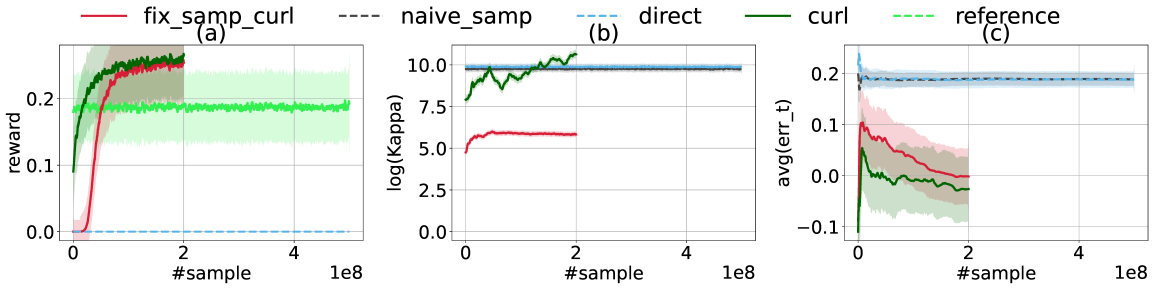

Figure 7: OKD, with seed 2018011309.

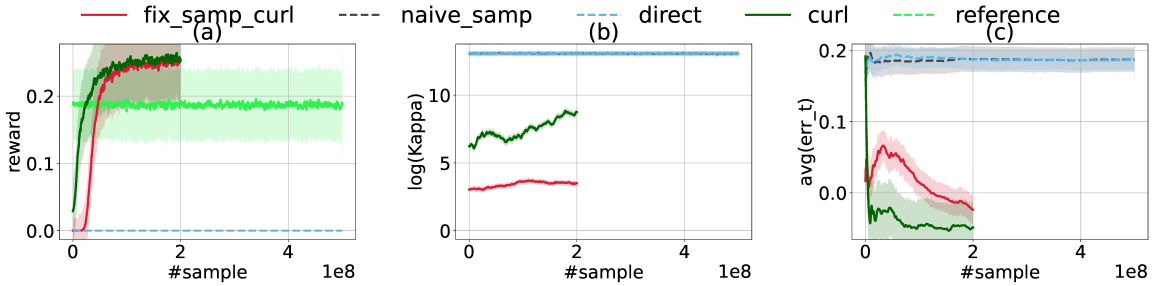

Figure 8: OKD, with seed 20000308.

Appendix E. Technical details and lemmas

E.1. Natural Policy Gradient for LMDP

This section is a complement to Sec. 3. We give details about the correctness of Natural Policy Gradient for LMDP.

Thm. 11 is the finite-horizon Policy Gradient Theorem for LMDP, which takes the mixing weight $\{w_m\}$ into consideration.

According to Agarwal et al. [1], the unconstrained, full-information NPG update weight satisfies $F(\theta_t)g_t = \nabla_\theta V^{t,\lambda}$. Lem. 12 and Lem. 13 together show that: it is equivalent to finding a minimizer of the fitting compatible function approximation loss (Def. 2).

Theorem 11 (Policy Gradient Theorem for LMDP) *For any policy π_θ parameterized by θ, and any $1 \leq m \leq M$,*

$$\nabla_\theta \left(\mathop{\mathbb{E}}_{s_0 \sim \nu_m} \left[V_{m,H}^{\pi_\theta,\lambda}(s_0) \right] \right) = \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a \sim d_{m,H-h}^{\theta}} \left[Q_{m,h}^{\pi_\theta,\lambda}(s,a) \nabla_\theta \ln \pi_\theta(a|s) \right].$$

As a result,

$$\nabla_\theta V^{\pi_\theta,\lambda} = \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a\sim d^\theta_{m,H-h}} \left[Q^{\pi_\theta,\lambda}_{m,h}(s,a) \nabla_\theta \ln \pi_\theta(a|s) \right].$$

Proof For any $1 \le h \le H$ and $s \in \mathcal{S}$, since $V^{\pi_\theta,\lambda}_{m,h}(s) = \sum_{a\in\mathcal{A}} \pi_\theta(a|s) Q^{\pi_\theta,\lambda}_{m,h}(s,a)$, we have

$$\nabla_\theta V^{\pi_\theta,\lambda}_{m,h}(s) = \sum_{a\in\mathcal{A}} \left(Q^{\pi_\theta,\lambda}_{m,h}(s,a) \nabla_\theta \pi_\theta(a|s) + \pi_\theta(a|s) \nabla_\theta Q^{\pi_\theta,\lambda}_{m,h}(s,a) \right).$$

Hence

$$\begin{aligned}
\sum_{h=1}^{H} \sum_{s\in\mathcal{S}} d^\theta_{m,H-h}(s) \nabla_\theta V^{\pi_\theta,\lambda}_{m,h}(s) &= \sum_{h=1}^{H} \sum_{s\in\mathcal{S}} d^\theta_{m,H-h}(s) \sum_{a\in\mathcal{A}} \left(Q^{\pi_\theta,\lambda}_{m,h}(s,a) \nabla_\theta \pi_\theta(a|s) + \pi_\theta(a|s) \nabla_\theta Q^{\pi_\theta,\lambda}_{m,h}(s,a) \right) \\
&= \sum_{h=1}^{H} \sum_{s\in\mathcal{S}} d^\theta_{m,H-h}(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) Q^{\pi_\theta,\lambda}_{m,h}(s,a) \nabla_\theta \ln \pi_\theta(a|s) \\
&\quad + \sum_{h=1}^{H} \sum_{s\in\mathcal{S}} d^\theta_{m,H-h}(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) \nabla_\theta Q^{\pi_\theta,\lambda}_{m,h}(s,a) \\
&= \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a\sim d^\theta_{m,H-h}} \left[Q^{\pi_\theta,\lambda}_{m,h}(s,a) \nabla_\theta \ln \pi_\theta(a|s) \right] \\
&\quad + \sum_{h=1}^{H} \sum_{s\in\mathcal{S}} d^\theta_{m,H-h}(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) \nabla_\theta Q^{\pi_\theta,\lambda}_{m,h}(s,a).
\end{aligned}$$

Next we focus on the second term. From the Bellman equation,

$$\begin{aligned}
\nabla_\theta Q^{\pi_\theta,\lambda}_{m,h}(s,a) &= \nabla_\theta \left(r_\theta(s,a) - \lambda \ln \pi_\theta(a|s) + \sum_{s'\in\mathcal{S}} P(s'|s,a) V^{\pi_\theta,\lambda}_{m,h-1}(s') \right) \\
&= -\lambda \nabla_\theta \ln \pi_\theta(a|s) + \sum_{s'\in\mathcal{S}} P(s'|s,a) \nabla_\theta V^{\pi_\theta,\lambda}_{m,h-1}(s').
\end{aligned}$$

Particularly, $\nabla_\theta Q^{\pi,\lambda}_{i,1}(s,a) = -\lambda \nabla_\theta \ln \pi_\theta(a|s)$. So

$$\begin{aligned}
&\sum_{h=1}^{H} \sum_{s\in\mathcal{S}} d^\theta_{m,H-h}(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) \nabla_\theta Q^{\pi_\theta,\lambda}_{m,h}(s,a) \\
&= \sum_{h=1}^{H} \sum_{s\in\mathcal{S}} d^\theta_{m,H-h}(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) \left(-\lambda \nabla_\theta \ln \pi_\theta(a|s) + \sum_{s'\in\mathcal{S}} P(s'|s,a) \nabla_\theta V^{\pi_\theta,\lambda}_{m,h-1}(s') \right) \\
&= -\lambda \sum_{h=1}^{H} \sum_{s\in\mathcal{S}} d^\theta_{m,H-h}(s) \underbrace{\sum_{a\in\mathcal{A}} \nabla_\theta \pi_\theta(a|s)}_{=\mathbf{0}} + \sum_{h=2}^{H} \sum_{s'\in\mathcal{S}} \nabla_\theta V^{\pi_\theta,\lambda}_{m,h-1}(s') \underbrace{\sum_{s\in\mathcal{S}} d^\theta_{m,H-h}(s) \sum_{a\in\mathcal{A}} \pi_\theta(a|s) P(s'|s,a)}_{=d^\theta_{m,H-h+1}(s')}
\end{aligned}$$

$$= \sum_{h=2}^{H} \sum_{s' \in \mathcal{S}} d^{\theta}_{m,H-h+1}(s') \nabla_\theta V^{\pi_\theta,\lambda}_{m,h-1}(s')$$
$$= \sum_{h=1}^{H} \sum_{s \in \mathcal{S}} d^{\theta}_{m,H-h}(s) \nabla_\theta V^{\pi_\theta,\lambda}_{m,h}(s) - \sum_{s_0 \in \mathcal{S}} \nu_m(s_0) \nabla_\theta V^{\pi_\theta,\lambda}_{m,H}(s_0),$$

where we used the definition of d and ν_m. So by rearranging the terms, we complete the proof. ■

Lemma 12 *Suppose* $\Gamma \in \mathbb{R}^{n \times m}, D = \mathrm{diag}(d_1, d_2, \ldots, d_m) \in \mathbb{R}^{m \times m}$ *where* $d_i \geq 0$ *and* $q \in \mathbb{R}^m$, *then* $x = (\Gamma D \Gamma^\top)^\dagger \Gamma D q$ *is a solution to the equation* $\Gamma D \Gamma^\top x = \Gamma D q$.

Proof Denote $D^{1/2} = \mathrm{diag}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_m}), P = \Gamma D^{1/2}, p = D^{1/2} q$, then the equation is reduced to $PP^\top x = Pp$. Suppose the singular value decomposition of P is $U \Sigma V^\top$ where $U \in \mathbb{R}^{n \times n}, \Sigma \in \mathbb{R}^{n \times m}, V \in \mathbb{R}^{m \times m}$ where U and V are unitary, and singular values are $\sigma_1, \sigma_2, \ldots, \sigma_k$. So $PP^\top = U(\Sigma\Sigma^\top)U^\top$ and $(PP^\top)^\dagger = U(\Sigma\Sigma^\top)^\dagger U^\top$. Notice that

$$\Sigma\Sigma^\top = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_k^2, 0, \ldots, 0) \in \mathbb{R}^{n \times n},$$

we can then derive the pseudo-inverse of this particular diagonal matrix as

$$(\Sigma\Sigma^\top)^\dagger = \mathrm{diag}(\sigma_1^{-2}, \sigma_2^{-2}, \ldots, \sigma_k^{-2}, 0, \ldots, 0).$$

It is then easy to verify that $(\Sigma\Sigma^\top)(\Sigma\Sigma^\top)^\dagger \Sigma = \Sigma$. Finally,

$$\begin{aligned} PP^\top x &= (PP^\top)[(PP^\top)^\dagger Pp] \\ &= U(\Sigma\Sigma^\top)U^\top U(\Sigma\Sigma^\top)^\dagger U^\top U \Sigma V^\top p \\ &= U(\Sigma\Sigma^\top)(\Sigma\Sigma^\top)^\dagger \Sigma V^\top p \\ &= U \Sigma V^\top p \\ &= Pp. \end{aligned}$$

This completes the proof. ■

Lemma 13 (NPG Update Rule) *The update rule* $\theta \leftarrow \theta + \eta F(\theta)^\dagger \nabla_\theta V^{\pi_\theta,\lambda}$ *where*

$$F(\theta) = \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a \sim d^{\theta}_{m,H-h}} \left[\nabla_\theta \ln \pi_\theta(a|s) \left(\nabla_\theta \ln \pi_\theta(a|s) \right)^\top \right]$$

is equivalent to $\theta \leftarrow \theta + \eta g^\star$, *where* $g^\star$ *is a minimizer of the function*

$$L(g) = \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a \sim d^{\theta}_{m,H-h}} \left[\left(A^{\pi_\theta,\lambda}_{m,h}(s,a) - g^\top \nabla_\theta \ln \pi_\theta(a|s) \right)^2 \right].$$

Proof

$$\nabla_g L(g) = -2 \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a\sim d^{\theta}_{m,H-h}} \left[\left(A^{\pi_\theta,\lambda}_{m,h}(s,a) - g^\top \nabla_\theta \ln \pi_\theta(a|s) \right) \nabla_\theta \ln \pi_\theta(a|s) \right].$$

Suppose $g^\star$ is any minimizer of $L(g)$, we have $\nabla_g L(g^\star) = \mathbf{0}$, hence

$$\begin{aligned}
&\sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a\sim d^{\theta}_{m,H-h}} \left[\left(g^{\star\top} \nabla_\theta \ln \pi_\theta(a|s) \right) \nabla_\theta \ln \pi_\theta(a|s) \right] \\
&= \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a\sim d^{\theta}_{m,H-h}} \left[A^{\pi_\theta,\lambda}_{m,h}(s,a) \nabla_\theta \ln \pi_\theta(a|s) \right] \\
&= \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a\sim d^{\theta}_{m,H-h}} \left[Q^{\pi_\theta,\lambda}_{m,h}(s,a) \nabla_\theta \ln \pi_\theta(a|s) \right].
\end{aligned}$$

Since $(u^\top v)v = (vv^\top)u$, then

$$F(\theta) g^\star = \nabla_\theta V^{\pi_\theta,\lambda}.$$

Now we assign $1, 2, \ldots, MHSA$ as indices to all $(m, h, s, a) \in \{1, \ldots, M\} \times \{1, \ldots, H\} \times \mathcal{S} \times \mathcal{A}$, and set

$$\begin{aligned}
\gamma_j &= \nabla_\theta \ln \pi_\theta(a|s), \\
d_j &= w_m d^{\theta}_{m,H-h}(s,a), \\
q_j &= Q^{\pi_\theta,\lambda}_{m,h}(s,a),
\end{aligned}$$

where j is the index assigned to (m, h, s, a). Then $F(\theta) = \Phi D \Phi^\top$ and $\nabla_\theta V^\theta = \Phi D q$ where

$$\begin{aligned}
\Gamma &= [\gamma_1, \gamma_2, \ldots, \gamma_{MHSA}] \in \mathbb{R}^{d \times MHSA}, \\
D &= \operatorname{diag}(d_1, d_2, \ldots, d_{MHSA}) \in \mathbb{R}^{MHSA \times MHSA}, \\
q &= [q_1, q_2, \ldots, q_{MHSA}]^\top \in \mathbb{R}^{MHSA}.
\end{aligned}$$

We now conclude the proof by utilizing Lem. 12. ■

E.2. Auxiliary lemmas used in the main results

Lemma 14 (Performance Difference Lemma) *For any two policies π_1 and π_2, and any $1 \le m \le M$,*

$$\mathop{\mathbb{E}}_{s_0\sim\nu_m} \left[V^{\pi_1,\lambda}_{m,H}(s_0) - V^{\pi_2,\lambda}_{m,H}(s_0) \right] = \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a\sim d^{\pi_1}_{m,H-h}} \left[A^{\pi_2,\lambda}_{m,h}(s,a) + \lambda \ln \frac{\pi_2(a|s)}{\pi_1(a|s)} \right].$$

As a result,

$$V^{\pi_1,\lambda} - V^{\pi_2,\lambda} = \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a\sim d^{\pi_1}_{m,H-h}} \left[A^{\pi_2,\lambda}_{m,h}(s,a) + \lambda \ln \frac{\pi_2(a|s)}{\pi_1(a|s)} \right].$$

Proof First we fix s_0. By definition of the value function, we have

$$
\begin{aligned}
&V_{m,H}^{\pi_1,\lambda}(s_0) - V_{m,H}^{\pi_2,\lambda}(s_0) \\
&= \mathbb{E}\left[\sum_{h=0}^{H-1} r_m(s_h,a_h) - \lambda \ln \pi_1(a_h|s_h) \;\middle|\; \mathcal{M}_m, \pi_1, s_0\right] - V_{m,H}^{\pi_2,\lambda}(s_0) \\
&= \mathbb{E}\left[\sum_{h=0}^{H-1} r_m(s_h,a_h) - \lambda \ln \pi_1(a_h|s_h) + V_{m,H+1-h}^{\pi_2,\lambda}(s_{h+1}) - V_{m,H-h}^{\pi_2,\lambda}(s_h) \;\middle|\; \mathcal{M}_m, \pi_1, s_0\right] \\
&= \mathbb{E}\left[\sum_{h=0}^{H-1} \mathbb{E}\left[r_m(s_h,a_h) - \lambda \ln \pi_2(a_h|s_h) + V_{m,H+1-h}^{\pi_2,\lambda}(s_{h+1}) \;\middle|\; \mathcal{M}_m, \pi_2, s_h, a_h\right] \;\middle|\; \mathcal{M}_m, \pi_1, s_0\right] \\
&\quad + \mathbb{E}\left[\sum_{h=0}^{H-1} -V_{m,H-h}^{\pi_2,\lambda}(s_h) + \lambda \ln \frac{\pi_2(a_h|s_h)}{\pi_1(a_h|s_h)} \;\middle|\; \mathcal{M}_m, \pi_1, s_0\right],
\end{aligned}
$$

where the last step uses law of iterated expectations. Since

$$
\mathbb{E}\left[r_m(s_h,a_h) - \lambda \ln \pi_2(a_h|s_h) + V_{m,H+1-h}^{\pi_2,\lambda}(s_{h+1}) \;\middle|\; \mathcal{M}_m, \pi_2, s_h, a_h\right] = Q_{m,H-h}^{\pi_2,\lambda}(s_h,a_h),
$$

we have

$$
\begin{aligned}
V_{m,H}^{\pi_1,\lambda}(s_0) - V_{m,H}^{\pi_2,\lambda}(s_0) &= \mathbb{E}\left[\sum_{h=0}^{H-1} Q_{m,H-h}^{\pi_2,\lambda}(s_h,a_h) - V_{m,H-h}^{\pi_2,\lambda}(s_h) + \lambda \ln \frac{\pi_2(a_h|s_h)}{\pi_1(a_h|s_h)} \;\middle|\; \mathcal{M}_m, \pi_1, s_0\right] \\
&= \mathbb{E}\left[\sum_{h=0}^{H-1} A_{m,H-h}^{\pi_2,\lambda}(s_h,a_h) + \lambda \ln \frac{\pi_2(a_h|s_h)}{\pi_1(a_h|s_h)} \;\middle|\; \mathcal{M}_m, \pi_1, s_0\right].
\end{aligned}
$$

By taking expectation over s_0, we have

$$
\begin{aligned}
\mathop{\mathbb{E}}_{s_0\sim\nu_m}\left[V_{m,H}^{\pi_1,\lambda}(s_0) - V_{m,H}^{\pi_2,\lambda}(s_0)\right] &= \mathbb{E}\left[\sum_{h=0}^{H-1} A_{m,H-h}^{\pi_2,\lambda}(s_h,a_h) + \lambda \ln \frac{\pi_2(a_h|s_h)}{\pi_1(a_h|s_h)} \;\middle|\; \mathcal{M}_m, \pi_1\right] \\
&= \sum_{h=0}^{H-1} \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} d_{m,h}^{\pi_1}(s,a)\left(A_{m,H-h}^{\pi_2,\lambda}(s,a) + \lambda \ln \frac{\pi_2(a|s)}{\pi_1(a|s)}\right).
\end{aligned}
$$

The proof is completed by reversing the order of h. ■

Lemma 15 (Lyapunov Drift) *Recall definitions in Def. 8 and 5. We have that:*

$$
\Phi(\pi_{t+1}) - \Phi(\pi_t) \le -\eta\lambda\Phi(\pi_t) + \eta\,\mathrm{err}_t - \eta\left(V^{\star,\lambda} - V^{t,\lambda}\right) + \frac{\eta^2 B^2 \|g_t\|_2^2}{2}.
$$

Proof Denote $\Phi_t := \Phi(\pi_t)$. This proof follows a similar manner as in that of Lem. 6 in Cayci et al. [7]. By smoothness (see Rem. 6.7 in Agarwal et al. [1]),

$$
\ln \frac{\pi_t(a|s)}{\pi_{t+1}(a|s)} \le (\theta_t - \theta_{t+1})^\top \nabla_\theta \ln \pi_t(a|s) + \frac{B^2}{2}\|\theta_{t+1} - \theta_t\|_2^2
$$

$$= -\eta g_t^\top \nabla_\theta \ln \pi_t(a|s) + \frac{\eta^2 B^2 \|g_t\|_2^2}{2}.$$

By the definition of Φ,

$$\begin{aligned}\Phi_{t+1} - \Phi_t &= \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{(s,a)\sim d^\star_{m,H-h}} \left[\ln \frac{\pi_t(a|s)}{\pi_{t+1}(a|s)}\right] \\ &\le -\eta \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{(s,a)\sim d^\star_{m,H-h}} \left[g_t^\top \nabla_\theta \ln \pi_t(a|s)\right] + \frac{\eta^2 B^2 \|g_t\|_2^2}{2}.\end{aligned}$$

By the definition of err_t, Lem. 14 and again the definition of Φ, we finally have

$$\begin{aligned}\Phi_{t+1} - \Phi_t &\le \eta \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{(s,a)\sim d^\star_{m,H-h}} \left[A^{t,\lambda}_{m,h}(s,a) - g_t^\top \nabla_\theta \ln \pi_t(a|s)\right] \\ &\quad - \eta \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{(s,a)\sim d^\star_{m,H-h}} \left[A^{t,\lambda}_{m,h}(s,a) + \lambda \ln \frac{\pi_t(a|s)}{\pi^\star(a|s)}\right] \\ &\quad - \eta\lambda \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{(s,a)\sim d^\star_{m,H-h}} \left[\ln \frac{\pi^\star(a|s)}{\pi_t(a|s)}\right] + \frac{\eta^2 B^2 \|g_t\|_2^2}{2} \\ &= \eta\, \mathrm{err}_t - \eta\left(V^{\star,\lambda} - V^{t,\lambda}\right) - \eta\lambda\Phi_t + \frac{\eta^2 B^2 \|g_t\|_2^2}{2},\end{aligned}$$

which completes the proof. ■

Lemma 16 *Recall that $g_t^\star$ is the true minimizer of $L(g;\theta_t,d^t)$ in domain $\mathcal{G}$. err_t defined in Def. 5 satisfies*

$$\mathrm{err}_t \le \sqrt{HL(g_t^\star;\theta_t,d^\star)} + \sqrt{H\kappa(L(g_t;\theta_t,d^t) - L(g_t^\star;\theta_t,d^t))}.$$

Proof The proof is similar to that of Thm. 6.1 in Agarwal et al. [1]. We make the following decomposition of err_t:

$$\begin{aligned}\mathrm{err}_t = &\underbrace{\sum_{m=1}^{M} w_m \sum_{h=0}^{H-1} \mathop{\mathbb{E}}_{(s,a)\sim d^\star_{m,h}} \left[A^{t,\lambda}_{m,h}(s,a) - g_t^{\star\top} \nabla_\theta \ln \pi_t(a|s)\right]}_{①} \\ &+ \underbrace{\sum_{m=1}^{M} w_m \sum_{h=0}^{H-1} \mathop{\mathbb{E}}_{(s,a)\sim d^\star_{m,h}} \left[(g_t^\star - g_t)^\top \nabla_\theta \ln \pi_t(a|s)\right]}_{②}.\end{aligned}$$

Since $\sum_{m=1}^{M} w_m \sum_{h=0}^{H-1} \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} d^\star_{m,h}(s,a) = H$, normalize the coefficients and apply Jensen's inequality, then

$$① \le \sqrt{\sum_{m=1}^{M} w_m \sum_{h=0}^{H-1} \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} d^\star_{m,h}(s,a)} \cdot \sqrt{\sum_{m=1}^{M} w_m \sum_{h=0}^{H-1} \mathop{\mathbb{E}}_{(s,a)\sim d^\star_{m,h}} \left[\left(A^{t,\lambda}_{m,h}(s,a) - g_t^{\star\top} \nabla_\theta \ln \pi_t(a|s)\right)^2\right]}$$

$$= \sqrt{HL(g_t^\star; \theta_t, d^\star)}.$$

Similarly,

$$\begin{aligned}
② &\leq \sqrt{H \sum_{m=1}^{M} w_m \sum_{h=0}^{H-1} \mathop{\mathbb{E}}_{(s,a)\sim d^\star_{m,h}} \left[\left((g_t^\star - g_t)^\top \nabla_\theta \ln \pi_t(a|s) \right)^2 \right]} \\
&= \sqrt{H \sum_{m=1}^{M} w_m \sum_{h=0}^{H-1} \mathop{\mathbb{E}}_{(s,a)\sim d^\star_{m,h}} \left[(g_t^\star - g_t)^\top \nabla_\theta \ln \pi_t(a|s) (\nabla_\theta \ln \pi_t(a|s))^\top (g_t^\star - g_t) \right]} \\
&\overset{\text{(i)}}{=} \sqrt{H \|g_t^\star - g_t\|^2_{\Sigma^t_{d^\star}}} \\
&\leq \sqrt{H\kappa \|g_t^\star - g_t\|^2_{\Sigma_t}},
\end{aligned}$$

where in (i), for vector v, denote $\|v\|_A = \sqrt{v^\top A v}$ for a symmetric positive semi-definite matrix A. Due to that $g_t^\star$ minimizes $L(g; \theta_t, d^t)$ over the set $\mathcal{G}$, the first-order optimality condition implies that

$$(g - g_t^\star)^\top \nabla_g L(g_t^\star; \theta_t, d^t) \geq 0$$

for any g. Therefore,

$$\begin{aligned}
&L(g; \theta_t, d^t) - L(g_t^\star; \theta_t, d^t) \\
&= \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a\sim d^t_{m,H-h}} \left[\left(A^{t,\lambda}_{m,h}(s,a) - g_t^{\star\top} \nabla \ln \pi_t(a|s) + (g_t^\star - g)^\top \nabla \ln \pi_t(a|s) \right)^2 \right] - L(g_t^\star; \theta_t, d^t) \\
&= \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a\sim d^t_{m,H-h}} \left[\left((g_t^\star - g)^\top \nabla_\theta \ln \pi_t(a|s) \right)^2 \right] \\
&\quad + (g - g_t^\star)^\top \left(-2 \sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a\sim d^t_{m,H-h}} \left[\left(A^{t,\lambda}_{m,h}(s,a) - g_t^{\star\top} \nabla_\theta \ln \pi_t(a|s) \right) \nabla_\theta \ln \pi_t(a|s) \right] \right) \\
&= \|g_t^\star - g\|^2_{\Sigma_t} + (g - g_t^\star)^\top \nabla_g L(g_t^\star; \theta_t, d^t) \\
&\geq \|g_t^\star - g\|^2_{\Sigma_t}.
\end{aligned}$$

So finally we have

$$\text{err}_t \leq \sqrt{HL(g_t^\star; \theta_t, d^\star)} + \sqrt{H\kappa (L(g_t; \theta_t, d^t) - L(g_t^\star; \theta_t, d^t))}.$$

This completes the proof. ■

E.3. Bounding ϵ_{stat}

Lemma 17 (Hoeffding's Inequality) *Suppose $X_1, X_2, \ldots, X_n$ are i.i.d. random variables taking values in $[a, b]$, with expectation μ. Let $\bar{X}$ denote their average, then for any $\epsilon \geq 0$,*

$$\mathbb{P}\left(\left| \bar{X} - \mu \right| \geq \epsilon \right) \leq 2 \exp\left(-\frac{2n\epsilon^2}{(b-a)^2} \right).$$

Lemma 18 *For any policy π, any state $s \in \mathcal{S}$ and any $U \geq \ln|\mathcal{A}| - 1$,*

$$0 \leq \sum_{a\in\mathcal{A}} \pi(a|s) \ln \frac{1}{\pi(a|s)} - \sum_{a\in\mathcal{A}} \pi(a|s) \min\left\{\ln \frac{1}{\pi(a|s)}, U\right\} \leq \frac{|\mathcal{A}|}{\mathrm{e}^{U+1}}.$$

Proof The first inequality is straightforward, so we focus on the second part. Set $\mathcal{A}' = \{a \in \mathcal{A} : \ln \frac{1}{\pi(a|s)} > U\} = \{a \in \mathcal{A} : \pi(a|s) < \frac{1}{\mathrm{e}^U}\}$ and $p = \sum_{a\in\mathcal{A}'} \pi(a|s)$, then

$$\begin{aligned}
\sum_{a\in\mathcal{A}} \pi(a|s) \ln \frac{1}{\pi(a|s)} - \sum_{a\in\mathcal{A}} \pi(a|s) \min\left\{\ln \frac{1}{\pi(a|s)}, U\right\} &= \sum_{a\in\mathcal{A}'} \pi(a|s) \ln \frac{1}{\pi(a|s)} - \sum_{a\in\mathcal{A}'} \pi(a|s)U \\
&= p \sum_{a\in\mathcal{A}'} \frac{\pi(a|s)}{p} \ln \frac{1}{\pi(a|s)} - pU \\
&\leq p \ln\left(\sum_{a\in\mathcal{A}'} \frac{\pi(a|s)}{p} \frac{1}{\pi(a|s)}\right) - pU \\
&\leq p \ln \frac{|\mathcal{A}|}{p} - pU,
\end{aligned}$$

where the penultimate step comes from concavity of $\ln x$ and Jensen's inequality. Let $f(p) = p \ln \frac{|\mathcal{A}|}{p} - pU$, then $f'(p) = \ln|\mathcal{A}| - U - 1 - \ln p$. Recall that $U \geq \ln|\mathcal{A}| - 1$, so $f(p)$ increases when $p \in (0, \frac{|\mathcal{A}|}{\mathrm{e}^{U+1}})$ and decreases when $p \in (\frac{|\mathcal{A}|}{\mathrm{e}^{U+1}}, 1)$. Since $f(\frac{|\mathcal{A}|}{\mathrm{e}^{U+1}}) = \frac{|\mathcal{A}|}{\mathrm{e}^{U+1}}$ we complete the proof. ■

Lemma 19 (Loss Function Concentration) *If set π_s = None and $U \geq \ln|\mathcal{A}| - 1$, then with probability $1 - 2(T+1)\exp\left(-\frac{2N\epsilon^2}{C^2}\right)$, the update weight sequence of Alg. 1 satisfies: for any $0 \leq t \leq T$,*

$$L(\widehat{g}_t; \theta_t, d^{\theta_t}) - L(g_t^\star; \theta_t, d^{\theta_t}) \leq 2\epsilon + \frac{8\lambda GB|\mathcal{A}|}{\mathrm{e}^{U+1}},$$

where

$$C = 16HGB[1 + \lambda U + H(1 + \lambda \ln|\mathcal{A}|)] + 4HG^2B^2.$$

If $\pi_s \neq$ None and $\lambda = 0$, then with probability $1 - 2(T+1)\exp\left(-\frac{2N\epsilon^2}{C^2}\right)$, the update weight sequence of Alg. 1 satisfies: for any $0 \leq t \leq T$,

$$L(\widehat{g}_t; \theta_t, \widetilde{d}^{\pi_s}) - L(g_t^\star; \theta_t, \widetilde{d}^{\pi_s}) \leq 2\epsilon,$$

where

$$C = 16H^2GB + 4HG^2B^2.$$

Proof We first prove the $\pi_s = \text{None}$ case. For time step t, Alg. 1 samples HN trajectories. Abusing the notation, denote

$$\widehat{F}_t = \frac{1}{N}\sum_{n=1}^{N}\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_{n,h}|s_{n,h}) \left(\nabla_\theta \ln \pi_\theta(a_{n,h}|s_{n,h})\right)^\top,$$

$$\widehat{\nabla}_t = \frac{1}{N}\sum_{n=1}^{N}\sum_{h=0}^{H-1} \widehat{A}_{n,H-h}(s_{n,h},a_{n,h}) \nabla_\theta \ln \pi_\theta(a_{n,h}|s_{n,h}),$$

$$\widehat{L}(g) = \underbrace{\sum_{m=1}^{M} w_m \sum_{h=1}^{H} \mathop{\mathbb{E}}_{s,a\sim d^{\theta_t}_{m,H-h}} \left[A^{t,\lambda}_{m,h}(s,a)^2\right]}_{①} + \underbrace{g^\top \widehat{F}_t g - 2g^\top \widehat{\nabla}_t}_{②}.$$

Notice that ① is a constant. From Alg. 1, $\widehat{g}_t$ is the minimizer of ② (hence $\widehat{L}(g)$) inside the ball $\mathcal{G}$. From $\nabla_\theta \ln \pi_\theta(a|s) = \phi(s,a) - \mathbb{E}_{a'\sim\pi_\theta(\cdot|s)}[\phi(s,a')]$, $\|\phi(s,a)\|_2 \le B$, $\|g\|_2 \le G$, we know that $\left|g^\top \nabla_\theta \ln \pi_\theta(a|s)\right| \le 2GB$. So $0 \le g^\top \widehat{F}_t g \le 4HG^2B^2$. From Alg. 2, we know that any sampled $\widehat{A}$ satisfies $|\widehat{A}| \le 2[1+\lambda U + H(1+\lambda \ln|\mathcal{A}|)]$. So $|g^\top \widehat{\nabla}_t| \le 4HGB[1+\lambda U + H(1+\lambda\ln|\mathcal{A}|)]$. We first have that

$$-8HGB[1+\lambda U + H(1+\lambda\ln|\mathcal{A}|)] \le ② \le 8HGB[1+\lambda U + H(1+\lambda\ln|\mathcal{A}|)] + 4HG^2B^2. \tag{2}$$

To apply any standard concentration inequality, we next need to calculate the expectation of ②. According to Monte Carlo sampling and Lem. 18, for any $1 \le m \le M, 1 \le h \le H$ and $(s,a) \in \mathcal{S}\times\mathcal{A}$, we have

$$A^{t,\lambda}_{m,h}(s,a) - \frac{\lambda|\mathcal{A}|}{\mathrm{e}^{U+1}} \le \mathbb{E}\left[\widehat{A}^{t,\lambda}_{m,h}(s,a)\right] \le A^{t,\lambda}_{m,h}(s,a).$$

Denote ∇_t as the exact policy gradient at time step t, then

$$\begin{aligned}\left|\mathbb{E}\left[g^\top\widehat{\nabla}_t\right] - g^\top\nabla_t\right| &\le \|g\|_2 \left\|\mathbb{E}\left[\widehat{\nabla}_t\right] - \nabla_t\right\|_2 \\ &\le \|g\|_2 \cdot H\|\nabla_\theta \ln\pi_\theta(a|s)\|_2 \left\|\mathbb{E}\left[\widehat{A}(s,a)\right] - A(s,a)\right\|_\infty \\ &\le \frac{2\lambda GB|\mathcal{A}|}{\mathrm{e}^{U+1}}.\end{aligned}$$

Since Monte Carlo sampling correctly estimates state-action visitation distribution, $\mathbb{E}\left[\widehat{F}_t\right] = F(\theta_t)$. Notice that $g^\top\widehat{F}_t g$ is linear in entries of $\widehat{F}_t$, we have $\mathbb{E}\left[g^\top\widehat{F}_t g\right] = g^\top F(\theta_t) g$. Now we are in the position to show that

$$\left|\mathbb{E}\left[\widehat{L}(g)\right] - L(g)\right| \le \frac{4\lambda GB|\mathcal{A}|}{\mathrm{e}^{U+1}}.$$

Hoeffding's inequality (Lem. 17) gives

$$\mathbb{P}\left(\left|\widehat{L}(g) - \mathbb{E}\left[\widehat{L}(g)\right]\right| \ge \epsilon\right) \le 2\exp\left(-\frac{2N\epsilon^2}{C^2}\right).$$

where from Eq. 2,

$$C = 16HGB[1 + \lambda U + H(1 + \lambda \ln |\mathcal{A}|)] + 4HG^2B^2.$$

After applying union bound for all t, with probability $1 - 2(T+1)\exp\left(-\frac{2N\epsilon^2}{C^2}\right)$ the following holds for any $g \in \mathcal{G}$:

$$\left|\widehat{L}(g; \theta_t, d^{\theta_t}) - L(g; \theta_t, d^{\theta_t})\right| \leq \epsilon + \frac{4\lambda GB|\mathcal{A}|}{\mathrm{e}^{U+1}}.$$

Hence

$$\begin{aligned} L(\widehat{g}_t; \theta_t, d^{\theta_t}) &\leq \widehat{L}(\widehat{g}_t; \theta_t, d^{\theta_t}) + \epsilon + \frac{4\lambda GB|\mathcal{A}|}{\mathrm{e}^{U+1}} \\ &\leq \widehat{L}(g_t^\star; \theta_t, d^{\theta_t}) + \epsilon + \frac{4\lambda GB|\mathcal{A}|}{\mathrm{e}^{U+1}} \\ &\leq L(g_t^\star; \theta_t, d^{\theta_t}) + 2\epsilon + \frac{8\lambda GB|\mathcal{A}|}{\mathrm{e}^{U+1}}. \end{aligned}$$

For $\pi_s \neq$ None and $\lambda = 0$, we notice that $|\widehat{A}| \leq 2H$ and hence $-8H^2GB \leq ② \leq 8H^2GB + 4HG^2B^2$. Moreover, $\mathbb{E}\left[\widehat{A}^{t,\lambda}_{m,h}(s,a)\right] = A^{t,\lambda}_{m,h}(s,a)$. So by slightly modifying the proof we can get the result. ■

Abbreviation	Detailed setup	Script
`fix_samp_curl`	**Fix**ed **samp**ler **cur**riculum **l**earning. In the warm-up phase, train a policy π_s from scratch (with zero initialization in parameters) using a small environment E'. In the final phase, change to the true environment E, use π_s as the sampler policy to train a policy from scratch.	Run Alg. 3 with $samp =$ `pi_s` and $\lambda = 0$.
`fix_samp_curl_reg`	The same as `fix_samp_curl`, but add entropy **reg**ularization to both phases.	Run Alg. 3 with $samp =$ `pi_s` and $\lambda \neq 0$.
`direct`	**Direct** learning. Only the final phase. Train a policy from scratch directly in E.	Run Alg. 1 with $\theta_0 = 0^d$, $\pi_s =$ None and $\lambda = 0$.
`direct_reg`	The same as `direct`, but add entropy **reg**ularization.	Run Alg. 1 with $\theta_0 = 0^d$, $\pi_s =$ None and $\lambda \neq 0$.
`naive_samp`	Learning with the **naïve samp**ler. Only the final phase. Use the naïve random policy as the sampler to train a policy from scratch in E.	Run Alg. 1 with $\theta_0 = 0^d$, $\pi_s =$ naïve random policy and $\lambda = 0$.
`naive_samp_reg`	The same as `naive_samp`, but add entropy **reg**ularization.	Run Alg. 1 with $\theta_0 = 0^d$, $\pi_s =$ naive random policy and $\lambda \neq 0$.
`curl`	**Cur**riculum **l**earning. In the warm-up phase, train a policy π_s from scratch in E'. In the final phase, change to E and continue on training π_s.	Run Alg. 3 with $samp =$ `pi_t` and $\lambda = 0$.
`curl_reg`	The same as `curl`, but add entropy **reg**ularization.	Run Alg. 3 with $samp =$ `pi_t` and $\lambda \neq 0$.
`reference`	This is the **reference** policy. For SP, it is exactly the optimal policy since it can be calculated. For OKD, it is a bang-per-buck policy and is not the optimal policy (whose exact form is not clear).	N/A

Table 1: Detailed setups for each training scheme.