A Dual-SPMA Framework for Convex MDPs

Shervin Khamooshian **Ahmed Magd** **Pegah Aryadoost** **Danielle Nguyen**
Simon Fraser University
{ska309, ams80, paa40, tdn8}@sfu.ca

Abstract

This project empirically studies a unified framework for solving Convex Markov Decision Processes (CMDPs) by combining two key ideas: a Fenchel Duality reformulation and the Softmax Policy Mirror Ascent (SPMA) algorithm. CMDPs generalize standard Reinforcement Learning (RL) to handle objectives with convex constraints and multi-objective trade-offs, but directly solving them is computationally hard due to their high-dimensional constrained space. Using Fenchel duality, we reformulate the CMDP into a min–max problem that alternates between a convex optimization over dual variables and a policy optimization step solvable via SPMA. The project implements this dual–SPMA framework, evaluates its convergence and efficiency, and provides empirical insights into the practicality of dual formulations for CMDPs when combined with efficient policy optimization methods.

1 Motivation

While Reinforcement Learning (RL) has demonstrated remarkable success across many domains, the standard formulation remains insufficient for capturing complex real-world objectives. Scenarios involving constraints, multi-objective trade-offs, or imitation learning often require more expressive formulations than conventional RL. The Convex Markov Decision Process (CMDP) framework extends RL to handle such cases by introducing convex objectives and constraints.

Solving CMDPs directly is difficult since it involves optimization over a high-dimensional constrained space of stationary distributions. To address this, prior work has employed Fenchel Duality to reformulate the primal CMDP minimization into a min-max optimization problem that alternates between two simpler subproblems: (i) a *dual update*, maximizing a convex function with respect to the dual variables y, and (ii) a *policy update*, solving a standard RL task with a y-shaped reward using established policy optimization methods.

To solve the policy subproblem, we adopt the Softmax Policy Mirror Ascent (SPMA) algorithm Asad et al. [2024], a principled approach with provable linear convergence for convex, smooth objectives.

While Fenchel Duality and SPMA exist, their integration in a unified empirical framework is underexplored. The empirical validation of this theoretically-promising dual decomposition remains uninvestigated, a gap this project addresses.

Our project thus aims to empirically implement and analyze this combined framework, developing a solver that is both theoretically grounded and computationally efficient.

2 Related Work

RL formulations beyond linear rewards. Typical MDP objectives can be written as maximizing a linear reward expectation over the occupancy measure Zahavy et al. [2021]:

$$\max_{d_\pi} \langle r(s,a), d_\pi(s,a) \rangle.$$

However, many practical settings are not captured by a simple inner product over state–action pairs. For example, entropy-regularized methods such as SAC Haarnoja et al. [2018] introduce diversity via an entropy-regularized objective (e.g., $f(d) = -\langle r, d\rangle - \alpha H(\pi_d)$), and constrained policy optimization Achiam et al. [2017] imposes inequality constraints (e.g., $\langle c_i, d_\pi\rangle \leq B_i$ for one or more cost functions c_i).

CMDP Reformulation via Fenchel Duality. A broad class of these objectives can be reformulated as *convex minimization problems* over the occupancy measure, $f(d_\pi)$, yielding the general *Convex MDP (CMDP)* formulation Zahavy et al. [2021]:

$$\min_{d_\pi \in \mathcal{K}} f(d_\pi).$$

The standard RL problem, $\max_{d_\pi} \langle r, d_\pi\rangle$, is just one simple instance of this, where $f(d_\pi) = \langle -r, d_\pi\rangle$. To solve the *general* problem, we use **Fenchel duality** to rewrite $f(d_\pi)$ in its conjugate form: $f(d_\pi) = \max_y\{\langle y, d_\pi\rangle - f^*(y)\}$, where y is the dual variable and f^* is the Fenchel conjugate. This transforms the original minimization into the **minimax saddle-point problem** Miryoosefi et al. [2019]:

$$\min_{d_\pi \in \mathcal{K}} \max_{y} \langle y, d_\pi\rangle - f^*(y).$$

Such saddle-point problems motivate descent–ascent procedures alternating between the minimization and maximization subproblems Lin et al. [2019], Ying et al. [2024].

Optimization roles: dual vs. policy player. For the *dual* (maximization) updates, convex first-order methods apply; when strong convexity holds, accelerated rates (e.g., Nesterov) are available Nesterov [1983]. In practice, full gradients can be costly, so SGD/Adam variants are common Kingma and Ba [2017]. For the *policy* (minimization) side, each fixed dual iterate induces a shaped-reward RL problem solvable with standard policy optimization.

Policy-gradient geometry and SPMA. Geometry-aware updates refine policy gradients via trust regions or mirror maps. TRPO enforces a KL trust region with monotonic improvement Schulman et al. [2015]; PPO relaxes this with clipping but can be hyperparameter sensitive Schulman et al. [2017], Lascu et al. [2025]. MDPO casts updates as mirror descent Tomar et al. [2020], providing a foundation for later geometry-aware methods. Building on this, SPMA operates in *logit* space with a log-sum-exp mirror map, achieving $O(\log(1/\epsilon))$ convergence and strong empirical performance, and reduces policy updates to convex softmax classification under linear function approximation Asad et al. [2024].

Summary. Prior work (i) shows many RL objectives can be expressed as convex CMDPs via Fenchel duality; (ii) motivates alternating descent–ascent between dual and policy players; and (iii) develops geometry-aware policy optimization where SPMA offers both theoretical and practical advantages. Our study integrates these strands empirically: a dual update for the convex side and SPMA for the policy side within a unified CMDP solver.

3 Problem Formulation

MDP and occupancies. We consider a discounted infinite-horizon MDP $(\mathcal{S}, \mathcal{A}, P, \rho, \gamma)$ with $\gamma \in (0, 1)$, start-state distribution ρ, and transition kernel $P(\cdot|s, a)$. For a stationary policy π, its discounted state–action occupancy measure is

$$d_\pi(s, a) \;=\; (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \Pr_\pi(s_t = s, a_t = a).$$

The feasible occupancies form a convex polytope $\mathcal{D}$ defined by standard flow constraints (full form moved for space to Appendix B). Any $d \in \mathcal{D}$ induces a policy via $\pi(a|s) = \frac{d(s,a)}{\sum_{a'} d(s,a')}$ when the denominator is nonzero.

Convex CMDP objective. Our goal is to minimize a convex function $f : \mathcal{D} \to \mathbb{R} \cup \{+\infty\}$ of the occupancy:

$$\min_{\pi} f(d_\pi) \qquad \Longleftrightarrow \qquad \min_{d \in \mathcal{D}} f(d). \tag{1}$$

Examples include constrained safety and entropy-regularized RL (details in Appendix A).

Fenchel duality and saddle formulation. Let f^* denote the Fenchel conjugate of f, $f^*(y) = \sup_{x \in \mathcal{D}} \langle y, x \rangle - f(x)$. By the Fenchel–Moreau representation,

$$f(d_\pi) = \max_{y \in \mathcal{Y}} \langle y, d_\pi \rangle - f^*(y),$$

for a suitable dual domain $\mathcal{Y}$. Thus (1) is equivalent to

$$\min_{\pi} \max_{y \in \mathcal{Y}} \mathcal{L}(\pi, y) \ := \ \langle y, d_\pi \rangle - f^*(y). \tag{2}$$

This is advantageous because for fixed y we obtain a standard RL problem, and for fixed π the y-update is a convex optimization. For fixed y, the term $\langle y, d_\pi \rangle$ corresponds to an RL problem with shaped reward ($r_y(s,a) = -y(s,a)$ in tabular form; or $r_y(s,a) = \phi(s,a)^\top y$ with linear features).

Regularity assumptions. We adopt standard conditions (communicating MDP; f proper/closed/-convex; compact $\mathcal{D}$) ensuring existence of a saddle point and well-posedness; see Appendix B for statements and implications.

First-order structure. When f^* is differentiable (else use subgradients), the dual gradient is $\nabla_y \mathcal{L}(\pi, y) = d_\pi - \nabla f^*(y)$. For a fixed y, the policy player's objective is to solve $\min_{\pi_\theta} \langle y, d_{\pi_\theta} \rangle$. This is equivalent to maximizing the standard RL objective $J_y(\pi_\theta) := \langle -y, d_{\pi_\theta} \rangle$ using the shaped reward $r_y = -y$. The gradients for this saddle-point problem are thus:

$$\nabla_y \mathcal{L}(\pi, y) = d_\pi - \nabla f^*(y), \qquad \nabla_\theta J_y(\pi_\theta) = \mathbb{E}_{\pi_\theta}\big[\nabla_\theta \log \pi_\theta(a|s)\, Q_{r_y}^{\pi_\theta}(s,a)\big].$$

Our solution framework applies mirror ascent to both players in this min-max game. For the dual (maximization) step, we apply a general convex first-order mirror ascent method over $\mathcal{Y}$. For the policy (primal) step, we maximize $J_y(\pi_\theta)$ by employing Softmax Policy Mirror Ascent (SPMA), which is a specific mirror ascent algorithm using the log-sum-exp mirror map. Estimators and implementation details are in Appendix C.

4 Plan

We follow a four-phase plan (see Appendix, Fig. 1) with milestones on Nov. 10, Dec. 2, and Dec. 15. Tasks run in parallel with clear handovers to keep theory, algorithms, and experiments aligned.

Phase 1: Parallel Foundations (Oct. 22–Nov. 10). **A** reviews literature on convex MDPs, constrained policy gradients, and related optimization; **B** defines/implements the environment and sanity-checks it with a standard RL baseline; **C** implements the SPMA-based main algorithm under the CMDP formulation; **D** builds comparative CMDP baselines (e.g., constrained PG). **Milestone (Nov. 10):** review, environment, and both algorithmic branches completed and validated.

Phase 2: Integration (Nov. 10–Nov. 17). **A** prepares slides; **B+C** integrate the environment with the main algorithm and run end-to-end tests; **B+D** repeat integration with comparative methods to check consistency.

Phase 3: Ablation & Presentation (Nov. 17–Dec. 2). **C** runs ablations over RL/optimization choices; **D** varies CMDP objectives; **A+B** finalize slides and incorporate results as C/D complete analyses. **Milestone (Dec. 2):** ablations finished and included in the presentation.

Phase 4: Reporting (Dec. 2–Dec. 15). After the Dec. 2 presentation, all members draft, refine, and finalize the report, integrating motivation, methods, and empirical analysis into a cohesive document. **Milestone (Dec. 15):** final report submitted.

Work Division & Coordination. Leads: **A** (theory/writing), **B** (environments/integration), **C** (algorithms), **D** (comparative methods). Weekly syncs maintain visibility; results are cross-validated for correctness and reproducibility.

Expected Outcomes. (i) Primary and comparative CMDP implementations; (ii) verified environment and integrations; (iii) systematic ablations across algorithms/objectives; (iv) a complete presentation and technical report summarizing theoretical and empirical insights.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 22–31. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/achiam17a.html`.

Reza Asad, Reza Babanezhad Harikandeh, Issam H. Laradji, Nicolas Le Roux, and Sharan Vaswani. Fast convergence of softmax policy mirror ascent for bandits & tabular MDPs. In *OPT 2024: Optimization for Machine Learning*, 2024. URL `https://openreview.net/forum?id=f5OjNMXIik`.

Anas Barakat, Souradip Chakraborty, Pengcheng Yu, Pratap Tokekar, and Amrit Singh Bedi. On the global optimality of policy gradient methods in general utility reinforcement learning. In *Advances in Neural Information Processing Systems*, 2024.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018. URL `http://arxiv.org/abs/1801.01290`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL `https://arxiv.org/abs/1412.6980`.

Razvan-Andrei Lascu, David Šiška, and Łukasz Szpruch. Ppo in the fisher-rao geometry, 2025. URL `https://arxiv.org/abs/2506.03757`.

Tianyi Lin, Chi Jin, and Michael I. Jordan. On gradient descent ascent for nonconvex-concave minimax problems. *CoRR*, abs/1906.00331, 2019. URL `http://arxiv.org/abs/1906.00331`.

Sobhan Miryoosefi, Kianté Brantley, Hal Daumé III, Miroslav Dudík, and Robert E. Schapire. Reinforcement learning with convex constraints. *CoRR*, abs/1906.09323, 2019. URL `http://arxiv.org/abs/1906.09323`.

Y. Nesterov. A method for solving the convex programming problem with convergence rate o(1/k2), 1983. URL `https://cir.nii.ac.jp/crid/1370862715914709505`.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL `http://arxiv.org/abs/1502.05477`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL `http://arxiv.org/abs/1707.06347`.

Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. *CoRR*, abs/2005.09814, 2020. URL `https://arxiv.org/abs/2005.09814`.

Donghao Ying, Mengzi Amy Guo, Hyunin Lee, Yuhao Ding, Javad Lavaei, and Zuo-Jun Max Shen. Policy-based primal-dual methods for concave cmdp with variance reduction, 2024. URL `https://arxiv.org/abs/2205.10715`.

Tom Zahavy, Brendan O'Donoghue, Guillaume Desjardins, and Satinder Singh. Reward is enough for convex mdps. *Advances in Neural Information Processing Systems*, 34:25746–25759, 2021.

Appendix

A Convex Objective Examples

A.1 Constrained Safety

$f(d) = -\langle r, d\rangle + \lambda \max\{0, \langle c, d\rangle - \tau\}$, where c encodes costs (e.g., collision risk) and τ is a safety threshold. This formulation allows reward maximization while ensuring safety constraints are satisfied.

A.2 Entropy-Regularized RL

$f(d) = -\langle r, d\rangle - \alpha \cdot H(\pi_d)$, where $H(\pi_d) = -\sum_{s,a} d(s,a) \log \frac{d(s,a)}{\sum_{a'} d(s,a')}$ is the entropy of the policy induced by d, and $\alpha > 0$ controls exploration. This maximum-entropy framework encourages diverse behavior while maximizing reward and has been successfully applied in modern deep RL methods such as soft actor-critic.

B Regularity Assumptions and Their Implications

This appendix explains the regularity assumptions from Section 3 and their role in ensuring well-posedness and convergence of our convex MDP framework.

B.1 Overview of Assumptions

We assume:

1. The MDP is communicating.
2. The objective function f is proper, closed, and convex on $\mathcal{D}$
3. For discounted problems with $\gamma \in (0, 1)$, the feasible set $\mathcal{D}$ is nonempty and compact.

B.2 Communicating MDP

Definition. An MDP is communicating if every state can reach every other state under some policy: for all $(s, s') \in \mathcal{S} \times \mathcal{S}$, there exists policy π and finite T with $\Pr_\pi(s_T = s' | s_0 = s) > 0$.

Necessity and Consequences. *With* communicating assumption we obtain well-defined occupancy measures throughout the state space, meaningful policy gradients everywhere, and no dead-end states that trap policies; *without* it, consider a dead-end state s_{trap} with $P(s_{\text{trap}}|s_{\text{trap}}, a) = 1$ for all a where once entered, all occupancy mass concentrates there ($\lim_{t\to\infty} d_\pi(s_{\text{trap}}, \cdot) \to 1$), causing ill-defined policy gradients for other states, SPMA failures due to zero probabilities, and unbounded dual variables.

B.3 Properties of Objective Function f

B.3.1 Proper

Definition. Function f is proper if $f(d) > -\infty$ for all $d \in \mathcal{D}$ and $f(d) < +\infty$ for at least one d.

Necessity and Consequences. *With* properness the minimization problem is bounded below, at least one feasible solution exists, and the Fenchel conjugate $f^*(y) = \sup_{d\in\mathcal{D}} \langle y, d\rangle - f(d)$ is well-defined; *without* it, if $f(d) = -\infty$ for some d, optimization is unbounded below, if $f(d) = +\infty$ for all d, the problem is infeasible, and in either case the Fenchel conjugate becomes undefined.

B.3.2 Closed

Definition. Function f is closed (lower semicontinuous) if its epigraph epi$(f) = \{(d, \alpha) : f(d) \leq \alpha\}$ is closed, or equivalently, $f(d) \leq \liminf_{n\to\infty} f(d_n)$ for any sequence $\{d_n\} \to d$.

Necessity and Consequences. *With* closedness minimizers exist (infimum is attained), limit points of converging sequences remain feasible, and the Fenchel-Moreau duality $f = f^{**}$ holds, ensuring exact duality with zero gap; *without* it, the infimum may not be attained (optimal solution doesn't exist), converging sequences may have strictly better values than their limits, and Fenchel duality may have a gap, making our saddle formulation inexact.

B.3.3 Convex

Definition. Function f is convex if $f(\lambda d_1 + (1-\lambda)d_2) \leq \lambda f(d_1) + (1-\lambda)f(d_2)$ for all $d_1, d_2 \in \mathcal{D}$ and $\lambda \in [0, 1]$.

Necessity and Consequences. *With* convexity we obtain exact Fenchel duality (zero gap, so solving the dual solves the primal), any local minimum is global, subgradients exist everywhere, and the dual problem over y is convex with convergence guarantees; *without* it, a duality gap emerges ($\min_\pi \max_y \mathcal{L} \neq \max_y \min_\pi \mathcal{L}$) so our algorithm optimizes the wrong objective, multiple local minima trap optimization, SPMA's convergence guarantees fail, and saddle points may not exist.

Verification for Our Objectives. Our examples are convex:

- **Constrained safety**: $f(d) = -\langle r, d\rangle + \lambda \max\{0, \langle c, d\rangle - \tau\}$ is a sum of linear (convex) and composition of max (convex) with linear (convex), hence convex.
- **Entropy-regularized**: $f(d) = -\langle r, d\rangle - \alpha H(\pi_d)$ is a sum of linear and negative entropy (convex), hence convex.

B.4 Nonemptiness and Compactness of $\mathcal{D}$

Nonemptiness. For any MDP with $\gamma \in (0, 1)$, nonemptiness is automatic: any policy π induces a valid occupancy $d_\pi \in \mathcal{D}$. Without nonemptiness, the problem is vacuous.

Compactness. For discounted MDPs, $\mathcal{D}$ is compact: it is *bounded* because $\sum_{s,a} d_\pi(s, a) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t = 1$, so d_π lies in the unit simplex. It is *closed* because the flow constraints are linear equations, making $\mathcal{D}$ the intersection of the positive orthant (closed) with an affine subspace (closed).

Necessity and Consequences. *With* compactness saddle points exist (Minimax Theorem), minimizers exist (Extreme Value Theorem), iterates $\{(\pi_k, y_k)\}$ remain bounded with convergent subsequences, and dual variables stay bounded; *without* it, there is no saddle point guarantee, the infimum may not be attained, iterates may diverge to infinity, and convergence proofs break down.

B.5 Additional Conditions for Fast Convergence

Strong Convexity of f^*. If f^* is μ-strongly convex ($f^*(\lambda y_1 + (1-\lambda)y_2) \leq \lambda f^*(y_1) + (1-\lambda)f^*(y_2) - \frac{\mu}{2}\lambda(1-\lambda)\|y_1 - y_2\|^2$), then gradient methods achieve *linear* convergence $O((1-\mu/L)^k)$ instead of sublinear $O(1/k)$, with better conditioning and stable updates; if f^* lacks strong convexity, add regularization $f_\epsilon(d) = f(d) + \frac{\epsilon}{2}\|d\|^2$ to make f^*_ϵ strongly convex.

B.6 Summary

Together, these assumptions ensure: (1) the problem is well-posed with a unique solution, (2) Fenchel duality is exact with zero gap, (3) our alternating optimization converges to the global optimum, and (4) convergence is fast (linear rate under strong convexity). Without them, the algorithm may fail to converge, converge to suboptimal solutions, or optimize an incorrect objective.

C Practical Estimation of Occupancy Measures

While the problem formulation defines the occupancy measure $d_\pi(s, a)$ mathematically, implementing our algorithm requires *practical methods* to estimate d_π from simulation data. This appendix details

Assumption	With	Without
Communicating MDP	Well-defined occupancy; meaningful gradients	Degenerate distributions; trapped states
f proper	Bounded problem; feasible solution exists	Unbounded or infeasible
f closed	Minimizer exists; exact duality	Infimum not attained; duality gap
f convex	Exact duality; global optimality; convergence	Duality gap; local minima; wrong objective
$\mathcal{D}$ compact	Saddle point exists; bounded iterates	No saddle point; diverging iterates
f^* strongly convex	Linear convergence; stable updates	Slow sublinear convergence

Table 1: Summary of regularity assumptions and their implications.

two complementary approaches: one for tabular settings and one for function approximation. We also provide a detailed variance analysis and discuss connections to recent theoretical work.

C.1 Why We Need to Estimate d_π

The dual gradient from Section 3 requires the occupancy measure:

$$\nabla_y \mathcal{L}(\pi, y) = d_\pi - \nabla f^*(y).$$

While d_π has a formal definition as an infinite sum, we must estimate it from finite simulation data. The estimation method depends on the problem scale:

- **Tabular settings**: Direct Monte Carlo estimation (Section C.2)
- **Large/continuous spaces**: Linearized function approximation (Section C.3)

C.2 Tabular Case: Direct Occupancy Estimation

Setting. When the state-action space is small (e.g., fewer than 10,000 pairs) and discrete, we can explicitly track visits to each (s, a) pair.

Estimator. A Monte Carlo estimate of the discounted occupancy measure is obtained by running N trajectories (rollouts) and averaging the discounted visits:

$$\hat{d}_\pi(s,a) = \frac{1-\gamma}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i} \gamma^t \mathbf{1}[s_t^{(i)} = s, a_t^{(i)} = a],$$

where T_i is the length of the i-th trajectory, and $\mathbf{1}[\cdot]$ is the indicator function.

Theoretical Justification. This is an unbiased Monte Carlo estimator: $\mathbb{E}[\hat{d}_\pi(s,a)] = d_\pi(s,a)$. By the Law of Large Numbers, as $N \to \infty$, $\hat{d}_\pi(s,a) \to d_\pi(s,a)$.

Use in Algorithm. In the outer loop, after running SPMA to update the policy π:

1. Estimate $\hat{d}_\pi$ using the procedure above.
2. Compute the dual gradient: $\nabla_y \mathcal{L} = \hat{d}_\pi - \nabla f^*(y)$
3. Update $y \leftarrow y + \eta \cdot \nabla_y \mathcal{L}$ (or a mirror ascent step).

C.3 Function Approximation: Linearized Dual Variables

Setting. When the state-action space is large or continuous, we cannot enumerate all (s, a) pairs.

Dual-Variable Parameterization. Following the approach in Zahavy et al. [2021], Barakat et al. [2024], we parameterize the dual variable as:

$$y(s,a) = \phi(s,a)^\top w, \tag{3}$$

where $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ is a feature map and $w \in \mathbb{R}^d$ is the learnable parameter.

Key Identity: Linearization with Respect to w. The inner product $\langle y, d_\pi \rangle$ can be rewritten using the discounted return identity:

$$\langle y, d_\pi \rangle = (1-\gamma)\,\mathbb{E}_\tau \left[\sum_{t \geq 0} \gamma^t y(s_t, a_t) \right] \tag{4}$$

where $\tau \sim \pi$ denotes a trajectory sampled under policy π. After plugging in $y(s,a) = \phi(s,a)^\top w$:

$$\langle y, d_\pi \rangle = \langle w, \mu_\pi \rangle, \quad \text{where} \quad \mu_\pi := \mathbb{E}_{d_\pi}[\phi(s,a)]. \tag{5}$$

Key Insight (Linearization). Instead of estimating $d_\pi(s,a)$ for all (s,a) pairs (scaling as $|\mathcal{S}| \times |\mathcal{A}|$), we only need to estimate the feature expectation vector $\mu_\pi = \mathbb{E}_{d_\pi}[\phi] \in \mathbb{R}^d$.

Feature Expectation Estimator. We estimate μ_π by Monte Carlo:

$$\hat{\mu}_\pi = (1-\gamma) \cdot \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i} \gamma^t \phi(s_t^{(i)}, a_t^{(i)}). \tag{6}$$

Equivalently, defining the discounted feature sum for a single trajectory as $G_\phi := (1-\gamma) \sum_t \gamma^t \phi(s_t, a_t) \in \mathbb{R}^d$, then $\hat{\mu}_\pi = \frac{1}{N} \sum_{i=1}^{N} G_\phi^{(i)}$.

Dual Gradient in Function Approximation. The gradient of the Lagrangian with respect to w is:

$$\nabla_w \mathcal{L}(\pi, w) = \mu_\pi - \nabla f^*(w), \tag{7}$$

with μ_π replaced by the Monte Carlo estimator $\hat{\mu}_\pi$.

C.4 Variance Analysis: Dependence on Feature Dimension d

A key concern is that the estimation variance of our Monte Carlo estimator **grows with the feature dimension** d.

Setup. Suppose each feature coordinate is bounded: $|\phi_j(s,a)| \leq 1$ for all $j \in \{1, \ldots, d\}$. For one trajectory, define the discounted feature sum $G_\phi := (1-\gamma) \sum_t \gamma^t \phi(s_t, a_t) \in \mathbb{R}^d$. Then $\hat{\mu}_\pi$ is the empirical mean of G_ϕ across N rollouts.

Variance Bound. By standard concentration inequalities, the mean-squared error is:

$$\boxed{\mathbb{E}\left[\|\hat{\mu}_\pi - \mu_\pi\|_2^2 \right] = O\left(\frac{d}{N} \right)} \tag{8}$$

This means the vector gradient noise $\widehat{\nabla_w \mathcal{L}} - \nabla_w \mathcal{L} = \hat{\mu}_\pi - \mu_\pi$ has typical size $\|\hat{\mu}_\pi - \mu_\pi\|_2 \sim \sqrt{d/N}$.

Implications for Sample Complexity: variance explodes with feature dimension d. To keep the gradient noise at a constant level as d grows, we need $N = \Theta(d)$ trajectories per outer iteration.

Remark 1 (Scalar vs. Vector Estimation) *If we only cared about the* ***scalar*** *quantity $\langle w, \mu_\pi \rangle$ with $\|w\|$ bounded, variance could stay independent of d. However, for updating the whole vector w, we care about the full d-dimensional error, which scales with d.*

C.5 Connection to Qishi's "Convex MDPs" Note

The "Convex MDPs" note provided by Qishi is a compact version of the derivation in Section C.3.

Dual Parameterization. The note parameterizes the dual variable as $\lambda_\theta(s,a) = \langle \phi(s,a), \theta \rangle$, identical to our $y(s,a) = \phi(s,a)^\top w$.

FTRL-Style Dual Update. Using the discounted occupancy identity, the note rewrites the dual update as:

$$\lambda^k = \arg\max_{\theta \in \mathbb{R}^d} \left\{ (1-\gamma) \sum_{j=1}^{k} \mathbb{E}_{\pi^j} \left[\sum_t \gamma^t \phi(s_t, a_t) \right] \cdot \theta - k \cdot f^*(\langle \phi, \theta \rangle) \right\}, \tag{9}$$

and proposes to estimate those expectations by sampling from each policy π^j.

Structural Agreement. Both our Appendix C and Qishi's note say: "We need discounted feature expectations, so we estimate them by Monte Carlo." The variance issue (Section C.4) is implicit in that note but not analyzed there—our analysis makes it explicit.

C.6 What the Barakat et al. Paper Adds

The Barakat et al. "On the Global Optimality of Policy Gradient Methods" paper Barakat et al. [2024] addresses occupancy estimation in large spaces. Two key contributions are relevant.

(a) Low-Rank / Linear Structure of Occupancies. In Appendix C of Barakat et al. [2024], they show that in a **low-rank MDP** there exist "density features" $\mu(s) \in \mathbb{R}^d$ such that **every** occupancy measure lies in their span:

$$d_\pi(s) = \rho(s) + \langle \omega_\pi, \mu(s) \rangle \tag{10}$$

for some vector $\omega_\pi \in \mathbb{R}^d$. This is similar to our dual parameterization:

- **Our approach**: Dual is linear in *state-action* features: $y(s,a) = \phi(s,a)^\top w$
- **Their result**: Occupancy is affine in *state* density features: $d_\pi(s) = \rho(s) + \mu(s)^\top \omega_\pi$

Both say: *a huge object (function over $\mathcal{S} \times \mathcal{A}$ or $\mathcal{S}$) really lives in a d-dimensional subspace.*

(b) MLE-Based Occupancy Estimation with Error $\propto \sqrt{d/n}$. Instead of coordinate-wise Monte Carlo means, Barakat et al. treat the occupancy measure λ_π as a **parametric density** $p_\omega(x)$ (e.g., $p_\omega(x) = g(\omega^\top \phi(x))$) and fit ω via **maximum likelihood estimation (MLE)**.

Under mild assumptions, they prove (Proposition 1 in Barakat et al. [2024]):

$$\boxed{\|\hat{\lambda}_\pi - \lambda_\pi\|_1 \lesssim \sqrt{\frac{m \log n}{n}}} \tag{11}$$

with high probability, where m is the parameter dimension. The sample complexity to achieve TV error $\leq \varepsilon$ is $n = \tilde{O}(m/\varepsilon^2)$, **independent of** $|\mathcal{S}||\mathcal{A}|$.

Why Regression / MSE Fails in Large Spaces. Barakat et al. argue that MSE-style estimators can have errors that scale with the size of the space. In a toy example, MSE can be tiny while total variation distance is huge—this discrepancy scales with space size. This is exactly the "blow up" our Monte Carlo gradient suffers from.

C.7 Summary: Current Implementation and What to tweak

Our Current FA Estimator.

(a) We estimate the discounted feature expectation $\mu_\pi = \mathbb{E}_{d_\pi}[\phi(s,a)]$ via Monte Carlo (Equation 6).

(b) The dual gradient in FA is $\nabla_w \mathcal{L} = \mu_\pi - \nabla f^*(w)$ (Equation 7).

(c) Because $\mu_\pi \in \mathbb{R}^d$, variance scales like $O(d/N)$ in squared norm (Equation 8).

(d) **To keep gradient noise fixed, we need $N = \Theta(d)$ trajectories per iteration.**

Connection to Theoretical Work.

(a) **Qishi's note**: Uses the same linear FA and discounted-feature-identity. Our Appendix C expands this and gives a concrete Monte Carlo estimator.

(b) **Barakat et al.**: Shows occupancy measures have linear structure in low-rank MDPs. Uses MLE instead of MC means, getting TV error $O(\sqrt{d/n})$ independent of $|\mathcal{S}||\mathcal{A}|$.

To update in our current implementation.

(i) **Assume a low-rank MDP structure** and parameterize the occupancy directly as in Equation 10.

(ii) **Replace the MC feature-mean estimator** with an MLE-based occupancy estimator whose error scales as $O(\sqrt{d/n})$ in TV distance.

C.8 Complexity Comparison

Aspect	Tabular MC	FA (Ours, MC)	FA (MLE, Barakat)
What is estimated	$d_\pi(s,a)\ \forall(s,a)$	$\mu_\pi \in \mathbb{R}^d$	$\omega_\pi \in \mathbb{R}^m$
Storage	$O(\lvert\mathcal{S}\rvert \cdot \lvert\mathcal{A}\rvert)$	$O(d)$	$O(m)$
Estimation error	$O\left(\sqrt{\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{N}}\right)$	$O\left(\sqrt{\frac{d}{N}}\right)$	$O\left(\sqrt{\frac{m}{n}}\right)$
Error metric	ℓ_2	ℓ_2	TV (ℓ_1)
Depends on $\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert$?	Yes	No	No

Table 2: Complexity comparison across estimation methods.

Aspect	Original (Gradient Ascent)	Updated (FTRL)
Update style	$w \leftarrow w + \alpha\nabla_w\mathcal{L}$	$w = \text{cumsum}/(k \cdot \lambda)$
Feature expectations	Current: $\mathbb{E}_{\pi^k}[\phi]$	Cumulative: $\sum_{j=1}^{k} \mathbb{E}_{\pi^j}[\phi]$
Step-size	Requires tuning α	Implicit via f^* regularization
Theory connection	Gradient-based analysis	OCO regret $\rightarrow O(1/\sqrt{K})$
Memory	$O(d)$	$O(d)$

Table 3: Comparison of gradient ascent and FTRL dual update schemes.

C.9 Implementation Details

Feature Map Design. For continuous control tasks (e.g., Pendulum-v1), we use polynomial features:

$$\phi(s,a) = [s, a, s^2, a^2]^\top \in \mathbb{R}^d,$$

where $d = \dim(s) + \dim(a) + \dim(s) + \dim(a)$. For Pendulum-v1 specifically: $\dim(s) = 3$, $\dim(a) = 1$, so $d = 8$.

Listing 1: Updated FeatureEstimator class for FTRL with variance tracking

Updated `FeatureEstimatorFTRL` **Class.**

```python
class FeatureEstimatorFTRL:
    """
    Cumulative discounted feature expectations for FTRL updates.
    Tracks variance for diagnostics.
    """
    def __init__(self, phi_fn, d: int, gamma: float):
        self.phi_fn = phi_fn
        self.d = d
        self.gamma = gamma
        self.cumulative_sum = np.zeros(d, dtype=np.float64)
        self.iteration = 0
        # For variance tracking
        self.sum_of_squares = np.zeros(d, dtype=np.float64)

    def update_from_batch(self, obs, acts, dones):
        """Add this iteration's feature expectations to cumulative sum
            ."""
        gpow = 1.0
        episode_phi_sums = []
        current_sum = np.zeros(self.d, dtype=np.float64)

        for t in range(len(acts)):
            phi = self.phi_fn(obs[t], acts[t])
            current_sum += (1.0 - self.gamma) * gpow * phi
            gpow *= self.gamma
            if dones[t] > 0.5:
                episode_phi_sums.append(current_sum.copy())
                current_sum = np.zeros(self.d)
                gpow = 1.0

        if len(episode_phi_sums) > 0:
            ephi_hat = np.mean(episode_phi_sums, axis=0)
            self.cumulative_sum += ephi_hat
            for g in episode_phi_sums:
                self.sum_of_squares += g**2

        self.iteration += 1
        return ephi_hat if len(episode_phi_sums) > 0 else np.zeros(
            self.d)

    def get_ftrl_weight(self, lam: float):
        """Return FTRL solution: w = cumsum / (k * lam)."""
        if self.iteration == 0:
            return np.zeros(self.d)
        return self.cumulative_sum / (self.iteration * lam)

    def estimate_variance(self, N_total_episodes: int):
        """Estimate Var[G_phi], which scales as O(d/N)."""
        if N_total_episodes == 0:
            return np.zeros(self.d)
        mean_sq = self.sum_of_squares / N_total_episodes
        mean = self.cumulative_sum / self.iteration
        var_per_coord = mean_sq - mean**2
        total_var = np.sum(var_per_coord)
        return total_var, var_per_coord
```

D Timeline (Supplemental)

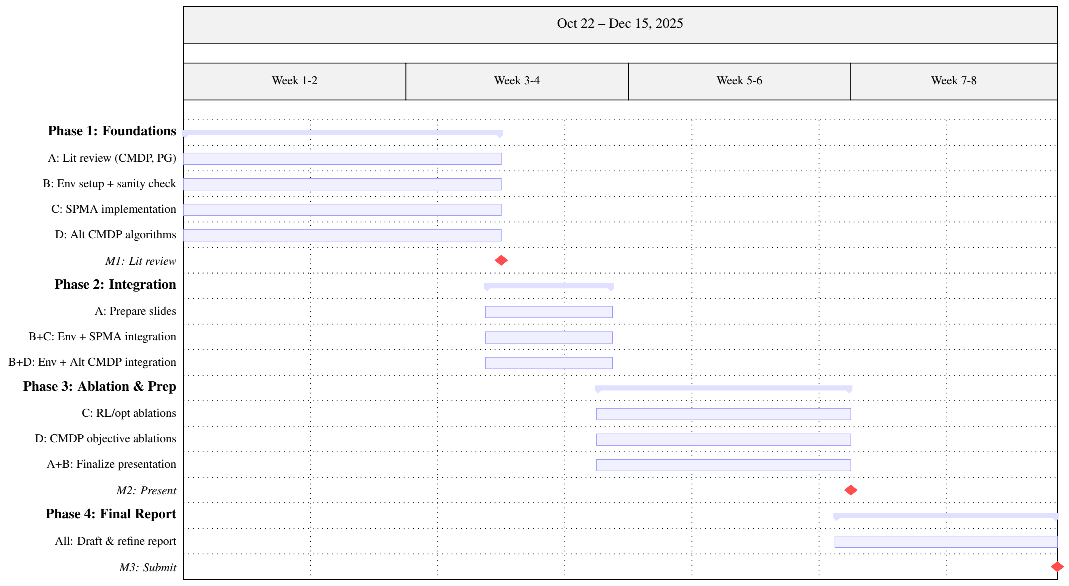

Figure 1: Project timeline with four phases and team coordination (A, B, C, D).