Learning in Stackelberg Games with Non-Myopic Agents


Nika Haghtalab
UC Berkeley


Thodoris Lykouris
MIT


Sloan Nietert
Cornell


Alex Wei
UC Berkeley

Principal-Agent Problems

- **Principal** commits to action; **agent** best responds
- Principal aims to find an optimal commitment
- Captures many settings of **economic design**

Defender vs. **attacker**
(Stackelberg security games)

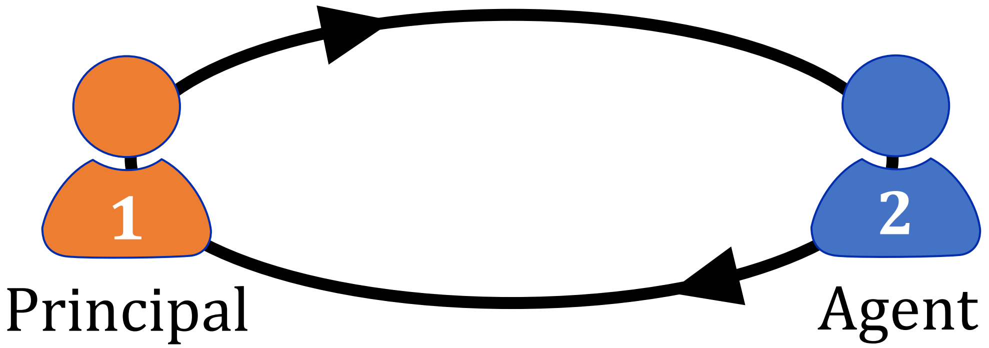
Seller vs. **buyer**
(Demand learning)

Decision-maker vs. **applicant**
(Strategic classification)

Principal-Agent Learning

But agent preferences are typically unknown!

⇒ Principal must **learn** and **adapt** from repeated interactions



Typical assumption: **myopic agent** who always best responds

⇒ Stationary learning problem with (semi-)bandit feedback

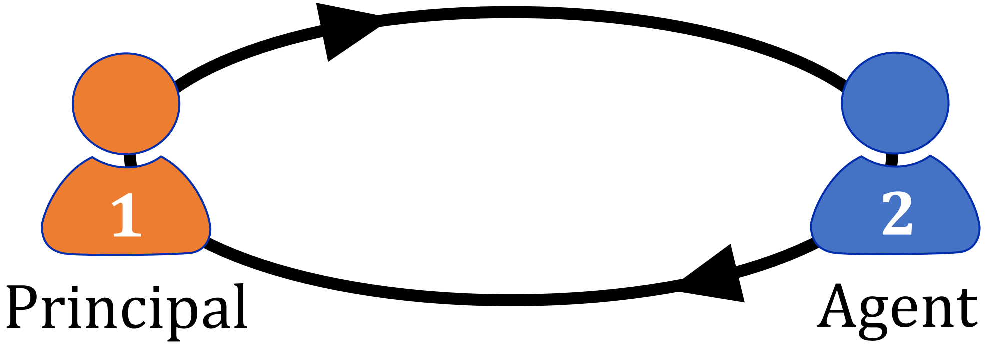
- Security games (Letchford et al., '09; Blum et al., '14), demand learning (Kleinberg-Leighton, '03; Besbes-Zeevi, '09), strategic classification (Dong et al., '18; Chen et al., '20)

Challenge: Non-Myopic Agents

What happens with **non-myopic** agents?

- Agent actions optimize long-run payoff
- Agents may not best respond—and may have incentive to mislead!

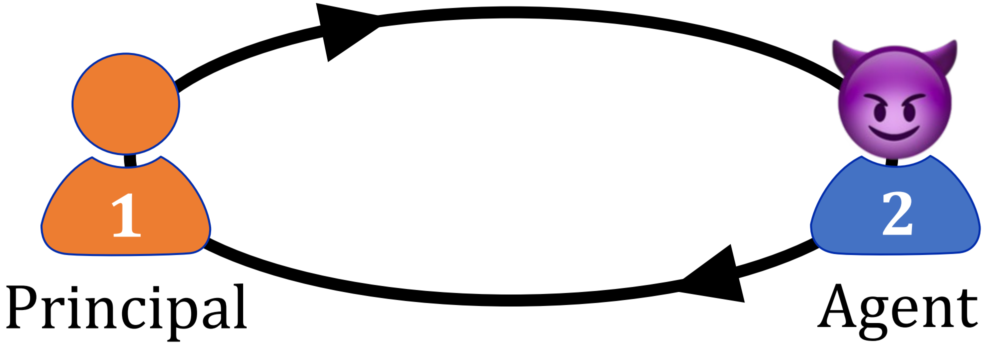
  (E.g., agent could mimic another type)



Challenge: Non-Myopic Agents

What happens with **non-myopic** agents?

- Agent actions optimize long-run payoff
- Agents may not best respond—and may have incentive to mislead!
  (E.g., agent could mimic another type)

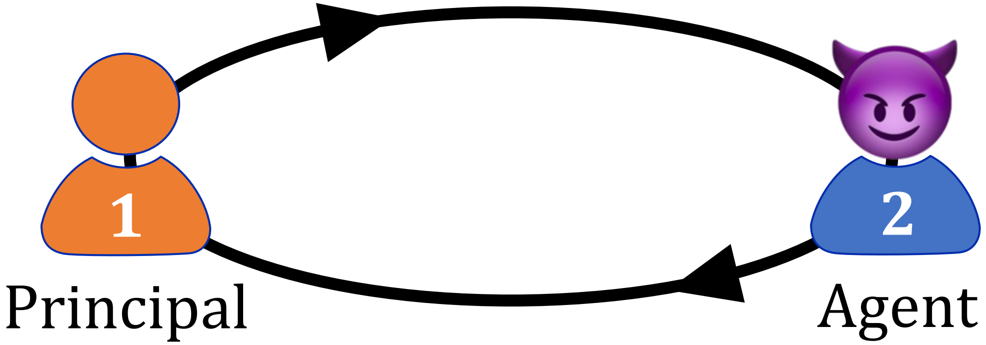


Challenge: Non-Myopic Agents

What happens with **non-myopic** agents?

- Agent actions optimize long-run payoff
- Agents may not best respond—and may have incentive to mislead!

  (E.g., agent could mimic another type)



Common model of non-myopic agents:

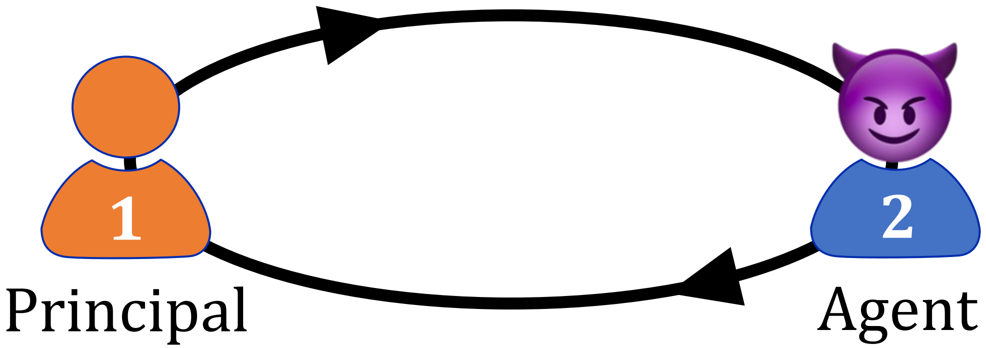
- Not arbitrarily patient (shorter-lived than principal; present bias)
- Receive discounted utility

$\Rightarrow$ Do not deviate from best response as much (or for as long)

Challenge: Non-Myopic Agents

What happens with **non-myopic** agents?

- Agent actions optimize long-run payoff
- Agents may not best respond—and may have incentive to mislead!
  (E.g., agent could mimic another type)



Questions

1. What are **principled ways** to learn from non-myopic agents?
2. How do insights for learning from myopic agents apply?

Our Contributions

Reduction framework: Non-myopic to myopic learning with:

- Minimal reactivity: Incentivize low deviation from best response
  - Achieved by delaying principal reaction to agent behavior
- Robustness: Learn effectively from approximate best responses

Our Contributions

Reduction framework: Non-myopic to myopic learning with:

- Minimal reactivity: Incentivize low deviation from best response
  - Achieved by delaying principal reaction to agent behavior
- Robustness: Learn effectively from approximate best responses

Principled reduction via delayed feedback and batched queries
(Almost) optimal algorithm for batched stochastic bandits

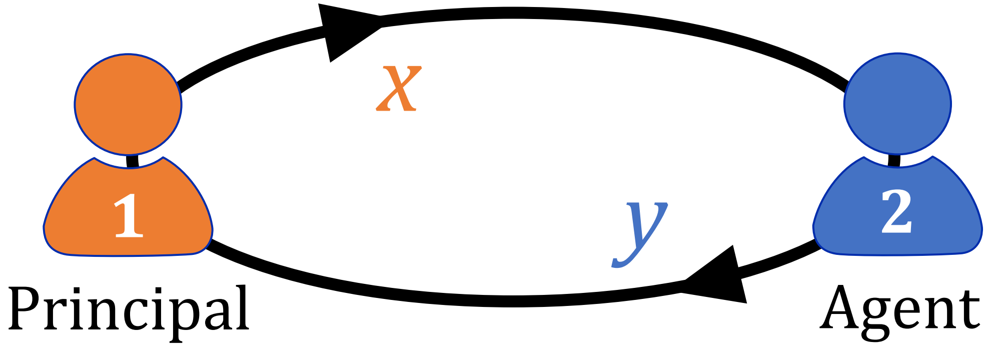
(Almost) optimal myopic learning for Stackelberg security games

Model

Principal-Agent Learning

Repeat over T rounds:

1. **Principal** commits to action $x \in \mathcal{X}$
2. **Agent** responds with action $y \in \mathcal{Y}$
3. Both parties observe x and y
4. Principal, agent receive payoffs $u(x, y)$, $v(x, y)$, respectively



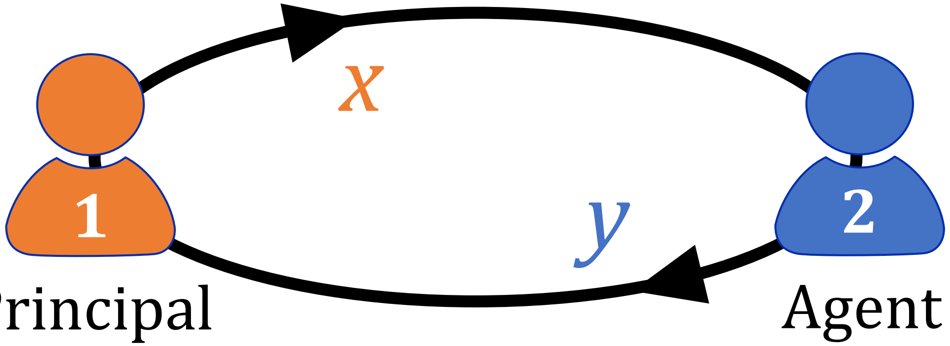
Agent behavior: Choose actions to maximize γ-discounted payoff

$$\mathbb{E}\left[\sum_t \gamma^t \cdot v(x_t, y_t) \mid \text{principal policy}\right].$$

Principal-Agent Learning

Repeat over T rounds:

1. **Principal** commits to action $x \in \mathcal{X}$
2. **Agent** responds with action $y \in \mathcal{Y}$
3. Both parties observe x and y
4. Principal, agent receive payoffs $u(x, y)$, $v(x, y)$, respectively



Principal goal: Minimize Stackelberg regret for **unknown** agent v:

Agent's best response

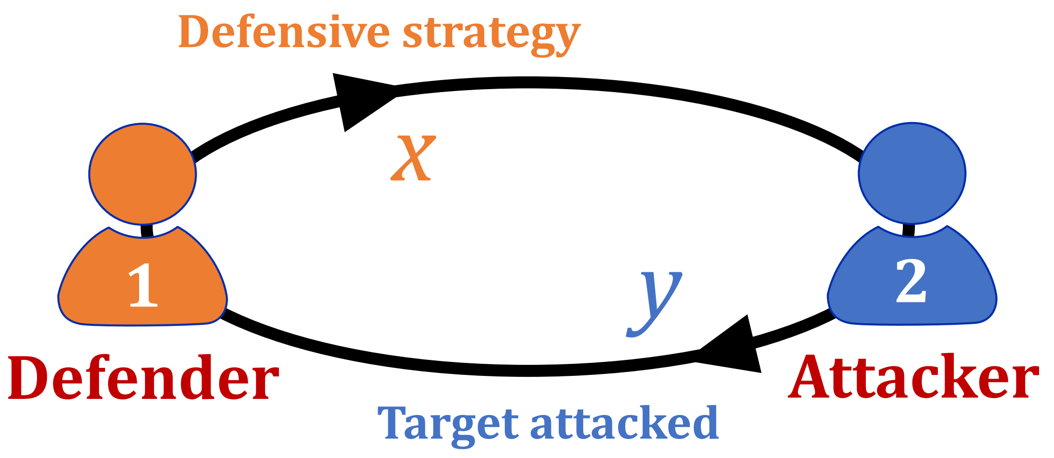
$$\underbrace{\max_{x \in \mathcal{X}} u(x, \mathrm{BR}(x))}_{\text{Stackelberg payoff}} - \underbrace{\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} u(x_t, y_t)\right]}_{\text{Algorithm payoff}}$$

Example: Stackelberg Security Games

Repeat over T rounds:

1. **Principal** commits to action $x \in \mathcal{X}$
2. **Agent** responds with action $y \in \mathcal{Y}$
3. Both parties observe x and y
4. Principal, agent receive payoffs $\underbrace{u(x, y), v(x, y)}$, respectively

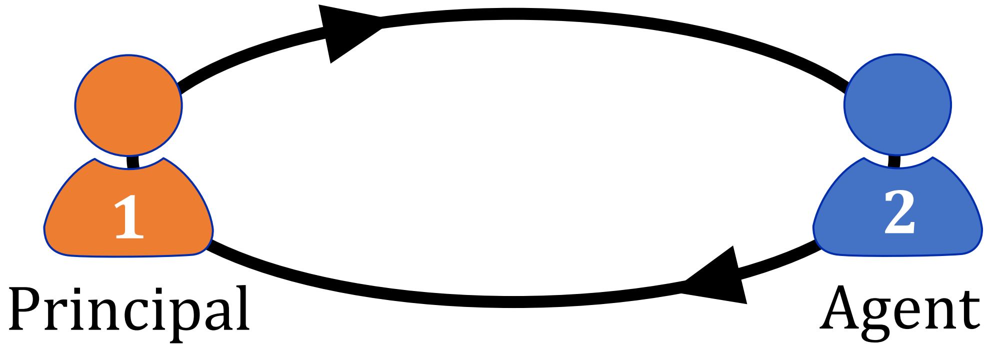
Expected payoff; depends only on attacked target and whether it was defended



Reduction

Non-Myopic Learning via Delayed Feedback

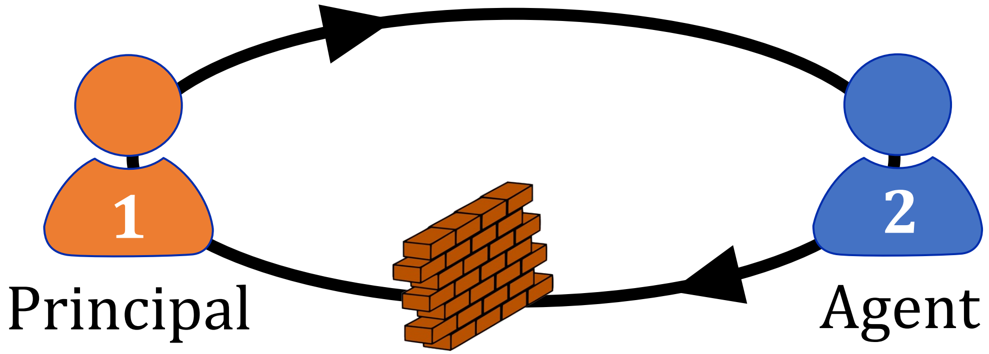
Policy is ***D*-delayed** if it only uses feedback from ≥ D rounds ago



Non-Myopic Learning via Delayed Feedback

Policy is ***D*-delayed** if it only uses feedback from ≥ D rounds ago

- Delay acts as information barrier!

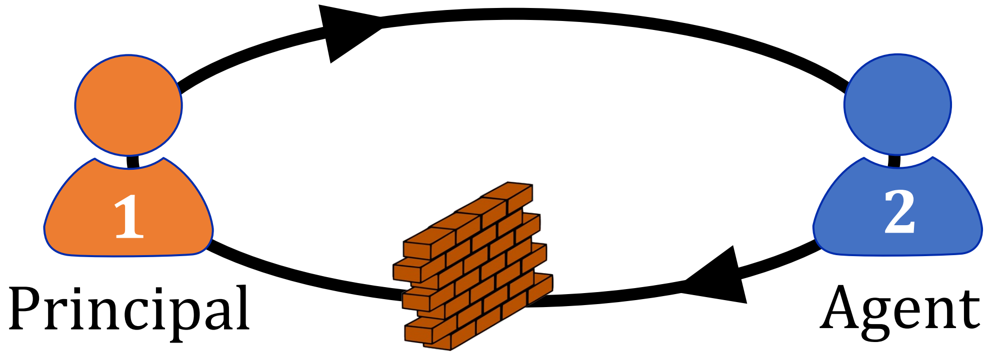


Non-Myopic Learning via Delayed Feedback

Policy is ***D*-delayed** if it only uses feedback from $\geq D$ rounds ago

- Delay acts as information barrier!

Principal delaying $\Rightarrow$ **lower reactivity**
$\Rightarrow$ less incentive for agent deviation



Proposition. If policy is D-delayed, then an γ-discounting agent will play an ε-approximate best response for $\varepsilon = \gamma^D/(1-\gamma)$:

$$v(x, y) \geq \max_{y' \in \mathcal{Y}} v(x, y') - \varepsilon$$

Non-Myopic Learning via Delayed Feedback

We reduce non-myopic learning to algorithm design desiderata:

1. **Robust** (to ε-approximate best response) bandit learning
2. Efficient bandit learning with ***D*-delayed feedback**

Minimize Stackelberg regret $\max_{x \in \mathcal{X}} u(x, \mathrm{BR}(x)) - \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} u(x_t, y_t)\right]$

x_t a function of $(x_1, y_1), \ldots, (x_{t-D}, y_{t-D})$

y_t is an ε-approximate best response

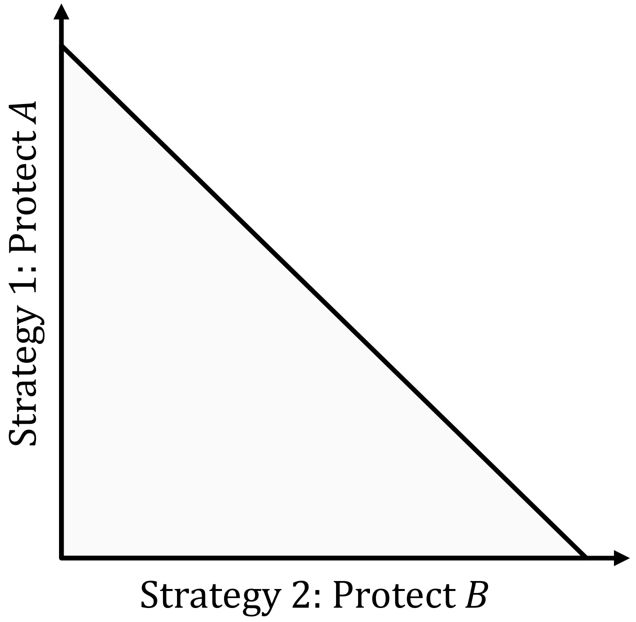
Robust Bandit Learning

(applied to Stackelberg security games)

Stackelberg security games

Targets = $\{A, B\}$

Defenses $\mathcal{X}$ = distributions over $\{\emptyset, A, B\}$



Stackelberg security games

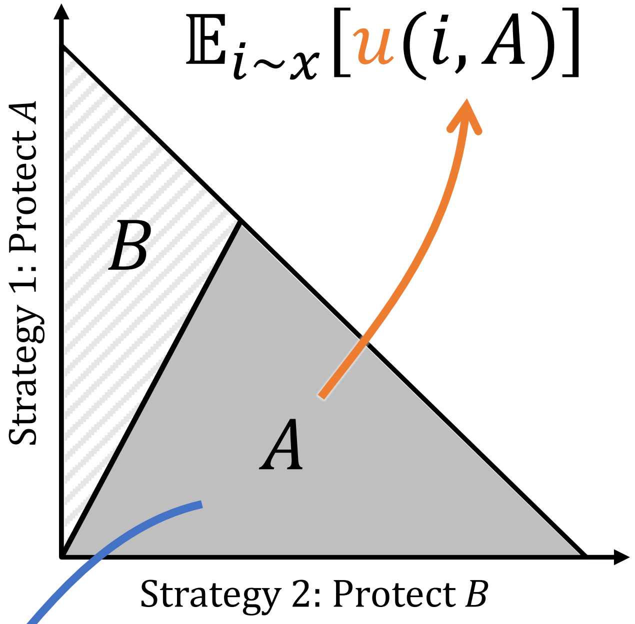
Targets = $\{A, B\}$

Defenses $\mathcal{X}$ = distributions over $\{\emptyset, A, B\}$

- Payoffs given by attacked target and whether it was defended

Defender objective:

$$\mathbb{E}_{i\sim x}[u(i, A)]$$

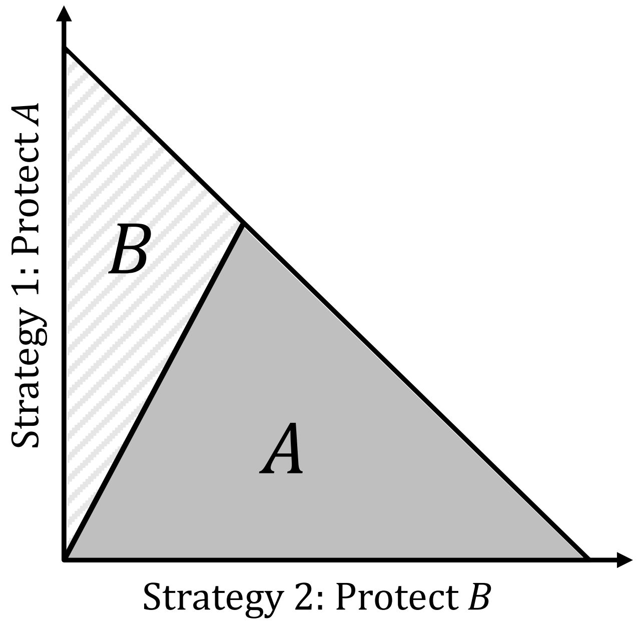


Agent choice:

$$\mathbb{E}_{i\sim x}[v(i, B)] \le \mathbb{E}_{i\sim x}[v(i, A)]$$

Robust (Myopic) Learning in Security Games

Identify **optimal algorithm** CLINCH for learning in security games



Traditionally:

- Run learning subroutine for each of n regions
- For each region, run generic (costly) learning algo

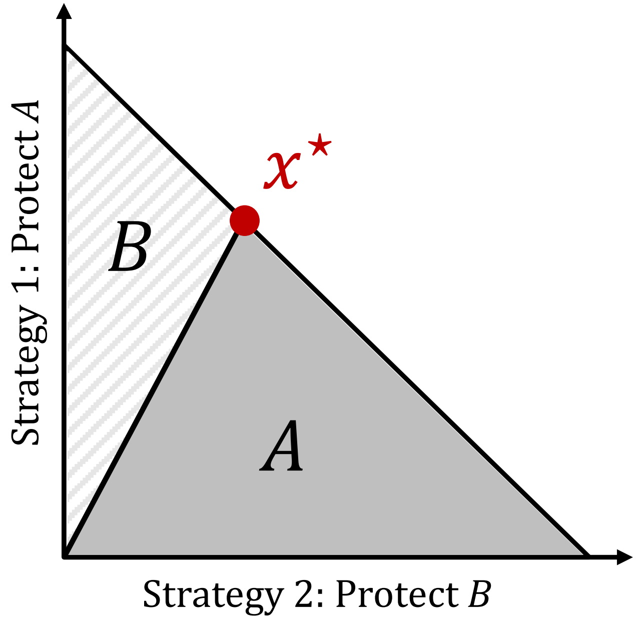
Our approach:

- Identify **structural properties** (⇒ solving single region suffices!)
- Apply **"cutting-plane"**-type optimization (Grünbaum's theorem)

Key Structural Property of Security Games

Principal's optimal action $x^\star \in \mathcal{X}$ satisfies:

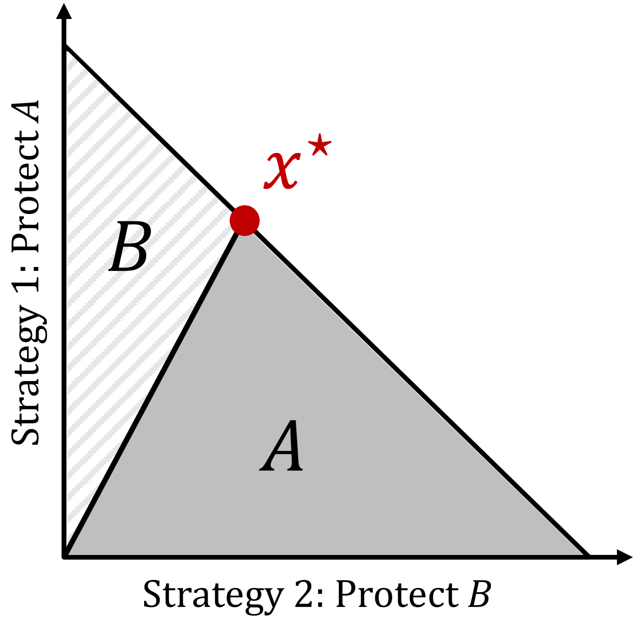
- Agent **indifferent** between all protected targets
- $x^\star$ also **minimizes agent's payoff**
- $x^\star$ is the intersection of **all non-empty regions**



Key Structural Property of Security Games

Principal's optimal action $x^\star \in \mathcal{X}$ satisfies:

- Agent **indifferent** between all protected targets
- $x^\star$ also **minimizes agent's payoff**
- $x^\star$ is the intersection of **all non-empty regions**



Robustness: feedback always points "towards" $x^\star$

Theorem. CLINCH finds a δ-approx. equilibrium with $O(N \log(N/\delta))$ queries to an (approx.) best response oracle

Bandits with Delayed Feedback

(applied to multi-armed bandits)

Efficient Bandit Learning with Delayed Feedback

Naïve solution: Repeat each action D times $\Rightarrow$ up to D times regret

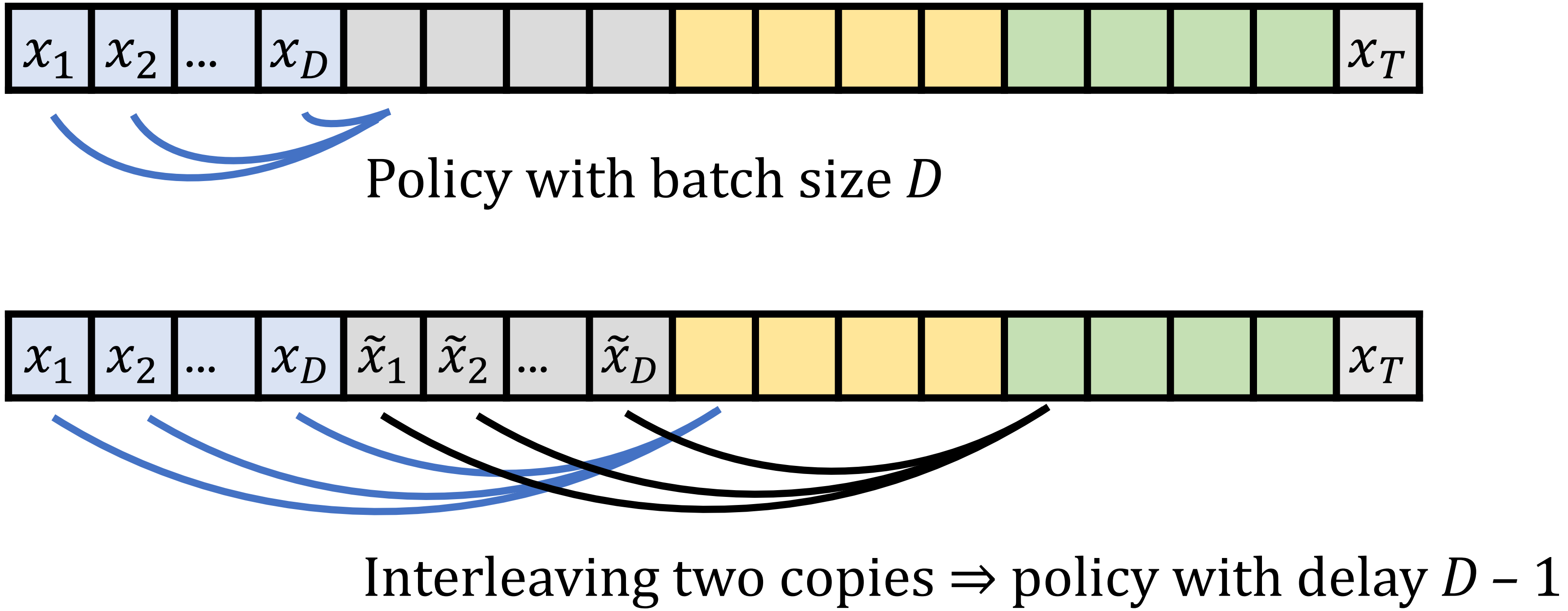
- Always works, but often suboptimal

Better: Show bandits with delays **equivalent to batched bandits**:

- Batch size B = submit B queries simultaneously
- Dependent delay sequences $\Rightarrow$ independent batches
- Well-studied setting (and thus a nice target for reduction)

Proposition. Algorithm with delay D and regret $R \Rightarrow$ algorithm with batch size D and regret $O(R)$, and vice versa.

Efficient Bandit Learning with Delayed Feedback



Proposition. Algorithm with delay D and regret $R \Rightarrow$ algorithm with batch size D and regret $O(R)$, and vice versa.

Stochastic Multi-Armed Bandits

Naïve B-batched regret bound: $B \cdot \log(T/B) \sum_i \frac{1}{\Delta_i}$

Idea: Exploration of ActiveArmElimination **parallelizable**

- Allows for parallelization within each batch
- Fewer "wasted" queries

Stochastic Multi-Armed Bandits

Naïve B-batched regret bound: $B \cdot \log(T/B) \sum_i \frac{1}{\Delta_i}$

Idea: Exploration of ActiveArmElimination **parallelizable**

- Allows for parallelization within each batch
- Fewer "wasted" queries

B-batched ActiveArmElimination (ours): $B \log K + \log(T) \sum_i \frac{1}{\Delta_i}$

$\Rightarrow$ For demand learning, handles much more patient agents

Summary

Reduction framework: Non-myopic to myopic learning with:

- Minimal reactivity: Incentivize low deviation from best response
  - Achieved by delaying principal reaction to agent behavior
- Robustness: Learn effectively from approximate best responses

Principled reduction via delayed feedback and batched queries
(Almost) optimal algorithm for batched stochastic bandits

(Almost) optimal myopic learning for Stackelberg security games