Project Notes

Softmax Policy Mirror Ascent (SPMA) & Convex MDPs

CMPT 409 F2025 – Dual SPMA Project

December 14, 2025

Contents

1 Softmax Policy Mirror Ascent (SPMA)

1.1 Objective Formulation

The SPMA algorithm operates in the z-space (logit space) rather than directly in the policy space. The objective is:

SPMA Objective

$$\max_{z \in \mathcal{Z}} J(z) = \max_{z \in \mathcal{Z}} J(h(z)) = \max_{\pi} J(\pi)$$

where the last term is the **original objective in RL**.

The mapping from z-space to policy space is given by the softmax:

$$h(z(s, \cdot)) := \frac{e^{z(s,a)}}{\sum_{a'} e^{z(s,a')}} = \text{softmax}(z)$$

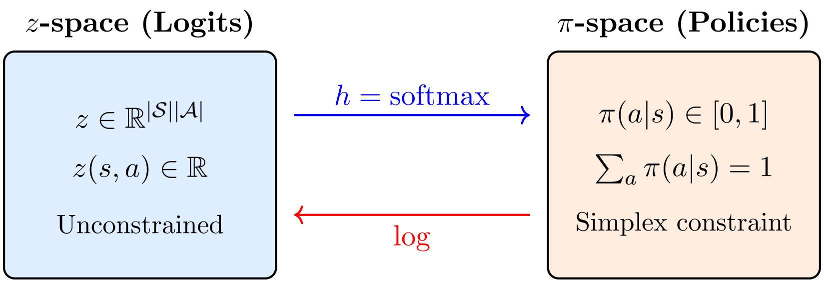


Figure 1: Relationship between z-space (logits) and π-space (policies)

1.2 Mirror Map and Bregman Divergence

1.2.1 Mirror Map Definition

The mirror map $\phi : \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} \to \mathbb{R}$ is defined as:

Weighted Log-Sum-Exp Mirror Map

$$\phi(z) := \sum_s d_\pi(s) \ln \left(\sum_a \exp(z(s,a)) \right) \in \mathbb{R}$$

where:

- $z \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ is the full logit vector
- $z(s, \cdot) \in \mathbb{R}^{|\mathcal{A}|}$ is the logit vector for state s
- $z(s,a) \in \mathbb{R}$ is the logit for state-action pair (s,a)
- $\phi : \mathbb{R}^{|\mathcal{S}||\mathcal{A}|} \to \mathbb{R}$ maps high-dimensional logits to a scalar

1.2.2 Gradient of the Mirror Map

The gradient of the mirror map is:

$$\nabla\phi(z)(s,a) = d_\pi(s)\frac{\exp(z(s,a))}{\sum_{a'}\exp(z(s,a'))} = d_\pi(s)\,\pi(a|s)$$

1.2.3 Bregman Divergence

The Bregman divergence induced by ϕ is:

$$D_\phi(z,z') := \phi(z) - \phi(z') - \langle\nabla\phi(z'), z - z'\rangle$$

This can be expressed in terms of KL divergence:

Key Connection

$$D_\phi(z,z') = \sum_s d_\pi(s)\,\mathrm{KL}\left(\pi_{z'}(\cdot|s)\,\|\,\pi_z(\cdot|s)\right)$$

1.3 Mirror Ascent Update

The update rule operates between the π-space and the z-space. The following diagram illustrates the mirror ascent process:

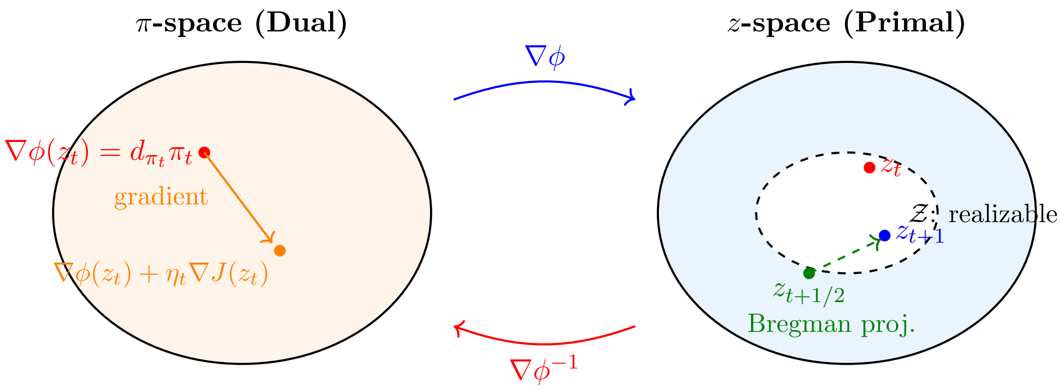


Figure 2: Mirror Ascent: The algorithm alternates between dual space (policy) updates and primal space (logit) projections. The Bregman projection ensures we stay in the realizable set $\mathcal{Z}$.

1.3.1 Dual Space Update

In the dual (π) space:

$$\nabla\phi(z_{t+1/2}) = \nabla\phi(z_t) + \eta_t\nabla J(z_t)$$

This implies:

$$d_{\pi_t}(s)\,\pi_{t+1/2}(a|s) = d_{\pi_t}(s)\,\pi_t(a|s) + \eta_t\,d_{\pi_t}(s)\,\pi_t(a|s)\,A_{\pi_t}(s,a)$$

Simplifying:

Policy Update Rule

$$\pi_{t+1/2}(a|s) = \pi_t(a|s)\,(1 + \eta_t\, A_{\pi_t}(s,a))$$

1.4 Projection to Realizable Set $\mathcal{Z}$

Since $z(s,a) := f_\theta(s,a)$ is parameterized, we need to project back to the realizable set:

$$z_{t+1} = \arg\min_{z\in\mathcal{Z}} D_\phi(z_{t+1/2}\,|\,z)$$

This is equivalent to:

$$\theta_{t+1} = \arg\min_\theta \sum_s d_\pi(s)\,\mathrm{KL}\left(\pi_{t+1/2}(\cdot|s)\,\|\,\pi_\theta(\cdot|s)\right)$$

Remark 1.1. $\tilde{\ell}_t$ is the **ideal surrogate** loss. In practice, we replace it with ℓ_t, which is a sampled approximation.

1.5 Practical Implementation

1.5.1 Unbiased Approximation

The practical update uses samples from d_π:

$$\theta_{t+1} \approx \arg\min_\theta \frac{1}{N}\sum_{s\sim d_\pi} \mathrm{KL}\left(\pi_{t+1/2}(\cdot|s)\,\|\,\pi_\theta(\cdot|s)\right)$$

The constant $\frac{1}{N}$ can be ignored in the $\arg\min$.

1.5.2 KL Divergence and Cross-Entropy

Recall that:

$$\mathrm{KL}(P\|Q) = \sum_i P(i)\log\frac{P(i)}{Q(i)} = \sum_i P(i)\,(\log P(i) - \log Q(i))$$

For our setting:

$$\ell_t = \sum_{s,a} \pi_{t+1/2}(a|s)\left(\log \pi_{t+1/2}(a|s) - \log \pi_\theta(a|s)\right)$$

Since the first term is constant in θ, we can ignore it in the $\arg\min$:

$$\ell_t = -\sum_{s,a}\pi_{t+1/2}(a|s)\log\pi_\theta(a|s) = H(\pi_{t+1/2},\pi_\theta) \quad \text{(cross-entropy)}$$

Final Update Rule

$$\theta_{t+1} = \arg\min_\theta H(\pi_{t+1/2},\pi_\theta)$$

1.6 Critical Issues in SPMA

Problem 1.1 (Expectation over All States)**.** The loss function requires expectation over all states, weighted by the true discounted occupancy measure d_π.

Problem 1.2 (Advantage Function Mismatch)**.**

Critical Issue in SPMA

The update requires:

$$\pi_{t+1/2} = \pi_t \left(1 + \eta_t \, A^{\pi_t}(s, \cdot)\right)$$

However, in practice we don't have $A^{\pi_t}(s, \cdot)$ for all actions—we only have $A^{\pi_t}(s, a)$ for the sampled action a.

Note: This may or may not be an issue—the estimator might still be unbiased.

2 Convex Markov Decision Processes (CMDP)

2.1 Problem Formulation

CMDP Objective

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

For the **entropy objective**:

$$f(d_\pi) = d_\pi \cdot \log(d_\pi)$$

2.2 Fenchel Conjugate

Definition 2.1 (Fenchel Conjugate)**.** The Fenchel conjugate of f is defined as:

$$f^*(\lambda) := \sup_y \{\lambda \cdot y - f(y)\}$$

where y is a **dummy variable**.

2.2.1 Properties of f^*

1. $f^*(\lambda)$ is always **convex** (regardless of whether f is convex)
2. If f is convex, then $f^{**} = f$ (the biconjugate equals the original function)

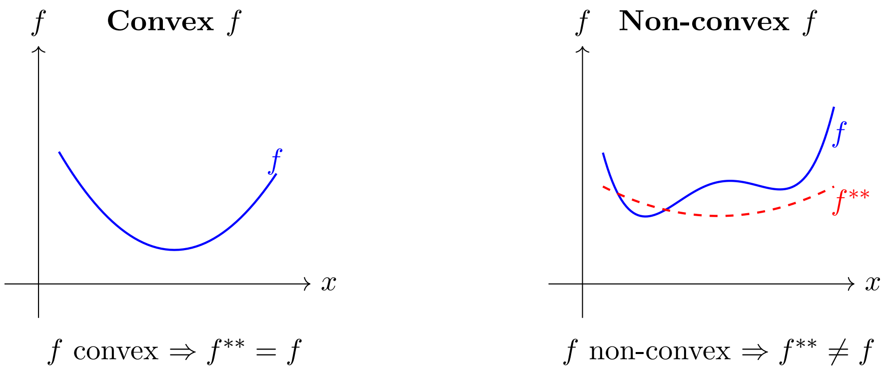


Figure 3: For convex functions, the biconjugate f^{**} equals f. For non-convex functions, f^{**} is the convex envelope (largest convex function below f).

2.3 Reformulation via Fenchel Conjugate

Using the Fenchel conjugate, we can write:

$$f^{**}(x) = \sup_\lambda \{x \cdot \lambda - f^*(\lambda)\} = f(x) \quad \text{for convex } f$$

Therefore:

$$f(d_\pi) = \sup_\lambda \{d_\pi \cdot \lambda - f^*(\lambda)\}$$

CMDP Objective (Saddle-Point Form)

$$\min_{d_\pi \in \mathcal{K}} f(d_\pi) = \min_{d_\pi \in \mathcal{K}} \max_{\lambda} \underbrace{\left[d_\pi \cdot \lambda - f^*(\lambda)\right]}_{=:L(d_\pi,\lambda)}$$

3 Meta-Algorithm for CMDP

3.1 Algorithm Overview

The meta-algorithm maintains:

- A sequence of policies: $\pi_1, \ldots, \pi_K$
- Their corresponding occupancy measures: $d^1, \ldots, d^K$
- Cost vectors: $\lambda_1, \ldots, \lambda_K$

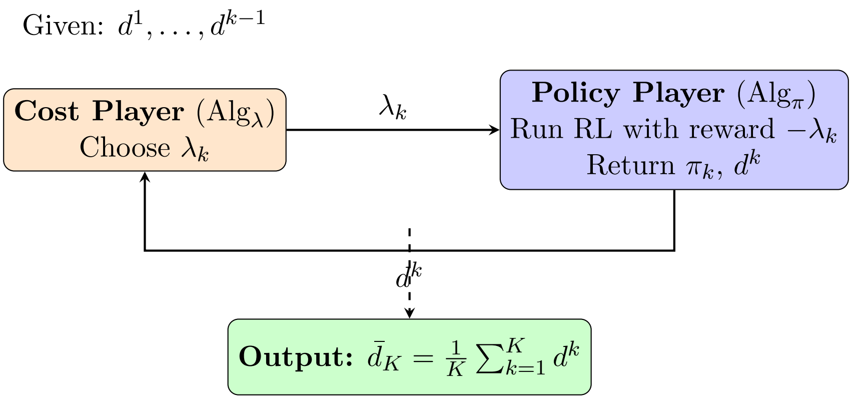


Figure 4: Meta-algorithm game between Cost Player and Policy Player

3.2 Algorithm Steps

CMDP Meta-Algorithm

1. **Initialize:** $\lambda_1 = 0$ or random value in Λ
2. **For** $k = 1, \ldots, K$:
   - **Cost Player** (Alg_λ): Given past $d^1, \ldots, d^{k-1}$, choose λ_k
   - **Policy Player** (Alg_π): Given cost λ_k, run an RL algorithm on reward $-\lambda_k$ and return π_k, d^k
3. **Output:** Average occupancies $\bar{d}_K = \frac{1}{K}\sum_{k=1}^{K} d^k$

3.3 Main Theorem

Theorem 3.1 (Convergence Guarantee)**.** If both players are low-regret online learners (in the OCO sense), then $\bar{d}_K$ converges to a solution of the convex MDP:

$$f(\bar{d}_K) - f_{\text{opt}} \leq O\left(\frac{1}{\sqrt{K}}\right)$$

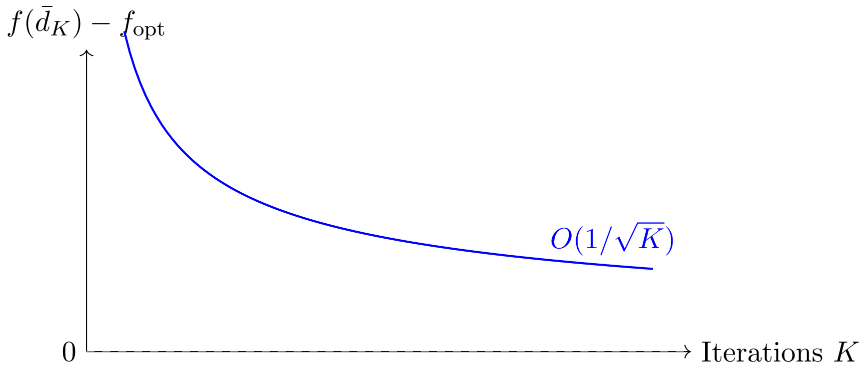


Figure 5: Convergence rate of the meta-algorithm

4 Cost Player: Follow The Leader (FTL)

4.1 FTL Update Rule

The cost player uses Follow The Leader:

FTL: Best Choice in Hindsight

$$\lambda_k = \arg\max_{\lambda} \sum_{j=1}^{k-1} L(d^j, \lambda)$$

Why Use FTL? (Important for Final Report)

Key advantages of Follow The Leader:

- FTL simplifies the problem by **eliminating the need to compute the Fenchel conjugate** $f^*(\lambda)$ directly
- Instead, we only need to compute $\nabla f(\bar{d}_{k-1})$, which is the gradient of the **original function**

Note: Do **NOT** mention Online Mirror Descent in the final report—we did not implement it. We only use FTL for the cost player.

4.2 Derivation of Optimal λ_k

Expanding the FTL objective:

$$\lambda_k = \arg\max_{\lambda} \sum_{j=1}^{k-1} \left[d^j \cdot \lambda - f^*(\lambda)\right] \tag{1}$$

$$= \arg\max_{\lambda} \left[\lambda \sum_{j=1}^{k-1} d^j - (k-1) f^*(\lambda)\right] \tag{2}$$

Defining the average occupancy $\bar{d}_{k-1} = \frac{1}{k-1}\sum_{j=1}^{k-1} d^j$:

$$\lambda_k = \arg\max_{\lambda} \left[\lambda(k-1)\bar{d}_{k-1} - (k-1)f^*(\lambda)\right] \tag{3}$$

$$= \arg\max_{\lambda}(k-1)\left[\lambda\bar{d}_{k-1} - f^*(\lambda)\right] \tag{4}$$

$$= \arg\max_{\lambda}\left[\lambda\bar{d}_{k-1} - f^*(\lambda)\right] \quad \text{(since } (k-1) > 0\text{)} \tag{5}$$

The last expression is exactly the **Fenchel conjugate**:

$$\sup_{\lambda}\left[\lambda \cdot d - f^*(\lambda)\right]$$

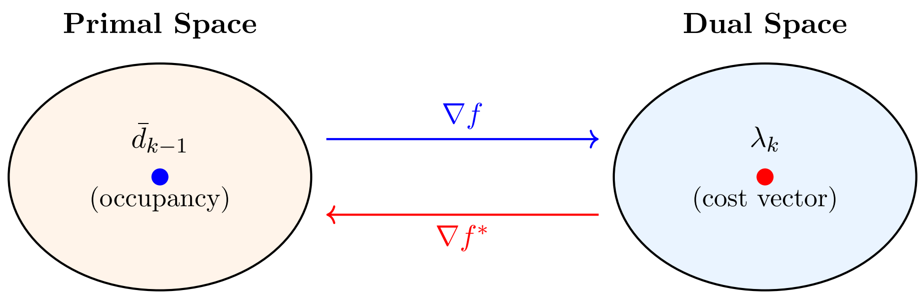
4.3 Finding Optimal λ

Setting the gradient to zero:

$$\nabla_\lambda\left[\lambda\bar{d}_{k-1} - f^*(\lambda)\right] = 0$$

$$\Rightarrow \quad \bar{d}_{k-1} - \nabla f^*(\hat{\lambda}) = 0$$

$$\Rightarrow \quad \nabla f^*(\hat{\lambda}) = \bar{d}_{k-1}$$



$$\boxed{\lambda_k = \nabla f(\bar{d}_{k-1}) \text{ and } (\nabla f^*)^{-1} = \nabla f}$$

Figure 6: Primal-Dual relationship: The gradient of f maps primal to dual, and vice versa.

Using the property that $(\nabla f^*)^{-1} = \nabla f$ (primal-dual relationship):

Optimal λ_k (Key Result)

$$\lambda_k = (\nabla f^*)^{-1}(\bar{d}_{k-1}) = \nabla f(\bar{d}_{k-1})$$

This connects the dual variable λ to the primal gradient!

5 Policy Player and Entropy Objective

5.1 Policy Player

The policy player (Alg_π) runs standard RL with reward $-\lambda_k$.

Why RL Works for the Policy Player

When minimizing w.r.t. d_π, we treat λ as a **constant**. Looking at our objective:

$$L(d_\pi, \lambda) = d_\pi \cdot \lambda - f^*(\lambda)$$

Since $f^*(\lambda)$ is constant w.r.t. d_π, it **does not affect the minimization** and can be ignored:

$$\min_{d_\pi} L(d_\pi, \lambda) = \min_{d_\pi} [d_\pi \cdot \lambda]$$

This is equivalent to an RL problem with **reward** $= -\lambda$ (negative because we minimize):

$$\min_{d_\pi} \langle d_\pi, \lambda\rangle \quad \Longleftrightarrow \quad \max_{d_\pi} \langle d_\pi, -\lambda\rangle$$

Key insight: The Policy Player just runs standard RL with cost vector λ_k as the (negative) reward!

5.2 Entropy-Based Objective Function

The objective function with entropy regularization is:

Objective Function

$$f(d_\pi) = -\beta\langle d_\pi, r\rangle + (1-\beta) \underbrace{d_\pi \log(d_\pi)}_{=H(d_\pi)\ \text{(entropy)}}$$

5.3 Gradient of the Objective

Gradient for Computing λ_k

$$\nabla f(d_\pi) = -\beta r + (1-\beta)\left(\log(d_\pi) + 1\right)$$

This gradient is used for computing the optimal λ_k via:

$$\lambda_k = \nabla f(\bar{d}_{k-1})$$

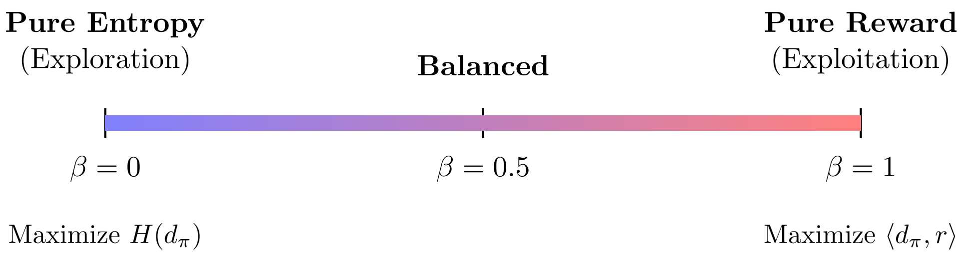


Figure 7: The β parameter controls the exploration-exploitation trade-off.

5.4 Role of β Parameter

Exploration vs. Exploitation Trade-off

- $\beta = 1$: Pure reward maximization (exploitation)

$$f(d_\pi) = -\langle d_\pi, r \rangle$$

- $\beta = 0$: Pure entropy maximization (exploration)

$$f(d_\pi) = d_\pi \log(d_\pi)$$

- $0 < \beta < 1$: Balanced objective combining both goals

5.5 Expected Behavior Validation

Experimental Validation (For Discussion Section)

The implementation validates the theory by observing:

- When $\beta = 1$: Return **increases** $\rightarrow$ agent maximizes reward (exploitation works)
- When $\beta = 0$: Entropy **increases** $\rightarrow$ agent explores the environment
- When β increases from 0 to 1: Entropy **decreases** gradually

This confirms that the Convex MDP formulation with entropy objective behaves as theoretically expected!

6 Practical Limitations & Discussion Points

6.1 Occupancy Measure Estimation

The algorithm requires computing/estimating the occupancy measure d_π:

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

Practical Limitation: d_π Estimation

Current approach: Run the environment many times and estimate d_π based on visited state-action pairs.
Limitations:

- Theoretically requires **infinite** time steps for exact d_π
- Works reasonably well in **small discrete environments** (e.g., 8×8 Frozen Lake)
- Becomes **very difficult** in larger environments
- **Impossible** in continuous state/action spaces

6.2 Future Work Directions

1. **Scalable d_π estimation:** Develop methods that don't require exhaustive environment sampling
2. **Continuous settings:** Extend the framework to continuous state-action spaces
3. **Function approximation:** Use neural networks to approximate d_π instead of tabular methods

6.3 Report Writing Notes

Important Notes for Final Report

- **Main focus:** Convex MDP formulation and FTL algorithm
- **SPMA:** Keep it brief—literature review level is sufficient; it's not the main contribution
- **Do NOT mention:** Online Mirror Descent (not implemented)
- **Do mention:** FTL simplifies the problem by eliminating Fenchel conjugate computation
- **Discussion should include:**
  - Expected behavior validation (β experiments)
  - Limitations of d_π estimation
  - Future work on continuous settings

7 Summary of Key Equations

Component	**Equation**
SPMA Objective	$\max_{z \in \mathcal{Z}} J(z) = \max_{\pi} J(\pi)$
Mirror Map	$\phi(z) = \sum_s d_\pi(s) \ln \left(\sum_a e^{z(s,a)} \right)$
Policy Update	$\pi_{t+1/2}(a\|s) = \pi_t(a\|s)\,(1 + \eta_t A_{\pi_t}(s,a))$
Projection	$\theta_{t+1} = \arg\min_\theta H(\pi_{t+1/2}, \pi_\theta)$
CMDP Objective	$\min_{d_\pi \in \mathcal{K}} \max_\lambda \left[d_\pi \cdot \lambda - f^*(\lambda)\right]$
FTL Update	$\lambda_k = \nabla f(\bar{d}_{k-1})$
Entropy Objective	$f(d_\pi) = -\beta\langle d_\pi, r\rangle + (1-\beta) d_\pi \log(d_\pi)$
Gradient	$\nabla f(d_\pi) = -\beta r + (1-\beta)(\log(d_\pi) + 1)$

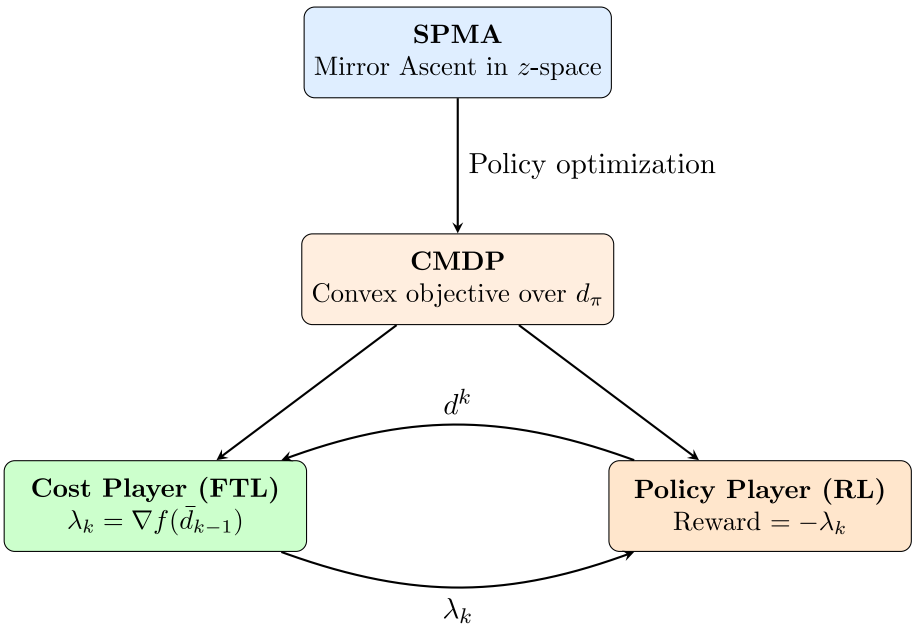


Figure 8: Overall algorithm structure showing how SPMA, CMDP, and the two-player game connect.

End of Project Notes