EXACT FAST BATCH SIMULATION FOR TABULAR REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Simulation is a fundamental computational primitive in reinforcement learning (RL), yet conventional simulation explicitly generates individual trajectories even when downstream procedures use only aggregate statistics. To address this, we develop an exact fast-simulation framework for finite-horizon tabular Markov decision processes. Our framework has two complementary modes. In direct batch simulation, a batch is represented by its aggregate Markov flow. With sufficient parallel simulation resources, this flow can be obtained by trajectory aggregation; when such simulation is unavailable or costly but the initial state and transition distributions are directly accessible, we instead generate an identically distributed flow through forward Markov-flow sampling without materializing individual trajectories. The latter reduces the simulator-side computational dependence on batch size m from $O(m)$ to $O(1)$. In adaptive batch simulation, when batch length is determined by a data-dependent condition, exact multivariate-hypergeometric splitting recursively refines a candidate Markov flow while preserving the conditional law, reducing the cost dependence on m from $O(m)$ to $O(\log m)$. Together, these modes accelerate simulation by keeping trajectories aggregated whenever possible and refining flows only when required to locate data-dependent boundaries. The framework applies broadly across simulator-based, offline, and online batch or stage-based RL, as illustrated with representative algorithms from each setting.

1 INTRODUCTION

Reinforcement learning (RL) (Sutton & Barto, 2018) has become a useful framework for sequential decision-making, with applications ranging from games (Vinyals et al., 2019) and robotics (Gu et al., 2017) to autonomous driving (Yurtsever et al., 2020). A canonical theoretical setting is the finite-horizon tabular Markov decision process (MDP), where an agent generates trajectories of H consecutive state-action transitions according to a policy and the underlying transition kernel.

Simulation is a fundamental computational primitive in RL. It is used to generate training data, construct offline datasets, evaluate learned policies, and empirically study algorithmic properties such as regret scaling, sample complexity, visitation frequencies, and switching behavior. The conventional simulation interface operates at the level of individual trajectories: it samples an initial state, repeatedly selects an action and draws a next state, materializes the resulting trajectory, and then repeats this procedure episode by episode.

This episode-level interface can be unnecessarily restrictive for RL procedures that naturally operate in batches. In particular, the sampling policy often remains unchanged over multiple consecutive episodes before the learner performs its next update. We formalize this structure as follows.

Definition 1.1 (Batch and batch-implementable update)**.** *A batch is a consecutive collection of episodes generated under the same Markov policy π. The policy may depend on the pre-batch learner state, which collects all information maintained by the learning algorithm before the batch begins, but remains unchanged within the batch. The batch endpoint may be predetermined or be the first episode at which a specified data-dependent boundary condition is met.*

A learner update is batch-implementable if it can be computed exactly from the pre-batch learner state, including any accumulators retained from previous batches, and aggregate information from the current batch, without explicitly processing its episodes one by one.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Many RL procedures naturally admit this form of batching. Simulator-based methods may repeatedly query the same transition distributions only to construct empirical models or statistics (Gheshlaghi Azar et al., 2013; Sidford et al., 2018; Li et al., 2024b); in this case, the raw next-state samples are ultimately used only through empirical transition counts. Offline RL often operates on trajectories collected under a fixed behavior policy (Yin et al., 2021; Yin & Wang, 2021; Shi et al., 2022); here, many downstream computations depend only on aggregate occupancy, visitation, and transition statistics. Batch or stage-based online algorithms may retain the same decision rule while accumulating samples until an update is triggered (Bai et al., 2019; Zhang et al., 2020; Qiao et al., 2022; Zhang et al., 2022; Zheng et al., 2024; 2025a; Zhang et al., 2025b); between updates, the trajectories are often used only to update visit counts, empirical moments, or other batch-wise accumulators. In all three settings, explicitly generating every individual sample before forming the required aggregate summaries can introduce unnecessary simulator-side computation.

This issue becomes particularly important when RL simulation is performed at scale. In theoretical studies, large numbers of episodes, repeated random seeds, multiple MDP instances, and extensive parameter sweeps are often required to reveal finite-sample scaling laws or distinguish closely related algorithmic behaviors. For example, recent tabular RL experiments (Zhang et al., 2025a) simulate millions of episodes in order to examine regret and switching behavior. Large simulation workloads also arise in practical RL systems: modern simulators such as Isaac Gym train policies using thousands of parallel environments and tens to hundreds of millions of environment steps (Makoviychuk et al., 2021). To handle such workloads, systems including Isaac Gym (Makoviychuk et al., 2021), Brax (Freeman et al., 2021), and EnvPool (Weng et al., 2022) improve throughput by parallelizing conventional environment stepping. These approaches substantially accelerate trajectory-level simulation, but still explicitly generate individual transitions. This motivates a complementary direction for finite-horizon tabular MDPs:

Can we exploit aggregate batch structure to accelerate simulation while exactly preserving the distribution needed by the learning procedure?

We answer this question affirmatively through two simulation modes, depending on how the batch endpoint is determined. When the endpoint is known in advance, we use *direct batch simulation*. A predetermined batch is represented by its *Markov-flow counts*: state counts, state-action counts, and state-action-next-state transition counts. There are two ways to obtain this flow. When sufficient parallel simulation resources are available, the trajectories can be generated concurrently and then aggregated into the Markov flow. When such parallel simulation is unavailable or costly, but the initial state and transition distributions are accessible, we instead generate an identically distributed Markov flow directly, without materializing individual trajectories, using forward Markov-flow sampling. This reduces the simulator-side computational dependence on the batch size from $O(m)$ to $O(1)$.

When the batch endpoint is determined by a data-dependent boundary condition and is therefore unknown in advance, we use *adaptive batch simulation*. This situation arises in online algorithms whose next update is triggered, for example, when an accumulated visit count reaches a stage boundary or a reference statistic is refreshed. Under the current policy, we first simulate a large candidate batch in aggregate form and then use exact multivariate-hypergeometric splitting to recursively locate the first boundary episode. The aggregate flow up to this boundary is retained, the corresponding learner updates are applied, and simulation restarts from the updated learner state. Because each split preserves the correct conditional law, the resulting procedure is distributionally identical to conventional sequential simulation, while locating the boundary within a candidate batch of m episodes requires only $O(\log m)$ refinement levels instead of $O(m)$ episode-wise inspections.

The resulting principle is simple: *represent trajectory batches by aggregate Markov flows whenever possible, and apply exact hypergeometric splitting only when a data-dependent boundary must be located*. Although our main results are stated for finite-horizon tabular MDPs, the construction is not tied to episodic dynamics and also applies to independent finite-length rollouts of tabular MDPs.

Our contributions. Our main contributions are summarized as follows.

Exact direct batch simulation. We formulate the Markov-flow representation of a batch in Section 3.1. When sufficient parallel simulation resources are available, the batch can be generated concurrently and aggregated directly into its Markov flow. When such parallel simulation is unavailable or costly, but the initial state and transition distributions are accessible, we instead show that the aggregate state, state-action, and transition counts of m episodes can be sampled directly

through forward Markov-flow sampling without materializing individual trajectories, while exactly preserving the joint flow distribution of conventional episode-by-episode simulation (Theorem 3.1). Any flow-measurable batch accumulator can then be computed from the resulting aggregate counts. Under the computational convention of Section 4, this direct flow-sampling implementation reduces the simulator-side dependence on m from $O(m)$ to $O(1)$ (see Proposition 4.1).

Exact adaptive batch simulation. For a prefix-monotone data-dependent boundary, we develop a multivariate-hypergeometric splitting procedure in Section 3.2 that samples exact conditional subflows of a generated aggregate flow. Recursive binary splitting locates the first boundary in at most $\lceil \log_2 m \rceil$ refinements and recovers the aggregate flow up to that boundary (Theorem 3.5). Under the boundary compatibility condition specified in Section 2, restarting after each update exactly reproduces the joint distribution of the post-boundary learner states. When candidate flows are generated by forward Markov-flow sampling, the simulator-side dependence on m is reduced from $O(m)$ to $O(\log m)$ (see Proposition 4.2 for details).

Applications across representative RL settings. In Section 5, we demonstrate the framework in simulator-based, offline, and online RL through model-based planning with a generative model (Li et al., 2024b), VPVI (Yin & Wang, 2021), and stage-based UCB-Advantage (Zhang et al., 2020), respectively. Model-based planning uses direct transition-count sampling, VPVI uses direct Markov-flow generation for a fixed offline dataset, and UCB-Advantage uses adaptive batch simulation to locate data-dependent update boundaries. Our experiments further demonstrate the practical computational efficiency of our framework compared with conventional episode-by-episode simulation.

Technical novelty. A key technical ingredient of our analysis is a conditional labeling representation of an aggregate Markov flow. Conditional on the full flow, all compatible labeled trajectory sequences are uniformly distributed and admit a sequential combinatorial representation (Lemmas D.1 and D.2). Building on this representation, we derive multivariate hypergeometric decompositions that recover the exact conditional law of a contiguous subbatch despite dependencies across successive MDP steps (Lemmas D.3–D.5). We further show that this conditional structure is preserved under recursive refinement, enabling repeated splitting and establishing distributional equivalence with sequential simulation across successive data-dependent boundaries.

2 Problem Formulation

Finite-horizon tabular Markov decision process. We consider a finite-horizon tabular Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, P, r, \nu)$, where $\mathcal{S}$ is the state space with $|\mathcal{S}| = S$, $\mathcal{A}$ is the action space with $|\mathcal{A}| = A$, and H is the horizon. The transition kernel is $P = \{P_h\}_{h=1}^H$, where $P_h(\cdot \mid s, a)$ specifies the distribution of the next state given (s, a) at step h. The reward function is $r = \{r_h\}_{h=1}^H$, where each $r_h : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is deterministic, and $\nu \in \Delta(\mathcal{S})$ denotes the initial-state distribution.

At the beginning of each episode, $X_1 \sim \nu$. At step $h \in [H]$, the agent observes $X_h \in \mathcal{S}$, selects $A_h \in \mathcal{A}$, receives $r_h(X_h, A_h)$, and transitions according to $X_{h+1} \sim P_h(\cdot \mid X_h, A_h)$. A Markov policy $\pi = \{\pi_h\}_{h=1}^H$ satisfies $\pi_h(\cdot \mid s) \in \Delta(\mathcal{A})$. Whenever a policy is defined greedily from Q-values, ties are broken according to a fixed deterministic rule. For a fixed policy π, one episode generates $\omega = (X_1, A_1, \ldots, X_H, A_H, X_{H+1})$, and for any realization $\omega = (x_1, a_1, \ldots, x_H, a_H, x_{H+1})$,

$$\mathbb{P}_\pi(\omega) = \nu(x_1) \prod_{h=1}^H \pi_h(a_h \mid x_h) P_h(x_{h+1} \mid x_h, a_h). \tag{1}$$

Markov-flow counts. Fix the learner state Z at the beginning of a batch as in Definition 1.1, and let $\pi = \pi(Z)$ be the policy used throughout the batch. Conditional on Z, consider m labeled trajectories $\omega_1, \ldots, \omega_m \overset{\text{iid}}{\sim} \mathbb{P}_\pi$. Writing $\omega_i = (X_{i,1}, A_{i,1}, \ldots, X_{i,H}, A_{i,H}, X_{i,H+1})$, define

$$N_h(s) = \sum_{i=1}^m \mathbb{I}\{X_{i,h} = s\}, \qquad C_h(s, a) = \sum_{i=1}^m \mathbb{I}\{X_{i,h} = s, A_{i,h} = a\},$$
$$E_h(s, a, s') = \sum_{i=1}^m \mathbb{I}\{X_{i,h} = s, A_{i,h} = a, X_{i,h+1} = s'\}.$$

We call $F_m = (N_1, C, E)$ the *Markov flow* of the batch, where $C = \{C_h(s, a)\}_{h,s,a}$ and $E = \{E_h(s, a, s')\}_{h,s,a,s'}$. For two Markov flows $F = (N_1, C, E)$ and $F' = (N_1', C', E')$, addition and subtraction are defined as $F \pm F' = (N_1 \pm N_1',\ C \pm C',\ E \pm E')$, where $(N_1 \pm N_1')(s) = N_1(s) \pm N_1'(s)$, $(C \pm C')_h(s, a) = C_h(s, a) \pm C_h'(s, a)$, $(E \pm E')_h(s, a, s') = E_h(s, a, s') \pm E_h'(s, a, s')$.

Flow-measurable accumulators. Given the pre-batch learner state Z, we call a batch statistic or accumulator *flow measurable* if it can be written as a deterministic measurable function $\phi(Z, F_m)$. Examples include state-action visitation counts, empirical transition kernels, and sums of a fixed function of the next state, such as

$$\sum_{i=1}^{m} \mathbb{I}\{X_{i,h} = s, A_{i,h} = a\} f_{h+1}(X_{i,h+1}) = \sum_{s'} E_h(s, a, s') f_{h+1}(s'), \tag{2}$$

whenever f_{h+1} is fixed by Z throughout the batch. Thus, whenever the aggregate information required by a batch-implementable update is flow measurable, the update can be carried out directly from F_m without reconstructing individual trajectories.

Adaptive boundaries and boundary compatibility condition. For adaptive simulation, let G_j denote the Markov flow of the first j episodes generated under the current policy. A trigger $g(Z, G_j) \in \{0, 1\}$ is *prefix monotone* if $g(Z, G_j) = 1$ implies $g(Z, G_{j'}) = 1$ for every $j' \geq j$. Cumulative threshold rules, including staggered visit-count or stage boundaries associated with different state-action-step triples, satisfy this property. Define the first boundary by

$$\tau = \inf\{j \geq 1 : g(Z, G_j) = 1\}, \tag{3}$$

with $\inf \emptyset = \infty$. Once the boundary is identified, adaptive simulation retains the aggregate flow G_τ of all episodes up to the boundary. We require that all learner updates triggered at this boundary can be computed exactly from (Z, G_τ) and that postponing these updates until G_τ has been recovered does not alter any subsequent action selection or accumulator contribution within the boundary episode. We refer to these requirements collectively as the *boundary compatibility condition*. After applying the triggered updates, simulation restarts from the resulting learner state.

3 Exact Fast Simulation Framework

3.1 Direct Batch Simulation

We first consider a predetermined batch of size m, with the goal of obtaining its Markov flow F_m. The same aggregate-count perspective also applies to repeated generative-model queries. We consider two complementary ways of obtaining F_m.

Parallel aggregation. When sufficient parallel simulation resources are available, the m episodes of the batch can be generated concurrently and aggregated into F_m.

Forward Markov-flow sampling. When such parallel simulation is unavailable or costly, but the transition kernels and initial-state distribution are directly accessible, we can instead generate an identically distributed flow without materializing individual trajectories using the forward Markov-flow sampling procedure developed below.

Sample the initial-state counts according to

$$N_1(\cdot) \sim \text{Multinomial}(m, \nu). \tag{4}$$

Then, perform the sampling and propagation steps in Equations (5) to (7) for $h = 1, \ldots, H$ to produce the Markov flow $F_m = (N_1, C, E)$.

Conditional on N_h, sample

$$C_h(s, \cdot) \mid N_h \sim \text{Multinomial}\left(N_h(s), \pi_h(\cdot \mid s)\right), \qquad s \in \mathcal{S}, \tag{5}$$

independently across s. Conditional on C_h, sample

$$E_h(s, a, \cdot) \mid C_h \sim \text{Multinomial}\left(C_h(s, a), P_h(\cdot \mid s, a)\right), \qquad (s, a) \in \mathcal{S} \times \mathcal{A}, \tag{6}$$

independently across (s, a), and propagate

$$N_{h+1}(s') = \sum_{s,a} E_h(s, a, s'), \qquad s' \in \mathcal{S}. \tag{7}$$

Theorem 3.1 (Exactness of direct Markov-flow sampling)**.** *Let $F_m^{\text{iid}} = (N_1^{\text{iid}}, C^{\text{iid}}, E^{\text{iid}})$ be the Markov flow obtained from m independent trajectories $\omega_1, \ldots, \omega_m \overset{\text{iid}}{\sim} \mathbb{P}_\pi$. Let $F_m^{\text{fc}} = (N_1^{\text{fc}}, C^{\text{fc}}, E^{\text{fc}})$ be generated by Equations* (4) *to* (7)*. Then $F_m^{\text{fc}} \overset{d}{=} F_m^{\text{iid}}$.*

Equivalently, the joint law of $(N_1, C_1, E_1, \ldots, C_H, E_H)$ is identical under direct Markov-flow sampling and conventional episode-by-episode simulation.

A complete proof is deferred to Appendix C. The exactness of the Markov flow immediately extends to any computation that depends on the batch only through its flow.

Corollary 3.2. *Fix the pre-batch learner state Z. For any measurable map ϕ of Z and the Markov flow, $\phi(Z, F_m^{\mathrm{fc}}) \overset{d}{=} \phi(Z, F_m^{\mathrm{iid}})$. Therefore, any flow-measurable accumulator or batch-implementable update that depends on the batch only through F_m is distributionally identical under direct batch simulation and conventional episode-by-episode simulation.*

3.2 Adaptive Batch Simulation

When the batch endpoint is determined by a data-dependent boundary, its location is unknown in advance. We therefore first obtain the Markov flow of a fixed-size candidate batch under the current policy using direct batch simulation and, if the candidate batch triggers the boundary, recursively refine its flow through exact multivariate-hypergeometric splitting to locate the earliest triggering episode.

Exact multivariate-hypergeometric splitting. Let a parent flow $F_m = (N_1, C, E)$ represent m trajectories, and choose any $m_L \in \{0, \dots, m\}$ with $m_R = m - m_L$. For a count vector b, let $Y \sim \mathrm{MHG}(b, n)$ denote a multivariate hypergeometric random vector obtained by sampling n objects without replacement from categories with counts b.

Define the left child flow recursively by first sampling

$$N_1^L(\cdot) \sim \mathrm{MHG}(N_1(\cdot), m_L)\,. \tag{8}$$

Then, repeat the following procedure for $h = 1, \dots, H$. Conditional on N_h^L, sample

$$C_h^L(s, \cdot) \sim \mathrm{MHG}\big(C_h(s, \cdot), N_h^L(s)\big)\,, \qquad s \in \mathcal{S}, \tag{9}$$

independently across s and, conditional on C_h^L, sample

$$E_h^L(s, a, \cdot) \sim \mathrm{MHG}\big(E_h(s, a, \cdot), C_h^L(s, a)\big)\,, \qquad (s, a) \in \mathcal{S} \times \mathcal{A}, \tag{10}$$

independently across (s, a). Finally, propagate

$$N_{h+1}^L(s') = \sum\nolimits_{s,a} E_h^L(s, a, s'), \qquad s' \in \mathcal{S}.$$

The right child flow is obtained by subtraction, $F_{m_R}^R = F_m - F_{m_L}^L$.

Theorem 3.3 (Conditional correctness of a split)**.** *Let $\omega_1, \dots, \omega_m \overset{\mathrm{iid}}{\sim} \mathbb{P}_\pi$, and let F_m be their aggregate Markov flow. Fix any $m_L \in \{0, \dots, m\}$ and let $m_R = m - m_L$. Denote by $F_{m_L}^{\mathrm{true},L}$ and $F_{m_R}^{\mathrm{true},R}$ the Markov flows of the first m_L and remaining m_R trajectories, respectively, under conventional episode-by-episode simulation. Let $F_{m_L}^{\mathrm{split},L}$ be the left child generated by Equations* (8) *to* (10)*, and define the right child by $F_{m_R}^{\mathrm{split},R} = F_m - F_{m_L}^{\mathrm{split},L}$. Then, conditional on F_m,*

$$\big(F_{m_L}^{\mathrm{split},L}, F_{m_R}^{\mathrm{split},R}\big) \mid F_m \overset{d}{=} \big(F_{m_L}^{\mathrm{true},L}, F_{m_R}^{\mathrm{true},R}\big) \mid F_m.$$

In particular, each child has the correct conditional distribution corresponding to its true sub-batch.

A complete proof is deferred to Appendix D.

Remark 3.4 (Deterministic-policy specialization)**.** *Suppose π is deterministic, and write $\pi_h(s) \in \mathcal{A}$ for the unique action selected at state s and step h. Then Equation* (5) *is deterministic: $C_h(s, \pi_h(s)) = N_h(s)$ and $C_h(s, a) = 0$ for $a \neq \pi_h(s)$, so Equation* (6) *need only be sampled for the active row $(s, \pi_h(s))$. Likewise, in multivariate-hypergeometric splitting, Equation* (9) *is deterministic and Equation* (10) *need only be sampled for the active action. In these deterministic-policy specializations, we represent the inactive action rows implicitly as zero rather than materializing them. This changes neither the Markov flow represented nor the exactness statements above.*

Binary search for the first boundary. For adaptive simulation, we specialize the general splitting rule above to approximately equal child sizes. Algorithm 1 maintains an accepted prefix flow F^{pre} that is known not to have triggered the boundary. At each iteration, the current batch is split at its midpoint. If appending the left child triggers g, the first boundary lies in the left child; otherwise, the left child is accepted, and the search continues in the right child. Thus, the unresolved batch size decreases by approximately one half at each iteration. The exactness of Algorithm 1 is stated in the following theorem and proved in Appendix E.

Theorem 3.5 (Exact adaptive batch simulation)**.** *Fix Z and suppose the policy remains equal to $\pi(Z)$ until the first boundary defined in Equation* (3)*. Let $\widetilde{G}$ denote the flow returned by Algorithm 1. Then*

$$\big(g(Z, F_m), \widetilde{G}\big) \overset{d}{=} \big(\mathbb{I}\{\tau \le m\}, G_{\tau \wedge m}\big).$$

Moreover, the algorithm uses at most $\lceil \log_2 m \rceil$ conditional splits. If the boundary compatibility condition holds, repeatedly applying the procedure at successive boundaries produces the same joint distribution over post-boundary learner states as conventional simulation.

Algorithm 1 Exact adaptive batch simulation by binary refinement

Require: Learner state Z, candidate size m, policy $\pi(Z)$, prefix-monotone trigger g
Sample candidate flow F_m by direct batch simulation
if $g(Z, F_m) = 0$ **then**
  return F_m
end if
$(F, n) \leftarrow (F_m, m)$ and $F^{\mathrm{pre}} \leftarrow 0$
while $n > 1$ **do**
  $n_L \leftarrow \lfloor n/2 \rfloor$ and $n_R \leftarrow n - n_L$
  Split (F, n) into exact prefix and suffix flows (F^L, n_L) and (F^R, n_R)
  if $g(Z, F^{\mathrm{pre}} + F^L) = 1$ **then**
    $(F, n) \leftarrow (F^L, n_L)$
  else
    $F^{\mathrm{pre}} \leftarrow F^{\mathrm{pre}} + F^L$, $(F, n) \leftarrow (F^R, n_R)$
  end if
end while
return boundary flow $\widetilde{G}_\tau = F^{\mathrm{pre}} + F$

4 Computational Cost

The results in Theorems 3.1, 3.3 and 3.5 establish that the proposed procedures exactly preserve the distribution induced by conventional episode-by-episode simulation. We now compare the simulator-side computational cost required to generate these equivalent outputs. For direct batch simulation, our cost analysis focuses on the forward Markov-flow sampling implementation.

Computational convention. Throughout this section, we assume that basic arithmetic on the counts, sampling from binomial and hypergeometric distributions, and sampling a single environment transition each have $O(1)$ cost. Consequently, a q-category multinomial or multivariate-hypergeometric draw costs $O(q)$, independent of the total count.

For a general Markov policy, a forward Markov-flow sample requires one S-category multinomial draw, HS draws over A action categories, and HSA draws over S next-state categories. Thus, its simulator-side computational cost is $C_{\mathrm{flow}} = O(S + HSA + HS^2A) = O(HS^2A)$, independent of the batch size m. The multivariate-hypergeometric splitting procedure in Section 3.2 has the same computational cost. Under the deterministic-policy specialization of Remark 3.4, the action-count draws disappear, and only one S-category transition row is sampled for each (s, h). With inactive action rows represented implicitly as zero, both forward flow sampling and multivariate-hypergeometric splitting therefore have simulator-side cost $C^{\mathrm{det}}_{\mathrm{flow}} = O(S + HS^2) = O(HS^2)$.

For the applications in Section 5, learner updates are computed from aggregate visitation, transition, or moment statistics and introduce no additional dependence on m. In all three applications, the learner-side cost per update is at most $O(HS^2A)$ and is accounted for separately when relevant.

One apparent exception is the broad class of Q-learning methods based on the visit-dependent learning rate $\alpha_t = (H + 1)/(H + t)$ (Jin et al., 2018). A direct implementation applies the learning-rate update after individual visits and therefore appears to retain learner-side computation linear in the batch size m. However, suitably designed round-based variants can aggregate not only the cumulative learning-rate effect but also the trajectory-dependent statistics and confidence bonuses required by the update. As shown in Appendix G using the single-agent specialization of FedQ-Bernstein (Zheng et al., 2024) as a representative example, these quantities can be computed from aggregate round statistics and pre- and post-round visit counts with cost independent of m. Thus, visit-dependent learning rates do not inherently preclude batch-size-independent learner updates.

4.1 Cost of Direct Batch Simulation

Direct batch simulation applies whenever the sampling rule remains fixed over a predetermined batch, and the required output is flow measurable. This includes repeated simulator-based queries used to construct empirical transition models (Gheshlaghi Azar et al., 2013; Sidford et al., 2018; Li et al., 2024b), synthetic offline data generated under a fixed behavior policy (Yin et al., 2021; Yin & Wang, 2021; Shi et al., 2022; Li et al., 2024a), and online algorithms with predetermined batch or policy-update schedules (Qiao et al., 2022; Zhang et al., 2022). In each case, the resulting samples can be represented and processed through aggregate flow statistics, while forward Markov-flow sampling can generate these statistics directly without materializing individual trajectories or transitions.

Proposition 4.1 (Computational cost of direct batch simulation)**.** *For a predetermined batch of m episodes, conventional episode-by-episode simulation has simulator-side computational cost $\Theta(mH)$, whereas direct Markov-flow sampling has cost $C_{\mathrm{flow}} = O(HS^2A)$. If the sampling policy is deterministic, the specialization in Remark 3.4 has cost $C_{\mathrm{flow}}^{\mathrm{det}} = O(HS^2)$.*

Thus, the simulator-side computational cost improves from $O(m)$ to $O(1)$ in its dependence on the batch size m. By Theorem 3.1 and Corollary 3.2, this reduction preserves the distribution of the Markov flow and any flow-measurable computation without generating the individual trajectories.

4.2 Cost of Adaptive Batch Simulation

Adaptive batch simulation is relevant to online algorithms whose policy updates are triggered by data-dependent visitation thresholds (Bai et al., 2019; Zhang et al., 2020). Suppose a run of $K \geq 2$ episodes contains a random number B of adaptive batches, possibly including a terminal partial batch truncated at episode K, with $B \leq B_{\max}$ almost surely. Conventional episode-by-episode simulation generates trajectories sequentially and therefore has simulator-side computational cost $\Theta(HK)$.

Our adaptive procedure instead locates each data-dependent boundary through binary refinement. If k episodes have already been accepted, we choose the candidate size as $m = K - k$. Thus, the initial forward Markov-flow sample has cost C_{flow}, and locating the boundary requires at most $\lceil \log_2 K \rceil$ conditional splits, each with the same computational cost.

Proposition 4.2 (Computational cost of adaptive batch simulation)**.** *Suppose each candidate flow in Algorithm 1 is generated by forward Markov-flow sampling, and the number of data-dependent batches satisfies $B \leq B_{\max}$ almost surely for some deterministic $B_{\max}$. Conventional episode-by-episode simulation incurs simulator-side computational cost $\Theta(HK)$, whereas Algorithm 1 has simulator-side computational cost $O(B_{\max}HS^2A \log K)$. If the policy in every adaptive batch is deterministic, the corresponding cost is $O(B_{\max}HS^2 \log K)$.*

By Theorems 3.3 and 3.5, this reduction preserves the exact distribution of every boundary flow. In stage-based tabular algorithms with geometrically increasing visitation thresholds (Bai et al., 2019; Zhang et al., 2020), the number of updates typically admits a deterministic upper bound logarithmic in K, up to factors in (H, S, A). Thus, when $B_{\max} = O(\log K)$ in its dependence on K, the simulator-side dependence on the number of episodes becomes $O(\log^2 K)$ rather than $O(K)$.

If evaluating the prefix-monotone trigger costs C_g per call, Algorithm 1 makes $O(\log K)$ trigger evaluations per adaptive batch, giving total cost $O(B_{\max}(C_{\mathrm{flow}} + C_g) \log K)$. For UCB-Advantage in Section 5, each adaptive-batch policy is deterministic. Hence, for every (s, h), only the active action $a = \pi_h(s)$ can receive additional visits as the candidate prefix grows, while all inactive-action counts remain zero. The stage-threshold trigger therefore inspects only the HS active state-action-step tuples, while the reference-value part contains only HS state-step conditions. Thus, $C_g = O(HS)$, and together with $C_{\mathrm{flow}}^{\mathrm{det}} = O(HS^2)$, the total deterministic-policy cost remains $O(B_{\max}HS^2 \log K)$.

5 Representative Applications

We now illustrate how the fast-simulation framework applies to three RL settings: simulator-based RL, offline RL, and online RL. The role of simulation differs across these settings. In simulator-based RL, repeated generative-model queries can be sampled directly in aggregate form. In offline RL, datasets generated under a fixed behavior policy can be sampled directly as aggregate Markov flows. In online RL, where the sampling policy evolves with the learner, adaptive batch simulation can fast-forward through intervals between data-dependent update boundaries. Together, these examples demonstrate the applicability of our framework across these three RL settings.

5.1 SIMULATOR-BASED RL: MODEL-BASED PLANNING WITH A GENERATIVE MODEL

Consider model-based planning with a generative model (Li et al., 2024b). Suppose each (s, a, h) is queried exactly J times. Under conventional simulation, the simulator independently draws J next states from $P_h(\cdot \mid s, a)$ one by one and uses them to construct the empirical transition model. Repeating this procedure for all (s, a, h) requires $\Theta(SAHJ)$ simulator-side computational cost.

For this application, we assume simulator-side access to the transition probability vectors $P_h(\cdot \mid s, a)$, which need not be available to the learning algorithm. Under this access model, direct transition-count sampling, corresponding to Equation (6), produces an empirical transition model with the same distribution as conventional generative-model sampling without generating the J individual next-state samples. Specifically, with $C_h(s, a) = J$, Equation (6) gives

$$E_h(s, a, \cdot) \sim \text{Multinomial}(J, P_h(\cdot \mid s, a)), \text{ and thus } \widehat{P}_h(s' \mid s, a) = E_h(s, a, s')/J.$$

Thus, the resulting $\widehat{P}_h(\cdot \mid s, a)$, and hence the full empirical model $\widehat{P}$, has the same distribution as under conventional generative-model sampling. Furthermore, by Proposition 4.1, direct batch simulation has simulator-side cost $O(S^2AH)$, eliminating the dependence on the prescribed query count J while leaving the model-based planning procedure unchanged.

5.2 OFFLINE RL: VANILLA PESSIMISTIC VALUE ITERATION (VPVI)

Consider the offline RL algorithm VPVI (Yin & Wang, 2021). Suppose an offline dataset consists of K trajectories collected under a fixed behavior policy μ. Under conventional simulation, these trajectories are generated one episode at a time, incurring $\Theta(HK)$ simulator-side computational cost.

VPVI builds the empirical transition model from the dataset as $\widehat{P}_h(s' \mid s, a) = E_h(s, a, s')/C_h(s, a)$ whenever $C_h(s, a) > 0$. Under our deterministic-reward setting, the empirical rewards are also determined by the state-action counts. The subsequent pessimistic value iteration depends on the dataset only through these empirical quantities and the visitation counts $C_h(s, a)$.

Since the behavior policy μ is fixed and the dataset size K is predetermined, the aggregate dataset statistics required by VPVI can instead be generated by direct batch simulation. Applying Equations (4) to (7) with policy μ and batch size K directly generates the Markov flow $F_K = (N_1, C, E)$, from which all quantities required by VPVI can be computed without materializing individual trajectories. By Theorem 3.1 and Corollary 3.2, the resulting value estimates and learned policy have the same distribution as under conventional episode-by-episode data generation.

By Proposition 4.1, the simulator-side computational cost is reduced from $\Theta(HK)$ to $O(HS^2A)$, eliminating its dependence on the dataset size K, while leaving the VPVI computation unchanged. If the behavior policy μ is deterministic, the cost further reduces to $O(HS^2)$ by Remark 3.4.

5.3 ONLINE RL: UCB-ADVANTAGE

Consider the online RL algorithm UCB-Advantage (Zhang et al., 2020). Under conventional simulation, the algorithm generates K trajectories sequentially, incurring $\Theta(HK)$ simulator-side computational cost. Unlike the previous two settings, the policy is updated adaptively based on observed data, so the entire interaction cannot be generated as a single predetermined batch. Between updates, however, UCB-Advantage follows the fixed deterministic greedy policy $\pi_h(s) \in \arg\max_{a \in \mathcal{A}} Q_h(s, a)$.

UCB-Advantage updates its value estimates only when prescribed visitation thresholds are reached. For each (s, a, h), visits are partitioned into geometrically growing stages with lengths satisfying $e_1 = H$ and $e_{j+1} = \lfloor (1 + 1/H)e_j \rfloor$. At each stage end, the estimates are updated, and the intra-stage accumulators are reset. The reference value $V_h^{\text{ref}}(s)$ is also updated at prescribed state-level visitation thresholds. We treat these stage and reference-value updates as adaptive boundaries.

Let $n_h(s, a)$ and $n_h(s) = \sum_{a \in \mathcal{A}} n_h(s, a)$ denote the cumulative state-action and state visit counts before the current batch, respectively, and let $b_h^{\text{stage}}(s, a)$ and $b_h^{\text{ref}}(s)$ denote the next active stage and reference thresholds. For the prefix flow G_j, let $C_h^{(j)}(s, a)$ and $N_h^{(j)}(s)$ denote its state-action and state counts, respectively, and define $g_{\text{UCB}}(Z, G_j)$ as

$$\mathbb{I}\big\{\exists (s, a, h) : n_h(s, a) + C_h^{(j)}(s, a) \geq b_h^{\text{stage}}(s, a) \text{ or } \exists (s, h) : n_h(s) + N_h^{(j)}(s) \geq b_h^{\text{ref}}(s)\big\}.$$

Since these visitation counts are nondecreasing in j, the trigger is prefix monotone. Moreover, within the current batch only $C_h^{(j)}(s, \pi_h(s))$ can change for each (s, h); all $C_h^{(j)}(s, a)$ with $a \neq \pi_h(s)$ remain zero. Hence the trigger can be evaluated in $O(HS)$ time as described in Section 4.

Between consecutive boundaries, the policy and the value quantities entering the relevant accumulators remain fixed. Hence the visitation counts and first- and second-moment accumulators are flow measurable through Equation (2). Together with the active thresholds stored in the pre-boundary learner state Z, the flow G_τ therefore determines all updates triggered at the boundary.

UCB-Advantage performs triggered updates immediately at step h of the boundary episode, so the second requirement must be verified. Such an update modifies only Q_h, V_h, and possibly V_h^{ref}, while decisions in the same episode use quantities indexed by $h+1, h+2, \ldots$. The step-h accumulators depend only on V_{h+1} and V_{h+1}^{ref}, which are likewise unaffected. Hence postponing these updates does not alter any subsequent action or accumulator contribution within the boundary episode. Replaying the triggered updates in increasing order of h, with Q_h and V_h updated before $V_h^{\mathrm{ref}}(s)$ when both updates coincide, therefore yields the same post-episode learner state.

Thus, the boundary compatibility condition holds, and adaptive batch simulation can be applied between successive boundaries. By Theorem 3.5, Algorithm 1 reproduces the joint distribution of the post-boundary learner-state sequence under conventional episode-by-episode simulation.

Finally, each (s, a, h) has at most $O(H \log K)$ stages, so the number of adaptive boundaries is $O(H^2 SA \log K)$ including reference-value updates, and hence $B_{\max} = O(H^2 SA \log K)$. Since each batch policy is deterministic, the flow-sampling cost is $O(B_{\max} HS^2 \log K)$ by Proposition 4.2; the $O(HS)$ trigger evaluations are dominated by this cost. Therefore, the simulator-side cost is $O(H^3 S^3 A \log^2 K)$, compared with $\Theta(HK)$ for conventional simulation. Thus, the dependence on K is reduced to $O(\log^2 K)$ while preserving the joint distribution of the post-boundary learner states.

6 Empirical Evaluation

We evaluate our framework on the three applications in Section 5, comparing conventional episode-by-episode simulation with our exact fast simulator across four tabular MDP scales. We examine whether the two simulators produce consistent learner-relevant outcomes and compare their computational costs at different budgets. Full experimental settings and results are deferred to Appendix B.

For model-based planning and VPVI, direct batch simulation produces learning outcomes closely consistent with conventional simulation, including value errors, policy gaps, and empirical transition errors, in agreement with the distributional equivalence established in Corollary 3.2. At the same time, conventional simulation time grows approximately linearly with the number of queries or trajectories, whereas the runtime of direct Markov-flow sampling remains nearly constant with respect to the batch size, consistent with Proposition 4.1. See Appendices B.1 and B.2.

For UCB-Advantage, adaptive batch simulation closely reproduces both cumulative regret and the data-dependent batch-update behavior of conventional simulation, consistent with the distributional equivalence in Theorem 3.5. Although adaptive splitting introduces additional overhead at small budgets, its runtime grows much more slowly with the number of episodes, consistent with Proposition 4.2. In particular, once the algorithm reaches the accuracy regime where the average regret falls below 0.01, the fast simulator is consistently faster across all tested scales, with the computational advantage increasing at larger budgets. Details are provided in Appendix B.3. Overall, the experiments support both the distributional equivalence and computational improvements predicted by our theory.

7 Conclusion

We present an exact fast simulation framework for tabular RL that preserves the relevant distributions of the learning procedure. For predetermined batches, Markov flows can be obtained through parallel trajectory aggregation or, with direct access to the initial-state and transition distributions, through forward Markov-flow sampling without materializing individual trajectories. Exact hypergeometric splitting extends the framework to data-dependent boundaries through adaptive refinement. Under forward Markov-flow sampling, the simulator-side dependence on batch size is reduced from linear to constant for direct simulation and logarithmic for adaptive simulation. Applications to simulator-based, offline, and online RL demonstrate the applicability of our framework.

AI USE STATEMENT

AI tools were used for language polishing, presentation refinement, mathematics proofreading, assistance with method implementation and code debugging, and support in refining the experimental design. All scientific ideas, problem formulation, mathematical claims, proof strategies and key arguments, and interpretation of results were independently developed and verified by the authors.

ETHICS STATEMENT

This work does not involve human subjects, personal data, or real-world experiments requiring ethical approval. The empirical evaluation is conducted entirely on synthetic tabular MDPs. We have followed the guidelines outlined in the ICLR Code of Ethics and have aimed to ensure transparency, integrity, and fairness throughout the research process. We do not foresee any direct ethical concerns or potential harms arising from this study.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide the formal problem setup, simulation framework, and computational assumptions in the main text, together with complete proofs of the main theoretical results in the appendix. The experimental section describes the synthetic tabular MDPs, representative RL algorithms, implementation settings, and evaluation protocol used to compare conventional episode-by-episode simulation with our fast batch-simulation framework. The code for method implementation is included in the supplementary materials.

REFERENCES

Yu Bai, Tengyang Xie, Nan Jiang, and Yu-Xiang Wang. Provably efficient q-learning with low switching cost. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/473803f0f2ebd77d83ee60daaa61f381-Paper.pdf.

Vincent-Pierre Berges, Andrew Szot, Devendra Singh Chaplot, Aaron Gokaslan, Roozbeh Mottaghi, Dhruv Batra, and Eric Undersander. Galactic: Scaling end-to-end reinforcement learning for rearrangement at 100k steps-per-second. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13767–13777, 2023. doi: 10.1109/CVPR52729.2023.01323.

Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/d1f491a404d6854880943e5c3cd9ca25-Paper-round1.pdf.

Mohammad Gheshlaghi Azar, Rémi Munos, and Hilbert J Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3): 325–349, 2013.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3389–3396, 2017. doi: 10.1109/ICRA.2017.7989385.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *Advances in Neural Information Processing Systems*, 31, 2018.

Shivaram Kalyanakrishnan, Sheel Shah, and Santhosh Kumar Guguloth. A view of the certainty-equivalence method for pac rl as an application of the trajectory tree method, 2025. URL https://arxiv.org/abs/2501.02652.

Sanath Kumar Krishnamurthy, Tanmay Gangwani, Sumeet Katariya, Branislav Kveton, Shrey Modi, and Anshuka Rangi. Selective uncertainty propagation in offline rl. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 17974–17982, 2025.

Gen Li, Laixi Shi, Yuxin Chen, Yuejie Chi, and Yuting Wei. Settling the sample complexity of model-based offline reinforcement learning. *The Annals of Statistics*, 52(1):233 – 260, 2024a. doi: 10.1214/23-AOS2342. URL https://doi.org/10.1214/23-AOS2342.

Gen Li, Yuting Wei, Yuejie Chi, and Yuxin Chen. Breaking the sample size barrier in model-based reinforcement learning with a generative model. *Operations Research*, 72(1):203–221, 2024b.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu based physics simulation for robot learning. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/28dd2c7955ce926456240b2ff0100bde-Paper-round2.pdf.

Duc Thien Nguyen, Akshat Kumar, and Hoong Chuin Lau. Credit assignment for collective multiagent rl with global rewards. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/94bb077f18daa6620efa5cf6e6f178d2-Paper.pdf.

Aleksei Petrenko, Zhehui Huang, Tushar Kumar, Gaurav Sukhatme, and Vladlen Koltun. Sample factory: Egocentric 3D control from pixels at 100000 FPS with asynchronous reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7652–7662. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/petrenko20a.html.

Dan Qiao, Ming Yin, Ming Min, and Yu-Xiang Wang. Sample-efficient reinforcement learning with loglog(T) switching cost. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 18031–18061. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/qiao22a.html.

Brennan Shacklett, Erik Wijmans, Aleksei Petrenko, Manolis Savva, Dhruv Batra, Vladlen Koltun, and Kayvon Fatahalian. Large batch simulation for deep reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=cP5IcoAkfKa.

Daniel Sheldon and Thomas Dietterich. Collective graphical models. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/fccb3cdc9acc14a6e70a12f74560c026-Paper.pdf.

Laixi Shi, Gen Li, Yuting Wei, Yuxin Chen, and Yuejie Chi. Pessimistic q-learning for offline reinforcement learning: Towards optimal sample complexity. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 19967–20025. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/shi22c.html.

Aaron Sidford, Mengdi Wang, Xian Wu, Lin Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving markov decision processes with a generative model. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/bb03e43ffe34eeb242a2ee4a4f125e56-Paper.pdf.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, Zhongwen Xu, and Shuicheng Yan. Envpool: A highly parallel reinforcement learning environment execution engine. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22409–22421. Curran Associates, Inc., 2022. doi: 10.52202/068431-1628. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8caaf08e49ddbad6694fae067442ee21-Paper-Datasets_and_Benchmarks.pdf.

Nuoya Xiong, Zhaoran Wang, and Zhuoran Yang. A general framework for sequential decision-making under adaptivity constraints. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54792–54830. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/xiong24d.html.

Ming Yin and Yu-Xiang Wang. Towards instance-optimal offline reinforcement learning with pessimism. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 4065–4078. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/212ab20dbdf4191cbcdcf015511783f4-Paper.pdf.

Ming Yin, Yu Bai, and Yu-Xiang Wang. Near-optimal offline reinforcement learning via double variance reduction. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 7677–7688. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/3f24bb08a5741e4197af64e1f93a5029-Paper.pdf.

Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020. doi: 10.1109/ACCESS.2020.2983149.

Haochen Zhang, Zhong Zheng, and Lingzhou Xue. Gap-dependent bounds for federated q-learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 77107–77151. PMLR, 13–19 Jul 2025a. URL https://proceedings.mlr.press/v267/zhang25dl.html.

Haochen Zhang, Zhong Zheng, and Lingzhou Xue. Regret-optimal q-learning with low cost for single-agent and federated reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 128711–128799. Curran Associates, Inc., 2025b. doi: 10.52202/085713-4288. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/bb203e938836544655996d1bb94a0fd7-Paper-Conference.pdf.

Haochen Zhang, Zhong Zheng, and Lingzhou Xue. Q-learning with fine-grained gap-dependent regret. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 48118–48162, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/4f8d1e02a057beb7920618ac73788078-Paper-Conference.pdf.

Zihan Zhang, Yuan Zhou, and Xiangyang Ji. Almost optimal model-free reinforcement learning via reference-advantage decomposition. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 15198–15207. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/ad71c82b22f4f65b9398f76d8be4c615-Paper.pdf.

Zihan Zhang, Yuhang Jiang, Yuan Zhou, and Xiangyang Ji. Near-optimal regret bounds for multi-batch reinforcement learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24586–24596. Curran Associates, Inc., 2022. doi: 10.52202/068431-1785. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9bcd1fa0c05e5f25ba7a1261f1852e82-Paper-Conference.pdf.

Zhong Zheng, Fengyu Gao, Lingzhou Xue, and Jing Yang. Federated q-learning: Linear regret speedup with low communication cost. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 57399–57443, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/fe318a2b6c699808019a456b706cd845-Paper-Conference.pdf.

Zhong Zheng, Haochen Zhang, and Lingzhou Xue. Federated q-learning with reference-advantage decomposition: Almost optimal regret and logarithmic communication cost. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 22423–22478, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/3882ca2c952276247fe9a993193b00e4-Paper-Conference.pdf.

Zhong Zheng, Haochen Zhang, and Lingzhou Xue. Gap-dependent bounds for q-learning using reference-advantage decomposition. In *The Thirteenth International Conference on Learning Representations*, 2025b.

The appendices are organized as follows. Appendix A discusses related work. Appendix B provides the complete experimental setup and results. Appendix C proves the exactness of direct Markov-flow sampling in Theorem 3.1, Appendix D establishes the conditional correctness of exact flow splitting in Theorem 3.3, and Appendix E proves the exactness of adaptive batch simulation in Theorem 3.5. Appendix F reviews the learning algorithms used in Section 5, and Appendix G discusses further applications to Q-learning with visit-dependent learning rates and federated RL.

A Related Work

Parallel and batched RL simulation. Efficient RL systems improve data-generation throughput by executing many environments concurrently. Sample Factory (Petrenko et al., 2020) combines asynchronous sampling with GPU-based inference. Large Batch Simulation (Shacklett et al., 2021) jointly designs rendering and simulation around batches of requests, sharing scene resources and amortizing rendering, data-loading, and synchronization costs. Galactic (Berges et al., 2023) further optimizes physics, rendering, inference, and learning together for embodied RL. These systems generate environment transitions efficiently through parallelism and batching. Our framework addresses a complementary opportunity in finite tabular MDPs: directly sampling the aggregate statistics of many episodes when the learning procedure does not require their individual trajectories.

Simulation with aggregate counts. Collective graphical models perform probabilistic inference on aggregate sufficient statistics of an individual-level graphical model (Sheldon & Dietterich, 2011). In collective multiagent RL, aggregate state, state-action, and transition counts are used to represent population-level dynamics and perform count-based learner updates (Nguyen et al., 2018). These count-based perspectives are related to our Markov-flow representation. Our framework instead studies exact batch simulation across independent episodes of a learning procedure, formalized through a flow-measurable interface. Beyond direct generation of aggregate statistics, we introduce exact conditional splitting of a previously sampled multi-step flow to locate the first data-dependent update boundary. Under the stated compatibility conditions, updating the learner and restarting simulation preserves the joint distribution of its boundary learner states.

The preceding discussion positions our framework relative to existing simulation methods. We next discuss its relationship to simulator-based, offline, and online RL, together with extensions to federated RL.

Simulator-based RL. Generative models allow transition samples to be queried from specified state-action pairs or state-action-step triples. Prior work studies the sample and computational complexity of learning under this interface (Sidford et al., 2018). Model-based methods estimate transition kernels from repeated queries and plan in the resulting empirical MDP; Li et al. (2024b) establish guarantees covering finite-horizon settings, and Kalyanakrishnan et al. (2025) revisit certainty-equivalence planning through the trajectory tree method. We study the simulator-side cost of realizing the statistics required by a prescribed learning procedure. With simulator-side access to transition probability vectors, the empirical transition counts associated with repeated queries can be sampled directly. This preserves their joint distribution, with computational costs measured under the convention in Section 4.

Offline RL. Offline RL learns from a fixed dataset without further environment interaction. In finite-horizon tabular MDPs, pessimistic value-iteration methods use empirical visitation and transition statistics to construct policies (Yin & Wang, 2021; Li et al., 2024a). The tabular implementation of selective uncertainty propagation also uses empirical transition models and count-based bonuses (Krishnamurthy et al., 2025). These methods motivate direct generation of the statistics needed from synthetic offline datasets. In our VPVI example, the behavior policy and dataset size are fixed, and, under deterministic rewards, the dataset enters the learner only through visitation and transition counts. Direct Markov-flow sampling therefore generates required data statistics without materializing individual trajectories, preserving the distribution of the value estimates and policy.

Batch and low-switching online RL. Another line of work reduces online adaptivity by limiting policy updates. Bai et al. (2019) study tabular Q-learning with low switching cost, while UCB-Advantage (Zhang et al., 2020) combines reference-advantage decomposition with stage-wise updates. Subsequent work further reduces switching complexity (Qiao et al., 2022), studies multi-batch RL with predetermined update schedules (Zhang et al., 2022), and develops a general framework for rare

policy switching and batch learning (Xiong et al., 2024). Recent model-free algorithms improve the tradeoffs among regret, burn-in, and switching costs (Zheng et al., 2025b; Zhang et al., 2025b; 2026). These methods motivate simulation over intervals with a fixed policy. Low switching alone, however, does not ensure flow measurability: visit-index-dependent weights or sequential updates may require additional trajectory-order information. Our adaptive method applies when the required statistics are flow measurable, the trigger is prefix monotone, and the boundary compatibility condition holds. UCB-Advantage provides a concrete example in Section 5.

Federated RL. Event-triggered federated Q-learning achieves logarithmic communication cost in tabular episodic MDPs (Zheng et al., 2024; 2025a), and subsequent work establishes gap-dependent regret guarantees (Zhang et al., 2025a). These settings provide natural applications of our exact batch simulation framework. In Appendix G, we verify the required conditions for FedQ-Bernstein and FedQ-Advantage, illustrating both round-based aggregate updates and distributed local exploration.

B Numerical Experiments

We empirically evaluate our fast simulation framework on model-based planning with a generative model, offline Vanilla Pessimistic Value Iteration (VPVI), and online UCB-Advantage.[1] In particular, we examine whether the learner-relevant quantities produced by the two simulation methods are consistent with the theoretical distributional equivalence and whether the reduced simulation complexity yields substantial computational gains. We consider four tabular MDP sizes:

$$(H, S, A) \in \{(2, 3, 3), (5, 5, 5), (7, 8, 6), (10, 15, 10)\}.$$

For each size, the fixed-MDP experiment holds the underlying MDP fixed and uses ten independent simulation replicates per method to assess the distributional equivalence of learner-relevant quantities. The separate cross-MDP timing experiment uses ten independently generated MDP instances, with the conventional episode-by-episode simulation and exact fast simulation methods evaluated on the same instance and simulation seed, to assess computational efficiency and scaling across problem instances. For each (h, s, a), the transition probability vector $P_h(\cdot \mid s, a)$ is sampled independently from a symmetric Dirichlet distribution with unit concentration; deterministic rewards are drawn independently from $\mathrm{Unif}[0, 1]$, and the initial-state distribution is uniform.

Measurement and uncertainty. Both implementations use the same programming environment, learner implementation, and boundary checks, differing only in the simulation procedure. The simulator time covers only the generation of transition samples or episode batches and the construction of the aggregate statistics passed to the learner. The learner time covers the subsequent algorithmic update, including policy-value computation for performance tracking whenever the policy changes. MDP construction, file I/O, and plotting are excluded from both timings.

For short-running configurations, identified by a pilot wall time below 0.5 seconds, we use two warm-up runs followed by five timed blocks, with each block repeating fresh runs until approximately 0.5 seconds of total runtime is accumulated; the reported time is the median of the per-run runtimes across the five blocks.

Tables report sample means with 95% percentile bootstrap confidence intervals (CIs) for learning statistics and sample medians with 95% percentile bootstrap CIs for timing results, based on 10,000 resamples. Numerical entries in the tables are reported to three significant digits, with scientific notation used where appropriate. For learning statistics, the conventional and fast methods use independent simulation replicates on the same fixed MDP. Confidence intervals for each mean and for the difference in means are therefore constructed by resampling the two groups independently. For timing, the ten MDP instances serve as the sampling units; median runtimes and median same-instance runtime ratios are bootstrapped across instances.

Budget grid. For the distributional comparisons, both the conventional and fast simulators are evaluated on the common grids $J = 10^1, \ldots, 10^6$ for model-based planning and $K = 10^1, \ldots, 10^8$ for VPVI and UCB-Advantage. The timing experiments use the same measured ranges for the conventional simulator, while the fast simulator is further evaluated at every power of ten up to 10^{15}.

[1] The experiments were split by algorithm, scale, and either simulation replicate (fixed-MDP comparison) or MDP instance (cross-MDP timing). Each task ran on one core of an Intel Xeon Gold 6430 with 4 GB of allocated memory and finished within 15 hours.

The extended timing range is motivated by sample-complexity guarantees for the corresponding learning problems. For model-based planning, $\widetilde{O}(H^3/\epsilon^2)$ queries per (h, s, a) suffice to obtain a uniformly ϵ-accurate value function. For VPVI, $\widetilde{O}(H^4SC^\star/\epsilon^2)$ episodes suffice to learn a policy with value gap at most ϵ under the standard coverage condition, where $C^\star$ bounds the state-action occupancy ratio between an optimal policy and the behavior policy. For UCB-Advantage, $\widetilde{O}(H^3SA/\epsilon^2)$ episodes suffice to learn an ϵ-optimal policy. For our largest problem scale, setting $\epsilon = 10^{-4}$ yields aggregate query or episode counts on the order of 10^{13}–10^{14} for planning and UCB-Advantage, before logarithmic and confidence factors, while the VPVI scale additionally depends on the coverage coefficient $C^\star \geq 1$. These estimates motivate probing substantially larger simulation budgets. Accordingly, we extend the fast-simulator timing experiments up to 10^{15} as a large-count stress test.

Because running the conventional simulator beyond the measured range would incur prohibitive computational cost, we extrapolate its runtime under a linear-scaling assumption with respect to the number of queries or episodes. Specifically, starting from the largest directly measured budget, we multiply the measured runtime by ten for each additional order-of-magnitude increase in the budget; all extrapolated entries are marked by $\approx$. This assumption is supported empirically by the approximately linear growth of the measured conventional runtimes over the observed range, and theoretically by the fact that conventional simulation performs work proportional to the number of generated queries or episodes. In contrast, all reported fast-simulation runtimes are obtained from direct measurements. We define speedup as the ratio of the conventional runtime to the fast-simulation runtime, so a value greater than one indicates that the fast simulator is faster.

B.1 Model-Based Planning with a Generative Model

The planning budget J is applied to every (h, s, a), resulting in a total of $HSAJ$ generative-model queries. The conventional and fast methods differ only in how the transition counts are generated; the same planning routine is applied afterward. We use the average optimal-value error

$$E_V(J) = \frac{1}{HS} \sum\nolimits_{h=1}^{H} \sum\nolimits_{s=1}^{S} \left| \widehat{V}_h^\star(s) - V_h^\star(s) \right|,$$

where $\widehat{V}^\star$ and $V^\star$ are the optimal value functions of the empirical and true MDPs, respectively. This continuous statistic directly measures the accuracy of the empirical planning solution. Since the empirical transition counts have the same joint distribution under conventional and fast simulation, the resulting empirical MDP, $\widehat{V}^\star$, and hence $E_V(J)$ are also equal in distribution.

Tables 1–4 show that the conventional and fast simulators achieve very similar value errors across all four problem scales and budgets. Most bootstrap confidence intervals for the difference in mean error contain zero, with 21 of the 24 intervals including zero. These results are consistent with the distributional equivalence predicted by our theory.

The timing results in Tables 5–8 show that conventional simulation is substantially slower and its runtime grows approximately linearly with the query budget J. In contrast, our fast method uses direct batch simulation to sample the transition counts directly, so its simulator runtime remains nearly unchanged as J increases, consistent with our complexity result in Proposition 4.1 that removes the dependence on J. Moreover, the learner time remains below 0.007 seconds in all tested configurations and is substantially smaller than the conventional simulation time at large scales and budgets. This shows that simulation is the dominant computational bottleneck in this regime, making the reduction in simulator-side complexity practically significant.

Table 1: Average optimal-value error for $(H, S, A) = (2, 3, 3)$. Entries report the mean and 95% percentile bootstrap confidence interval over ten independent simulation seeds on a fixed MDP. The error difference is defined as fast minus conventional.

Budget	Conventional	Fast	Difference
10^1	1.41×10^{-2} $[1.08\times10^{-2}, 1.79\times10^{-2}]$	1.08×10^{-2} $[8.27\times10^{-3}, 1.35\times10^{-2}]$	-3.29×10^{-3} $[-7.79\times10^{-3}, 1.01\times10^{-3}]$
10^2	3.58×10^{-3} $[2.72\times10^{-3}, 4.42\times10^{-3}]$	3.21×10^{-3} $[2.37\times10^{-3}, 4.03\times10^{-3}]$	-3.67×10^{-4} $[-1.59\times10^{-3}, 8.24\times10^{-4}]$
10^3	1.30×10^{-3} $[8.96\times10^{-4}, 1.72\times10^{-3}]$	8.97×10^{-4} $[6.69\times10^{-4}, 1.14\times10^{-3}]$	-4.06×10^{-4} $[-8.84\times10^{-4}, 6.27\times10^{-5}]$
10^4	3.79×10^{-4} $[2.91\times10^{-4}, 4.69\times10^{-4}]$	2.90×10^{-4} $[2.34\times10^{-4}, 3.37\times10^{-4}]$	-8.91×10^{-5} $[-1.96\times10^{-4}, 1.17\times10^{-5}]$
10^5	1.25×10^{-4} $[9.90\times10^{-5}, 1.52\times10^{-4}]$	8.20×10^{-5} $[6.27\times10^{-5}, 9.99\times10^{-5}]$	-4.28×10^{-5} $[-7.67\times10^{-5}, -1.16\times10^{-5}]$
10^6	3.51×10^{-5} $[2.38\times10^{-5}, 4.70\times10^{-5}]$	2.49×10^{-5} $[1.80\times10^{-5}, 3.14\times10^{-5}]$	-1.02×10^{-5} $[-2.39\times10^{-5}, 2.96\times10^{-6}]$

Table 2: Average optimal-value error for $(H, S, A) = (5, 5, 5)$. Entries report the mean and 95% percentile bootstrap confidence interval over ten independent simulation seeds on a fixed MDP. The error difference is defined as fast minus conventional.

Budget	Conventional	Fast	Difference
10^1	2.48×10^{-2} $[2.06\times10^{-2}, 2.92\times10^{-2}]$	1.98×10^{-2} $[1.74\times10^{-2}, 2.21\times10^{-2}]$	-4.94×10^{-3} $[-9.89\times10^{-3}, -2.74\times10^{-4}]$
10^2	6.99×10^{-3} $[6.15\times10^{-3}, 7.77\times10^{-3}]$	8.55×10^{-3} $[6.91\times10^{-3}, 1.04\times10^{-2}]$	1.56×10^{-3} $[-2.88\times10^{-4}, 3.62\times10^{-3}]$
10^3	1.91×10^{-3} $[1.57\times10^{-3}, 2.30\times10^{-3}]$	2.07×10^{-3} $[1.71\times10^{-3}, 2.54\times10^{-3}]$	1.55×10^{-4} $[-3.81\times10^{-4}, 7.39\times10^{-4}]$
10^4	7.60×10^{-4} $[6.21\times10^{-4}, 8.89\times10^{-4}]$	6.34×10^{-4} $[5.12\times10^{-4}, 8.00\times10^{-4}]$	-1.27×10^{-4} $[-3.13\times10^{-4}, 9.20\times10^{-5}]$
10^5	2.08×10^{-4} $[1.77\times10^{-4}, 2.36\times10^{-4}]$	2.18×10^{-4} $[1.93\times10^{-4}, 2.45\times10^{-4}]$	1.04×10^{-5} $[-2.74\times10^{-5}, 5.09\times10^{-5}]$
10^6	6.40×10^{-5} $[5.51\times10^{-5}, 7.41\times10^{-5}]$	6.45×10^{-5} $[5.68\times10^{-5}, 7.30\times10^{-5}]$	4.12×10^{-7} $[-1.23\times10^{-5}, 1.27\times10^{-5}]$

Table 3: Average optimal-value error for $(H, S, A) = (7, 8, 6)$. Entries report the mean and 95% percentile bootstrap confidence interval over ten independent simulation seeds on a fixed MDP. The error difference is defined as fast minus conventional.

Budget	Conventional	Fast	Difference
10^1	2.58×10^{-2} $[2.26\times10^{-2}, 2.90\times10^{-2}]$	2.91×10^{-2} $[2.49\times10^{-2}, 3.39\times10^{-2}]$	3.29×10^{-3} $[-2.15\times10^{-3}, 9.02\times10^{-3}]$
10^2	8.29×10^{-3} $[7.45\times10^{-3}, 9.13\times10^{-3}]$	7.45×10^{-3} $[6.70\times10^{-3}, 8.44\times10^{-3}]$	-8.43×10^{-4} $[-1.97\times10^{-3}, 4.28\times10^{-4}]$
10^3	2.53×10^{-3} $[2.31\times10^{-3}, 2.78\times10^{-3}]$	3.07×10^{-3} $[2.67\times10^{-3}, 3.47\times10^{-3}]$	5.48×10^{-4} $[8.15\times10^{-5}, 1.00\times10^{-3}]$
10^4	8.01×10^{-4} $[6.89\times10^{-4}, 9.29\times10^{-4}]$	8.81×10^{-4} $[7.68\times10^{-4}, 1.03\times10^{-3}]$	8.01×10^{-5} $[-8.82\times10^{-5}, 2.64\times10^{-4}]$
10^5	2.68×10^{-4} $[2.45\times10^{-4}, 2.92\times10^{-4}]$	2.58×10^{-4} $[2.31\times10^{-4}, 2.90\times10^{-4}]$	-1.01×10^{-5} $[-4.69\times10^{-5}, 2.88\times10^{-5}]$
10^6	7.63×10^{-5} $[6.71\times10^{-5}, 8.64\times10^{-5}]$	8.77×10^{-5} $[7.68\times10^{-5}, 9.92\times10^{-5}]$	1.14×10^{-5} $[-3.45\times10^{-6}, 2.66\times10^{-5}]$

Table 4: Average optimal-value error for $(H, S, A) = (10, 15, 10)$. Entries report the mean and 95% percentile bootstrap confidence interval over ten independent simulation seeds on a fixed MDP. The error difference is defined as fast minus conventional.

Budget	Conventional	Fast	Difference
10^1	1.95×10^{-2} $[1.76\times10^{-2}, 2.23\times10^{-2}]$	1.70×10^{-2} $[1.53\times10^{-2}, 1.94\times10^{-2}]$	-2.52×10^{-3} $[-5.85\times10^{-3}, 6.62\times10^{-4}]$
10^2	5.24×10^{-3} $[4.92\times10^{-3}, 5.64\times10^{-3}]$	5.73×10^{-3} $[5.34\times10^{-3}, 6.14\times10^{-3}]$	4.81×10^{-4} $[-6.87\times10^{-5}, 1.02\times10^{-3}]$
10^3	1.85×10^{-3} $[1.63\times10^{-3}, 2.15\times10^{-3}]$	1.64×10^{-3} $[1.58\times10^{-3}, 1.71\times10^{-3}]$	-2.05×10^{-4} $[-5.07\times10^{-4}, 2.85\times10^{-5}]$
10^4	5.33×10^{-4} $[5.09\times10^{-4}, 5.58\times10^{-4}]$	5.44×10^{-4} $[4.94\times10^{-4}, 6.04\times10^{-4}]$	1.11×10^{-5} $[-4.46\times10^{-5}, 7.48\times10^{-5}]$
10^5	1.74×10^{-4} $[1.61\times10^{-4}, 1.92\times10^{-4}]$	1.78×10^{-4} $[1.69\times10^{-4}, 1.87\times10^{-4}]$	3.43×10^{-6} $[-1.63\times10^{-5}, 2.02\times10^{-5}]$
10^6	5.40×10^{-5} $[5.07\times10^{-5}, 5.75\times10^{-5}]$	5.37×10^{-5} $[5.02\times10^{-5}, 5.71\times10^{-5}]$	-3.20×10^{-7} $[-5.20\times10^{-6}, 4.42\times10^{-6}]$

Table 5: Simulator time for $(H, S, A) = (2, 3, 3)$, in seconds. Entries are medians with 95% percentile bootstrap CIs over ten MDP instances; speedup is the median of paired time ratios. Conventional entries beyond the last measured 10^6 budget are marked $\approx$ and extrapolated by a factor of ten per additional power of ten.

Budget	Conventional (s)	Fast (s)	Speedup
10^1	3.01×10^{-4} $[2.61\times10^{-4}, 3.33\times10^{-4}]$	2.77×10^{-5} $[2.38\times10^{-5}, 3.04\times10^{-5}]$	1.09×10^{1} $[1.09\times10^{1}, 1.10\times10^{1}]$
10^2	2.81×10^{-3} $[2.51\times10^{-3}, 3.18\times10^{-3}]$	2.93×10^{-5} $[2.56\times10^{-5}, 3.29\times10^{-5}]$	9.76×10^{1} $[9.58\times10^{1}, 9.88\times10^{1}]$
10^3	2.91×10^{-2} $[2.51\times10^{-2}, 3.20\times10^{-2}]$	2.85×10^{-5} $[2.44\times10^{-5}, 3.11\times10^{-5}]$	1.03×10^{3} $[1.02\times10^{3}, 1.03\times10^{3}]$
10^4	2.88×10^{-1} $[2.51\times10^{-1}, 3.21\times10^{-1}]$	2.82×10^{-5} $[2.40\times10^{-5}, 3.06\times10^{-5}]$	1.04×10^{4} $[1.03\times10^{4}, 1.07\times10^{4}]$
10^5	2.90 $[2.52, 3.24]$	2.72×10^{-5} $[2.37\times10^{-5}, 3.10\times10^{-5}]$	1.06×10^{5} $[1.05\times10^{5}, 1.08\times10^{5}]$
10^6	2.89×10^{1} $[2.54\times10^{1}, 3.23\times10^{1}]$	2.77×10^{-5} $[2.38\times10^{-5}, 3.03\times10^{-5}]$	1.07×10^{6} $[1.05\times10^{6}, 1.07\times10^{6}]$
10^7	$\approx 2.89\times10^{2}$	2.79×10^{-5} $[2.38\times10^{-5}, 3.08\times10^{-5}]$	$\approx 1.06\times10^{7}$
10^8	$\approx 2.89\times10^{3}$	2.77×10^{-5} $[2.40\times10^{-5}, 3.05\times10^{-5}]$	$\approx 1.06\times10^{8}$
10^9	$\approx 2.89\times10^{4}$	2.79×10^{-5} $[2.37\times10^{-5}, 3.02\times10^{-5}]$	$\approx 1.06\times10^{9}$
10^{10}	$\approx 2.89\times10^{5}$	2.79×10^{-5} $[2.37\times10^{-5}, 3.06\times10^{-5}]$	$\approx 1.06\times10^{10}$
10^{11}	$\approx 2.89\times10^{6}$	2.75×10^{-5} $[2.38\times10^{-5}, 3.03\times10^{-5}]$	$\approx 1.07\times10^{11}$
10^{12}	$\approx 2.89\times10^{7}$	2.79×10^{-5} $[2.37\times10^{-5}, 3.03\times10^{-5}]$	$\approx 1.07\times10^{12}$
10^{13}	$\approx 2.89\times10^{8}$	2.78×10^{-5} $[2.37\times10^{-5}, 3.09\times10^{-5}]$	$\approx 1.06\times10^{13}$
10^{14}	$\approx 2.89\times10^{9}$	2.72×10^{-5} $[2.37\times10^{-5}, 3.04\times10^{-5}]$	$\approx 1.06\times10^{14}$
10^{15}	$\approx 2.89\times10^{10}$	2.76×10^{-5} $[2.38\times10^{-5}, 3.01\times10^{-5}]$	$\approx 1.07\times10^{15}$

Table 6: Simulator time for $(H, S, A) = (5, 5, 5)$, in seconds. Entries are medians with 95% percentile bootstrap CIs over ten MDP instances; speedup is the median of paired time ratios. Conventional entries beyond the last measured 10^6 budget are marked $\approx$ and extrapolated by a factor of ten per additional power of ten.

Budget	Conventional (s)	Fast (s)	Speedup
10^1	2.25×10^{-3} $[1.81\times10^{-3}, 2.33\times10^{-3}]$	1.70×10^{-4} $[1.64\times10^{-4}, 1.93\times10^{-4}]$	1.14×10^{1} $[1.12\times10^{1}, 1.35\times10^{1}]$
10^2	2.17×10^{-2} $[1.76\times10^{-2}, 2.24\times10^{-2}]$	1.90×10^{-4} $[1.84\times10^{-4}, 2.21\times10^{-4}]$	9.82×10^{1} $[9.50\times10^{1}, 1.17\times10^{2}]$
10^3	2.18×10^{-1} $[1.75\times10^{-1}, 2.25\times10^{-1}]$	1.81×10^{-4} $[1.76\times10^{-4}, 2.16\times10^{-4}]$	1.03×10^{3} $[1.00\times10^{3}, 1.22\times10^{3}]$
10^4	2.20 $[1.77, 2.26]$	1.76×10^{-4} $[1.70\times10^{-4}, 2.05\times10^{-4}]$	1.07×10^{4} $[1.05\times10^{4}, 1.28\times10^{4}]$
10^5	2.19×10^{1} $[1.77\times10^{1}, 2.26\times10^{1}]$	1.73×10^{-4} $[1.68\times10^{-4}, 1.99\times10^{-4}]$	1.07×10^{5} $[1.06\times10^{5}, 1.29\times10^{5}]$
10^6	2.12×10^{2} $[1.79\times10^{2}, 2.22\times10^{2}]$	1.73×10^{-4} $[1.67\times10^{-4}, 1.98\times10^{-4}]$	1.07×10^{6} $[1.06\times10^{6}, 1.24\times10^{6}]$
10^7	$\approx 2.12\times10^{3}$	1.74×10^{-4} $[1.68\times10^{-4}, 1.96\times10^{-4}]$	$\approx 1.08\times10^{7}$
10^8	$\approx 2.12\times10^{4}$	1.76×10^{-4} $[1.69\times10^{-4}, 2.08\times10^{-4}]$	$\approx 1.07\times10^{8}$
10^9	$\approx 2.12\times10^{5}$	1.74×10^{-4} $[1.68\times10^{-4}, 2.08\times10^{-4}]$	$\approx 1.07\times10^{9}$
10^{10}	$\approx 2.12\times10^{6}$	1.73×10^{-4} $[1.67\times10^{-4}, 2.08\times10^{-4}]$	$\approx 1.07\times10^{10}$
10^{11}	$\approx 2.12\times10^{7}$	1.73×10^{-4} $[1.67\times10^{-4}, 2.10\times10^{-4}]$	$\approx 1.07\times10^{11}$
10^{12}	$\approx 2.12\times10^{8}$	1.74×10^{-4} $[1.69\times10^{-4}, 2.10\times10^{-4}]$	$\approx 1.06\times10^{12}$
10^{13}	$\approx 2.12\times10^{9}$	1.72×10^{-4} $[1.66\times10^{-4}, 2.08\times10^{-4}]$	$\approx 1.07\times10^{13}$
10^{14}	$\approx 2.12\times10^{10}$	1.71×10^{-4} $[1.67\times10^{-4}, 2.10\times10^{-4}]$	$\approx 1.07\times10^{14}$
10^{15}	$\approx 2.12\times10^{11}$	1.71×10^{-4} $[1.67\times10^{-4}, 2.10\times10^{-4}]$	$\approx 1.06\times10^{15}$

Table 7: Simulator time for $(H, S, A) = (7, 8, 6)$, in seconds. Entries are medians with 95% percentile bootstrap CIs over ten MDP instances; speedup is the median of paired time ratios. Conventional entries beyond the last measured 10^6 budget are marked $\approx$ and extrapolated by a factor of ten per additional power of ten.

Budget	Conventional (s)	Fast (s)	Speedup
10^1	5.44×10^{-3} $[5.29\times10^{-3}, 5.72\times10^{-3}]$	5.04×10^{-4} $[4.92\times10^{-4}, 5.31\times10^{-4}]$	1.08×10^{1} $[1.05\times10^{1}, 1.12\times10^{1}]$
10^2	5.10×10^{-2} $[5.01\times10^{-2}, 5.34\times10^{-2}]$	6.40×10^{-4} $[6.19\times10^{-4}, 6.70\times10^{-4}]$	7.94×10^{1} $[7.77\times10^{1}, 8.49\times10^{1}]$
10^3	5.16×10^{-1} $[5.10\times10^{-1}, 5.35\times10^{-1}]$	6.10×10^{-4} $[5.93\times10^{-4}, 6.27\times10^{-4}]$	8.64×10^{2} $[8.33\times10^{2}, 8.87\times10^{2}]$
10^4	5.28 $[5.26, 5.51]$	5.66×10^{-4} $[5.47\times10^{-4}, 5.88\times10^{-4}]$	9.44×10^{3} $[9.17\times10^{3}, 9.91\times10^{3}]$
10^5	5.27×10^{1} $[5.18\times10^{1}, 5.48\times10^{1}]$	5.50×10^{-4} $[5.37\times10^{-4}, 5.67\times10^{-4}]$	9.61×10^{4} $[9.41\times10^{4}, 1.00\times10^{5}]$
10^6	5.27×10^{2} $[5.17\times10^{2}, 5.51\times10^{2}]$	5.42×10^{-4} $[5.33\times10^{-4}, 5.73\times10^{-4}]$	9.83×10^{5} $[9.38\times10^{5}, 9.94\times10^{5}]$
10^7	$\approx 5.27\times10^{3}$	5.53×10^{-4} $[5.36\times10^{-4}, 5.70\times10^{-4}]$	$\approx 9.58\times10^{6}$
10^8	$\approx 5.27\times10^{4}$	5.48×10^{-4} $[5.26\times10^{-4}, 5.71\times10^{-4}]$	$\approx 9.73\times10^{7}$
10^9	$\approx 5.27\times10^{5}$	5.49×10^{-4} $[5.33\times10^{-4}, 5.64\times10^{-4}]$	$\approx 9.68\times10^{8}$
10^{10}	$\approx 5.27\times10^{6}$	5.48×10^{-4} $[5.36\times10^{-4}, 5.62\times10^{-4}]$	$\approx 9.68\times10^{9}$
10^{11}	$\approx 5.27\times10^{7}$	5.48×10^{-4} $[5.29\times10^{-4}, 5.62\times10^{-4}]$	$\approx 9.74\times10^{10}$
10^{12}	$\approx 5.27\times10^{8}$	5.53×10^{-4} $[5.34\times10^{-4}, 5.70\times10^{-4}]$	$\approx 9.76\times10^{11}$
10^{13}	$\approx 5.27\times10^{9}$	5.46×10^{-4} $[5.39\times10^{-4}, 5.62\times10^{-4}]$	$\approx 9.60\times10^{12}$
10^{14}	$\approx 5.27\times10^{10}$	5.50×10^{-4} $[5.37\times10^{-4}, 5.63\times10^{-4}]$	$\approx 9.49\times10^{13}$
10^{15}	$\approx 5.27\times10^{11}$	5.43×10^{-4} $[5.29\times10^{-4}, 5.64\times10^{-4}]$	$\approx 9.78\times10^{14}$

Table 8: Simulator time for $(H, S, A) = (10, 15, 10)$, in seconds. Entries are medians with 95% percentile bootstrap CIs over ten MDP instances; speedup is the median of paired time ratios. Conventional entries beyond the last measured 10^6 budget are marked $\approx$ and extrapolated by a factor of ten per additional power of ten.

Budget	Conventional (s)	Fast (s)	Speedup
10^1	2.69×10^{-2} $[2.26\times10^{-2}, 2.76\times10^{-2}]$	3.17×10^{-3} $[2.62\times10^{-3}, 3.29\times10^{-3}]$	8.53 $[8.36, 8.58]$
10^2	2.57×10^{-1} $[2.21\times10^{-1}, 2.67\times10^{-1}]$	4.58×10^{-3} $[3.74\times10^{-3}, 4.75\times10^{-3}]$	5.70×10^{1} $[5.63\times10^{1}, 5.85\times10^{1}]$
10^3	2.57 $[2.27, 2.72]$	5.19×10^{-3} $[4.23\times10^{-3}, 5.35\times10^{-3}]$	5.13×10^{2} $[5.02\times10^{2}, 5.25\times10^{2}]$
10^4	2.57×10^{1} $[2.19\times10^{1}, 2.71\times10^{1}]$	4.26×10^{-3} $[3.47\times10^{-3}, 4.40\times10^{-3}]$	6.27×10^{3} $[6.13\times10^{3}, 6.30\times10^{3}]$
10^5	2.64×10^{2} $[2.23\times10^{2}, 2.71\times10^{2}]$	3.85×10^{-3} $[3.17\times10^{-3}, 4.00\times10^{-3}]$	6.90×10^{4} $[6.77\times10^{4}, 7.01\times10^{4}]$
10^6	2.61×10^{3} $[2.16\times10^{3}, 2.69\times10^{3}]$	3.71×10^{-3} $[3.03\times10^{-3}, 3.86\times10^{-3}]$	6.98×10^{5} $[6.94\times10^{5}, 7.13\times10^{5}]$
10^7	$\approx 2.61\times10^{4}$	3.68×10^{-3} $[3.02\times10^{-3}, 3.83\times10^{-3}]$	$\approx 7.04\times10^{6}$
10^8	$\approx 2.61\times10^{5}$	3.70×10^{-3} $[3.00\times10^{-3}, 3.83\times10^{-3}]$	$\approx 7.03\times10^{7}$
10^9	$\approx 2.61\times10^{6}$	3.69×10^{-3} $[3.00\times10^{-3}, 3.82\times10^{-3}]$	$\approx 7.07\times10^{8}$
10^{10}	$\approx 2.61\times10^{7}$	3.68×10^{-3} $[3.05\times10^{-3}, 3.81\times10^{-3}]$	$\approx 7.05\times10^{9}$
10^{11}	$\approx 2.61\times10^{8}$	3.68×10^{-3} $[3.03\times10^{-3}, 3.83\times10^{-3}]$	$\approx 7.05\times10^{10}$
10^{12}	$\approx 2.61\times10^{9}$	3.68×10^{-3} $[3.01\times10^{-3}, 3.82\times10^{-3}]$	$\approx 7.06\times10^{11}$
10^{13}	$\approx 2.61\times10^{10}$	3.65×10^{-3} $[3.00\times10^{-3}, 3.82\times10^{-3}]$	$\approx 7.10\times10^{12}$
10^{14}	$\approx 2.61\times10^{11}$	3.65×10^{-3} $[3.03\times10^{-3}, 3.80\times10^{-3}]$	$\approx 7.10\times10^{13}$
10^{15}	$\approx 2.61\times10^{12}$	3.65×10^{-3} $[3.02\times10^{-3}, 3.82\times10^{-3}]$	$\approx 7.13\times10^{14}$

B.2 Offline Vanilla Pessimistic Value Iteration

We use a uniform behavior policy over actions to generate an offline dataset. Under conventional simulation, the offline dataset is generated trajectory by trajectory. In contrast, the direct batch simulation mode of our framework samples the aggregate Markov flow for the full batch of K trajectories and directly recovers the visitation and transition counts required by VPVI. The same VPVI update is then applied to the resulting empirical statistics. The pessimism multiplier c and the log-confidence factor ι in Algorithm 3 are both set to one.

Learning performance is evaluated by the policy gap in the true MDP, averaged over the uniform initial-state distribution:

$$G_\pi(K) = \mathbb{E}_{s\sim\nu}\big[V_1^\star(s) - V_1^{\widehat{\pi}}(s)\big].$$

The policy-gap results are reported in Tables 9–12. Across the reported budgets, the conventional and fast simulators produce nearly identical policy gaps at the same K: for most settings, the confidence intervals for their differences either contain zero or collapse to zero (31 out of 32 settings). This behavior is consistent with the distributional equivalence established in Corollary 3.2. Indeed, since VPVI depends on the offline dataset only through the visitation and transition statistics contained in the Markov flow, equality in distribution of the generated flows induces equality in distribution of the resulting learned policies and hence of their policy gaps.

The policy gap, however, can become exactly zero at relatively modest budgets, particularly for smaller MDPs, once the learned policy is already optimal. Beyond this point, it no longer distinguishes between empirical models of different accuracies. We therefore additionally examine the visitation-weighted transition error

$$E_P(K) = \frac{\sum_{h,s,a} C_h(s,a)\left\|\widehat{P}_h(\cdot \mid s,a) - P_h(\cdot \mid s,a)\right\|_1}{\sum_{h,s,a} C_h(s,a)}.$$

This quantity measures the discrepancy between the empirical transition model passed to VPVI and the true transition model, weighted by how frequently each row is represented in the sampled dataset. It therefore provides a more fine-grained diagnostic of the sampled empirical model, complementing the policy gap, which measures the quality of the final learned policy.

The transition-error results are reported in Tables 13–16. The conventional and fast simulators exhibit very similar weighted transition errors across budgets, with almost all confidence intervals for their differences containing zero. Since $E_P(K)$ is itself a function of the sampled visitation and transition counts, this agreement is also consistent with the distributional equivalence in Corollary 3.2.

The timing results in Tables 17–20 show different computational scaling. Conventional simulation generates the K trajectories individually, and its runtime grows approximately linearly with K. In contrast, our fast method uses direct batch simulation to generate the aggregate Markov flow, eliminating the simulator-side dependence on K. Beyond the small-budget regime, its runtime remains approximately flat over moderate-to-large budgets, consistent with Proposition 4.1. Moreover, the learner time remains below 0.002 seconds in all tested configurations and is substantially smaller than the conventional simulation time at large scales and budgets. This shows that simulation is the dominant computational bottleneck in this regime, making the reduction in simulator-side complexity practically significant.

Table 9: Expected policy gap for $(H, S, A) = (2, 3, 3)$. Entries report the mean and 95% percentile bootstrap confidence interval over ten independent simulation seeds on a fixed MDP. The difference is defined as fast minus conventional.

Budget	Conventional	Fast	Difference
10^1	9.20×10^{-1} $[9.20\times10^{-1}, 9.20\times10^{-1}]$	9.20×10^{-1} $[9.20\times10^{-1}, 9.20\times10^{-1}]$	0 [0, 0]
10^2	2.89×10^{-2} $[9.48\times10^{-3}, 5.29\times10^{-2}]$	6.39×10^{-2} $[4.98\times10^{-2}, 7.89\times10^{-2}]$	3.50×10^{-2} $[6.42\times10^{-3}, 6.15\times10^{-2}]$
10^3	0 [0, 0]	0 [0, 0]	0 [0, 0]
10^4	0 [0, 0]	0 [0, 0]	0 [0, 0]
10^5	0 [0, 0]	0 [0, 0]	0 [0, 0]
10^6	0 [0, 0]	0 [0, 0]	0 [0, 0]
10^7	0 [0, 0]	0 [0, 0]	0 [0, 0]
10^8	0 [0, 0]	0 [0, 0]	0 [0, 0]

Table 10: Expected policy gap for $(H, S, A) = (5, 5, 5)$. Entries report the mean and 95% percentile bootstrap confidence interval over ten independent simulation seeds on a fixed MDP. The difference is defined as fast minus conventional.

Budget	Conventional	Fast	Difference
10^1	$1.79\,[1.79, 1.79]$	$1.79\,[1.79, 1.79]$	$0\,[0, 0]$
10^2	$1.79\,[1.79, 1.79]$	$1.79\,[1.79, 1.79]$	$0\,[0, 0]$
10^3	$1.21\times10^{-1}\,[9.11\times10^{-2}, 1.49\times10^{-1}]$	$9.35\times10^{-2}\,[5.11\times10^{-2}, 1.39\times10^{-1}]$	$-2.72\times10^{-2}\,[-7.89\times10^{-2}, 2.71\times10^{-2}]$
10^4	$4.14\times10^{-4}\,[1.04\times10^{-4}, 7.25\times10^{-4}]$	$3.11\times10^{-4}\,[0, 6.21\times10^{-4}]$	$-1.04\times10^{-4}\,[-5.18\times10^{-4}, 3.11\times10^{-4}]$
10^5	$0\,[0, 0]$	$0\,[0, 0]$	$0\,[0, 0]$
10^6	$0\,[0, 0]$	$0\,[0, 0]$	$0\,[0, 0]$
10^7	$0\,[0, 0]$	$0\,[0, 0]$	$0\,[0, 0]$
10^8	$0\,[0, 0]$	$0\,[0, 0]$	$0\,[0, 0]$

Table 11: Expected policy gap for $(H, S, A) = (7, 8, 6)$. Entries report the mean and 95% percentile bootstrap confidence interval over ten independent simulation seeds on a fixed MDP. The difference is defined as fast minus conventional.

Budget	Conventional	Fast	Difference
10^1	$2.60\,[2.60, 2.60]$	$2.60\,[2.60, 2.60]$	$0\,[0, 0]$
10^2	$2.60\,[2.60, 2.60]$	$2.60\,[2.60, 2.60]$	$0\,[0, 0]$
10^3	$2.60\,[2.60, 2.60]$	$2.60\,[2.60, 2.60]$	$0\,[0, 0]$
10^4	$8.57\times10^{-3}\,[5.52\times10^{-3}, 1.18\times10^{-2}]$	$8.36\times10^{-3}\,[5.73\times10^{-3}, 1.11\times10^{-2}]$	$-2.10\times10^{-4}\,[-4.35\times10^{-3}, 3.82\times10^{-3}]$
10^5	$0\,[0, 0]$	$0\,[0, 0]$	$0\,[0, 0]$
10^6	$0\,[0, 0]$	$0\,[0, 0]$	$0\,[0, 0]$
10^7	$0\,[0, 0]$	$0\,[0, 0]$	$0\,[0, 0]$
10^8	$0\,[0, 0]$	$0\,[0, 0]$	$0\,[0, 0]$

Table 12: Expected policy gap for $(H, S, A) = (10, 15, 10)$. Entries report the mean and 95% percentile bootstrap confidence interval over ten independent simulation seeds on a fixed MDP. The difference is defined as fast minus conventional.

Budget	Conventional	Fast	Difference
10^1	$4.06\,[4.06, 4.06]$	$4.06\,[4.06, 4.06]$	$0\,[0, 0]$
10^2	$4.06\,[4.06, 4.06]$	$4.06\,[4.06, 4.06]$	$0\,[0, 0]$
10^3	$4.06\,[4.06, 4.06]$	$4.06\,[4.06, 4.06]$	$0\,[0, 0]$
10^4	$4.06\,[4.06, 4.06]$	$4.06\,[4.06, 4.06]$	$0\,[0, 0]$
10^5	$4.01\times10^{-3}\,[2.93\times10^{-3}, 5.13\times10^{-3}]$	$4.25\times10^{-3}\,[3.26\times10^{-3}, 5.27\times10^{-3}]$	$2.32\times10^{-4}\,[-1.19\times10^{-3}, 1.69\times10^{-3}]$
10^6	$1.62\times10^{-4}\,[7.37\times10^{-5}, 2.54\times10^{-4}]$	$6.09\times10^{-5}\,[2.39\times10^{-5}, 1.17\times10^{-4}]$	$-1.01\times10^{-4}\,[-2.06\times10^{-4}, 1.16\times10^{-6}]$
10^7	$6.82\times10^{-6}\,[0, 1.70\times10^{-5}]$	$3.41\times10^{-6}\,[0, 1.02\times10^{-5}]$	$-3.41\times10^{-6}\,[-1.36\times10^{-5}, 6.82\times10^{-6}]$
10^8	$0\,[0, 0]$	$0\,[0, 0]$	$0\,[0, 0]$

Table 13: Visit-weighted transition L_1 error for $(H, S, A) = (2, 3, 3)$. Entries report the mean and 95% percentile bootstrap confidence interval over ten independent simulation seeds on a fixed MDP. The error difference is defined as fast minus conventional.

Budget	Conventional	Fast	Difference
10^1	$7.64\times10^{-1}\,[6.87\times10^{-1}, 8.42\times10^{-1}]$	$8.20\times10^{-1}\,[7.24\times10^{-1}, 9.13\times10^{-1}]$	$5.62\times10^{-2}\,[-6.30\times10^{-2}, 1.77\times10^{-1}]$
10^2	$2.64\times10^{-1}\,[2.40\times10^{-1}, 2.90\times10^{-1}]$	$2.91\times10^{-1}\,[2.68\times10^{-1}, 3.12\times10^{-1}]$	$2.65\times10^{-2}\,[-7.67\times10^{-3}, 5.93\times10^{-2}]$
10^3	$8.54\times10^{-2}\,[7.81\times10^{-2}, 9.21\times10^{-2}]$	$9.52\times10^{-2}\,[9.01\times10^{-2}, 1.00\times10^{-1}]$	$9.84\times10^{-3}\,[1.32\times10^{-3}, 1.85\times10^{-2}]$
10^4	$3.00\times10^{-2}\,[2.84\times10^{-2}, 3.15\times10^{-2}]$	$2.94\times10^{-2}\,[2.73\times10^{-2}, 3.15\times10^{-2}]$	$-5.82\times10^{-4}\,[-3.22\times10^{-3}, 2.06\times10^{-3}]$
10^5	$9.12\times10^{-3}\,[8.31\times10^{-3}, 9.96\times10^{-3}]$	$8.42\times10^{-3}\,[7.79\times10^{-3}, 9.10\times10^{-3}]$	$-6.99\times10^{-4}\,[-1.73\times10^{-3}, 3.56\times10^{-4}]$
10^6	$2.76\times10^{-3}\,[2.57\times10^{-3}, 2.96\times10^{-3}]$	$2.56\times10^{-3}\,[2.31\times10^{-3}, 2.84\times10^{-3}]$	$-2.07\times10^{-4}\,[-5.31\times10^{-4}, 1.33\times10^{-4}]$
10^7	$9.45\times10^{-4}\,[8.39\times10^{-4}, 1.06\times10^{-3}]$	$8.18\times10^{-4}\,[7.41\times10^{-4}, 9.05\times10^{-4}]$	$-1.27\times10^{-4}\,[-2.60\times10^{-4}, 1.08\times10^{-5}]$
10^8	$2.88\times10^{-4}\,[2.68\times10^{-4}, 3.10\times10^{-4}]$	$2.59\times10^{-4}\,[2.35\times10^{-4}, 2.88\times10^{-4}]$	$-2.87\times10^{-5}\,[-6.08\times10^{-5}, 6.21\times10^{-6}]$

Table 14: Visit-weighted transition L_1 error for $(H, S, A) = (5, 5, 5)$. Entries report the mean and 95% percentile bootstrap confidence interval over ten independent simulation seeds on a fixed MDP. The error difference is defined as fast minus conventional.

Budget	Conventional	Fast	Difference
10^1	$1.23\,[1.19, 1.27]$	$1.24\,[1.21, 1.26]$	$3.90\times10^{-3}\,[-4.22\times10^{-2}, 5.25\times10^{-2}]$
10^2	$6.72\times10^{-1}\,[6.65\times10^{-1}, 6.78\times10^{-1}]$	$6.90\times10^{-1}\,[6.76\times10^{-1}, 7.05\times10^{-1}]$	$1.85\times10^{-2}\,[2.22\times10^{-3}, 3.42\times10^{-2}]$
10^3	$2.15\times10^{-1}\,[2.10\times10^{-1}, 2.20\times10^{-1}]$	$2.20\times10^{-1}\,[2.17\times10^{-1}, 2.24\times10^{-1}]$	$4.95\times10^{-3}\,[-1.14\times10^{-3}, 1.09\times10^{-2}]$
10^4	$7.10\times10^{-2}\,[6.96\times10^{-2}, 7.25\times10^{-2}]$	$6.92\times10^{-2}\,[6.73\times10^{-2}, 7.11\times10^{-2}]$	$-1.89\times10^{-3}\,[-4.21\times10^{-3}, 5.84\times10^{-4}]$
10^5	$2.18\times10^{-2}\,[2.13\times10^{-2}, 2.25\times10^{-2}]$	$2.20\times10^{-2}\,[2.15\times10^{-2}, 2.25\times10^{-2}]$	$1.42\times10^{-4}\,[-6.62\times10^{-4}, 9.08\times10^{-4}]$
10^6	$6.81\times10^{-3}\,[6.63\times10^{-3}, 6.99\times10^{-3}]$	$6.75\times10^{-3}\,[6.61\times10^{-3}, 6.91\times10^{-3}]$	$-5.27\times10^{-5}\,[-2.96\times10^{-4}, 1.80\times10^{-4}]$
10^7	$2.17\times10^{-3}\,[2.09\times10^{-3}, 2.24\times10^{-3}]$	$2.18\times10^{-3}\,[2.15\times10^{-3}, 2.21\times10^{-3}]$	$1.22\times10^{-5}\,[-6.61\times10^{-5}, 9.26\times10^{-5}]$
10^8	$6.90\times10^{-4}\,[6.80\times10^{-4}, 7.00\times10^{-4}]$	$6.84\times10^{-4}\,[6.70\times10^{-4}, 6.98\times10^{-4}]$	$-6.11\times10^{-6}\,[-2.34\times10^{-5}, 1.17\times10^{-5}]$

Table 15: Visit-weighted transition L_1 error for $(H, S, A) = (7, 8, 6)$. Entries report the mean and 95% percentile bootstrap confidence interval over ten independent simulation seeds on a fixed MDP. The error difference is defined as fast minus conventional.

Budget	Conventional	Fast	Difference
10^1	$1.51\,[1.48, 1.54]$	$1.50\,[1.48, 1.52]$	$-1.14\times10^{-2}\,[-5.21\times10^{-2}, 2.70\times10^{-2}]$
10^2	$1.10\,[1.08, 1.11]$	$1.11\,[1.10, 1.12]$	$9.43\times10^{-3}\,[-9.93\times10^{-3}, 2.79\times10^{-2}]$
10^3	$4.03\times10^{-1}\,[3.98\times10^{-1}, 4.09\times10^{-1}]$	$4.06\times10^{-1}\,[4.02\times10^{-1}, 4.11\times10^{-1}]$	$2.97\times10^{-3}\,[-4.40\times10^{-3}, 1.02\times10^{-2}]$
10^4	$1.28\times10^{-1}\,[1.27\times10^{-1}, 1.29\times10^{-1}]$	$1.29\times10^{-1}\,[1.28\times10^{-1}, 1.30\times10^{-1}]$	$3.99\times10^{-4}\,[-8.50\times10^{-4}, 1.58\times10^{-3}]$
10^5	$4.12\times10^{-2}\,[4.07\times10^{-2}, 4.17\times10^{-2}]$	$4.04\times10^{-2}\,[3.98\times10^{-2}, 4.09\times10^{-2}]$	$-7.26\times10^{-4}\,[-1.54\times10^{-3}, -1.61\times10^{-5}]$
10^6	$1.29\times10^{-2}\,[1.28\times10^{-2}, 1.30\times10^{-2}]$	$1.28\times10^{-2}\,[1.27\times10^{-2}, 1.30\times10^{-2}]$	$-2.16\times10^{-5}\,[-2.17\times10^{-4}, 1.64\times10^{-4}]$
10^7	$4.03\times10^{-3}\,[3.99\times10^{-3}, 4.07\times10^{-3}]$	$4.06\times10^{-3}\,[4.02\times10^{-3}, 4.09\times10^{-3}]$	$3.19\times10^{-5}\,[-1.88\times10^{-5}, 8.16\times10^{-5}]$
10^8	$1.28\times10^{-3}\,[1.27\times10^{-3}, 1.29\times10^{-3}]$	$1.28\times10^{-3}\,[1.27\times10^{-3}, 1.29\times10^{-3}]$	$-1.54\times10^{-7}\,[-1.48\times10^{-5}, 1.40\times10^{-5}]$

Table 16: Visit-weighted transition L_1 error for $(H, S, A) = (10, 15, 10)$. Entries report the mean and 95% percentile bootstrap confidence interval over ten independent simulation seeds on a fixed MDP. The error difference is defined as fast minus conventional.

Budget	Conventional	Fast	Difference
10^1	$1.74\,[1.74, 1.75]$	$1.73\,[1.72, 1.74]$	$-9.56\times10^{-3}\,[-2.11\times10^{-2}, 1.65\times10^{-3}]$
10^2	$1.62\,[1.62, 1.62]$	$1.62\,[1.61, 1.62]$	$-4.61\times10^{-3}\,[-9.54\times10^{-3}, 1.63\times10^{-4}]$
10^3	$9.70\times10^{-1}\,[9.67\times10^{-1}, 9.73\times10^{-1}]$	$9.72\times10^{-1}\,[9.70\times10^{-1}, 9.75\times10^{-1}]$	$2.89\times10^{-3}\,[-1.85\times10^{-3}, 6.95\times10^{-3}]$
10^4	$3.20\times10^{-1}\,[3.19\times10^{-1}, 3.21\times10^{-1}]$	$3.21\times10^{-1}\,[3.20\times10^{-1}, 3.22\times10^{-1}]$	$7.83\times10^{-4}\,[-4.90\times10^{-4}, 2.21\times10^{-3}]$
10^5	$1.02\times10^{-1}\,[1.01\times10^{-1}, 1.02\times10^{-1}]$	$1.02\times10^{-1}\,[1.01\times10^{-1}, 1.02\times10^{-1}]$	$1.04\times10^{-4}\,[-4.42\times10^{-4}, 6.80\times10^{-4}]$
10^6	$3.23\times10^{-2}\,[3.22\times10^{-2}, 3.24\times10^{-2}]$	$3.21\times10^{-2}\,[3.20\times10^{-2}, 3.22\times10^{-2}]$	$-1.44\times10^{-4}\,[-2.89\times10^{-4}, 2.50\times10^{-6}]$
10^7	$1.02\times10^{-2}\,[1.01\times10^{-2}, 1.02\times10^{-2}]$	$1.01\times10^{-2}\,[1.01\times10^{-2}, 1.02\times10^{-2}]$	$-1.36\times10^{-5}\,[-6.97\times10^{-5}, 4.30\times10^{-5}]$
10^8	$3.22\times10^{-3}\,[3.20\times10^{-3}, 3.23\times10^{-3}]$	$3.21\times10^{-3}\,[3.20\times10^{-3}, 3.22\times10^{-3}]$	$-5.47\times10^{-6}\,[-2.36\times10^{-5}, 1.22\times10^{-5}]$

Table 17: Simulator time for $(H, S, A) = (2, 3, 3)$, in seconds. Entries are medians with 95% percentile bootstrap CIs over ten MDP instances; speedup is the median of paired time ratios. Conventional entries beyond the last measured 10^8 budget are marked $\approx$ and extrapolated by a factor of ten per additional power of ten.

Budget	Conventional (s)	Fast (s)	Speedup
10^1	$1.05\times10^{-4}\,[9.41\times10^{-5}, 1.07\times10^{-4}]$	$4.71\times10^{-5}\,[4.15\times10^{-5}, 4.88\times10^{-5}]$	$2.21\,[2.15, 2.25]$
10^2	$9.80\times10^{-4}\,[8.74\times10^{-4}, 1.00\times10^{-3}]$	$5.95\times10^{-5}\,[5.13\times10^{-5}, 5.98\times10^{-5}]$	$1.68\times10^{1}\,[1.64\times10^{1}, 1.70\times10^{1}]$
10^3	$9.93\times10^{-3}\,[8.72\times10^{-3}, 9.98\times10^{-3}]$	$6.11\times10^{-5}\,[5.33\times10^{-5}, 6.20\times10^{-5}]$	$1.63\times10^{2}\,[1.59\times10^{2}, 1.64\times10^{2}]$
10^4	$1.00\times10^{-1}\,[8.77\times10^{-2}, 1.01\times10^{-1}]$	$5.95\times10^{-5}\,[5.17\times10^{-5}, 6.06\times10^{-5}]$	$1.67\times10^{3}\,[1.64\times10^{3}, 1.71\times10^{3}]$
10^5	$9.74\times10^{-1}\,[8.81\times10^{-1}, 9.97\times10^{-1}]$	$5.88\times10^{-5}\,[5.08\times10^{-5}, 5.95\times10^{-5}]$	$1.67\times10^{4}\,[1.64\times10^{4}, 1.74\times10^{4}]$
10^6	$9.82\,[8.92, 1.01\times10^{1}]$	$5.89\times10^{-5}\,[5.08\times10^{-5}, 5.94\times10^{-5}]$	$1.70\times10^{5}\,[1.68\times10^{5}, 1.74\times10^{5}]$
10^7	$9.75\times10^{1}\,[8.76\times10^{1}, 9.91\times10^{1}]$	$5.83\times10^{-5}\,[5.09\times10^{-5}, 5.90\times10^{-5}]$	$1.67\times10^{6}\,[1.66\times10^{6}, 1.74\times10^{6}]$
10^8	$9.82\times10^{2}\,[8.81\times10^{2}, 1.00\times10^{3}]$	$5.79\times10^{-5}\,[5.02\times10^{-5}, 5.89\times10^{-5}]$	$1.71\times10^{7}\,[1.69\times10^{7}, 1.75\times10^{7}]$
10^9	$\approx 9.82\times10^{3}$	$5.79\times10^{-5}\,[5.01\times10^{-5}, 5.92\times10^{-5}]$	$\approx 1.71\times10^{8}$
10^{10}	$\approx 9.82\times10^{4}$	$5.85\times10^{-5}\,[5.08\times10^{-5}, 5.98\times10^{-5}]$	$\approx 1.68\times10^{9}$
10^{11}	$\approx 9.82\times10^{5}$	$5.80\times10^{-5}\,[5.03\times10^{-5}, 5.94\times10^{-5}]$	$\approx 1.70\times10^{10}$
10^{12}	$\approx 9.82\times10^{6}$	$5.77\times10^{-5}\,[5.03\times10^{-5}, 5.94\times10^{-5}]$	$\approx 1.72\times10^{11}$
10^{13}	$\approx 9.82\times10^{7}$	$5.83\times10^{-5}\,[5.03\times10^{-5}, 5.91\times10^{-5}]$	$\approx 1.71\times10^{12}$
10^{14}	$\approx 9.82\times10^{8}$	$5.81\times10^{-5}\,[5.04\times10^{-5}, 5.89\times10^{-5}]$	$\approx 1.72\times10^{13}$
10^{15}	$\approx 9.82\times10^{9}$	$5.79\times10^{-5}\,[5.03\times10^{-5}, 5.93\times10^{-5}]$	$\approx 1.71\times10^{14}$

Table 18: Simulator time for $(H, S, A) = (5, 5, 5)$, in seconds. Entries are medians with 95% percentile bootstrap CIs over ten MDP instances; speedup is the median of paired time ratios. Conventional entries beyond the last measured 10^8 budget are marked $\approx$ and extrapolated by a factor of ten per additional power of ten.

Budget	Conventional (s)	Fast (s)	Speedup
10^1	1.94×10^{-4} $[1.92\times10^{-4}, 2.09\times10^{-4}]$	1.29×10^{-4} $[1.20\times10^{-4}, 1.45\times10^{-4}]$	1.49 $[1.42, 1.63]$
10^2	1.88×10^{-3} $[1.86\times10^{-3}, 2.05\times10^{-3}]$	2.70×10^{-4} $[2.62\times10^{-4}, 3.06\times10^{-4}]$	6.81 $[6.62, 7.12]$
10^3	1.86×10^{-2} $[1.84\times10^{-2}, 2.03\times10^{-2}]$	3.00×10^{-4} $[2.83\times10^{-4}, 3.31\times10^{-4}]$	6.18×10^{1} $[6.05\times10^{1}, 6.51\times10^{1}]$
10^4	1.87×10^{-1} $[1.83\times10^{-1}, 2.04\times10^{-1}]$	3.09×10^{-4} $[2.89\times10^{-4}, 3.41\times10^{-4}]$	6.23×10^{2} $[5.95\times10^{2}, 6.32\times10^{2}]$
10^5	1.89 $[1.85, 2.01]$	2.90×10^{-4} $[2.78\times10^{-4}, 3.34\times10^{-4}]$	6.43×10^{3} $[6.04\times10^{3}, 6.79\times10^{3}]$
10^6	1.91×10^{1} $[1.86\times10^{1}, 2.06\times10^{1}]$	2.87×10^{-4} $[2.71\times10^{-4}, 3.27\times10^{-4}]$	6.60×10^{4} $[6.25\times10^{4}, 6.79\times10^{4}]$
10^7	1.86×10^{2} $[1.82\times10^{2}, 2.03\times10^{2}]$	2.83×10^{-4} $[2.73\times10^{-4}, 3.21\times10^{-4}]$	6.49×10^{5} $[6.18\times10^{5}, 6.75\times10^{5}]$
10^8	1.92×10^{3} $[1.90\times10^{3}, 2.06\times10^{3}]$	2.81×10^{-4} $[2.72\times10^{-4}, 3.23\times10^{-4}]$	6.80×10^{6} $[6.27\times10^{6}, 7.04\times10^{6}]$
10^9	$\approx 1.92\times10^{4}$	2.79×10^{-4} $[2.70\times10^{-4}, 3.20\times10^{-4}]$	$\approx 6.80\times10^{7}$
10^{10}	$\approx 1.92\times10^{5}$	2.83×10^{-4} $[2.73\times10^{-4}, 3.14\times10^{-4}]$	$\approx 6.77\times10^{8}$
10^{11}	$\approx 1.92\times10^{6}$	2.82×10^{-4} $[2.72\times10^{-4}, 3.22\times10^{-4}]$	$\approx 6.74\times10^{9}$
10^{12}	$\approx 1.92\times10^{7}$	2.82×10^{-4} $[2.74\times10^{-4}, 3.25\times10^{-4}]$	$\approx 6.73\times10^{10}$
10^{13}	$\approx 1.92\times10^{8}$	2.82×10^{-4} $[2.75\times10^{-4}, 3.26\times10^{-4}]$	$\approx 6.74\times10^{11}$
10^{14}	$\approx 1.92\times10^{9}$	2.83×10^{-4} $[2.76\times10^{-4}, 3.26\times10^{-4}]$	$\approx 6.78\times10^{12}$
10^{15}	$\approx 1.92\times10^{10}$	2.97×10^{-4} $[2.75\times10^{-4}, 3.26\times10^{-4}]$	$\approx 6.50\times10^{13}$

Table 19: Simulator time for $(H, S, A) = (7, 8, 6)$, in seconds. Entries are medians with 95% percentile bootstrap CIs over ten MDP instances; speedup is the median of paired time ratios. Conventional entries beyond the last measured 10^8 budget are marked $\approx$ and extrapolated by a factor of ten per additional power of ten.

Budget	Conventional (s)	Fast (s)	Speedup
10^1	3.07×10^{-4} $[2.77\times10^{-4}, 3.08\times10^{-4}]$	2.72×10^{-4} $[2.54\times10^{-4}, 2.82\times10^{-4}]$	1.10 $[1.07, 1.13]$
10^2	3.01×10^{-3} $[2.70\times10^{-3}, 3.03\times10^{-3}]$	7.98×10^{-4} $[7.44\times10^{-4}, 8.06\times10^{-4}]$	3.73 $[3.66, 3.78]$
10^3	3.01×10^{-2} $[2.69\times10^{-2}, 3.02\times10^{-2}]$	1.04×10^{-3} $[9.73\times10^{-4}, 1.05\times10^{-3}]$	2.87×10^{1} $[2.77\times10^{1}, 2.90\times10^{1}]$
10^4	3.00×10^{-1} $[2.67\times10^{-1}, 3.01\times10^{-1}]$	1.17×10^{-3} $[1.10\times10^{-3}, 1.19\times10^{-3}]$	2.52×10^{2} $[2.43\times10^{2}, 2.55\times10^{2}]$
10^5	3.05 $[2.68, 3.06]$	1.07×10^{-3} $[1.01\times10^{-3}, 1.08\times10^{-3}]$	2.79×10^{3} $[2.66\times10^{3}, 2.85\times10^{3}]$
10^6	3.05×10^{1} $[2.72\times10^{1}, 3.06\times10^{1}]$	1.01×10^{-3} $[9.46\times10^{-4}, 1.01\times10^{-3}]$	3.01×10^{4} $[2.94\times10^{4}, 3.03\times10^{4}]$
10^7	3.04×10^{2} $[2.64\times10^{2}, 3.06\times10^{2}]$	9.94×10^{-4} $[9.33\times10^{-4}, 9.99\times10^{-4}]$	3.02×10^{5} $[2.83\times10^{5}, 3.07\times10^{5}]$
10^8	2.98×10^{3} $[2.64\times10^{3}, 3.01\times10^{3}]$	9.91×10^{-4} $[9.26\times10^{-4}, 9.96\times10^{-4}]$	2.99×10^{6} $[2.92\times10^{6}, 3.03\times10^{6}]$
10^9	$\approx 2.98\times10^{4}$	9.89×10^{-4} $[9.22\times10^{-4}, 9.97\times10^{-4}]$	$\approx 2.99\times10^{7}$
10^{10}	$\approx 2.98\times10^{5}$	9.92×10^{-4} $[9.23\times10^{-4}, 9.99\times10^{-4}]$	$\approx 2.98\times10^{8}$
10^{11}	$\approx 2.98\times10^{6}$	9.94×10^{-4} $[9.26\times10^{-4}, 1.00\times10^{-3}]$	$\approx 2.98\times10^{9}$
10^{12}	$\approx 2.98\times10^{7}$	9.93×10^{-4} $[9.26\times10^{-4}, 9.99\times10^{-4}]$	$\approx 2.98\times10^{10}$
10^{13}	$\approx 2.98\times10^{8}$	9.94×10^{-4} $[9.31\times10^{-4}, 1.00\times10^{-3}]$	$\approx 2.98\times10^{11}$
10^{14}	$\approx 2.98\times10^{9}$	9.93×10^{-4} $[9.44\times10^{-4}, 1.00\times10^{-3}]$	$\approx 2.97\times10^{12}$
10^{15}	$\approx 2.98\times10^{10}$	9.92×10^{-4} $[9.34\times10^{-4}, 1.00\times10^{-3}]$	$\approx 2.98\times10^{13}$

Table 20: Simulator time for $(H, S, A) = (10, 15, 10)$, in seconds. Entries are medians with 95% percentile bootstrap CIs over ten MDP instances; speedup is the median of paired time ratios. Conventional entries beyond the last measured 10^8 budget are marked $\approx$ and extrapolated by a factor of ten per additional power of ten.

Budget	Conventional (s)	Fast (s)	Speedup
10^1	3.43×10^{-4} $[3.36\times10^{-4}, 4.17\times10^{-4}]$	5.90×10^{-4} $[5.80\times10^{-4}, 6.22\times10^{-4}]$	5.98×10^{-1} $[5.72\times10^{-1}, 6.62\times10^{-1}]$
10^2	3.34×10^{-3} $[3.26\times10^{-3}, 3.82\times10^{-3}]$	2.09×10^{-3} $[2.05\times10^{-3}, 2.18\times10^{-3}]$	1.64 $[1.59, 1.77]$
10^3	3.32×10^{-2} $[3.25\times10^{-2}, 3.93\times10^{-2}]$	4.27×10^{-3} $[4.17\times10^{-3}, 4.40\times10^{-3}]$	8.02 $[7.76, 8.75]$
10^4	3.33×10^{-1} $[3.27\times10^{-1}, 3.93\times10^{-1}]$	5.53×10^{-3} $[5.37\times10^{-3}, 5.65\times10^{-3}]$	6.23×10^{1} $[6.02\times10^{1}, 6.75\times10^{1}]$
10^5	3.38 $[3.29, 3.84]$	6.59×10^{-3} $[6.34\times10^{-3}, 6.73\times10^{-3}]$	5.29×10^{2} $[5.05\times10^{2}, 5.71\times10^{2}]$
10^6	3.43×10^{1} $[3.39\times10^{1}, 4.04\times10^{1}]$	5.53×10^{-3} $[5.19\times10^{-3}, 5.70\times10^{-3}]$	6.67×10^{3} $[6.22\times10^{3}, 7.03\times10^{3}]$
10^7	3.49×10^{2} $[3.43\times10^{2}, 3.91\times10^{2}]$	4.97×10^{-3} $[4.71\times10^{-3}, 5.26\times10^{-3}]$	7.33×10^{4} $[6.95\times10^{4}, 7.62\times10^{4}]$
10^8	3.87×10^{3} $[3.84\times10^{3}, 4.07\times10^{3}]$	4.90×10^{-3} $[4.66\times10^{-3}, 5.16\times10^{-3}]$	8.03×10^{5} $[7.76\times10^{5}, 8.36\times10^{5}]$
10^9	$\approx 3.87\times10^{4}$	4.90×10^{-3} $[4.73\times10^{-3}, 5.10\times10^{-3}]$	$\approx 8.05\times10^{6}$
10^{10}	$\approx 3.87\times10^{5}$	4.93×10^{-3} $[4.75\times10^{-3}, 5.15\times10^{-3}]$	$\approx 7.87\times10^{7}$
10^{11}	$\approx 3.87\times10^{6}$	4.92×10^{-3} $[4.86\times10^{-3}, 5.10\times10^{-3}]$	$\approx 7.89\times10^{8}$
10^{12}	$\approx 3.87\times10^{7}$	4.96×10^{-3} $[4.85\times10^{-3}, 5.11\times10^{-3}]$	$\approx 7.81\times10^{9}$
10^{13}	$\approx 3.87\times10^{8}$	4.96×10^{-3} $[4.75\times10^{-3}, 5.10\times10^{-3}]$	$\approx 7.87\times10^{10}$
10^{14}	$\approx 3.87\times10^{9}$	4.91×10^{-3} $[4.63\times10^{-3}, 5.10\times10^{-3}]$	$\approx 8.02\times10^{11}$
10^{15}	$\approx 3.87\times10^{10}$	4.94×10^{-3} $[4.38\times10^{-3}, 5.10\times10^{-3}]$	$\approx 8.17\times10^{12}$

B.3 ONLINE UCB-ADVANTAGE

We next consider online UCB-Advantage, where the policy changes at data-dependent update boundaries. Under conventional simulation, episodes are generated sequentially, and learner updates are triggered as soon as the corresponding stage or reference-value thresholds are reached. In contrast, the adaptive mode of our fast simulation framework generates an aggregate Markov flow for a candidate batch of episodes under the current policy and uses hypergeometric splitting to locate the first update boundary. The aggregate statistics before the boundary are processed in batch, the boundary episode is handled according to the original learner update, and simulation then restarts under the updated policy. Between two consecutive update boundaries, the quantities entering a Q-update are determined by the learner state fixed at the beginning of the batch, including the relevant value functions and visit counts, together with the additive accumulators collected over the current stage. Hence, these quantities can be recovered from the aggregate flow without materializing the individual episodes. Both implementations exploit the deterministic action selection of UCB-Advantage when simulating transitions. The confidence factor ι and bonus confidence constants are fixed to one in Algorithm 4, and the reference-update threshold is set to HSA.

Learning performance is evaluated with regret. For each batch, let $N_1(s)$ denote the number of episodes whose initial state is s. The regret contributed by this batch is $\sum_{s\in\mathcal{S}} N_1(s)\big(V_1^\star(s)-V_1^\pi(s)\big)$, where $V_1^\star(s)$ is the optimal initial-step value in state s, and $V_1^\pi(s)$ is the value of policy π used throughout that batch. At each batch endpoint, including a possible terminal partial batch, the cumulative regret is obtained by summing the regret contributions from all batches up to that point.

The regret results are shown in Figure 1. For each method, we use ten independent replicates on the same fixed MDP, each consisting of $K = 10^8$ episodes, and record cumulative regret at every completed batch endpoint; straight segments are used only to interpolate between recorded endpoints for visualization. In each plot, the solid curve is the median over the ten replicates, with the shaded region showing the empirical 10th–90th percentile range. Across all four problem scales, the conventional and fast regret curves closely track each other over the common episode range. At the largest scale, $(H, S, A) = (10, 15, 10)$, the final median regrets at $K = 10^8$ are approximately 511861 and 511803, respectively. This close agreement is consistent with the distributional equivalence of the adaptive fast simulation procedure and conventional online simulation established by Theorem 3.5.

We further examine the number of processed adaptive batches in Figure 2. This statistic includes all processed batches, including a possible terminal partial batch at the prescribed episode budget, and provides a direct diagnostic of whether the fast simulator reproduces the batching behavior of the original algorithm. The conventional and fast methods again exhibit closely matched trajectories. At the largest scale, their final median batch counts are approximately 48773 and 48749, respectively.

The timing results in Tables 21–24 illustrate the cost introduced by adaptive splitting. Unlike direct batch simulation, the adaptive method must repeatedly locate data-dependent boundaries and restart simulation after learner updates, so it is not uniformly faster at small budgets. Conventional simulation grows approximately linearly with K, whereas the fast method replaces long episode-by-episode segments between updates with aggregate flow generation and boundary refinement, resulting in slower runtime growth at large budgets.

For each problem scale, the boldfaced budget in the corresponding timing table denotes the smallest episode count on the tested budget grid at which the mean value of $\mathrm{Regret}(K)/K$ for the conventional method falls below 0.01. At these accuracy-relevant budgets and beyond, the fast simulator is consistently faster than conventional simulation. For the largest scale, conventional simulation remains faster from $K = 10^2$ to 10^7, but at the last measured budget $K = 10^8$, its median simulator time is 3543 seconds compared with 2032 seconds for fast simulation, a speedup of about 1.72. The crossover point and runtime growth of the adaptive fast method vary with the MDP size and budget. At $K = 10^{15}$ for the largest scale $(10, 15, 10)$, the fast simulator remains directly measurable, with a median runtime of 7420 seconds, or approximately 2.06 hours. In contrast, the extrapolated conventional runtime at the same budget is approximately 3.54×10^{10} seconds, or about 1120 years, making direct conventional simulation computationally prohibitive. This simulator cost is far higher than the maximum learner time of about 38 seconds across all tested configurations, showing that simulation overwhelmingly dominates the computational cost in the large-budget regime. Overall, these results demonstrate that, despite the overhead of adaptive boundary refinement, our method yields substantial computational gains at large budgets by avoiding episode-by-episode simulation.

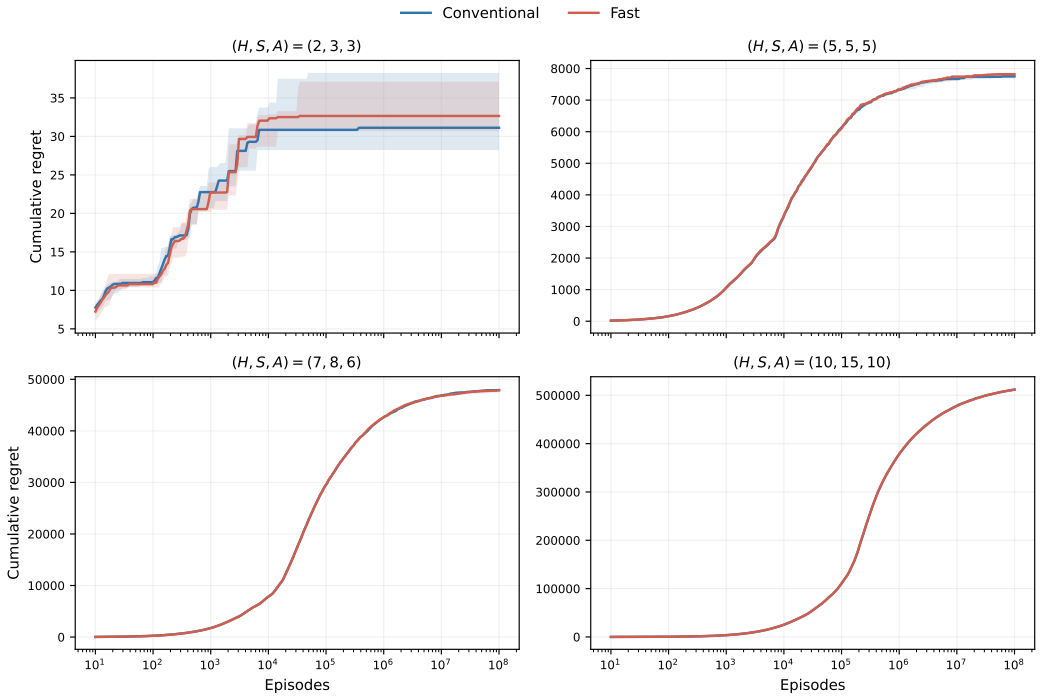

Figure 1: Cumulative UCB-Advantage regret on four fixed MDPs, one per scale. Conventional and fast simulation each use ten independent simulation seeds. Lines are seed medians and shaded bands are empirical 10th–90th percentiles. Intermediate regret values are visually interpolated between recorded batch endpoints.

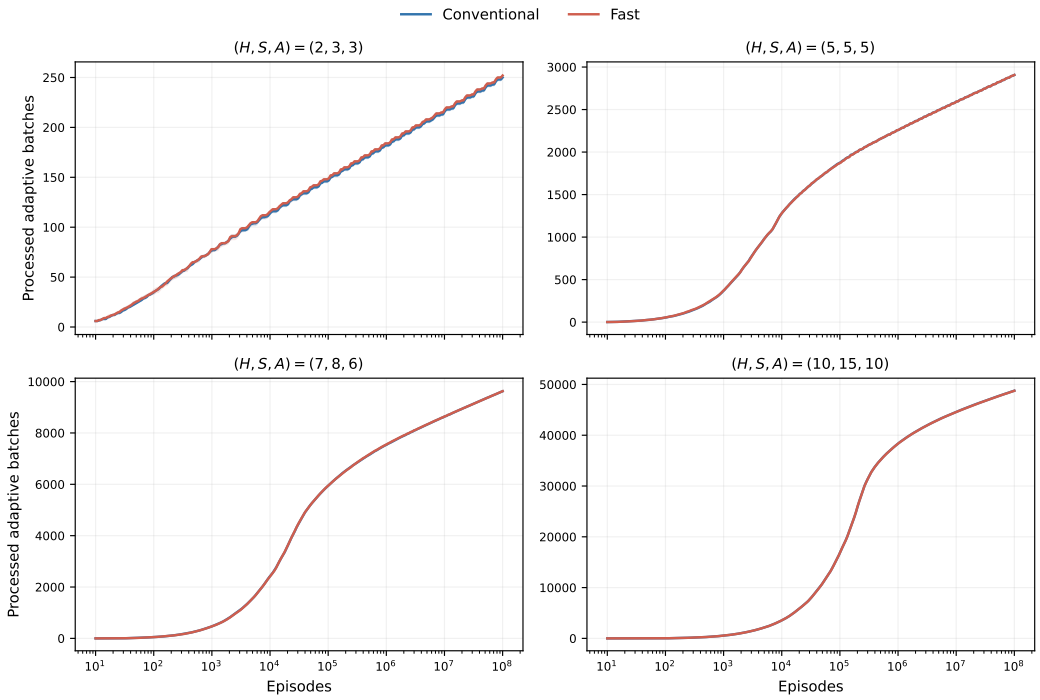

Figure 2: Processed adaptive batch count on the same four fixed MDPs. Lines are medians and shading is the empirical 10th–90th percentile range over ten simulation seeds per method. Counts change only at recorded batch endpoints.

Table 21: Simulator time for $(H, S, A) = (2, 3, 3)$, in seconds. Entries are medians with 95% percentile bootstrap CIs over ten MDP instances; speedup is the median of paired time ratios. Conventional entries beyond the last measured 10^8 budget are marked $\approx$ and extrapolated by a factor of ten per additional power of ten.

Budget	Conventional (s)	Fast (s)	Speedup
10^1	1.64×10^{-4} $[1.56\times10^{-4}, 1.81\times10^{-4}]$	8.18×10^{-4} $[7.76\times10^{-4}, 9.48\times10^{-4}]$	1.97×10^{-1} $[1.83\times10^{-1}, 2.27\times10^{-1}]$
10^2	1.41×10^{-3} $[1.38\times10^{-3}, 1.58\times10^{-3}]$	1.41×10^{-2} $[1.25\times10^{-2}, 1.62\times10^{-2}]$	1.04×10^{-1} $[9.63\times10^{-2}, 1.09\times10^{-1}]$
10^3	1.24×10^{-2} $[1.19\times10^{-2}, 1.38\times10^{-2}]$	5.60×10^{-2} $[5.25\times10^{-2}, 5.89\times10^{-2}]$	2.37×10^{-1} $[2.17\times10^{-1}, 2.47\times10^{-1}]$
$\mathbf{10^4}$	$\mathbf{1.20\times10^{-1}}$ $\mathbf{[1.16\times10^{-1}, 1.34\times10^{-1}]}$	$\mathbf{1.16\times10^{-1}}$ $\mathbf{[1.07\times10^{-1}, 1.25\times10^{-1}]}$	$\mathbf{1.10}$ $\mathbf{[1.01, 1.14]}$
10^5	1.20 $[1.16, 1.33]$	1.96×10^{-1} $[1.79\times10^{-1}, 2.04\times10^{-1}]$	6.58 $[5.90, 6.85]$
10^6	1.19×10^{1} $[1.14\times10^{1}, 1.33\times10^{1}]$	2.82×10^{-1} $[2.65\times10^{-1}, 2.94\times10^{-1}]$	4.40×10^{1} $[4.03\times10^{1}, 4.65\times10^{1}]$
10^7	1.20×10^{2} $[1.16\times10^{2}, 1.33\times10^{2}]$	3.89×10^{-1} $[3.69\times10^{-1}, 4.06\times10^{-1}]$	3.19×10^{2} $[3.02\times10^{2}, 3.36\times10^{2}]$
10^8	1.19×10^{3} $[1.18\times10^{3}, 1.36\times10^{3}]$	5.11×10^{-1} $[4.82\times10^{-1}, 5.36\times10^{-1}]$	2.44×10^{3} $[2.33\times10^{3}, 2.55\times10^{3}]$
10^9	$\approx 1.19\times10^{4}$	6.52×10^{-1} $[6.29\times10^{-1}, 6.88\times10^{-1}]$	$\approx 1.88\times10^{4}$
10^{10}	$\approx 1.19\times10^{5}$	8.05×10^{-1} $[7.74\times10^{-1}, 8.48\times10^{-1}]$	$\approx 1.53\times10^{5}$
10^{11}	$\approx 1.19\times10^{6}$	9.65×10^{-1} $[9.23\times10^{-1}, 1.02]$	$\approx 1.28\times10^{6}$
10^{12}	$\approx 1.19\times10^{7}$	1.14 $[1.10, 1.20]$	$\approx 1.07\times10^{7}$
10^{13}	$\approx 1.19\times10^{8}$	1.32 $[1.30, 1.40]$	$\approx 9.16\times10^{7}$
10^{14}	$\approx 1.19\times10^{9}$	1.52 $[1.48, 1.60]$	$\approx 7.99\times10^{8}$
10^{15}	$\approx 1.19\times10^{10}$	1.74 $[1.71, 1.85]$	$\approx 6.96\times10^{9}$

Table 22: Simulator time for $(H, S, A) = (5, 5, 5)$, in seconds. Entries are medians with 95% percentile bootstrap CIs over ten MDP instances; speedup is the median of paired time ratios. Conventional entries beyond the last measured 10^8 budget are marked $\approx$ and extrapolated by a factor of ten per additional power of ten.

Budget	Conventional (s)	Fast (s)	Speedup
10^1	2.41×10^{-4} $[2.22\times10^{-4}, 2.52\times10^{-4}]$	5.98×10^{-4} $[2.98\times10^{-4}, 7.18\times10^{-4}]$	4.08×10^{-1} $[3.44\times10^{-1}, 1.96]$
10^2	2.63×10^{-3} $[2.43\times10^{-3}, 2.72\times10^{-3}]$	5.11×10^{-2} $[4.82\times10^{-2}, 5.31\times10^{-2}]$	5.14×10^{-2} $[4.93\times10^{-2}, 5.36\times10^{-2}]$
10^3	2.47×10^{-2} $[2.33\times10^{-2}, 2.63\times10^{-2}]$	6.60×10^{-1} $[6.31\times10^{-1}, 7.25\times10^{-1}]$	3.63×10^{-2} $[3.54\times10^{-2}, 3.87\times10^{-2}]$
10^4	2.31×10^{-1} $[2.15\times10^{-1}, 2.44\times10^{-1}]$	3.65 $[3.52, 3.85]$	6.24×10^{-2} $[6.15\times10^{-2}, 6.35\times10^{-2}]$
10^5	2.22 $[2.16, 2.35]$	7.45 $[7.15, 7.84]$	2.98×10^{-1} $[2.93\times10^{-1}, 3.04\times10^{-1}]$
$\mathbf{10^6}$	$\mathbf{2.20\times10^{1}}$ $\mathbf{[2.13\times10^{1}, 2.33\times10^{1}]}$	$\mathbf{1.11\times10^{1}}$ $\mathbf{[1.06\times10^{1}, 1.19\times10^{1}]}$	$\mathbf{1.99}$ $\mathbf{[1.95, 2.03]}$
10^7	2.21×10^{2} $[2.19\times10^{2}, 2.32\times10^{2}]$	1.51×10^{1} $[1.46\times10^{1}, 1.62\times10^{1}]$	1.48×10^{1} $[1.44\times10^{1}, 1.51\times10^{1}]$
10^8	2.16×10^{3} $[2.12\times10^{3}, 2.31\times10^{3}]$	1.98×10^{1} $[1.91\times10^{1}, 2.09\times10^{1}]$	1.10×10^{2} $[1.08\times10^{2}, 1.14\times10^{2}]$
10^9	$\approx 2.16\times10^{4}$	2.51×10^{1} $[2.40\times10^{1}, 2.64\times10^{1}]$	$\approx 8.65\times10^{2}$
10^{10}	$\approx 2.16\times10^{5}$	3.10×10^{1} $[2.96\times10^{1}, 3.22\times10^{1}]$	$\approx 7.08\times10^{3}$
10^{11}	$\approx 2.16\times10^{6}$	3.70×10^{1} $[3.55\times10^{1}, 3.88\times10^{1}]$	$\approx 5.88\times10^{4}$
10^{12}	$\approx 2.16\times10^{7}$	4.39×10^{1} $[4.23\times10^{1}, 4.59\times10^{1}]$	$\approx 4.98\times10^{5}$
10^{13}	$\approx 2.16\times10^{8}$	5.20×10^{1} $[4.96\times10^{1}, 5.39\times10^{1}]$	$\approx 4.23\times10^{6}$
10^{14}	$\approx 2.16\times10^{9}$	5.95×10^{1} $[5.72\times10^{1}, 6.19\times10^{1}]$	$\approx 3.70\times10^{7}$
10^{15}	$\approx 2.16\times10^{10}$	6.79×10^{1} $[6.49\times10^{1}, 7.04\times10^{1}]$	$\approx 3.25\times10^{8}$

Table 23: Simulator time for $(H, S, A) = (7, 8, 6)$, in seconds. Entries are medians with 95% percentile bootstrap CIs over ten MDP instances; speedup is the median of paired time ratios. Conventional entries beyond the last measured 10^8 budget are marked $\approx$ and extrapolated by a factor of ten per additional power of ten.

Budget	Conventional (s)	Fast (s)	Speedup
10^1	2.53×10^{-4} $[2.50\times10^{-4}, 2.76\times10^{-4}]$	9.84×10^{-5} $[9.43\times10^{-5}, 1.11\times10^{-4}]$	2.59 $[2.40, 2.70]$
10^2	2.78×10^{-3} $[2.74\times10^{-3}, 3.00\times10^{-3}]$	6.05×10^{-2} $[5.90\times10^{-2}, 7.26\times10^{-2}]$	4.53×10^{-2} $[4.22\times10^{-2}, 4.70\times10^{-2}]$
10^3	2.74×10^{-2} $[2.71\times10^{-2}, 2.99\times10^{-2}]$	1.33 $[1.32, 1.66]$	2.05×10^{-2} $[1.87\times10^{-2}, 2.06\times10^{-2}]$
10^4	2.59×10^{-1} $[2.55\times10^{-1}, 2.82\times10^{-1}]$	1.10×10^{1} $[1.06\times10^{1}, 1.34\times10^{1}]$	2.33×10^{-2} $[2.18\times10^{-2}, 2.42\times10^{-2}]$
10^5	2.47 $[2.43, 2.67]$	4.02×10^{1} $[3.87\times10^{1}, 4.85\times10^{1}]$	6.07×10^{-2} $[5.62\times10^{-2}, 6.30\times10^{-2}]$
10^6	2.42×10^{1} $[2.39\times10^{1}, 2.54\times10^{1}]$	6.53×10^{1} $[6.24\times10^{1}, 7.88\times10^{1}]$	3.67×10^{-1} $[3.37\times10^{-1}, 3.85\times10^{-1}]$
$\mathbf{10^7}$	$\mathbf{2.41\times10^{2}}$ $\mathbf{[2.39\times10^{2}, 2.82\times10^{2}]}$	$\mathbf{9.07\times10^{1}}$ $\mathbf{[8.62\times10^{1}, 1.09\times10^{2}]}$	$\mathbf{2.66}$ $\mathbf{[2.52, 2.77]}$
10^8	2.31×10^{3} $[2.29\times10^{3}, 2.79\times10^{3}]$	1.17×10^{2} $[1.12\times10^{2}, 1.43\times10^{2}]$	1.96×10^{1} $[1.91\times10^{1}, 2.06\times10^{1}]$
10^9	$\approx 2.31\times10^{4}$	1.47×10^{2} $[1.40\times10^{2}, 1.81\times10^{2}]$	$\approx 1.55\times10^{2}$
10^{10}	$\approx 2.31\times10^{5}$	1.80×10^{2} $[1.73\times10^{2}, 2.18\times10^{2}]$	$\approx 1.28\times10^{3}$
10^{11}	$\approx 2.31\times10^{6}$	2.19×10^{2} $[2.09\times10^{2}, 2.69\times10^{2}]$	$\approx 1.05\times10^{4}$
10^{12}	$\approx 2.31\times10^{7}$	2.60×10^{2} $[2.50\times10^{2}, 3.20\times10^{2}]$	$\approx 8.83\times10^{4}$
10^{13}	$\approx 2.31\times10^{8}$	3.15×10^{2} $[2.95\times10^{2}, 3.78\times10^{2}]$	$\approx 7.43\times10^{5}$
10^{14}	$\approx 2.31\times10^{9}$	3.82×10^{2} $[3.59\times10^{2}, 4.41\times10^{2}]$	$\approx 6.23\times10^{6}$
10^{15}	$\approx 2.31\times10^{10}$	4.39×10^{2} $[4.17\times10^{2}, 5.09\times10^{2}]$	$\approx 5.40\times10^{7}$

Table 24: Simulator time for $(H, S, A) = (10, 15, 10)$, in seconds. Entries are medians with 95% percentile bootstrap CIs over ten MDP instances; speedup is the median of paired time ratios. Conventional entries beyond the last measured 10^8 budget are marked $\approx$ and extrapolated by a factor of ten per additional power of ten.

Budget	Conventional (s)	Fast (s)	Speedup
10^1	3.68×10^{-4} $[3.42\times10^{-4}, 3.91\times10^{-4}]$	2.14×10^{-4} $[2.01\times10^{-4}, 2.23\times10^{-4}]$	1.75 $[1.63, 1.81]$
10^2	3.75×10^{-3} $[3.47\times10^{-3}, 3.89\times10^{-3}]$	3.98×10^{-2} $[3.41\times10^{-2}, 4.37\times10^{-2}]$	9.52×10^{-2} $[8.86\times10^{-2}, 1.02\times10^{-1}]$
10^3	3.98×10^{-2} $[3.71\times10^{-2}, 4.22\times10^{-2}]$	4.39 $[4.03, 4.47]$	9.28×10^{-3} $[9.00\times10^{-3}, 9.57\times10^{-3}]$
10^4	3.86×10^{-1} $[3.56\times10^{-1}, 4.02\times10^{-1}]$	5.01×10^{1} $[4.62\times10^{1}, 5.11\times10^{1}]$	7.77×10^{-3} $[7.69\times10^{-3}, 7.94\times10^{-3}]$
10^5	3.67 $[3.43, 3.79]$	3.38×10^{2} $[3.09\times10^{2}, 3.44\times10^{2}]$	1.10×10^{-2} $[1.08\times10^{-2}, 1.12\times10^{-2}]$
10^6	3.56×10^{1} $[3.33\times10^{1}, 3.67\times10^{1}]$	1.06×10^{3} $[9.92\times10^{2}, 1.08\times10^{3}]$	3.39×10^{-2} $[3.32\times10^{-2}, 3.46\times10^{-2}]$
10^7	3.52×10^{2} $[3.27\times10^{2}, 3.64\times10^{2}]$	1.57×10^{3} $[1.48\times10^{3}, 1.63\times10^{3}]$	2.22×10^{-1} $[2.15\times10^{-1}, 2.31\times10^{-1}]$
$\mathbf{10^8}$	$\mathbf{3.54\times10^{3}}$ $\mathbf{[3.27\times10^{3}, 3.62\times10^{3}]}$	$\mathbf{2.03\times10^{3}}$ $\mathbf{[1.91\times10^{3}, 2.11\times10^{3}]}$	$\mathbf{1.72}$ $\mathbf{[1.66, 1.78]}$
10^9	$\approx 3.54\times10^{4}$	2.54×10^{3} $[2.37\times10^{3}, 2.63\times10^{3}]$	$\approx 1.38\times10^{1}$
10^{10}	$\approx 3.54\times10^{5}$	3.07×10^{3} $[2.95\times10^{3}, 3.13\times10^{3}]$	$\approx 1.15\times10^{2}$
10^{11}	$\approx 3.54\times10^{6}$	3.73×10^{3} $[3.61\times10^{3}, 3.91\times10^{3}]$	$\approx 9.35\times10^{2}$
10^{12}	$\approx 3.54\times10^{7}$	4.56×10^{3} $[4.32\times10^{3}, 4.72\times10^{3}]$	$\approx 7.69\times10^{3}$
10^{13}	$\approx 3.54\times10^{8}$	5.42×10^{3} $[5.16\times10^{3}, 5.60\times10^{3}]$	$\approx 6.48\times10^{4}$
10^{14}	$\approx 3.54\times10^{9}$	6.37×10^{3} $[6.14\times10^{3}, 6.56\times10^{3}]$	$\approx 5.52\times10^{5}$
10^{15}	$\approx 3.54\times10^{10}$	7.42×10^{3} $[7.25\times10^{3}, 7.62\times10^{3}]$	$\approx 4.74\times10^{6}$

B.3.1 Large-Population Multivariate-Hypergeometric Sampling

The adaptive fast simulator requires multivariate-hypergeometric (MHG) sampling when splitting an aggregate flow at a data-dependent boundary. An MHG draw can be generated through a sequence of conditional scalar hypergeometric draws. Specifically, let $b_1, \dots, b_q$ denote the color counts, with $N_{\mathrm{pop}} = \sum_{i=1}^{q} b_i$, and suppose that n items are drawn without replacement. Define

$$R_i = \sum_{j=i}^{q} b_j, \qquad n_1 = n, \qquad n_{i+1} = n_i - x_i.$$

We sequentially sample

$$x_i \sim \mathrm{Hypergeom}(R_i, b_i, n_i), \qquad i = 1, \dots, q-1,$$

and set $x_q = n_q$. The resulting joint probability is

$$\prod_{i=1}^{q-1} \frac{\binom{b_i}{x_i}\binom{R_i - b_i}{n_i - x_i}}{\binom{R_i}{n_i}} = \frac{\prod_{i=1}^{q}\binom{b_i}{x_i}}{\binom{N_{\mathrm{pop}}}{n}},$$

which is exactly the MHG probability mass function. Therefore, provided that each scalar hypergeometric draw is exact, this sequential construction produces the same MHG distribution as direct multivariate-hypergeometric sampling.

In our implementation, for total population size $N_{\mathrm{pop}} < 10^9$, we use NumPy's MHG sampler with `method="marginals"`, which implements a sequential construction based on scalar hypergeometric draws. Since this method requires $N_{\mathrm{pop}} < 10^9$, we use a custom large-population implementation when $N_{\mathrm{pop}} \geq 10^9$, retaining the same conditional factorization. Each scalar draw is generated using an HRUA-style ratio-of-uniforms rejection procedure. To reduce precision loss from subtracting large log-factorial terms, the compiled implementation evaluates the required log-probability ratios through stable log-factorial differences, with numerically ambiguous acceptance decisions recomputed using higher-precision arithmetic. The exactness result above concerns the probabilistic construction under exact scalar sampling; the custom sampler is its finite-precision numerical implementation. Thus, the large-population modification leaves the MHG factorization and distributional argument unchanged.

C Proof of Theorem 3.1

Proof. We prove equality of the joint flow law by induction over the step index h.

Under conventional episode-by-episode simulation, the initial states $X_{1,1}, \dots, X_{m,1}$ are i.i.d. from ν. Hence

$$N_1^{\mathrm{iid}}(\cdot) \sim \mathrm{Multinomial}(m, \nu),$$

which is exactly the law used in Equation (4).

Now fix $h \in [H]$ and suppose that the joint law of the aggregate flow up to the beginning of step h, and in particular the law of N_h, agrees under the two constructions. We compare the conditional laws used to extend this flow through step h.

Let

$$\mathcal{H}_h^- = \sigma(N_1, C_1, E_1, \ldots, C_{h-1}, E_{h-1}),$$

with $\mathcal{H}_1^- = \sigma(N_1)$, denote the aggregate-flow history up to the beginning of step h, and let

$$\mathcal{G}_h^- = \sigma(X_{i,1}, A_{i,1}, X_{i,2}, \ldots, A_{i,h-1}, X_{i,h} : i \in [m])$$

denote the corresponding finer labeled history. Then $\mathcal{H}_h^- \subseteq \mathcal{G}_h^-$, and N_h is $\mathcal{H}_h^-$-measurable.

For conventional simulation, conditional on $\mathcal{G}_h^-$, for each $s \in \mathcal{S}$, the labels satisfying $X_{i,h} = s$ choose their actions independently according to $\pi_h(\cdot \mid s)$. Therefore

$$C_h(s, \cdot) \mid \mathcal{G}_h^- \sim \text{Multinomial}(N_h(s), \pi_h(\cdot \mid s)),$$

independently across s. Since this conditional law depends on $\mathcal{G}_h^-$ only through the $\mathcal{H}_h^-$-measurable count vector N_h, the tower property yields

$$C_h(s, \cdot) \mid \mathcal{H}_h^- \sim \text{Multinomial}(N_h(s), \pi_h(\cdot \mid s)),$$

independently across s. This is exactly the conditional law in Equation (5).

Next, let

$$\mathcal{G}_h = \sigma(\mathcal{G}_h^-, A_{1,h}, \ldots, A_{m,h}) \qquad \text{and} \qquad \mathcal{H}_h = \sigma(\mathcal{H}_h^-, C_h).$$

Then $\mathcal{H}_h \subseteq \mathcal{G}_h$. Conditional on $\mathcal{G}_h$, for each $(s, a) \in \mathcal{S} \times \mathcal{A}$, the labels satisfying $(X_{i,h}, A_{i,h}) = (s, a)$ transition independently according to $P_h(\cdot \mid s, a)$. Thus

$$E_h(s, a, \cdot) \mid \mathcal{G}_h \sim \text{Multinomial}(C_h(s, a), P_h(\cdot \mid s, a)),$$

independently across (s, a). Since this conditional law depends on $\mathcal{G}_h$ only through the $\mathcal{H}_h$-measurable count array C_h, another application of the tower property gives

$$E_h(s, a, \cdot) \mid \mathcal{H}_h \sim \text{Multinomial}(C_h(s, a), P_h(\cdot \mid s, a)),$$

independently across (s, a), which is exactly the conditional law in Equation (6).

Finally, in both constructions the next state counts are the same deterministic function of the transition-flow counts,

$$N_{h+1}(s') = \sum_{s,a} E_h(s, a, s'), \qquad s' \in \mathcal{S},$$

as in Equation (7). Therefore, the conditional law of (C_h, E_h, N_{h+1}) given the previously generated aggregate flow is identical under conventional simulation and direct Markov-flow sampling. Induction over $h = 1, \ldots, H$ gives equality of the joint law of $(N_1, C_1, E_1, \ldots, C_H, E_H)$, and hence

$$F_m^{\text{fc}} \overset{d}{=} F_m^{\text{iid}}.$$

□

D Proof of Theorem 3.3

We first record the conditional combinatorial structure of a labeled batch given its aggregate flow. Throughout this section, let $\omega_1, \ldots, \omega_m \overset{\text{iid}}{\sim} \mathbb{P}_\pi$ and let F_m denote their aggregate Markov flow. Fix $I_L = \{1, \ldots, m_L\}$, and let $F_{m_L}^{\text{true},L}$ be the flow of the trajectories whose labels belong to I_L.

Lemma D.1 (Conditional uniformity given the full flow)**.** *Fix a flow realization $f = (N_1, C, E)$ such that $\mathbb{P}(F_m = f) > 0$. Conditional on $F_m = f$, all ordered labeled trajectory sequences $(\omega_1, \ldots, \omega_m)$ compatible with f are equally likely.*

Proof. For any ordered labeled trajectory sequence, independence of the episodes and Equation (1) give

$$\prod_{i=1}^{m} \mathbb{P}_\pi(\omega_i) = \prod_{s\in\mathcal{S}} \nu(s)^{N_1(s)} \prod_{h=1}^{H} \prod_{s,a} \pi_h(a \mid s)^{C_h(s,a)} \prod_{h=1}^{H} \prod_{s,a,s'} P_h(s' \mid s,a)^{E_h(s,a,s')}.$$

Here and below, we use the convention $0^0 = 1$.

For every sequence compatible with f, the right-hand side is the same because it depends only on the aggregate counts in f. Since $\mathbb{P}(F_m = f) > 0$, conditioning on $F_m = f$ therefore produces the uniform distribution over all compatible ordered labeled trajectory sequences. □

Lemma D.2 (Conditional labeling representation)**.** *Fix $f = (N_1, C, E)$ with $\mathbb{P}(F_m = f) > 0$. Conditional on $F_m = f$, the labeled trajectories can equivalently be generated by the following combinatorial procedure:*

1. *assign the m labels to initial states uniformly among all assignments having counts $N_1(s)$;*
2. *for each $h \in [H]$ and each state s, once the set of labels currently at s is known, assign those labels uniformly to outgoing categories $(a, s') \in \mathcal{A} \times \mathcal{S}$ with prescribed counts $E_h(s, a, s')$.*

Conditional on the label sets entering the rows at step h, the outgoing assignments are independent across states s.

Proof. The procedure generates only labeled trajectory sequences compatible with f. For any fixed compatible sequence, its probability under this combinatorial procedure is

$$\frac{1}{m!/\prod_s N_1(s)!} \prod_{h=1}^{H} \prod_{s} \frac{1}{N_h(s)!/\prod_{a,s'} E_h(s,a,s')!},$$

which depends only on f and not on the particular compatible sequence. Hence the procedure is uniform over all compatible sequences and is therefore identical to the conditional law in Lemma D.1. The row-wise conditional independence follows directly from the construction, because after the entering label sets are fixed, each state row is assigned separately subject only to its prescribed row counts. □

For the true left batch, write N_1^L, C_h^L, E_h^L for its corresponding counts. Before processing step h, define the aggregate left-flow sigma-field

$$\mathcal{H}_{h-1}^L = \sigma\big(N_1^L, E_1^L, \ldots, E_{h-1}^L\big),$$

with $\mathcal{H}_0^L = \sigma(N_1^L)$. Thus $\mathcal{H}_{h-1}^L$ determines $N_h^L(s)$ for every s. We will also use the finer labeled-history sigma-field

$$\mathcal{G}_h^- = \sigma(X_{i,1}, A_{i,1}, X_{i,2}, \ldots, A_{i,h-1}, X_{i,h} : i \in [m]),$$

where for $h = 1$ this means $\mathcal{G}_1^- = \sigma(X_{1,1}, \ldots, X_{m,1})$.

Lemma D.3 (Initial-state hypergeometric split)**.** *Conditional on F_m,*

$$N_1^L(\cdot) \sim \mathrm{MHG}(N_1(\cdot), m_L).$$

Proof. Fix a realization $F_m = f$ and write $n_s = N_1(s)$. By Lemma D.2, conditional on $F_m = f$, the m labels are assigned uniformly to the initial states among all assignments having counts $(n_s)_{s\in\mathcal{S}}$. The total number of such assignments is

$$\frac{m!}{\prod_{s\in\mathcal{S}} n_s!}.$$

Fix a feasible vector $y = (y_s)_{s\in\mathcal{S}}$ satisfying $0 \le y_s \le n_s$ and $\sum_s y_s = m_L$, and let $m_R = m - m_L$. The number of initial-state assignments for which the first m_L labels have category counts y_s is

$$\frac{m_L!}{\prod_s y_s!} \frac{m_R!}{\prod_s (n_s - y_s)!}.$$

Consequently,

$$\mathbb{P}\big(N_1^L(s) = y_s,\ \forall s \mid F_m = f\big) = \frac{\frac{m_L!}{\prod_s y_s!}\frac{m_R!}{\prod_s (n_s - y_s)!}}{\frac{m!}{\prod_s n_s!}} = \frac{m_L! m_R!}{m!}\prod_s \frac{n_s!}{y_s!(n_s - y_s)!}$$
$$= \frac{\prod_{s\in\mathcal{S}} \binom{n_s}{y_s}}{\binom{m}{m_L}}.$$

Since $n_s = N_1(s)$ under $F_m = f$, this is exactly $\mathrm{MHG}(N_1(\cdot), m_L)$. □

Lemma D.4 (Row-wise hypergeometric split)**.** *Fix $h \in [H]$. Conditional on F_m and $\mathcal{H}_{h-1}^L$, for each $s \in \mathcal{S}$,*

$$E_h^L(s,\cdot,\cdot) \sim \mathrm{MHG}\big(\{E_h(s,a,s')\}_{a\in\mathcal{A},\ s'\in\mathcal{S}}, N_h^L(s)\big),$$

and these arrays are conditionally independent across s.

Proof. Fix realizations $F_m = f$ and $\mathcal{H}_{h-1}^L = \eta$. These realizations determine the aggregate left state counts $N_h^L(s)$, but in general they do *not* determine which particular labels in I_L are currently at each state. This is the point at which it is useful to condition temporarily on the finer labeled history $\mathcal{G}_h^-$.

For each $s \in \mathcal{S}$, define

$$\mathcal{I}_h(s) = \{i \in [m] : X_{i,h} = s\}, \quad L_h(s) = \mathcal{I}_h(s) \cap I_L.$$

Conditional on $\mathcal{G}_h^-$, both sets are fixed, with

$$|\mathcal{I}_h(s)| = N_h(s), \qquad |L_h(s)| = N_h^L(s).$$

By Lemma D.2, conditional on $F_m = f$ and the entering label sets $\{\mathcal{I}_h(s)\}_{s\in\mathcal{S}}$, the labels in row (h,s) are assigned uniformly to the outgoing categories (a,s') with prescribed category counts $\{E_h(s,a,s')\}_{a,s'}$. The assignments for distinct states are conditionally independent.

Fix a state s and a feasible array $y_s = \{y_s(a,s')\}_{a,s'}$ satisfying

$$0 \le y_s(a,s') \le E_h(s,a,s'), \qquad \sum_{a,s'} y_s(a,s') = N_h^L(s).$$

The number of row assignments having the prescribed full row counts is

$$\frac{N_h(s)!}{\prod_{a,s'} E_h(s,a,s')!}.$$

Among them, the number for which the fixed left subset $L_h(s)$ receives category counts $y_s(a,s')$ is

$$\frac{N_h^L(s)!}{\prod_{a,s'} y_s(a,s')!}\frac{(N_h(s) - N_h^L(s))!}{\prod_{a,s'}(E_h(s,a,s') - y_s(a,s'))!}.$$

Therefore,

$$\mathbb{P}\big(E_h^L(s,a,s') = y_s(a,s')\ \forall a,s' \mid F_m = f, \mathcal{G}_h^-\big) = \frac{\frac{N_h^L(s)!}{\prod_{a,s'} y_s(a,s')!}\frac{(N_h(s)-N_h^L(s))!}{\prod_{a,s'}(E_h(s,a,s')-y_s(a,s'))!}}{\frac{N_h(s)!}{\prod_{a,s'} E_h(s,a,s')!}}$$
$$= \frac{\prod_{a,s'} \binom{E_h(s,a,s')}{y_s(a,s')}}{\binom{N_h(s)}{N_h^L(s)}}.$$

Thus, conditional on $F_m = f$ and $\mathcal{G}_h^-$,

$$E_h^L(s,\cdot,\cdot) \sim \mathrm{MHG}\big(\{E_h(s,a,s')\}_{a,s'}, N_h^L(s)\big).$$

Because the current outgoing row assignments are product-form across states, for any collection of feasible arrays $\{y_s\}_{s\in\mathcal{S}}$,

$$\mathbb{P}\big(E_h^L(s,\cdot,\cdot) = y_s,\ \forall s \mid F_m = f, \mathcal{G}_h^-\big) = \prod_{s\in\mathcal{S}} \frac{\prod_{a,s'} \binom{E_h(s,a,s')}{y_s(a,s')}}{\binom{N_h(s)}{N_h^L(s)}}.$$

Crucially, the right-hand side does not depend on the identities of the entering labels or any other aspect of $\mathcal{G}_h^-$; it depends on the past only through $N_h^L(s)$, which is already measurable with respect to $\mathcal{H}_{h-1}^L$. Since $\mathcal{H}_{h-1}^L \subseteq \mathcal{G}_h^-$, the tower property gives

$$\mathbb{P}\big(E_h^L(s,\cdot,\cdot) = y_s,\ \forall s \mid F_m, \mathcal{H}_{h-1}^L\big) = \mathbb{E}\big[\mathbb{P}\big(E_h^L(s,\cdot,\cdot) = y_s,\ \forall s \mid F_m, \mathcal{G}_h^-\big)\big|F_m, \mathcal{H}_{h-1}^L\big] = \prod_{s\in\mathcal{S}} \frac{\prod_{a,s'} \binom{E_h(s,a,s')}{y_s(a,s')}}{\binom{N_h(s)}{N_h^L(s)}}.$$

This is exactly the claimed row-wise multivariate-hypergeometric law and establishes the conditional independence across states. □

Lemma D.5 (Equivalence of row-wise and action-then-transition splitting)**.** *Fix $h \in [H]$ and $s \in \mathcal{S}$. Conditional on F_m and $\mathcal{H}_{h-1}^L$, the row-wise draw in Lemma D.4 is equivalent in distribution to first sampling*

$$C_h^L(s,\cdot) \sim \mathrm{MHG}\big(C_h(s,\cdot), N_h^L(s)\big)$$

and then, independently across $a \in \mathcal{A}$, sampling

$$E_h^L(s,a,\cdot) \sim \mathrm{MHG}\big(E_h(s,a,\cdot), C_h^L(s,a)\big)\,.$$

The two-stage implementation also preserves the conditional independence across states s.

Proof. Fix h and s, and abbreviate $e_{a,s'} = E_h(s,a,s')$ and $c_a = C_h(s,a) = \sum_{s'} e_{a,s'}$. Under the row-wise split, for every feasible array $y = \{y(a,s')\}_{a,s'}$ with $\sum_{a,s'} y(a,s') = N_h^L(s)$,

$$\mathbb{P}(Y = y) = \frac{\prod_{a,s'} \binom{e_{a,s'}}{y(a,s')}}{\binom{N_h(s)}{N_h^L(s)}}.$$

Let $Y_A(a) = \sum_{s'} Y(a,s')$. For any feasible action-total vector $r = (r_a)_a$, Vandermonde's identity gives

$$\mathbb{P}(Y_A(a) = r_a,\ \forall a) = \frac{\prod_a \sum_{\{y(a,s'):\sum_{s'} y(a,s') = r_a\}} \prod_{s'} \binom{e_{a,s'}}{y(a,s')}}{\binom{N_h(s)}{N_h^L(s)}} = \frac{\prod_a \binom{c_a}{r_a}}{\binom{N_h(s)}{N_h^L(s)}}.$$

Hence $Y_A \sim \mathrm{MHG}(C_h(s,\cdot), N_h^L(s))$. Moreover, conditional on $Y_A(a) = r_a$ for every a,

$$\mathbb{P}(Y(a,s') = y(a,s'),\ \forall a,s' \mid Y_A = r) = \prod_a \frac{\prod_{s'} \binom{e_{a,s'}}{y(a,s')}}{\binom{c_a}{r_a}},$$

which is exactly the product of the conditional draws $\mathrm{MHG}(E_h(s,a,\cdot), r_a)$ across a. Thus the row-wise and two-stage procedures have the same law. The independence across states follows from Lemma D.4. □

Proof of Theorem 3.3. We induct over h on the conditional law of the left flow.

By Lemma D.3, the split rule Equation (8) generates exactly the conditional law of the true left initial-state counts N_1^L given F_m. Suppose now that the split has generated the correct conditional law of $\mathcal{H}_{h-1}^L$. Conditional on $(F_m, \mathcal{H}_{h-1}^L)$, Lemma D.4 gives the correct row-wise law of E_h^L, and Lemma D.5 shows that this law is exactly implemented by first drawing Equation (9) and then drawing Equation (10). After E_h^L is sampled, flow conservation gives

$$N_{h+1}^L(s') = \sum_{s,a} E_h^L(s,a,s'),$$

which is the same deterministic relation satisfied by the true left sub-batch. Therefore the induction continues through all $h \in [H]$, yielding

$$F_{m_L}^{\mathrm{split},L} \mid F_m \overset{d}{=} F_{m_L}^{\mathrm{true},L} \mid F_m.$$

For the right child, conditional on the parent flow and the left child, all right counts are determined by subtraction:

$$F_{m_R}^{\mathrm{split},R} = F_m - F_{m_L}^{\mathrm{split},L}.$$

The same identity holds for the true partition of the first m_L and last m_R labeled trajectories. Therefore

$$\left(F_{m_L}^{\mathrm{split},L}, F_{m_R}^{\mathrm{split},R}\right) \mid F_m \overset{d}{=} \left(F_{m_L}^{\mathrm{true},L}, F_{m_R}^{\mathrm{true},R}\right) \mid F_m,$$

which proves the claimed joint conditional correctness. □

E Proof of Theorem 3.5

Proof. Fix Z and consider the sequence $\omega_1, \omega_2, \ldots$ generated by conventional episode-by-episode simulation while the policy remains equal to $\pi(Z)$. Let G_j be the Markov flow of its first j episodes and let

$$\tau = \inf\{j \geq 1 : g(Z, G_j) = 1\}.$$

We first consider the no-trigger case. By prefix monotonicity, $g(Z, G_m) = 0$ if and only if $\tau > m$. By Theorem 3.1, the sampled candidate flow F_m has exactly the same law as G_m. Therefore, the event $g(Z, F_m) = 0$ and the returned flow have the same joint distribution as the corresponding no-trigger event and prefix flow under conventional simulation.

Now suppose $g(Z, F_m) = 1$, corresponding under the common root-flow law to $\tau \leq m$. We justify the recursive use of Theorem 3.3 by maintaining the following invariant for the conventional i.i.d. episode sequence: conditional on the entire revealed-flow history, the labeled trajectories in the current retained node are uniform over all sequences compatible with that node's flow. At the root, this follows from Lemma D.1.

Suppose the invariant holds for a retained node with index interval I, and write F_I for its flow. Conditional on the revealed history, I and F_I are fixed, and the trajectories in I are uniform over all labeled sequences compatible with F_I. Thus Theorem 3.3 gives the exact joint conditional law of the flows of its left and right children. Conditional on the revealed history and both child flows, the labeled trajectories in the two children are independent, and each child is uniform over the sequences compatible with its own flow. The choice of which child to retain is determined by the revealed flows and the accepted prefix flow; it uses no additional information about the trajectories' labels. Consequently, revealing the child flows and making this choice leaves the invariant valid for the retained child. Since the splitting procedure uses fresh randomness at each refinement, induction over the refinement depth shows that every adaptive split has the same conditional law as the corresponding split of the conventional episode sequence.

It remains to verify that the binary search retains exactly the batch containing the first trigger. Throughout the search, maintain the invariant that (i) F^{pre} is the flow of a contiguous prefix that has not triggered, and (ii) the current node immediately follows this prefix and contains the first triggering episode. The invariant holds initially because $F^{\mathrm{pre}} = 0$ and the root batch contains a trigger. Suppose a current node is split into contiguous children (F^L, n_L) and (F^R, n_R). If $g(Z, F^{\mathrm{pre}} + F^L) = 1$, prefix monotonicity implies that the first trigger lies in the left child, so the algorithm retains F^L. Otherwise, $F^{\mathrm{pre}} + F^L$ has not triggered, and because the current node contains a trigger, the first trigger must lie in the right child; the algorithm therefore accepts F^L into F^{pre} and retains F^R. Thus the invariant is preserved.

When the current node has size one, it corresponds to the boundary episode at index τ, while F^{pre} is the flow of the first $\tau - 1$ episodes. Consequently, the returned boundary flow satisfies

$$F^{\mathrm{pre}} + F \overset{d}{=} G_\tau.$$

Thus Algorithm 1 returns the correct boundary-inclusive aggregate flow without reconstructing the boundary trajectory.

Each binary refinement replaces a node of size n by a child of size at most $\lceil n/2 \rceil$. After q splits, the retained node therefore has size at most $\lceil m/2^q \rceil$, so at most $\lceil \log_2 m \rceil$ splits are required to reach the boundary episode.

Finally, suppose the boundary compatibility condition in Section 2 holds. Since the returned flow $\widetilde{G}_\tau$ has the same distribution as G_τ under conventional simulation, applying the same boundary update

produces the same post-boundary learner state Z' and hence the same next policy. The unaccepted suffix of the candidate batch was generated under the old policy $\pi(Z)$ and is discarded. Fresh episodes generated under the updated policy $\pi(Z')$ therefore have the same conditional law as the episodes generated after the boundary by conventional simulation. Repeating this argument over successive boundaries proves equality in distribution of the sequence of post-boundary learner states. □

F Review of Algorithms Used in Section 5

In this section, we briefly review the three RL algorithms used in Section 5. We rewrite their notation to match the notation used throughout the paper whenever possible.

F.1 Model-Based Planning with a Generative Model

We first review the finite-horizon model-based planning with a generative model considered in Li et al. (2024b). A generative model allows the learner to query any state-action-step triple (s, a, h) and obtain an independent next-state sample from $P_h(\cdot \mid s, a)$. Suppose each triple is queried J times. Let $Y_{h,j}(s, a)$ denote the j-th sampled next state for (s, a, h). The empirical transition kernel is

$$\widehat{P}_h(s' \mid s, a) = \frac{1}{J} \sum_{j=1}^{J} \mathbb{I}\{Y_{h,j}(s, a) = s'\}.$$

Equivalently, if $E_h(s, a, s')$ denotes the resulting transition count, then

$$\widehat{P}_h(s' \mid s, a) = E_h(s, a, s')/J.$$

The algorithm constructs the empirical MDP using $\widehat{P}$ and the original reward function r, and then computes an optimal policy of this empirical MDP by backward dynamic programming.

Algorithm 2 Model-based planning with a generative model

Require: Generative model, query count J
1: **for** $h = 1, \ldots, H$, $s \in \mathcal{S}$, $a \in \mathcal{A}$ **do**
2: Draw $Y_{h,1}(s, a), \ldots, Y_{h,J}(s, a) \overset{\text{iid}}{\sim} P_h(\cdot \mid s, a)$
3: Set $\widehat{P}_h(s' \mid s, a) \leftarrow J^{-1} \sum_{j=1}^{J} \mathbb{I}\{Y_{h,j}(s, a) = s'\}$ for all $s' \in \mathcal{S}$
4: **end for**
5: $\widehat{V}_{H+1}(s) \leftarrow 0$ for all $s \in \mathcal{S}$
6: **for** $h = H, H-1, \ldots, 1$ **do**
7: $\widehat{Q}_h(s, a) \leftarrow r_h(s, a) + \sum_{s'} \widehat{P}_h(s' \mid s, a) \widehat{V}_{h+1}(s')$
8: $\widehat{V}_h(s) \leftarrow \max_{a \in \mathcal{A}} \widehat{Q}_h(s, a)$
9: $\widehat{\pi}_h(s) \leftarrow \arg\max_{a \in \mathcal{A}} \widehat{Q}_h(s, a)$
10: **end for**
11: **return** $\widehat{\pi} = \{\widehat{\pi}_h\}_{h=1}^{H}$

Thus, after sampling, the planning stage depends on the generative-model observations only through the empirical transition counts $E_h(s, a, s')$.

F.2 Vanilla Pessimistic Value Iteration

We next review Vanilla Pessimistic Value Iteration (VPVI) (Yin & Wang, 2021). Let the offline dataset consist of K trajectories generated by a fixed behavior policy μ. Using the notation of Section 2, $C_h(s, a)$ is the number of visits to (s, a) at step h, and $E_h(s, a, s')$ is the corresponding transition count. For $C_h(s, a) > 0$, VPVI forms the empirical transition kernel

$$\widehat{P}_h(s' \mid s, a) = \frac{E_h(s, a, s')}{C_h(s, a)}.$$

When $C_h(s, a) = 0$, the original algorithm sets $\widehat{P}_h(\cdot \mid s, a)$ to the uniform distribution. Under the deterministic-reward setting considered in this paper, the empirical reward equals $r_h(s, a)$ whenever $C_h(s, a) > 0$; for an unvisited pair, the original algorithm sets the empirical reward to zero.

Let $\iota = \log(HSA/\delta)$. VPVI uses a Hoeffding-type pessimistic penalty of the form

$$\Gamma_h(s,a) = \begin{cases} cH\iota/\sqrt{C_h(s,a)}, & C_h(s,a) > 0, \\ cH\iota, & C_h(s,a) = 0, \end{cases}$$

where $c > 0$ is an absolute constant. Starting from $\widehat{V}_{H+1} = 0$, it performs backward value iteration, subtracting Γ_h from the empirical Bellman update.

Algorithm 3 Vanilla Pessimistic Value Iteration

Require: Offline dataset of K trajectories, failure probability δ
Construct $C_h(s,a)$ and $E_h(s,a,s')$ from the dataset
for $h \in [H]$, $(s,a) \in \mathcal{S} \times \mathcal{A}$ **do**
  if $C_h(s,a) > 0$ **then**
    $\widehat{P}_h(s' \mid s,a) \leftarrow E_h(s,a,s')/C_h(s,a)$
    $\widehat{r}_h(s,a) \leftarrow r_h(s,a)$
  else
    $\widehat{P}_h(s' \mid s,a) \leftarrow 1/S$ and $\widehat{r}_h(s,a) \leftarrow 0$
  end if
end for
$\widehat{V}_{H+1}(s) \leftarrow 0$ for all $s \in \mathcal{S}$
$\iota \leftarrow \log(HSA/\delta)$
for $h = H, H-1, \ldots, 1$ **do**
  $\overline{Q}_h(s,a) \leftarrow \widehat{r}_h(s,a) + \sum_{s'} \widehat{P}_h(s' \mid s,a)\widehat{V}_{h+1}(s')$
  Set $\Gamma_h(s,a) \leftarrow cH\iota/\sqrt{C_h(s,a)}$ if $C_h(s,a) > 0$, and $\Gamma_h(s,a) \leftarrow cH\iota$ otherwise
  $Q_h(s,a) \leftarrow \max\{0, \min\{\overline{Q}_h(s,a) - \Gamma_h(s,a), H-h+1\}\}$
  $\widehat{\pi}_h(s) \leftarrow \arg\max_{a \in \mathcal{A}} Q_h(s,a)$
  $\widehat{V}_h(s) \leftarrow Q_h(s, \widehat{\pi}_h(s))$
end for
return $\widehat{\pi} = \{\widehat{\pi}_h\}_{h=1}^H$

In particular, under our deterministic-reward setting, the dataset enters VPVI only through the visitation and transition counts (C, E).

F.3 UCB-Advantage

Finally, we review UCB-Advantage (Zhang et al., 2020), a model-free online algorithm based on reference-advantage decomposition and stage-wise updates. For each (s,a,h), let $n_h(s,a)$ denote its cumulative number of visits and $\check{n}_h(s,a)$ the number of visits in the current stage. The stage lengths satisfy

$$e_1 = H, \quad e_{j+1} = \left\lfloor \left(1 + \frac{1}{H}\right) e_j \right\rfloor,$$

and we write

$$\mathcal{L} = \left\{ \sum_{i=1}^{j} e_i : j = 1, 2, \ldots \right\}$$

for the cumulative visit counts at which stage updates are triggered.

Besides Q_h and V_h, UCB-Advantage maintains a reference value V_h^{ref}. For a transition (s,a,s') at step h, the current-stage accumulators are updated as

$$\check{\mu}_h(s,a) \stackrel{+}{=} V_{h+1}(s') - V_{h+1}^{\mathrm{ref}}(s'),\ \check{\nu}_h(s,a) \stackrel{+}{=} V_{h+1}(s'),\ \check{\sigma}_h(s,a) \stackrel{+}{=} \left(V_{h+1}(s') - V_{h+1}^{\mathrm{ref}}(s')\right)^2,$$

while the global reference accumulators satisfy

$$\mu_h^{\mathrm{ref}}(s,a) \stackrel{+}{=} V_{h+1}^{\mathrm{ref}}(s'), \qquad \sigma_h^{\mathrm{ref}}(s,a) \stackrel{+}{=} \left(V_{h+1}^{\mathrm{ref}}(s')\right)^2.$$

The checked quantities are reset after each stage, whereas the reference accumulators use all samples collected so far.

At the end of a stage, writing $n = n_h(s,a)$ and $\check{n} = \check{n}_h(s,a)$, the algorithm constructs a Bernstein-type reference-advantage bonus

$$b_h(s,a) = c_1\sqrt{\frac{\sigma_h^{\mathrm{ref}}(s,a)/n - \left(\mu_h^{\mathrm{ref}}(s,a)/n\right)^2}{n}}\,\iota + c_2\sqrt{\frac{\check{\sigma}_h(s,a)/\check{n} - \left(\check{\mu}_h(s,a)/\check{n}\right)^2}{\check{n}}}\,\iota + c_3\left(\frac{H\iota}{n} + \frac{H\iota}{\check{n}} + \frac{H\iota^{3/4}}{n^{3/4}} + \frac{H\iota^{3/4}}{\check{n}^{3/4}}\right),$$

together with the Hoeffding-type bonus

$$\bar{b}_h(s,a) = 2\sqrt{\frac{H^2\iota}{\check{n}}}.$$

Here ι is a logarithmic confidence factor and $c_1, c_2, c_3 > 0$ are universal constants.

The Q-value is updated by taking the minimum of the standard stage-wise estimate, the reference-advantage estimate, and its previous value:

$$Q_h(s,a) \leftarrow \min\left\{ r_h(s,a) + \frac{\check{\nu}_h(s,a)}{\check{n}} + \bar{b}_h(s,a),\right.$$
$$\left. r_h(s,a) + \frac{\mu_h^{\mathrm{ref}}(s,a)}{n} + \frac{\check{\mu}_h(s,a)}{\check{n}} + b_h(s,a),\ Q_h(s,a)\right\}.$$

Afterward, $V_h(s) = \max_a Q_h(s,a)$ and the current-stage accumulators are reset. When the state-level visitation count reaches the threshold N_0, the reference value is refreshed as $V_h^{\mathrm{ref}}(s) \leftarrow V_h(s)$.

Two features are particularly relevant to Section 5. First, between stage or reference-value updates, the functions appearing in the accumulated target and moment statistics remain fixed. Second, within a boundary episode, a step-h update modifies only the step-h value tables, whereas subsequent decisions use tables indexed by $h+1, h+2, \ldots$. Hence, the stage-wise statistics can be accumulated using the corresponding pre-boundary functions while preserving the update order of the original algorithm.

Algorithm 4 UCB-Advantage

Require: Number of episodes K, stage schedule $\mathcal{L}$, reference threshold N_0

Initialize all accumulators to zero
For $h \in [H]$, set $Q_h(s,a) \leftarrow H-h+1$, $V_h(s) \leftarrow H-h+1$, and $V_h^{\mathrm{ref}}(s) \leftarrow H$
Set $V_{H+1}(s) \leftarrow 0$ and $V_{H+1}^{\mathrm{ref}}(s) \leftarrow 0$ for all $s \in \mathcal{S}$
for $k = 1, \ldots, K$ **do**
    Observe $X_{k,1}$
    for $h = 1, \ldots, H$ **do**
        $A_{k,h} \leftarrow \arg\max_{a\in\mathcal{A}} Q_h(X_{k,h}, a)$
        Observe $X_{k,h+1} \sim P_h(\cdot \mid X_{k,h}, A_{k,h})$
        Set $(s,a,s') \leftarrow (X_{k,h}, A_{k,h}, X_{k,h+1})$
        $n_h(s,a) \leftarrow n_h(s,a)+1$, $\check{n}_h(s,a) \leftarrow \check{n}_h(s,a)+1$
        Update $\check{\mu}_h, \check{\nu}_h, \check{\sigma}_h, \mu_h^{\mathrm{ref}}, \sigma_h^{\mathrm{ref}}$ using (s,a,s')
        if $n_h(s,a) \in \mathcal{L}$ **then**
            Compute $b_h(s,a)$ and $\bar{b}_h(s,a)$
            Update $Q_h(s,a)$ using the stage-wise and reference-advantage estimates
            $V_h(s) \leftarrow \max_{a'\in\mathcal{A}} Q_h(s,a')$
            Reset $\check{n}_h(s,a), \check{\mu}_h(s,a), \check{\nu}_h(s,a), \check{\sigma}_h(s,a)$ to zero
        end if
        if $\sum_{a'\in\mathcal{A}} n_h(s,a') = N_0$ **then**
            $V_h^{\mathrm{ref}}(s) \leftarrow V_h(s)$
        end if
    end for
end for

G Further Applications of the Exact Fast-Simulation Framework

The representative applications in Section 5 illustrate our framework in simulator-based, offline, and online RL. All learner-side arithmetic costs in this appendix are measured under the same unit-cost arithmetic convention as in Section 4. We next discuss two further extensions that highlight additional algorithmic structures compatible with fast Markov-flow simulation. The first considers a round-based single-agent specialization of FedQ-Bernstein (Zheng et al., 2024), showing that visit-dependent learning rates, trajectory-dependent moments, and Bernstein confidence bonuses can be implemented without learner-side computation scaling with the round size. The second extends the framework to federated RL, where multiple agents independently collect local experience between learning rounds.

G.1 Round-Based Q-Learning with Visit-Dependent Learning Rates

The visit-dependent learning rate

$$\alpha_t = \frac{H+1}{H+t}$$

was introduced in model-free episodic Q-learning by Jin et al. (2018) and is used by a broad class of subsequent Q-learning methods. A direct implementation updates the Q-value after individual visits and therefore appears to retain learner-side computation linear in the number of trajectories in a batch. We use the single-agent specialization of FedQ-Bernstein (Zheng et al., 2024) to show that this dependence is not inherent. When $M = 1$, the agent and server operations of the federated algorithm collapse into a single round-based learner: each round executes a fixed policy until an event-triggered visitation threshold is reached, accumulates first- and second-moment statistics, and performs round-end Q-updates, using single-visit updates only in the bounded low-count regime and aggregate updates thereafter. Importantly, the Bernstein bonus is also represented through aggregate endpoint quantities rather than a visit-indexed cumulative sum.

G.1.1 Algorithm Review

We specialize FedQ-Bernstein (Zheng et al., 2024) to $M = 1$. The learner proceeds in rounds indexed by k. At the beginning of round k, it maintains Q^k, V^k, cumulative visit counts $N_h^k(s,a)$, and historical first- and second-moment accumulators. The deterministic greedy policy

$$\pi_h^k(s) \in \arg\max_{a\in\mathcal{A}} Q_h^k(s,a)$$

and the value function V^k remain fixed throughout the round.

For each (s,a,h), define the round threshold

$$c_h^k(s,a) = \max\left\{1, \left\lfloor \frac{N_h^k(s,a)}{H(H+1)} \right\rfloor\right\}. \tag{11}$$

The learner repeatedly executes π^k until, at the end of an episode, the within-round visitation count of some (s,a,h) reaches $c_h^k(s,a)$. Let $C_h^k(s,a)$ denote the number of visits to (s,a,h) during round k. For the next states observed after these visits, define

$$S_{1,h}^k(s,a) = \sum_{i\in\mathcal{I}_h^k(s,a)} V_{h+1}^k(X_{i,h+1}), \qquad S_{2,h}^k(s,a) = \sum_{i\in\mathcal{I}_h^k(s,a)} \left(V_{h+1}^k(X_{i,h+1})\right)^2,$$

where $\mathcal{I}_h^k(s,a)$ is the set of trajectories in round k that visit (s,a,h). When $C_h^k(s,a) > 0$, write

$$\overline{V}_{h+1}^k(s,a) = \frac{S_{1,h}^k(s,a)}{C_h^k(s,a)}.$$

Following FedQ-Bernstein, the learner maintains cumulative first- and second-moment sums

$$W_{1,k+1}(s,a,h) = W_{1,k}(s,a,h) + S_{2,h}^k(s,a), \qquad W_{2,k+1}(s,a,h) = W_{2,k}(s,a,h) + S_{1,h}^k(s,a),$$

and forms the empirical variance

$$W_{k+1}(s,a,h) = \frac{W_{1,k+1}(s,a,h)}{N_h^{k+1}(s,a)} - \left(\frac{W_{2,k+1}(s,a,h)}{N_h^{k+1}(s,a)}\right)^2$$

whenever $N_h^{k+1}(s,a) > 0$.

Let ι denote the confidence parameter and $c' > 0$ the absolute constant used in FedQ-Bernstein. Specializing its Bernstein endpoint confidence quantity to $M = 1$, for $N_h^k(s,a) > 0$ define

$$\beta_h^{\mathrm{B},k}(s,a) = c' \min\left\{ \sqrt{\frac{H\iota}{N_h^k(s,a)}\left(W_k(s,a,h) + H\right)} + \frac{(\sqrt{H^7SA} + \sqrt{SAH^6})\iota}{N_h^k(s,a)}, \sqrt{\frac{H^3\iota}{N_h^k(s,a)}} \right\},$$

and set $\beta_h^{\mathrm{B},k}(s,a) = 0$ when $N_h^k(s,a) = 0$. Define $\beta_h^{\mathrm{B},k+1}(s,a)$ analogously using the post-round count $N_h^{k+1}(s,a)$ and variance statistic $W_{k+1}(s,a,h)$. Thus, each endpoint quantity is a deterministic function of the corresponding aggregate visit count and empirical variance and can be evaluated using a constant number of scalar arithmetic operations once these quantities are available.

For integers $n_1 \le n_2$, let $\alpha^{\mathrm{c}}(n_1, n_2) = \prod_{t=n_1}^{n_2}(1-\alpha_t)$. If a tuple has $n = N_h^k(s,a)$ visits before round k and receives $c = C_h^k(s,a)$ new visits, define

$$\alpha_{\mathrm{agg}}(n,c) = 1 - \alpha^{\mathrm{c}}(n+1, n+c). \tag{12}$$

The corresponding aggregate Bernstein bonus is

$$\widetilde{\beta}_h^k(s,a) = \beta_h^{\mathrm{B},k+1}(s,a) - \alpha^{\mathrm{c}}(n+1,n+c)\beta_h^{\mathrm{B},k}(s,a). \tag{13}$$

As in the original algorithm, let $i_0 = 2H(H+1)$ and distinguish two regimes. If $n < i_0$, Equation (11) gives $c_h^k(s,a) = 1$, so a fixed (s,a,h) can be visited at most once during the round. If such a visit occurs and its next state is s', writing $t = n+1$, the corresponding single-visit bonus is

$$\widetilde{b}_t(s,a,h) = \frac{\beta_h^{\mathrm{B},k+1}(s,a) - (1-\alpha_t)\beta_h^{\mathrm{B},k}(s,a)}{2\alpha_t},$$

and the update is

$$Q_h^{k+1}(s,a) = (1-\alpha_t)Q_h^k(s,a) + \alpha_t\left(r_h(s,a) + V_{h+1}^k(s') + \widetilde{b}_t(s,a,h)\right). \tag{14}$$

If $n \ge i_0$, all new visits in the round receive equal weight and the aggregate update becomes

$$Q_h^{k+1}(s,a) = (1-\alpha_{\mathrm{agg}})Q_h^k(s,a) + \alpha_{\mathrm{agg}}\left(r_h(s,a) + \overline{V}_{h+1}^k(s,a)\right) + \frac{\widetilde{\beta}_h^k(s,a)}{2}. \tag{15}$$

Tuples not visited during the round retain their previous Q-values. Finally,

$$V_h^{k+1}(s) = \min\left\{H, \max_{a\in\mathcal{A}} Q_h^{k+1}(s,a)\right\}, \qquad \pi_h^{k+1}(s) \in \arg\max_{a\in\mathcal{A}} Q_h^{k+1}(s,a).$$

Algorithm 5 summarizes this single-agent specialization. Since $M = 1$, local trajectory collection and global aggregation are performed by the same learner, and no central server or communication operation is required.

G.1.2 Application of Our Framework and Computational Cost

We now apply adaptive Markov-flow simulation to the single-agent FedQ-Bernstein specialization. The round-based learner itself continues according to its original event-triggered rule. For the computational comparison, we consider the first K episodes of this learning process. If the K-th episode occurs before the end of a communication round, we truncate the computational accounting at episode K without applying a round-end update that has not yet been triggered.

Let Z_k denote the complete learner state at the beginning of round k, including π^k, V^k, the cumulative visit counts, the historical moment accumulators, and the round thresholds. Let G_j denote the Markov flow of the first j trajectories generated during the round, and let $C_h^{(j)}(s,a)$ denote its state-action counts. The round terminates at the first prefix satisfying

$$g_{\mathrm{QB}}(Z_k, G_j) = \mathbb{I}\{\exists(s,a,h) : C_h^{(j)}(s,a) \ge c_h^k(s,a)\}.$$

Algorithm 5 Single-Agent Specialization of FedQ-Bernstein

Initialize $Q_h^1(s,a) = V_h^1(s) = H$, $N_h^1(s,a) = 0$, and $W_{1,1}(s,a,h) = W_{2,1}(s,a,h) = 0$
Set $V_{H+1}^k(s) = 0$ for all $s \in \mathcal{S}$ and all k
for learning rounds $k = 1, 2, \ldots$ **do**
  Set $\pi_h^k(s) \in \arg\max_{a \in \mathcal{A}} Q_h^k(s,a)$
  Set $c_h^k(s,a) \leftarrow \max\{1, \lfloor N_h^k(s,a)/(H(H+1)) \rfloor\}$ for all (s,a,h)
  Initialize $C_h^k(s,a)$, $S_{1,h}^k(s,a)$, and $S_{2,h}^k(s,a)$ to zero
  repeat
    Generate one trajectory under the fixed policy π^k
    Update $C_h^k(s,a)$, $S_{1,h}^k(s,a)$, and $S_{2,h}^k(s,a)$ along the trajectory
  until some $C_h^k(s,a) = c_h^k(s,a)$ at the end of an episode
  Set $N_h^{k+1}(s,a) \leftarrow N_h^k(s,a) + C_h^k(s,a)$ and update $W_{1,k+1}$, $W_{2,k+1}$, and W_{k+1}
  Set $Q^{k+1} \leftarrow Q^k$
  for each (s,a,h) with $C_h^k(s,a) > 0$ **do**
    Set $n \leftarrow N_h^k(s,a)$ and $c \leftarrow C_h^k(s,a)$
    if $n < i_0$ **then**
      Apply the single-visit update in Equation (14)
    else
      Compute α_{agg} and $\widetilde{\beta}_h^k(s,a)$ from Equations (12) and (13)
      Apply the aggregate update in Equation (15)
    end if
  end for
  Set $V_h^{k+1}(s) \leftarrow \min\{H, \max_a Q_h^{k+1}(s,a)\}$ and $\pi_h^{k+1}(s) \in \arg\max_a Q_h^{k+1}(s,a)$
end for

Since the thresholds are determined before the round and each $C_h^{(j)}(s,a)$ is nondecreasing in j, this trigger is prefix monotone. Because π^k is deterministic, only $C_h^{(j)}(s, \pi_h^k(s))$ can increase for each (s,h), so each trigger evaluation costs $O(HS)$. If κ episodes have already been accounted for, Algorithm 1 is called with candidate size $K - \kappa$. If no round boundary occurs before the remaining horizon, the algorithm returns the full remaining flow and the accounting stops at episode K without a round-end update. Otherwise, adaptive binary refinement locates the exact terminating episode, with the trigger cost dominated by the $O(HS^2)$ deterministic-policy split cost.

All trajectory-dependent statistics required by a completed round update are flow measurable. If $E_h^k(s,a,s')$ denotes the transition counts of the recovered boundary flow, then

$$S_{1,h}^k(s,a) = \sum_{s' \in \mathcal{S}} E_h^k(s,a,s') V_{h+1}^k(s'), \qquad S_{2,h}^k(s,a) = \sum_{s' \in \mathcal{S}} E_h^k(s,a,s') \left(V_{h+1}^k(s')\right)^2.$$

Because V^k is fixed throughout round k, the visitation counts, round means, cumulative first and second moments, and empirical variance statistics are deterministic functions of the pre-round learner state and the boundary flow.

The visit-dependent learning-rate coefficient can also be evaluated without processing the c new visits individually. Since $1 - \alpha_t = (t-1)/(H+t)$,

$$\alpha^{\mathrm{c}}(n+1, n+c) = \prod_{t=n+1}^{n+c} \frac{t-1}{H+t} = \prod_{r=0}^{H} \frac{n+r}{n+c+r}.$$

Thus, Equation (12) can be evaluated using only $H+1$ factors, independently of c.

The Bernstein bonus has the same aggregate structure. The boundary flow determines the post-round first and second moments and hence the endpoint quantity $\beta_h^{\mathrm{B},k+1}(s,a)$. Equation (13) then recovers the entire round bonus contribution from the pre-round and post-round endpoint quantities together with $\alpha^{\mathrm{c}}(n+1, n+c)$. No visit-indexed bonus sum needs to be evaluated.

The low-count regime is also compatible with the recovered flow. When $N_h^k(s,a) < i_0$, Equation (11) gives $c_h^k(s,a) = 1$. Hence, the round contains at most one new visit to a fixed (s,a,h), and if

such a visit occurs, its next state is uniquely determined by the transition-count vector $E_h^k(s, a, \cdot)$. The single-visit update in Equation (14) can therefore be reproduced exactly. In the high-count regime, Equation (15) depends only on the round mean, aggregate learning-rate coefficient, and endpoint Bernstein bonus, all of which are determined by the recovered flow.

All Q-, value-, and policy updates occur only after a round terminates, so the second requirement of the boundary compatibility condition holds automatically. By Theorem 3.5, replacing conventional trajectory generation within completed rounds by Algorithm 1, together with the possible terminal truncation at episode K, preserves the joint distribution of the completed-round learner states and the aggregate statistics of the terminal partial round up to episode K.

We finally compare simulator-side and overall computational costs. Let $R_{\max}(K)$ be a deterministic upper bound on the number of communication-round batches intersecting the first K episodes, with the possible terminal partial batch absorbed into the bound. Conventional episode-by-episode simulation has simulator-side cost $\Theta(HK)$. Maintaining the visit counts and first- and second-moment accumulators along the generated transitions also costs $O(HK)$. Because the within-round policy is deterministic, at most one action per (s, h) receives new visits; the round-end aggregate coefficients therefore cost $O(H^2S)$, while recomputing values and the greedy policy costs $O(HSA)$. Thus, its overall computational cost is

$$O\left(HK + R_{\max}(K)HS(H + A)\right).$$

Under adaptive Markov-flow simulation, one forward flow sample and at most $O(\log K)$ conditional splits are required for each round batch intersecting the horizon. Therefore, the simulator-side cost is

$$O\left(R_{\max}(K)C_{\text{flow}}^{\text{det}} \log K\right) = O\left(R_{\max}(K)HS^2 \log K\right).$$

Only the active action $\pi_h^k(s)$ can have nonzero within-round flow for a given (s, h). Hence the first- and second-moment contractions require $O(HS^2)$ arithmetic per completed round, and the aggregate learning-rate coefficients require at most $O(H^2S)$ arithmetic per completed round. The variance, endpoint-bonus, and Q-updates over visited tuples cost at most $O(HS)$, while recomputing the value function and greedy policy costs $O(HSA)$. Thus, the overall cost is

$$O\left(R_{\max}(K)HS^2 \log K + R_{\max}(K)HS(H + A)\right).$$

The communication-round bound of FedQ-Bernstein is the same as that of FedQ-Hoeffding (Zheng et al., 2024). Specializing this deterministic bound to $M = 1$ gives

$$R_{\max}(K) = O\left(H^3SA \log(1 + HK)\right).$$

Consequently, the simulator-side cost becomes

$$O\left(H^4S^3A \log(1 + HK) \log K\right),$$

while the overall computational cost is

$$O\left(H^4S^2A \log(1 + HK)\left[S \log K + H + A\right]\right).$$

Thus, conventional simulation has linear dependence on K in both simulator-side and overall computation cost, whereas adaptive Markov-flow simulation reduces both to $O(\log^2 K)$ dependence.

G.2 Federated RL: FedQ-Advantage

Federated RL provides a second extension of the framework. Between communication rounds, multiple agents independently interact with copies of the environment and subsequently transmit aggregate statistics to a central server. This structure is particularly compatible with Markov-flow simulation: if each agent uses a fixed policy during its local exploration batch and its communicated quantities depend only on aggregate transition statistics, the local trajectories need not be materialized individually.

We illustrate this through FedQ-Advantage (Zheng et al., 2025a). The algorithm separates communication rounds from stage-wise policy updates and uses reference-advantage decomposition for variance reduction. During a communication round, each agent accumulates visitation counts and first- and second-moment statistics of value functions that remain fixed throughout that round. A stage for a fixed (s, a, h) consists of consecutive complete communication rounds, so its statistics are obtained by accumulating the round-wise contributions evaluated with the value functions used when those transitions were collected.

G.2.1 ALGORITHM REVIEW

FedQ-Advantage considers M agents, each interacting with an independent copy of the same finite-horizon MDP. The learning process is divided into communication rounds indexed by k. At the beginning of round k, the server maintains the current Q-estimate Q^k, value function V^k, reference value $V^{\mathrm{ref},k}$, and the count and moment accumulators described below. It constructs the deterministic policy

$$\pi_h^k(s) \in \arg\max_{a\in\mathcal{A}} Q_h^k(s,a)$$

and broadcasts π^k, V^k, $V^{\mathrm{ref},k}$, and the relevant counters to the agents. Define

$$V_h^{\mathrm{adv},k} = V_h^k - V_h^{\mathrm{ref},k}.$$

These functions remain fixed throughout communication round k.

For each (s,a,h), let $N_h^k(s,a)$ denote the cumulative number of visits in all completed stages before round k, let $n_h^k(s,a)$ denote the size of the previous completed stage, and let $\widetilde{n}_h^k(s,a)$ denote the number of visits already accumulated in the current stage before round k. Let $I_h^{\mathrm{ren},k}(s,a)$ indicate whether round k is the first round of a new stage, with $I_h^{\mathrm{ren},1}(s,a)=1$. The communication threshold is

$$c_h^k(s,a) = \begin{cases} \left\lfloor n_h^k(s,a)/(MH) \right\rfloor, & n_h^k(s,a)>0 \text{ and } \widetilde{n}_h^k(s,a) > (1-1/H)n_h^k(s,a), \\ \max\left\{1, \left\lceil \left(n_h^k(s,a) - \widetilde{n}_h^k(s,a)\right)/M \right\rceil\right\}, & \text{otherwise.} \end{cases} \tag{16}$$

FedQ-Advantage allows both synchronization options. When $\mathrm{usyn} = \mathrm{FALSE}$, each agent m independently executes π^k until, at the end of an episode, its local visitation count for at least one (s,a,h) reaches $c_h^k(s,a)$, and the server initiates aggregation after every agent has reached one such local threshold. When $\mathrm{usyn} = \mathrm{TRUE}$, we analyze the episode-synchronized specialization in which all agents advance under a common episode clock; the round terminates as soon as any agent reaches one of its local thresholds, at which point all agents transmit their aggregate statistics.

Let $C_{m,h}^k(s,a)$ denote agent m's number of visits to (s,a,h) during round k. For a fixed function f, define

$$A_{m,h}^k(s,a;f) = \sum_{i\in\mathcal{I}_{m,h}^k(s,a)} f(X_{m,i,h+1}). \tag{17}$$

In addition to the visit counts, each agent transmits quantities of the form Equation (17) for

$$f \in \left\{ V_{h+1}^{\mathrm{ref},k}, V_{h+1}^{\mathrm{adv},k}, V_{h+1}^k, \left(V_{h+1}^{\mathrm{ref},k}\right)^2, \left(V_{h+1}^{\mathrm{adv},k}\right)^2 \right\}. \tag{18}$$

The server stores all-history reference accumulators $\mu_h^{\mathrm{ref},k}, \sigma_h^{\mathrm{ref},k}$ and current-stage accumulators $\mu_h^{\mathrm{adv},k}, \mu_h^{\mathrm{val},k}, \sigma_h^{\mathrm{adv},k}$, initialized to zero. At the end of round k, these are updated pointwise in (s,a,h) as

$$(\mu,\sigma)_h^{\mathrm{ref},k+1}(s,a) = (\mu,\sigma)_h^{\mathrm{ref},k}(s,a) + \sum_{m=1}^M \left(A_{m,h}^k(s,a;V_{h+1}^{\mathrm{ref},k}), A_{m,h}^k(s,a;(V_{h+1}^{\mathrm{ref},k})^2) \right), \tag{19}$$

$$\begin{aligned}(\mu^{\mathrm{adv}},\mu^{\mathrm{val}},\sigma^{\mathrm{adv}})_h^{k+1}(s,a) &= (1-I_h^{\mathrm{ren},k}(s,a))(\mu^{\mathrm{adv}},\mu^{\mathrm{val}},\sigma^{\mathrm{adv}})_h^k(s,a) \\ &\quad + \sum_{m=1}^M \left(A_{m,h}^k(s,a;V_{h+1}^{\mathrm{adv},k}), A_{m,h}^k(s,a;V_{h+1}^k), A_{m,h}^k(s,a;(V_{h+1}^{\mathrm{adv},k})^2) \right). \end{aligned} \tag{20}$$

Thus, when a stage spans multiple rounds, each round contributes the value or advantage function that was fixed in that round; no historical transition is reevaluated using a later value function.

After receiving the local messages, the server computes

$$\widehat{n}_h^{k+1}(s,a) = \widetilde{n}_h^k(s,a) + \sum_{m=1}^M C_{m,h}^k(s,a). \tag{21}$$

A stage renewal occurs when

$$\widehat{n}_h^{k+1}(s,a) \geq \mathbb{I}\{n_h^k(s,a)=0\}MH + \left(1+\frac{1}{H}\right) n_h^k(s,a). \tag{22}$$

Equivalently, with $I_h^{\mathrm{ren},k+1}(s,a)$ denoting the renewal indicator,

$$\left(n_h^{k+1}, \widetilde{n}_h^{k+1}, N_h^{k+1}\right)(s,a) = \begin{cases} \left(\widehat{n}_h^{k+1}(s,a), 0, N_h^k(s,a) + \widehat{n}_h^{k+1}(s,a)\right), & I_h^{\mathrm{ren},k+1}(s,a)=1, \\ \left(n_h^k(s,a), \widehat{n}_h^{k+1}(s,a), N_h^k(s,a)\right), & I_h^{\mathrm{ren},k+1}(s,a)=0. \end{cases} \tag{23}$$

For a renewed stage, the empirical means entering the two Q-estimates are

$$\overline{V}_{h,\mathrm{stage}}^{k+1}(s,a) = \frac{\mu_h^{\mathrm{val},k+1}(s,a)}{n_h^{k+1}(s,a)},\ \overline{V}_{h,\mathrm{ref}}^{k+1}(s,a) = \frac{\mu_h^{\mathrm{ref},k+1}(s,a)}{N_h^{k+1}(s,a)},$$

and

$$\overline{V}_{h,\mathrm{adv}}^{k+1}(s,a) = \frac{\mu_h^{\mathrm{adv},k+1}(s,a)}{n_h^{k+1}(s,a)}.$$

For confidence parameter $p \in (0,1)$, let $\iota = \log(2/p)$. Define

$$\widehat{\mathrm{Var}}_{h,\mathrm{ref}}^{k+1}(s,a) = \frac{\sigma_h^{\mathrm{ref},k+1}(s,a)}{N_h^{k+1}(s,a)} - \left(\frac{\mu_h^{\mathrm{ref},k+1}(s,a)}{N_h^{k+1}(s,a)}\right)^2$$

and

$$\widehat{\mathrm{Var}}_{h,\mathrm{adv}}^{k+1}(s,a) = \frac{\sigma_h^{\mathrm{adv},k+1}(s,a)}{n_h^{k+1}(s,a)} - \left(\frac{\mu_h^{\mathrm{adv},k+1}(s,a)}{n_h^{k+1}(s,a)}\right)^2.$$

The corresponding bonus terms are

$$b_h^{k+1,1}(s,a) = \sqrt{\frac{2H^2\iota}{n_h^{k+1}(s,a)}},$$

$$b_h^{k+1,2}(s,a) = 2\sqrt{\frac{\widehat{\mathrm{Var}}_{h,\mathrm{ref}}^{k+1}(s,a)\iota}{N_h^{k+1}(s,a)}} + 2\sqrt{\frac{\widehat{\mathrm{Var}}_{h,\mathrm{adv}}^{k+1}(s,a)\iota}{n_h^{k+1}(s,a)}} + 10H\left[\left(\frac{\iota}{N_h^{k+1}(s,a)}\right)^{3/4} + \left(\frac{\iota}{n_h^{k+1}(s,a)}\right)^{3/4} + \frac{\iota}{N_h^{k+1}(s,a)} + \frac{\iota}{n_h^{k+1}(s,a)}\right].$$

The resulting Hoeffding-type and reference-advantage estimates can be written as

$$Q_h^{k+1,1}(s,a) = r_h(s,a) + \overline{V}_{h,\mathrm{stage}}^{k+1}(s,a) + b_h^{k+1,1}(s,a), \tag{24}$$

$$Q_h^{k+1,2}(s,a) = r_h(s,a) + \overline{V}_{h,\mathrm{ref}}^{k+1}(s,a) + \overline{V}_{h,\mathrm{adv}}^{k+1}(s,a) + b_h^{k+1,2}(s,a). \tag{25}$$

The Q-estimate is updated only when the stage renews:

$$Q_h^{k+1}(s,a) = \begin{cases} \min\left\{Q_h^{k+1,1}(s,a), Q_h^{k+1,2}(s,a), Q_h^k(s,a)\right\}, & \text{if the stage renews,} \\ Q_h^k(s,a), & \text{otherwise.} \end{cases} \tag{26}$$

The server then sets

$$V_h^{k+1}(s) = \max_{a\in\mathcal{A}} Q_h^{k+1}(s,a), \qquad \pi_h^{k+1}(s) \in \arg\max_{a\in\mathcal{A}} Q_h^{k+1}(s,a).$$

The reference value at (s,h) is refreshed at most once, using the updated $V_h^{k+1}(s)$ in the first round for which

$$\sum_{a\in\mathcal{A}} N_h^{k+1}(s,a) \geq N_0.$$

Thus the reference threshold is based on visits from completed stages and does not include the unfinished-stage count $\widetilde{n}_h^{k+1}$.

Algorithm 6 summarizes FedQ-Advantage under the synchronization conventions described above.

Algorithm 6 FedQ-Advantage

Require: Number of agents M, synchronization flag $\mathrm{usyn} \in \{\mathrm{TRUE}, \mathrm{FALSE}\}$, total interaction-budget target MT with $T \geq H$, reference threshold N_0, confidence parameter p

Initialize $Q^1, V^1, V^{\mathrm{ref},1}$, $N^1, n^1, \widetilde{n}^1$, $I^{\mathrm{ren},1}$, and all moment accumulators
Set $U \leftarrow 0$ and $k \leftarrow 1$
while $U < MT$ **do**
  Set $\pi_h^k(s) \in \arg\max_{a \in \mathcal{A}} Q_h^k(s,a)$
  Compute $c_h^k(s,a)$ from Equation (16)
  Broadcast π^k, V^k, $V^{\mathrm{ref},k}$, and the relevant stage counters
  for all agents $m \in [M]$ **do**
    Initialize local counts and the five sums in Equation (18) to zero, and set $\ell_m^k \leftarrow 0$
  end for
  if $\mathrm{usyn} = \mathrm{FALSE}$ **then**
    for all agents $m \in [M]$ in parallel **do**
      repeat
        Generate one trajectory under the fixed policy π^k and set $\ell_m^k \leftarrow \ell_m^k + 1$
        Update $C_{m,h}^k(s,a)$ and the five local sums
      until some $C_{m,h}^k(s,a) = c_h^k(s,a)$ at the end of an episode
    end for
  else
    repeat
      for all agents $m \in [M]$ in parallel **do**
        Generate one trajectory under the fixed policy π^k and set $\ell_m^k \leftarrow \ell_m^k + 1$
        Update $C_{m,h}^k(s,a)$ and the five local sums
      end for
    until some agent reaches $C_{m,h}^k(s,a) = c_h^k(s,a)$ for some (s,a,h)
  end if
  $U \leftarrow U + H \sum_{m=1}^M \ell_m^k$
  All agents send their local counts and aggregate sums to the server
  Update the global moment accumulators using Equations (19) and (20)
  Compute $\widehat{n}_h^{k+1}(s,a)$ using Equation (21)
  for each (s,a,h) **do**
    if Equation (22) holds **then**
      Set $I_h^{\mathrm{ren},k+1}(s,a) \leftarrow 1$ and update $(n, \widetilde{n}, N)$ using Equation (23)
      Compute $Q_h^{k+1,1}, Q_h^{k+1,2}$ and update $Q_h^{k+1}(s,a)$ using Equations (24) to (26)
    else
      Set $I_h^{\mathrm{ren},k+1}(s,a) \leftarrow 0$ and update $(n, \widetilde{n}, N)$ using Equation (23)
      $Q_h^{k+1}(s,a) \leftarrow Q_h^k(s,a)$
    end if
  end for
  Update V^{k+1} and π^{k+1}
  Refresh $V_h^{\mathrm{ref}}(s)$ from $V_h^{k+1}(s)$ where $\sum_a N_h^{k+1}(s,a) \geq N_0$ is reached for the first time
  $k \leftarrow k + 1$
end while

G.2.2 Application of Our Framework and Computational Cost

Let Z_k denote the server state at the beginning of communication round k, including $Q^k, V^k, V^{\mathrm{ref},k}$, $N^k, n^k, \widetilde{n}^k$, the renewal indicators, and all historical moment accumulators. For agent m, let $G_{m,j}^k$ denote the Markov flow of its first j trajectories in the round, with state-action and transition counts $C_{m,h}^{k,(j)}$ and $E_{m,h}^{k,(j)}$, respectively. Since π^k is fixed and deterministic throughout the round, conditional on Z_k, the trajectories generated by each agent follow a fixed deterministic Markov policy.

The local statistics communicated to the server are flow measurable irrespective of the synchronization mechanism. For every f in Equation (18),

$$A^k_{m,h}(s,a;f) = \sum_{s' \in \mathcal{S}} E^k_{m,h}(s,a,s') f(s'). \tag{27}$$

The corresponding local visitation count is simply $C^k_{m,h}(s,a)$. Historical contributions from earlier rounds are already stored in the accumulators contained in Z_k, while the current boundary flow contributes only the round-k contractions in Equation (27). Therefore, stages spanning multiple communication rounds remain flow measurable even though the value functions may change between rounds. Once these local statistics have been recovered, the moment updates in Equations (19) and (20), the count recursion in Equation (23), the Q-updates in Equations (24) to (26), and the subsequent value, policy, and reference updates are deterministic functions of the pre-round server state and the local aggregate messages. Moreover, all such updates occur only after local exploration terminates, so the second requirement of the boundary compatibility condition holds automatically.

For agent m, the local stopping condition is

$$g_{m,k}(Z_k, G^k_{m,j}) = \mathbb{I}\left\{\exists (s,a,h) : C^{k,(j)}_{m,h}(s,a) \geq c^k_h(s,a)\right\}. \tag{28}$$

The thresholds are determined by Z_k before local exploration begins, and all local visitation counts are nondecreasing in j, so Equation (28) is prefix monotone. At the beginning of round k, define

$$L_k = 1 + \sum_{s \in \mathcal{S}} \left(c^k_1(s, \pi^k_1(s)) - 1\right). \tag{29}$$

This candidate size is guaranteed to contain the local stopping boundary. Indeed, after L_k episodes,

$$L_k = \sum_{s \in \mathcal{S}} C^{k,(L_k)}_{m,1}(s, \pi^k_1(s)).$$

If no local threshold had been reached by then, every term on the right would satisfy

$$C^{k,(L_k)}_{m,1}(s, \pi^k_1(s)) \leq c^k_1(s, \pi^k_1(s)) - 1,$$

which would imply $L_k \leq L_k - 1$, a contradiction. Thus Algorithm 1 can be called with candidate size L_k for every complete local round.

For the computational comparison, fix a deterministic budget parameter $T \geq H$ and terminate after the first complete communication round for which the total number of environment steps across all agents reaches MT. Thus T specifies an average per-agent budget target; individual agents need not generate the same number of episodes. Every round that is started is completed according to the original local stopping rules. Let U_k denote the total number of environment steps generated by all agents before round k. A new round is started only when $U_k < MT$.

The candidate size in Equation (29) is uniformly controlled before termination. From Equation (16), for every state s,

$$c^k_1(s, \pi^k_1(s)) - 1 \leq \frac{n^k_1(s, \pi^k_1(s))}{M}.$$

Moreover, $n^k_1(s, \pi^k_1(s))$ is no larger than the cumulative number of visits to the corresponding step-1 tuple before round k. Since every previous episode contributes exactly one step-1 visit,

$$\sum_{s \in \mathcal{S}} n^k_1(s, \pi^k_1(s)) \leq \frac{U_k}{H}.$$

Consequently,

$$L_k \leq 1 + \frac{U_k}{MH} < 1 + \frac{T}{H}. \tag{30}$$

Thus each agent generates at most L_k episodes in a communication round, under either synchronization mechanism. If ΔU_k denotes the total number of environment steps generated in round k, then

$$\Delta U_k \leq MHL_k < MT + MH.$$

Let U be the total number of environment steps when the procedure terminates. The final round starts with $U_k < MT$ and ends with $U \geq MT$, so

$$MT \leq U < 2MT + MH. \tag{31}$$

Define

$$\overline{T} = 2T + H. \tag{32}$$

Then $U < M\overline{T}$, and $T \geq H$ implies $\overline{T} \leq 3T$. Therefore, conventional trajectory-by-trajectory simulation has simulator-side cost $\Theta(U) = \Theta(MT)$. Maintaining the constant number of local visitation and moment statistics along the generated transitions also costs $O(MT)$. At the end of each communication round, aggregating the local statistics across the M agents costs $O(MHS)$, while checking stage renewals and performing the corresponding Q-, value-, policy-, and reference-function updates costs $O(HSA)$.

Without forced synchronization. Suppose $\mathrm{usyn} = \mathrm{FALSE}$. Each agent terminates its local exploration independently after the first episode at which one of its local visitation counts reaches the prescribed threshold. Because π^k is deterministic, only the active action $\pi_h^k(s)$ can acquire additional local visits for each (s, h), so each local trigger evaluation costs $O(HS)$. Conditional on Z_k, the M agents interact with independent copies of the MDP, so Algorithm 1 can be applied independently to each agent to recover its exact local boundary flow. Together with the flow-measurability argument above, this shows inductively over complete communication rounds that replacing conventional local trajectory generation by adaptive Markov-flow simulation preserves the joint distribution of the federated learning process under the aggregate stopping rule above.

By Equation (30), locating a local stopping boundary requires at most $O(\log(1 + T/H))$ conditional refinements. Since the local policy is deterministic, each forward flow sample and conditional split costs $C_{\mathrm{flow}}^{\mathrm{det}} = O(HS^2)$, while the $O(HS)$ local trigger evaluation is dominated. Applying the deterministic communication-round bound of Zheng et al. (2025a) with the per-agent-scale parameter $\overline{T}$, define

$$R_{\max}(\overline{T}) = SAH^2 + 4SAH^2(\log H + 3)\log\left(1 + \frac{\overline{T}}{SAH^3}\right). \tag{33}$$

Since the realized total interaction satisfies $U < M\overline{T}$, this bounds the number of communication rounds before termination. The simulator-side cost is therefore

$$O\left(MR_{\max}(\overline{T})HS^2 \log\left(1 + \frac{T}{H}\right)\right). \tag{34}$$

The local contractions in Equation (27) require $O(MHS^2)$ arithmetic per communication round, while server-side aggregation and the subsequent Q-, value-, policy-, and reference-function updates require $O(HS(M + A))$ additional computation per round. Hence, the overall computational cost is

$$O\left(MR_{\max}(\overline{T})HS^2 \log\left(1 + \frac{T}{H}\right) + R_{\max}(\overline{T})HS(M + A)\right).$$

For comparison, conventional trajectory-by-trajectory simulation has simulator-side cost $\Theta(MT)$ and overall computational cost

$$O\left(MT + R_{\max}(\overline{T})HS(M + A)\right).$$

Substituting Equation (33) into Equation (34) and using $\overline{T} \leq 3T$, the simulator-side cost is

$$O\left(MH^3S^3A\left[1 + (\log H + 3)\log\left(1 + \frac{T}{SAH^3}\right)\right]\log\left(1 + \frac{T}{H}\right)\right).$$

The corresponding overall computational cost is

$$O\left(H^3S^2A\left[1 + (\log H + 3)\log\left(1 + \frac{T}{SAH^3}\right)\right]\left[MS\log\left(1 + \frac{T}{H}\right) + M + A\right]\right).$$

With forced synchronization. We next consider $\mathrm{usyn} = \mathrm{TRUE}$ under a synchronized episode clock. In this case, a communication round terminates at the first episode for which any agent reaches one of its local thresholds. The joint stopping condition can therefore be written as

$$g_k^{\mathrm{sync}}\left(Z_k, G_{1,j}^k, \ldots, G_{M,j}^k\right) = \mathbb{I}\left\{\exists m, s, a, h : C_{m,h}^{k,(j)}(s, a) \geq c_h^k(s, a)\right\}.$$

This trigger is again prefix monotone. Since the common policy is deterministic, evaluating it over the M agents costs $O(MHS)$ per candidate prefix by checking only the active action at each (s, h). The same L_k in Equation (29) guarantees that the joint stopping boundary occurs no later than episode L_k. Adaptive binary refinement can therefore be performed jointly across the M local flows. At each candidate prefix, the M flows are conditionally split at the same episode index; conditional independence of the agents implies that these splits recover the correct joint conditional distribution.

Define the corresponding deterministic communication-round upper bound

$$R_{\mathrm{sync,max}}(\overline{T}) = MSAH^2 + 4MSAH^2(\log H + 3)\log\left(1 + \frac{\overline{T}}{SAH^3}\right). \tag{35}$$

Each joint refinement costs $O(MC_{\mathrm{flow}}^{\mathrm{det}}) = O(MHS^2)$, which dominates the $O(MHS)$ joint trigger evaluation. By Equation (30), at most $O(\log(1 + T/H))$ refinements are needed per round. Thus, the simulator-side cost is

$$O\left(MR_{\mathrm{sync,max}}(\overline{T})HS^2 \log\left(1 + \frac{T}{H}\right)\right),$$

and the overall computational cost is

$$O\left(MR_{\mathrm{sync,max}}(\overline{T})HS^2 \log\left(1 + \frac{T}{H}\right) + R_{\mathrm{sync,max}}(\overline{T})HS(M + A)\right).$$

Substituting Equation (35) and using $\overline{T} \le 3T$, the simulator-side cost is

$$O\left(M^2H^3S^3A\left[1 + (\log H + 3)\log\left(1 + \frac{T}{SAH^3}\right)\right]\log\left(1 + \frac{T}{H}\right)\right),$$

while the overall computational cost is

$$O\left(MH^3S^2A\left[1 + (\log H + 3)\log\left(1 + \frac{T}{SAH^3}\right)\right]\left[MS\log\left(1 + \frac{T}{H}\right) + M + A\right]\right).$$

For comparison, conventional trajectory-by-trajectory simulation again has simulator-side cost $\Theta(MT)$ and overall cost $O(MT + R_{\mathrm{sync,max}}(\overline{T})HS(M + A))$. Thus, under the no-forced-synchronization mechanism and the episode-synchronized specialization of forced synchronization, adaptive Markov-flow simulation replaces the linear simulator-side dependence on the average per-agent budget target with polylogarithmic dependence while preserving the corresponding complete-round update semantics.

G.3 Implications for Other Federated Q-Learning Algorithms

The two extensions above identify complementary structures that substantially broaden the applicability of our framework. The single-agent specialization of FedQ-Bernstein shows that visit-dependent learning rates, round-wise value moments, and Bernstein confidence bonuses can all be represented through aggregate round statistics and endpoint quantities, while FedQ-Advantage shows that adaptive Markov-flow simulation extends naturally to distributed local exploration. Together, these observations suggest a general route for extending fast simulation to a broader class of federated RL algorithms that organize local interaction into communication rounds and base global updates on flow-measurable aggregate statistics.