Pareto-Optimal Algorithms for Learning in Repeated Games

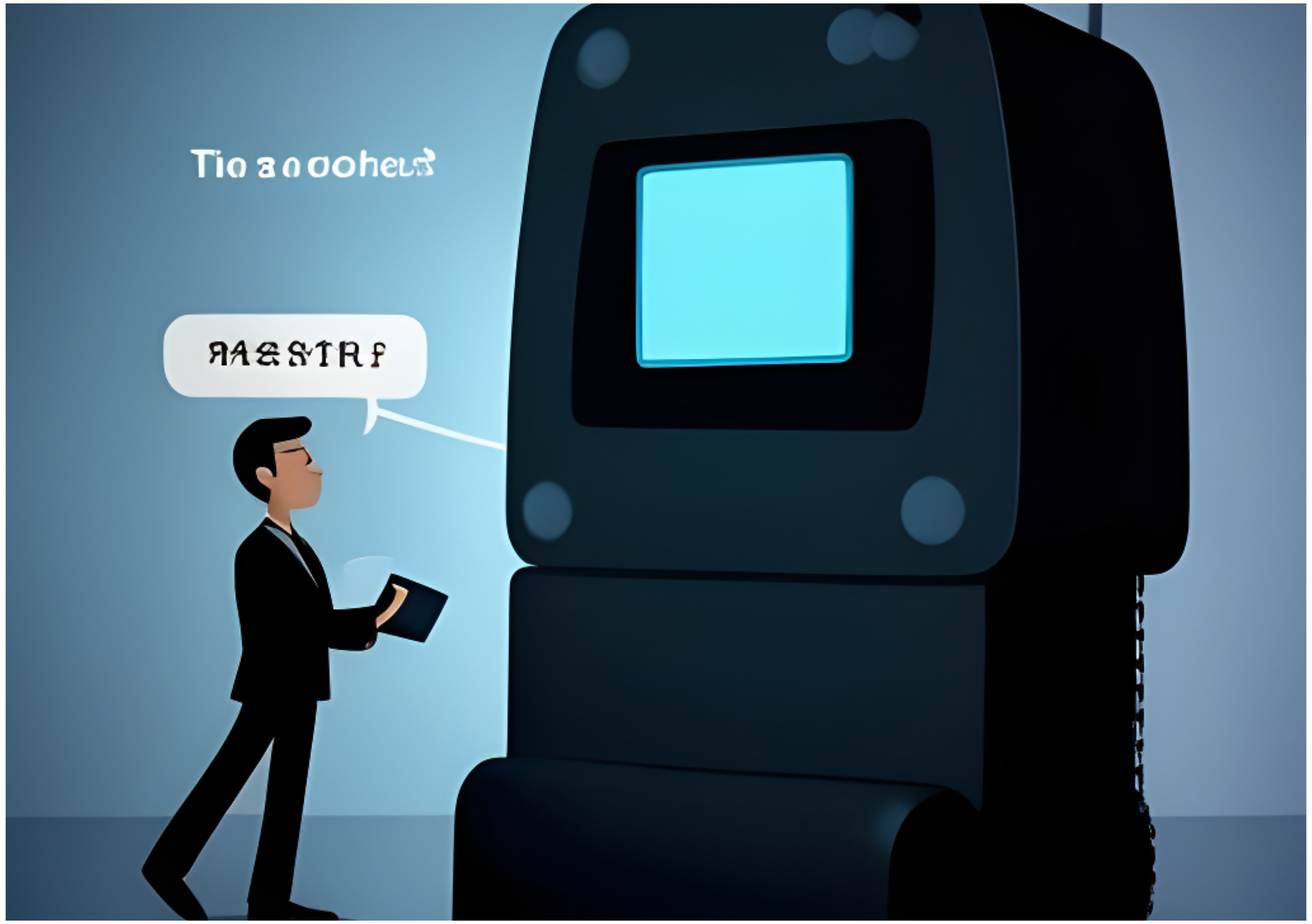
Talk at Central Applied Science, Meta

Natalie Collina, Upenn

Jon Schneider, Google Research

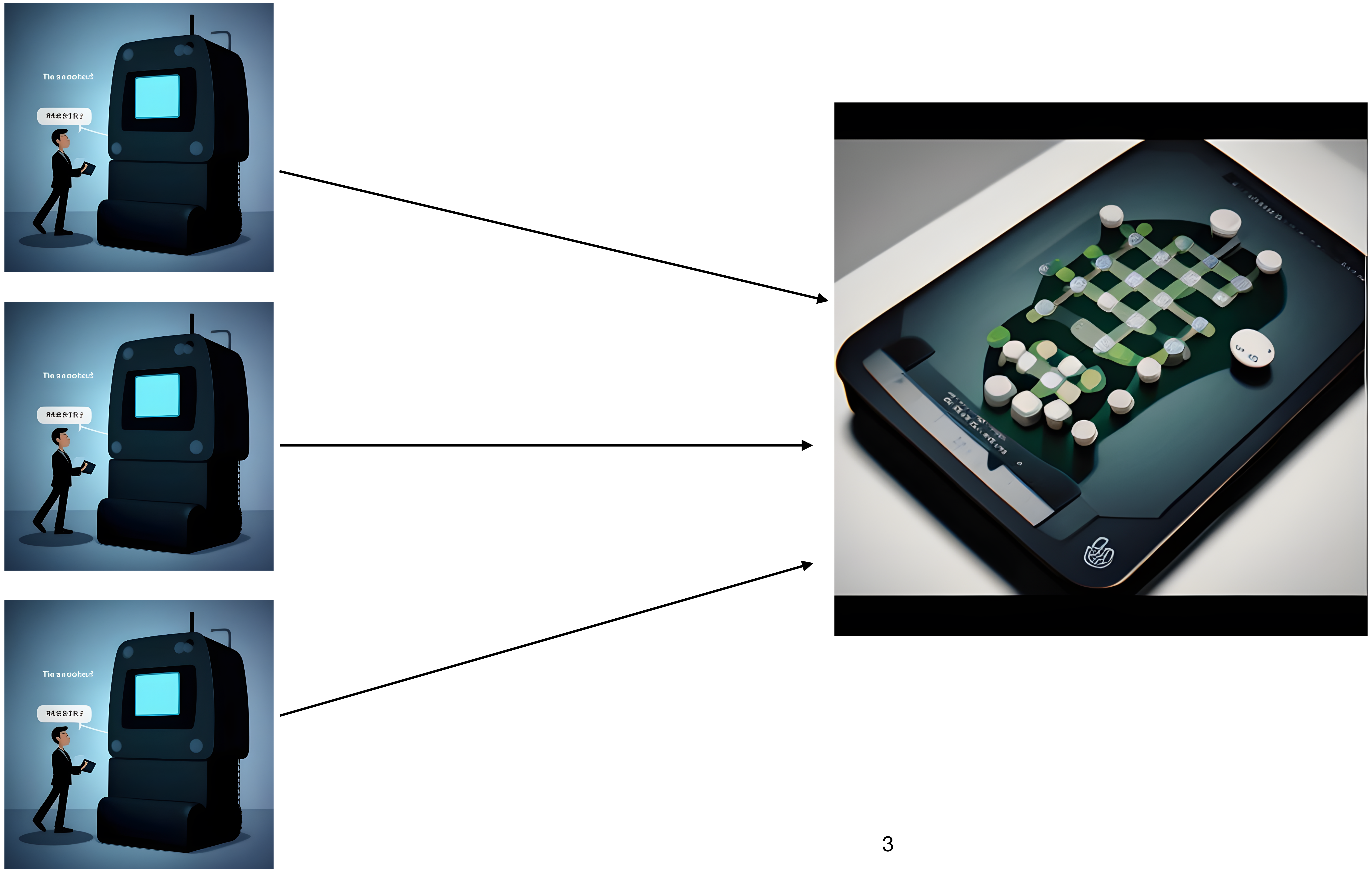
Motivation

Agents use Learning Algorithms to make Decisions



Motivation

Agents use Learning Algorithms to play games



Motivation

Agents use Learning Algorithms to play repeated games

Motivation

Repeated Ad-Auctions

We use an ad auction to determine the best ad to show to a person at a given point in time. The winning ad maximizes value for both people and businesses. Understanding the ad auction can help you understand your ad performance.

When do ad auctions take place?

Each time there's an opportunity to show an ad to someone, an auction takes place to determine which ad to show to that person. Billions of auctions take place everyday across the Facebook family of apps.

Who competes in each auction?

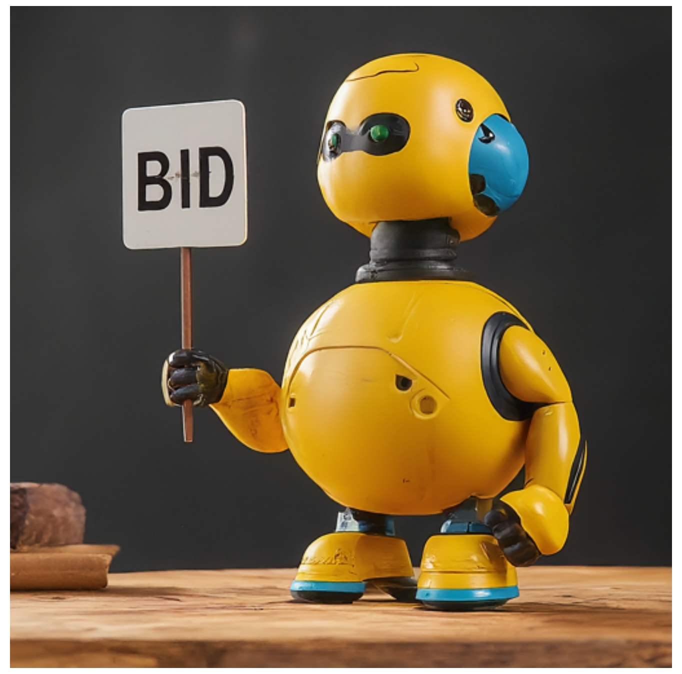
When advertisers create ads, they tell us who they want to show their ads to by defining a target audience. A person can fall into multiple target audiences. For example, one advertiser targets women who like skiing, while another advertiser targets all skiers who live in California. The same person (in this case, a female skier who lives in California) could fall into the target audience of both advertisers.

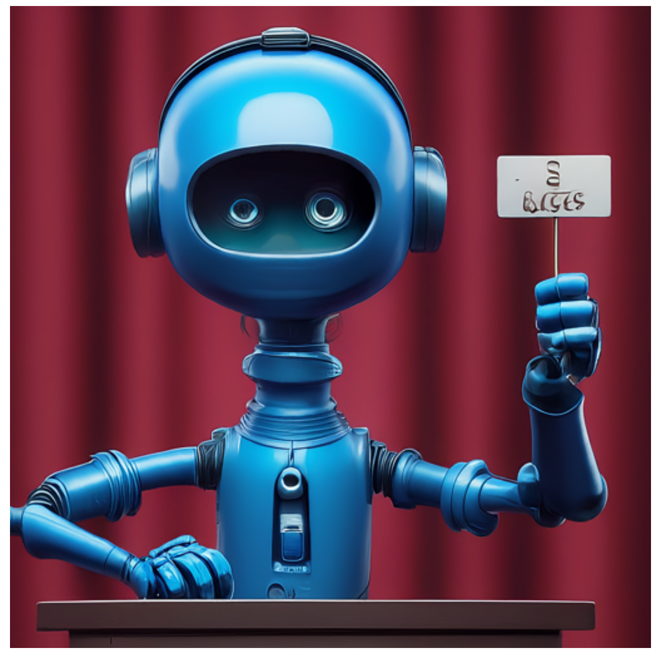
When there's an opportunity to show someone an ad, the ads with a target audience that the person belongs to are eligible to compete in the auction.

Motivation

Repeated Ad-Auctions

Advertising Slot





Automated
Auctioneer

Output

Winner + Price

Some Questions

- What are good learning algorithms to use?
- Existing Benchmark : No-Regret
- A New Criterion: Non-Manipulability
- Our Novel Criterion : Pareto-Optimality

- How might other agents respond to these learning algorithms?
- For eg: How should an auctioneer pick a dynamic pricing rule against certain bidding algorithms?

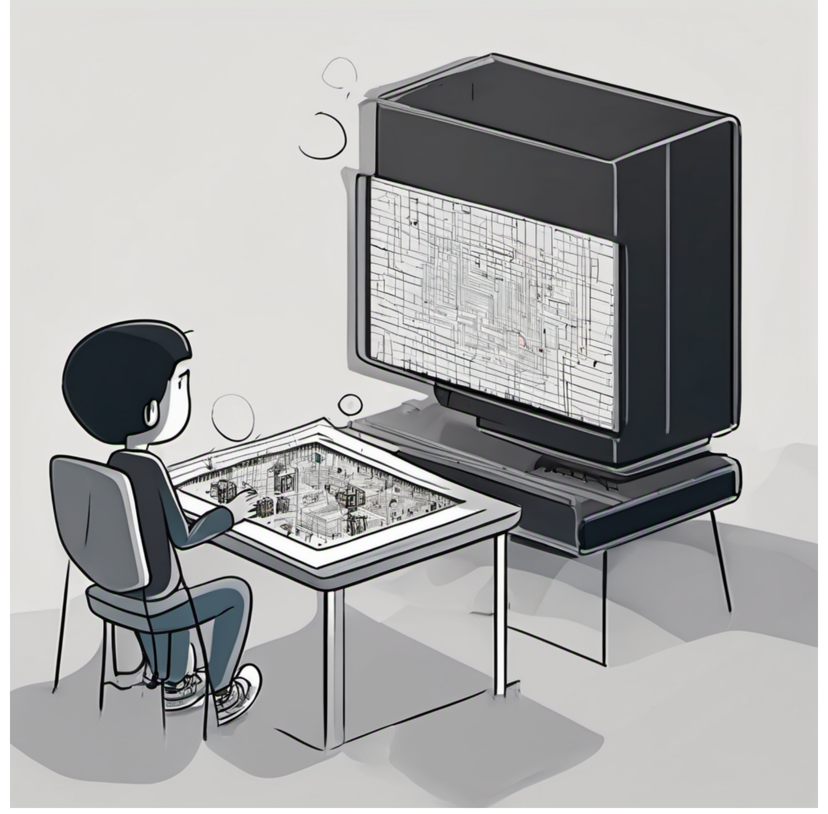
Model

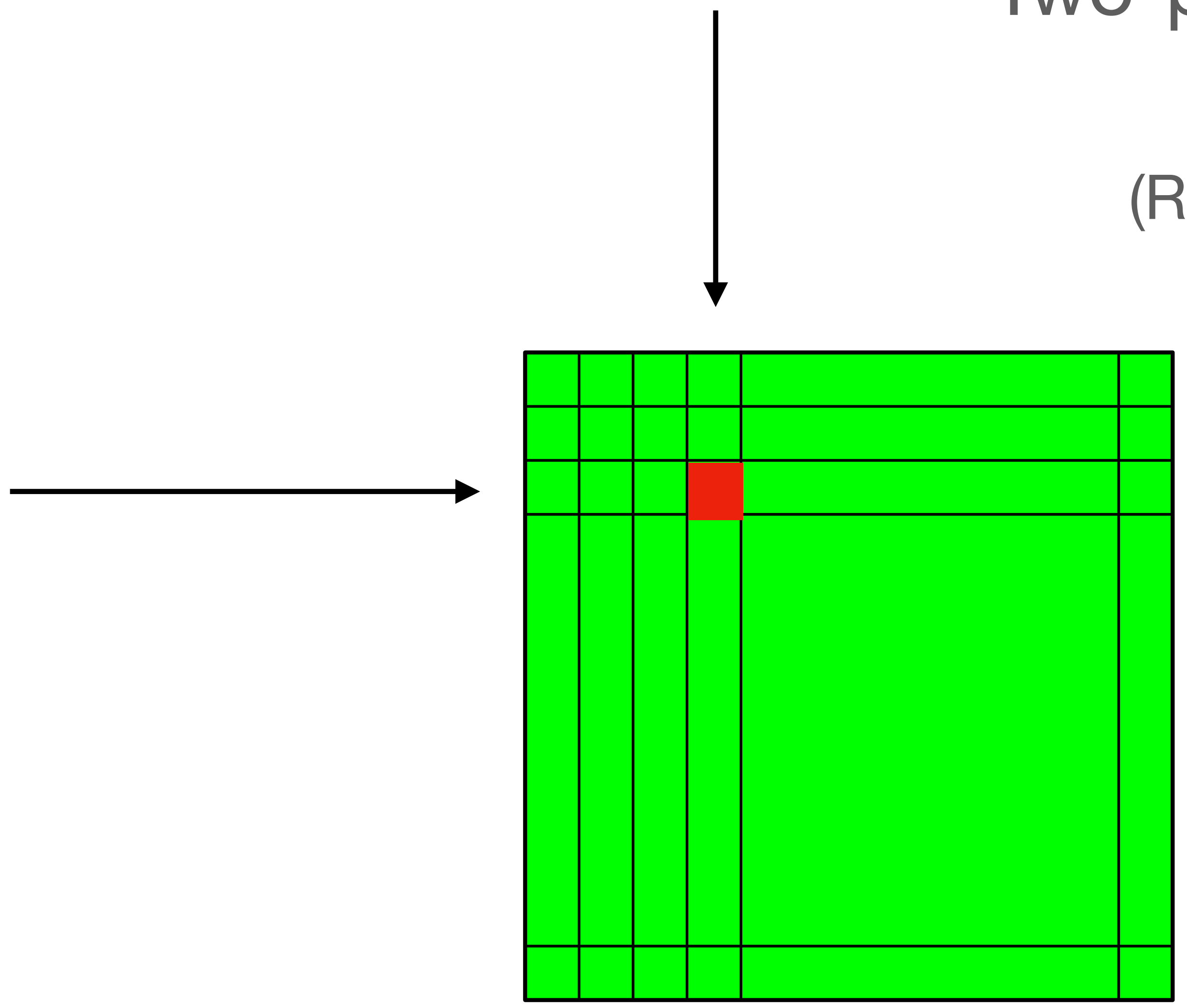
Model

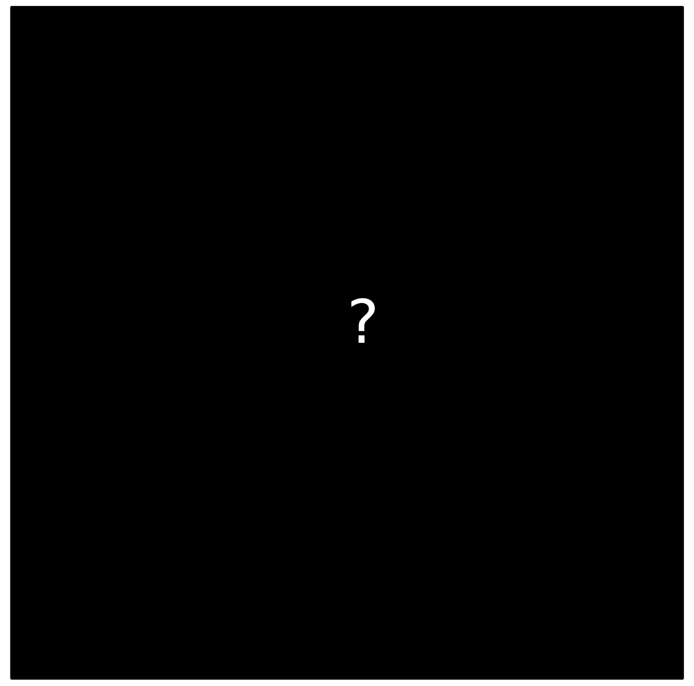
Two Players - Learner and Optimizer

Two-player bimatrix game

(Repeated T times)







The Learner knows their own payoff but not that of the optimizer

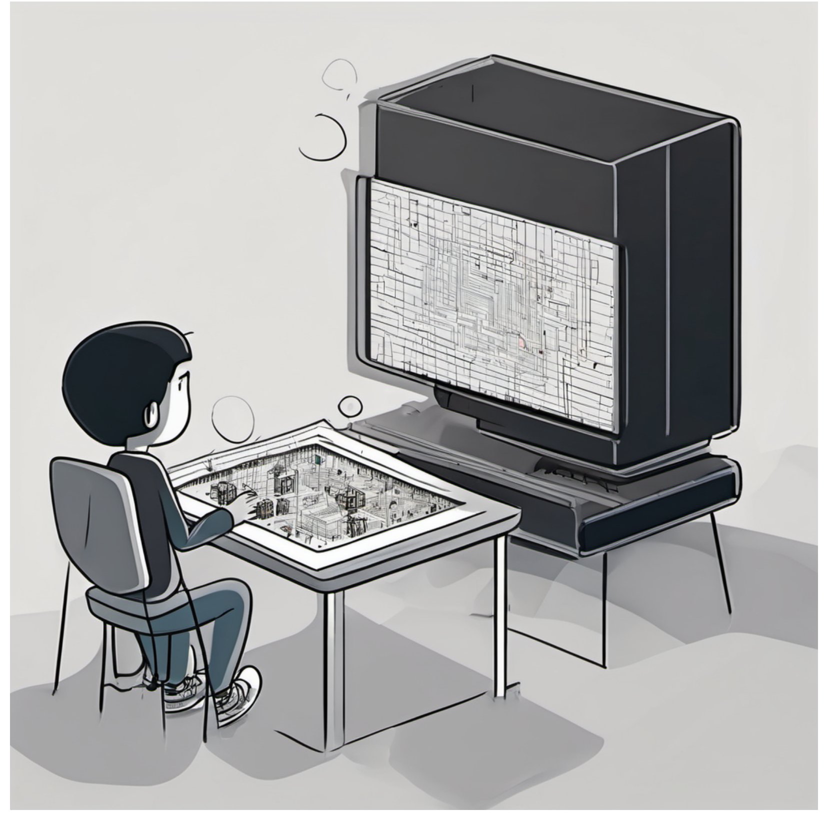
The optimizer has full information and best-responds (non-myopically)

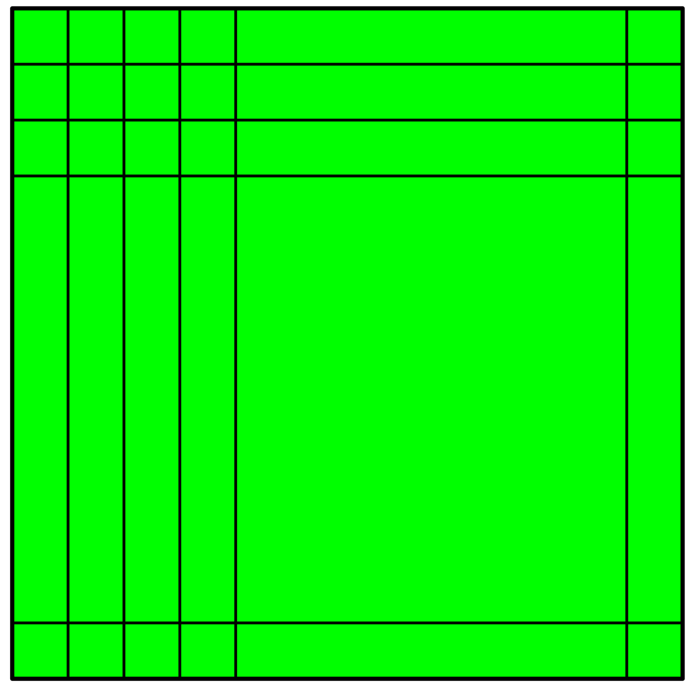
Model

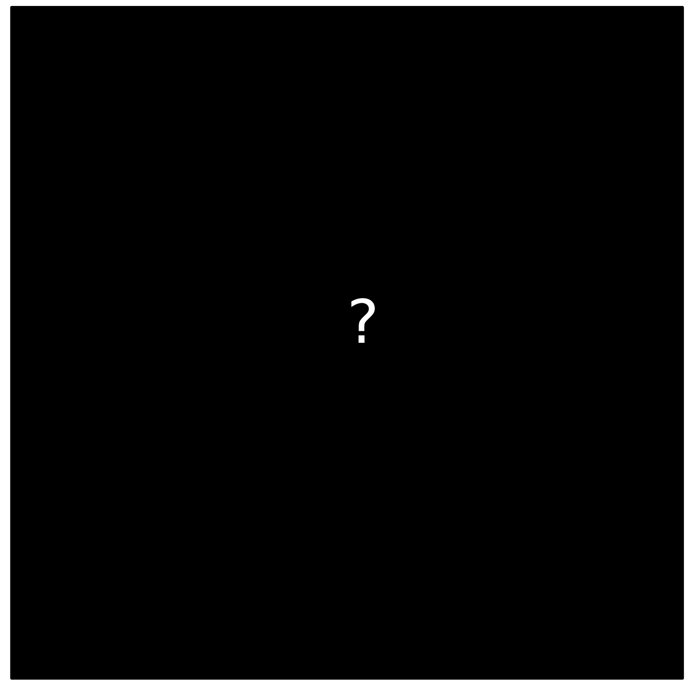
Two Players - Learner and Optimizer

Two-player bimatrix game

(Repeated T times)







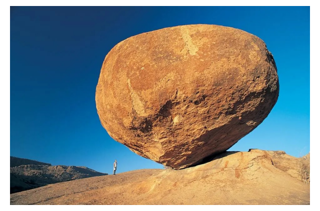
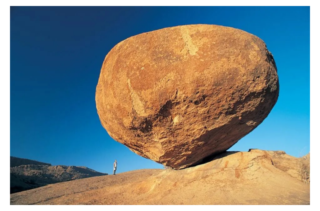
The Learner knows their own payoff but not that of the optimizer

The learner observes the action played by the optimizer in each round

The optimizer has full information and best-responds (non-myopically)

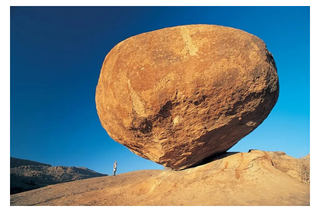
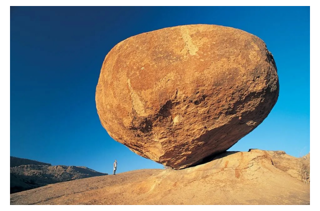
Example - The RPS game

A Two-Player Zero-sum Game

			
	0	-1	1
	1	0	-1
	-1	1	0

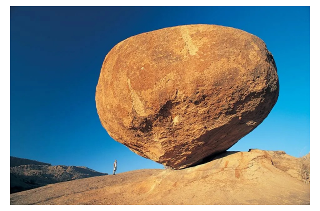
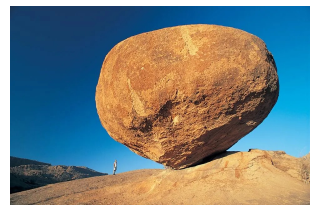
Learner Payoffs

Unknown to the Learner

			
	0	1	-1
	-1	0	1
	1	-1	0

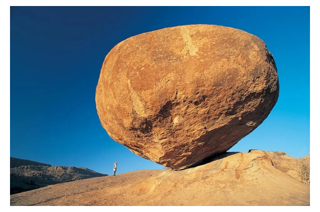
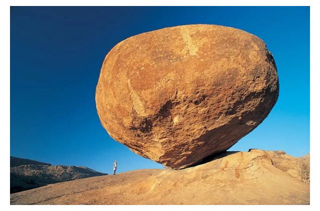
Optimizer Payoffs

Example - The RPS game

			
	0	-1	1
	1	0	-1
	-1	1	0

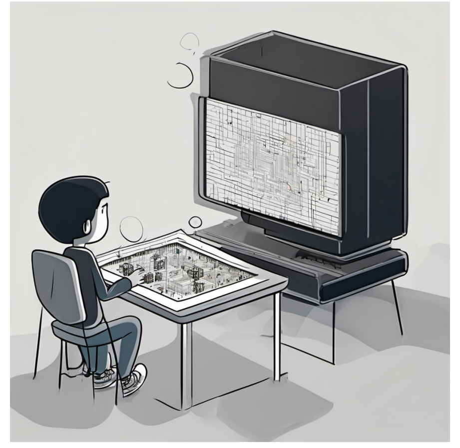
Learner Payoffs

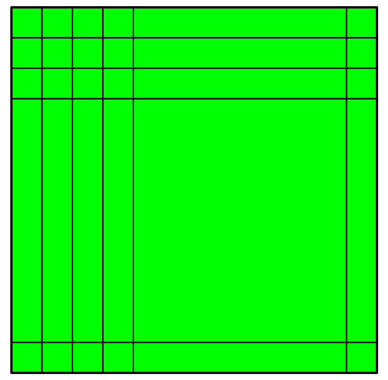
Unknown to the Learner

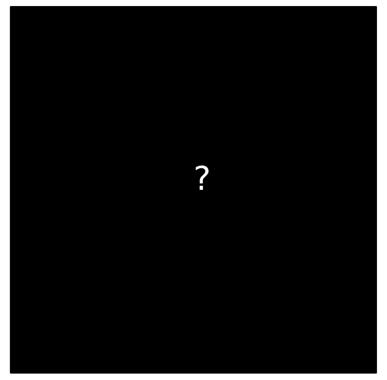
			
	1	0	0
	0	1	0
	0	0	1

Optimizer Payoffs

Model: Notation





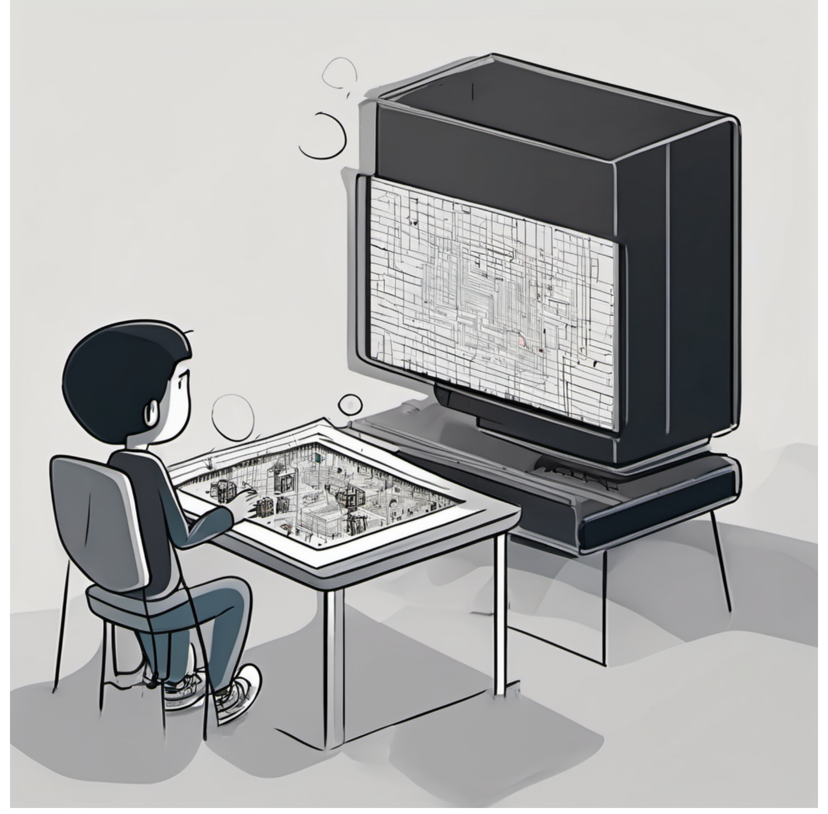


In Each Round

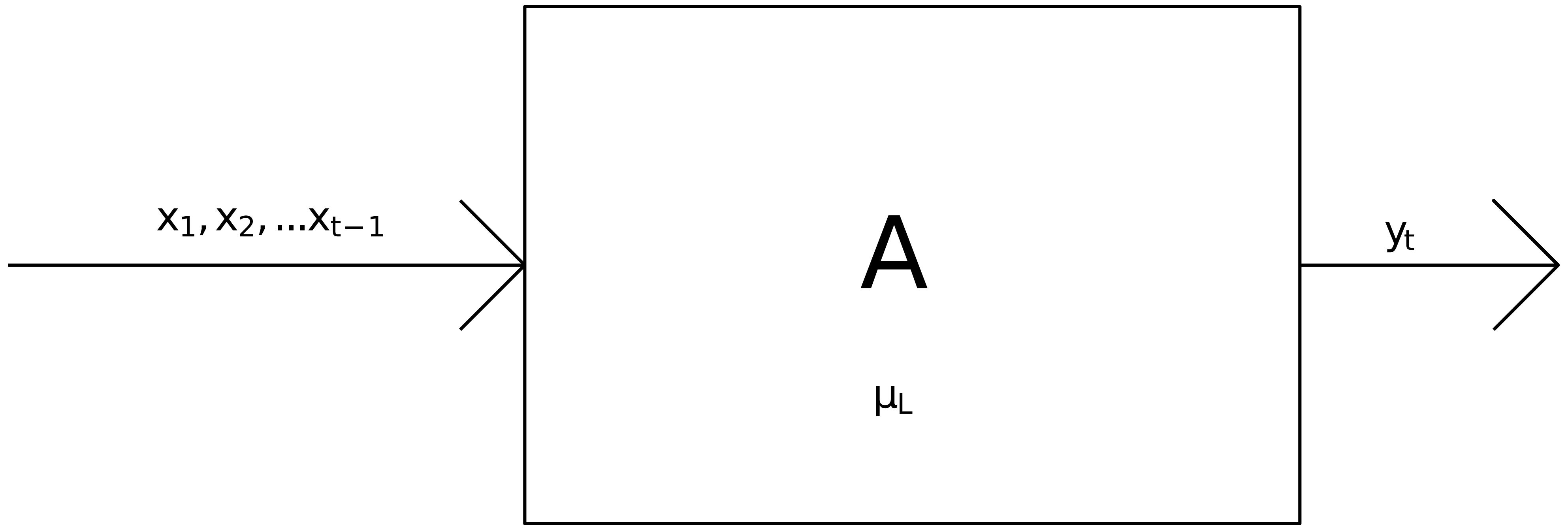
- The learner has action set Δ_m
- The optimizer has action set Δ_n
- They play actions y_t, x_t in the t-th round.
- Linear utility function u_L, u_O

Model : Learning Algorithms

The Learner Perspective



Without knowing u_O, the learner commit to an algorithm mapping (deterministically) from histories of play of length t-1 to a distribution y_t over actions in the t-th round



Model

The Optimizer Perspective

With full information (payoffs, learner algorithm), the optimizer plays a best-response sequence of actions

$$x_1, x_2 \cdots x_T \in \text{argmax}_{(x_1, x_2 \cdots x_T) \in \Delta_m^T} \frac{1}{T} \sum_{t=1}^{T} u_O(x_t, y_t)$$

Where $y_t = \mathscr{A}(x_1, x_2 \cdots x_{t-1})$

Model

Learner Payoff

With full information (payoffs, learner algorithm), the optimizer plays a best-response sequence of actions

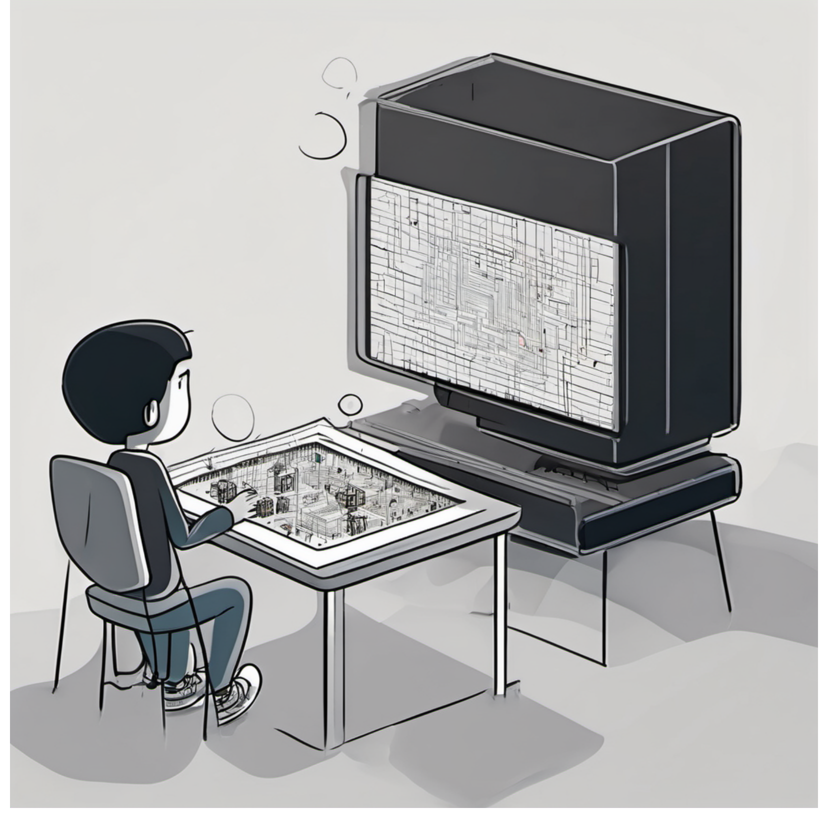
$$x_1, x_2 \cdots x_T \in \text{argmax}_{(x_1, x_2 \cdots x_T) \in \Delta_m^T} \frac{1}{T} \sum_{t=1}^{T} u_O(x_t, y_t)$$

Where $y_t = \mathscr{A}(x_1, x_2 \cdots x_{t-1})$

The learner gets payoff

$$V_L(\mathscr{A}, u_O, T) = \frac{1}{T} \sum_{t=1}^{T} u_L(x_t, y_t)$$

Model : The Stackelberg Perspective



The Learner Commits to a Learning Algorithm

The Optimizer plays a best-response sequence

The Learner wants to maximize their resulting payoff

A, Collina, Kearns - Solves the full information version of this problem

Our question - What is a good algorithm for the learning version?

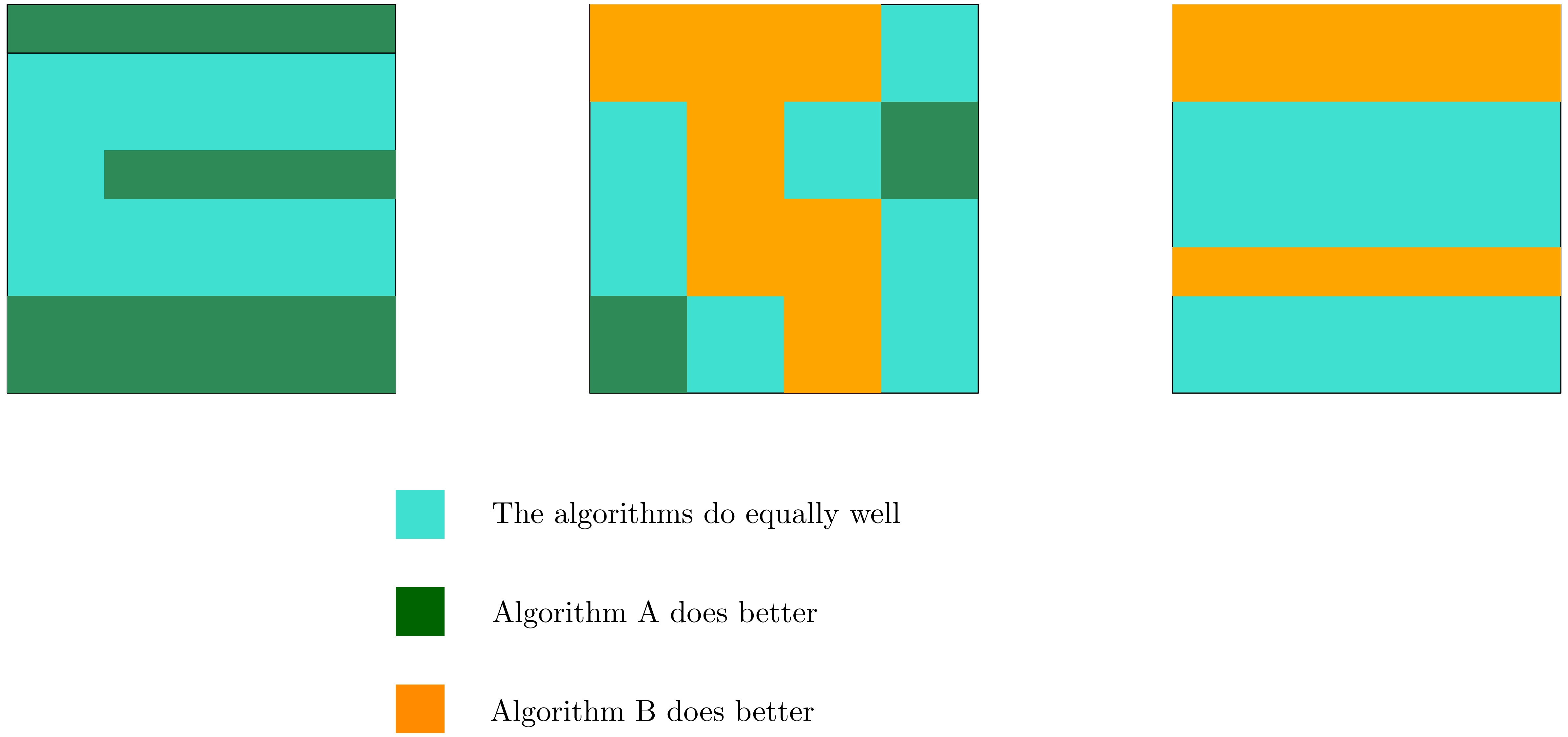
Pareto-Optimality

Re-defining optimality over all possible optimizers

A property of algorithms based upon a partial order over algorithms.Two Algorithms A and B$are compared over all possible optimizer payoffs

A property of algorithms based upon a partial order over algorithms.Two Algorithms A and B are compared over all possible optimizer payoffs

Three Scenarios:



Pareto-Optimality

Re-defining optimality over all possible optimizers

Algorithm A Pareto-dominates algorithm B for some payoff u_L if:

1. $\forall \mu_O : V_L(A, u_O) \geq V_L(B, u_O)$
2. $\exists \mu_O$ s.t. $V_L(A, u_O) > V_L(B, u_O)$

An algorithm is Pareto-Optimal if it is not Pareto-dominated

(All results are for positive measure sets and limit average payoffs)

A Basic Guarantee : No-Regret

Pick action
$y_t \in Y$

Get feedback from an adaptive adversary. $f_t : Y \to [-1,1]$

Objective : Guarantee that the performance is comparable to the single best action in hindsight

i.e. $$\sum_{t=1}^{T} f_t(y_t) \geq \max_{y^* \in Y} \sum_{t=1}^{T} f_t(y^*) - o(T)$$

Realized utility

Best response in hindsight

No-Regret : Applications

Algorithms exists given convexity

- Online Shortest Path Problem (All s-t paths)
- Online Classification (All classifiers in a concept class)
- Boosting Weak Classifiers (via Minimax Computation)
- Bidders behavior in online auctions is consistent with no-regret learning algorithms [Nekipelov et al., 2015]

No-Regret : Applications

Algorithms exists given convexity

In our setting : $\sum_{t=1}^{T} u_L(x_t, y_t) \geq \max_{y^* \in \Delta^n} \sum_{t=1}^{T} u_L(x_t, y^*) - o(T)$

For example: Rock, Paper and Scissors:

Learner Sequence

......

Optimizer Sequence

......

FTRL

A Popular class of No-Regret Algorithms

Given that R is continuous and strongly-convex, and $\eta_T = \frac{1}{o(T)}$:

$$y_t = \arg\max_{y \in \Delta^n} \left(\sum_{s=1}^{t-1} u_L(x_s, y) - \frac{R(y)}{\eta_T} \right)$$

All Follow-the-Regularized Leader type algorithms, including Multiplicative Weights (Hedge), Online Gradient Descent are Mean-Based No-Regret Algorithms

A Stronger Guarantee : No-Swap-Regret

Pick action
$y_t \in Y$

Get feedback. $f_t : Y \to [-1,1]$

Objective : Guarantee that the performance is comparable to any swap function in-hindsight

i.e. $$\sum_{t=1}^{T} f_t(y_t) \geq \max_{\pi : Y \to Y} \sum_{t=1}^{T} f_t(\pi(y_t)) - o(T)$$

No-Swap-Regret

A Stronger version of No-Regret

- Calibrated Forecasting
- Boosting for Regression
- Stronger Guarantees exist for context-based subsequences

Non-Manipulability

The optimizer has an asymptotic best-response that is just playing a static strategy over time

Is there an algorithm that has all three properties - No-Regret, Pareto-Optimality and Non-Manipulability?

Trivial to achieve any one property

Our Results

Main Results

Result 1:

All No-Swap-Regret algorithms are Pareto-Optimal and non-manipulable.

PS: The non-manipulability result was already proved by Deng et al. (2019), via a different sequence of arguments

Our Results

Main Results

Result 2:

Not all No-Regret algorithms are Pareto-optimal. Specifically, Follow-the-Regularized-Leader (FTRL) based algorithms (which includes Multiplicative Weights Update, Online Gradient Descent) are Pareto-dominated.

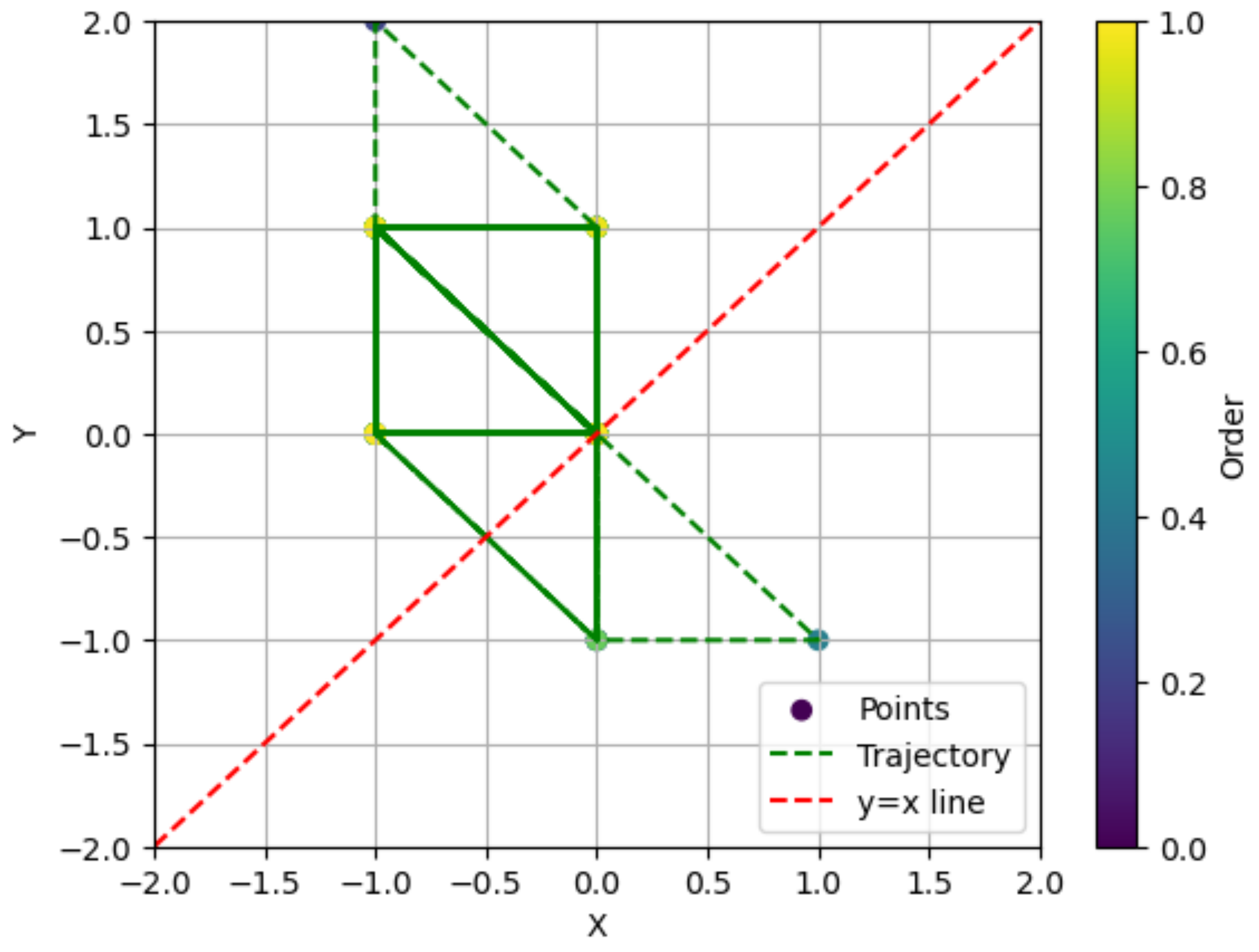
Our Results

Other Results

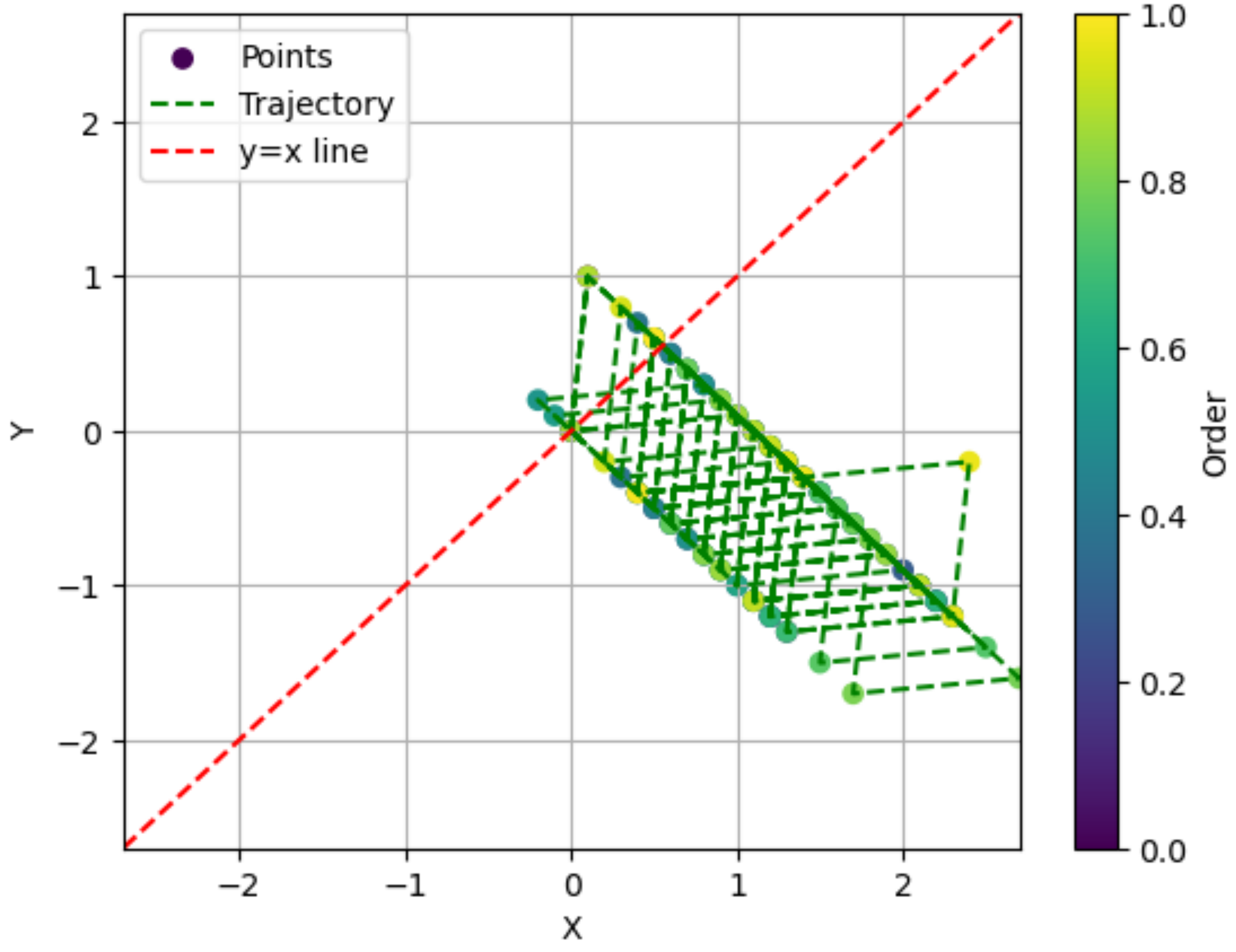
- A Geometric View of Algorithms
- A characterization of best-responses to mean-based no-regret algorithms (i.e. how to manipulate them)
- A characterization of Pareto-optimal No-Regret Algorithms

RL Experiment for Optimizer

Best-Response to Multiplicative Weights



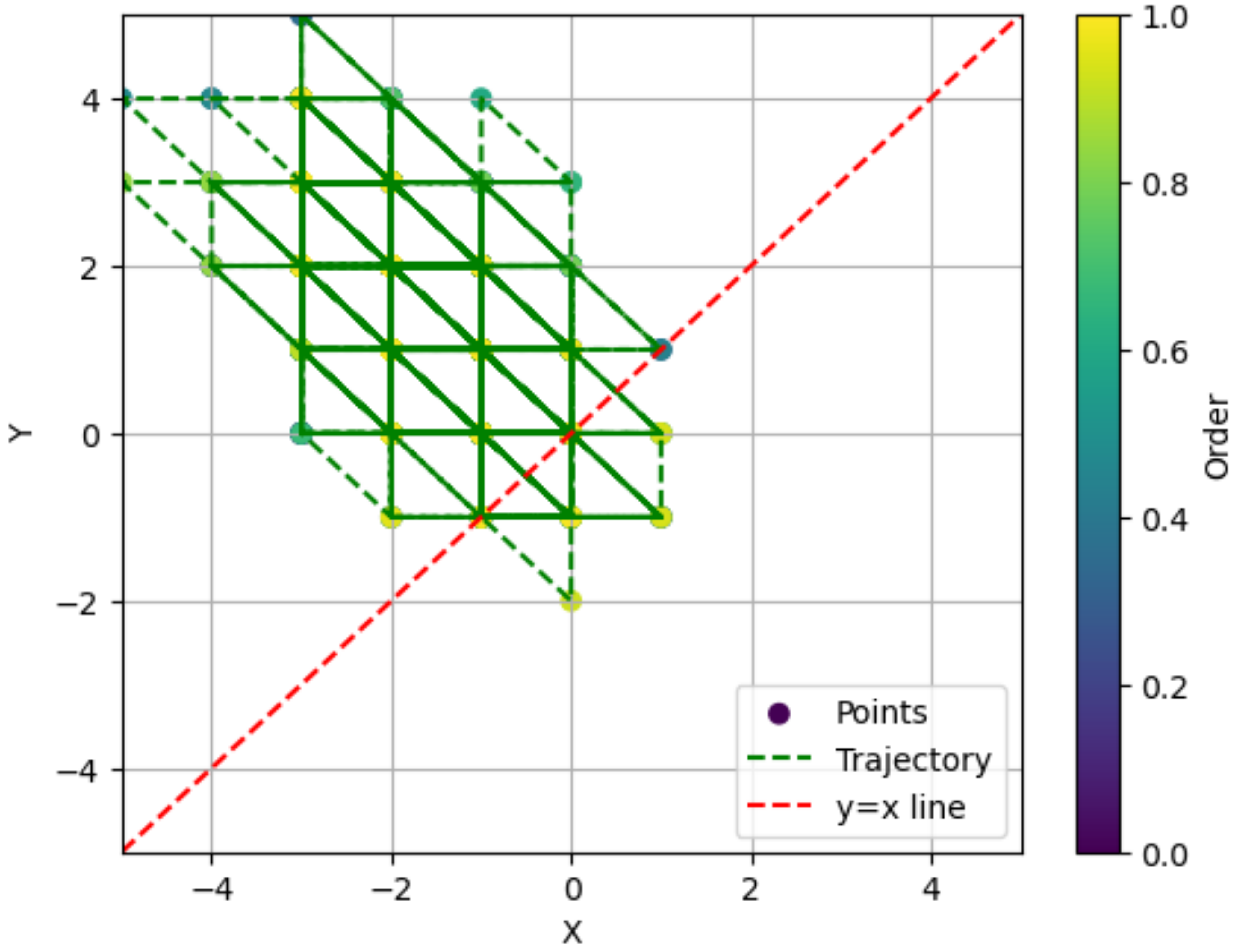
Rock, Paper, Scissors for T=1000



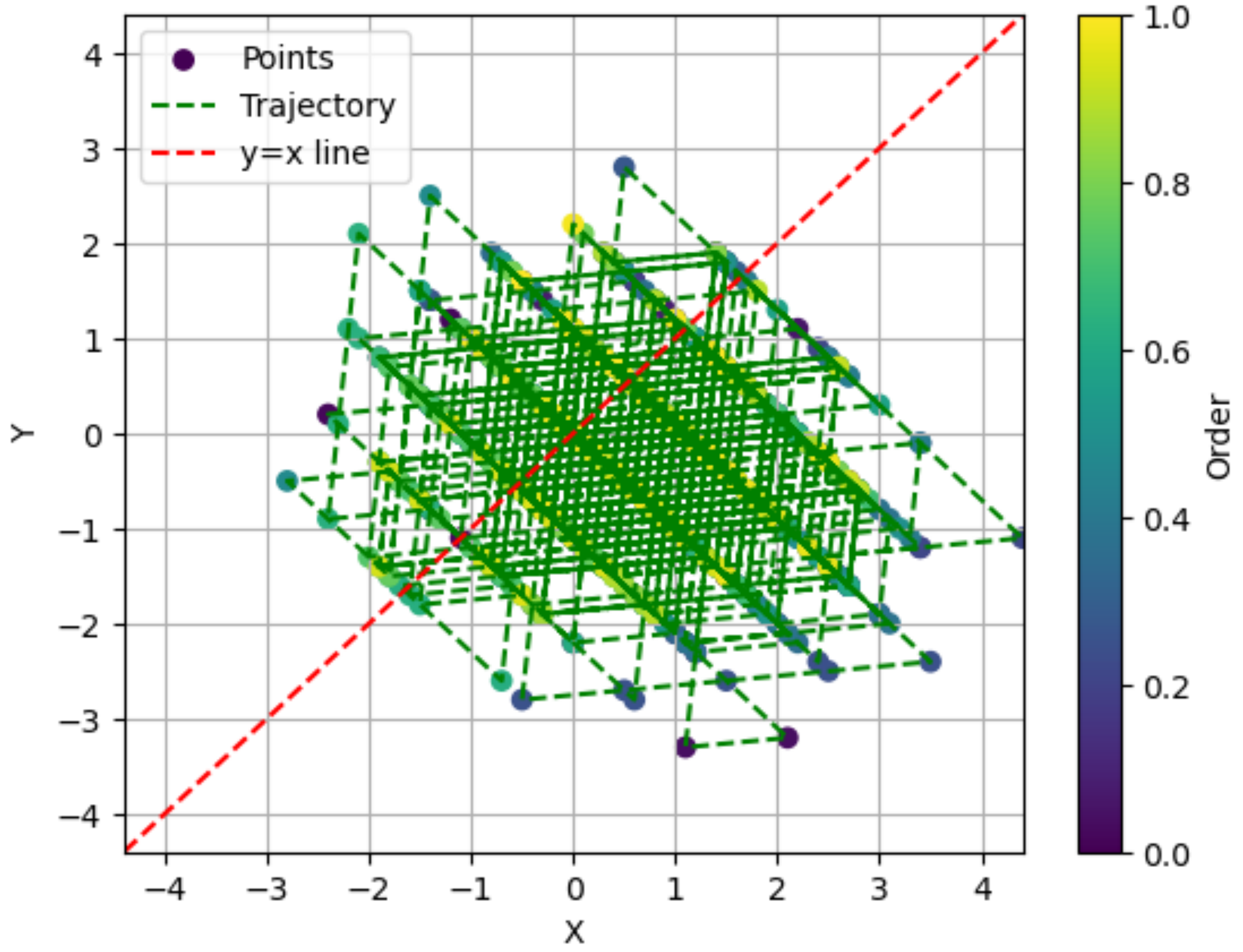
Modified RPS (Non zero sum) for T=100

RL Experiment for Optimizer

Best-Response to Multiplicative Weights



Rock, Paper, Scissors for T=1000



Modified RPS (Non zero sum) for T=1000

Talk Plan

- Geometric View of Learning Algorithms - Menus
- NSR is Non-Manipulable (Intuition)
- FTRL is Pareto-dominated (Intuition) (Time Permitting)
- Future Directions/ Related Work

Menus

Summaries of Play

Transcripts and Correlated Strategy Profiles (CSPs)

Transcript of Play

Sequence of action pairs
$\{x_t, y_t\}_{t=1}^{T}$

Correlated Strategy Profile

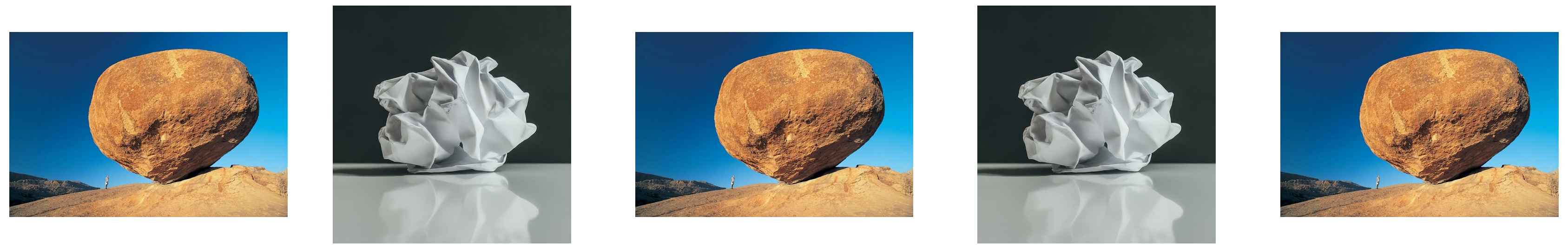
Empirical distribution over all resulting pure action pairs

$$\phi = \frac{1}{T}\sum_{t=1}^{T} x_t \otimes y_t$$

CSPs are sufficient to check for no-regret/ no-swap-regret

CSP : Example

Transcripts and Correlated Strategy Profiles (CSPs)

Learner Sequence  …… Algorithm : Mimic the optimizer

Optimizer Sequence  …… Sequence: Alternate Paper and Rock

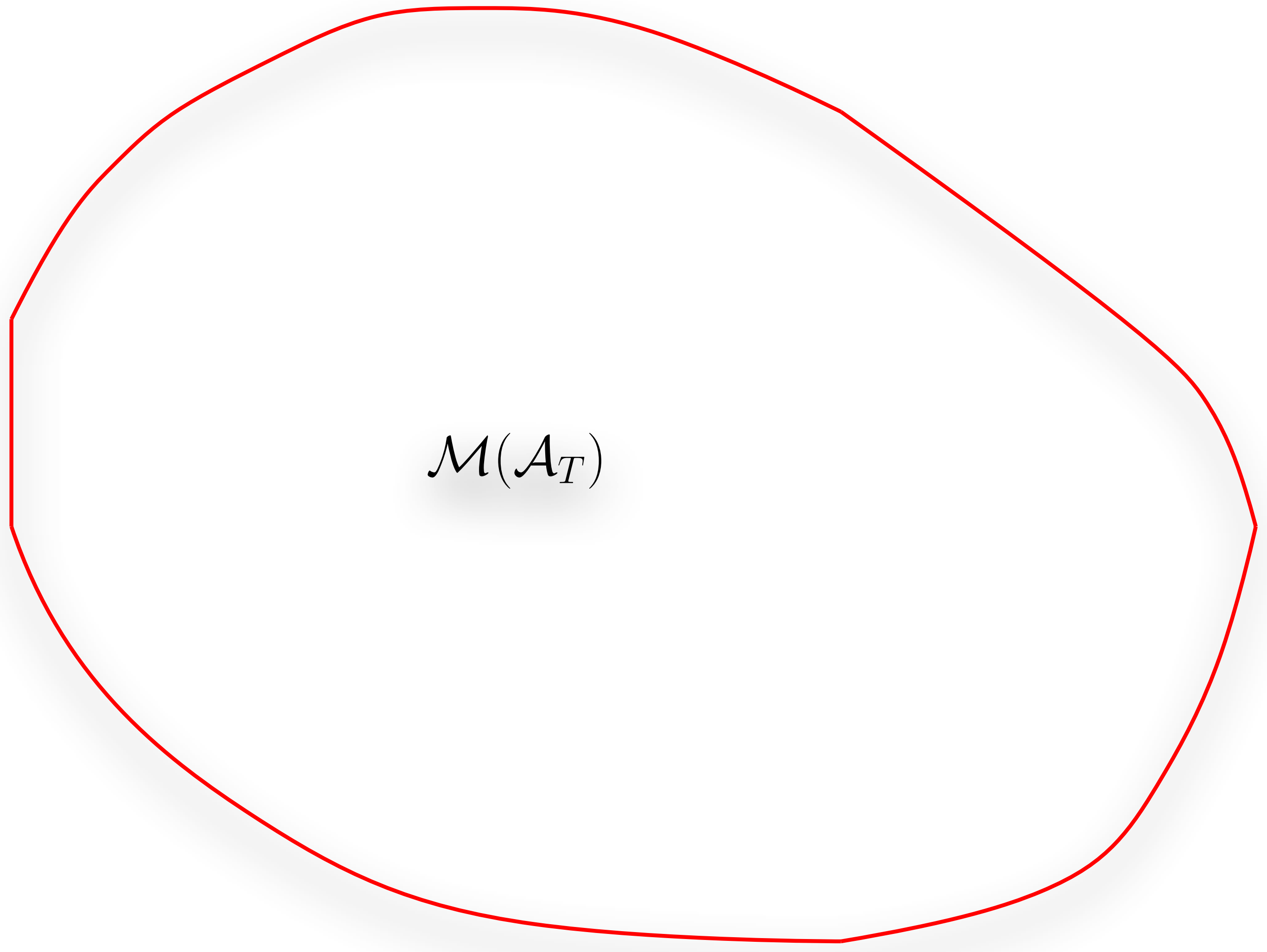
CSP: $\phi = 1/2(R \otimes P) + 1/2(P \otimes R)$

Menus

All possible CSPs

For every optimizer sequence $x_1, x_2, \cdots x_T$, record the induced CSP

Menu of an Algorithm : Take the convex hull of this set



Implicitly : The Limit Menu as $T \to \infty$

Menus: An Example

All possible CSPs

Learning Algorithm A1: Always play P

	A	B
P	X	X
Q	X	X

Menus: Example 1

All possible CSPs

Learning Algorithm A1: Always play P

	A	B
P	X	X
Q	X	X

Menus: An Example

All possible CSPs

Learning Algorithm A2: Play Q as long as the Optimizer has always played A. Otherwise, play P

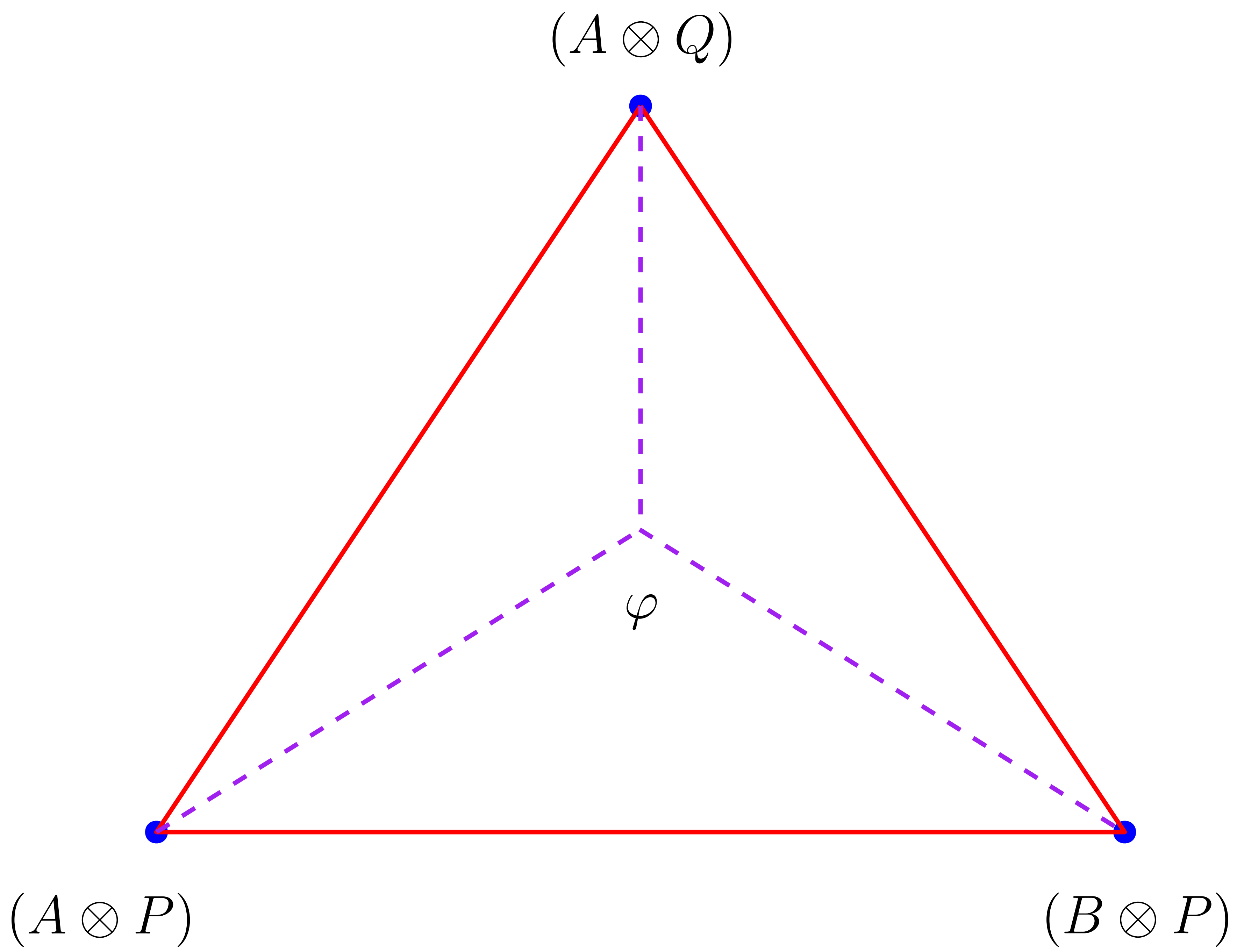
	A	**B**
P	X	X
Q	X	X

Menus: Example 2

All possible CSPs

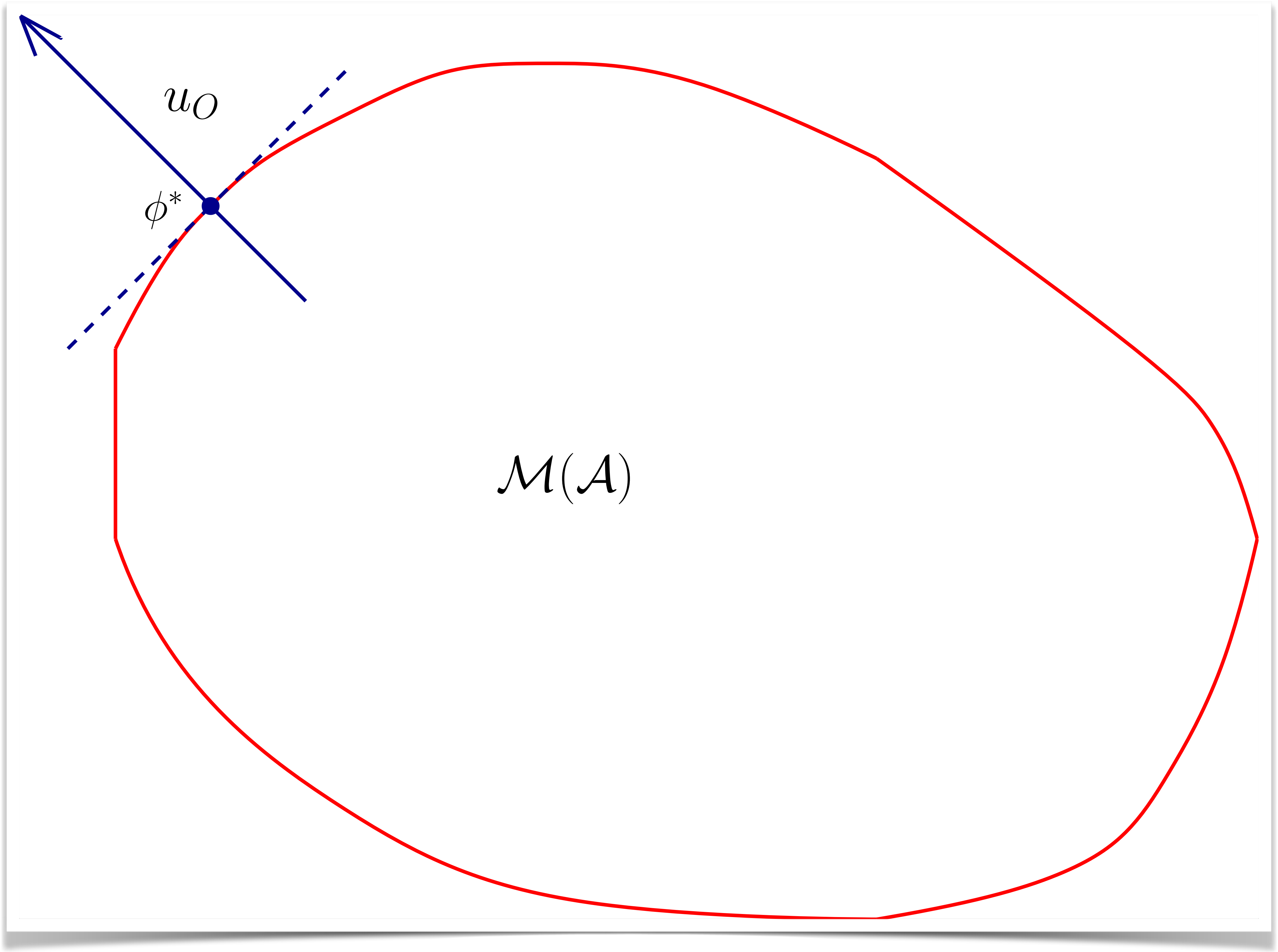
Learning Algorithm A2: Play Q as long as the Optimizer has always played A. Otherwise, play P

	A	B
P	X	X
Q	X	X



Menus are all you need

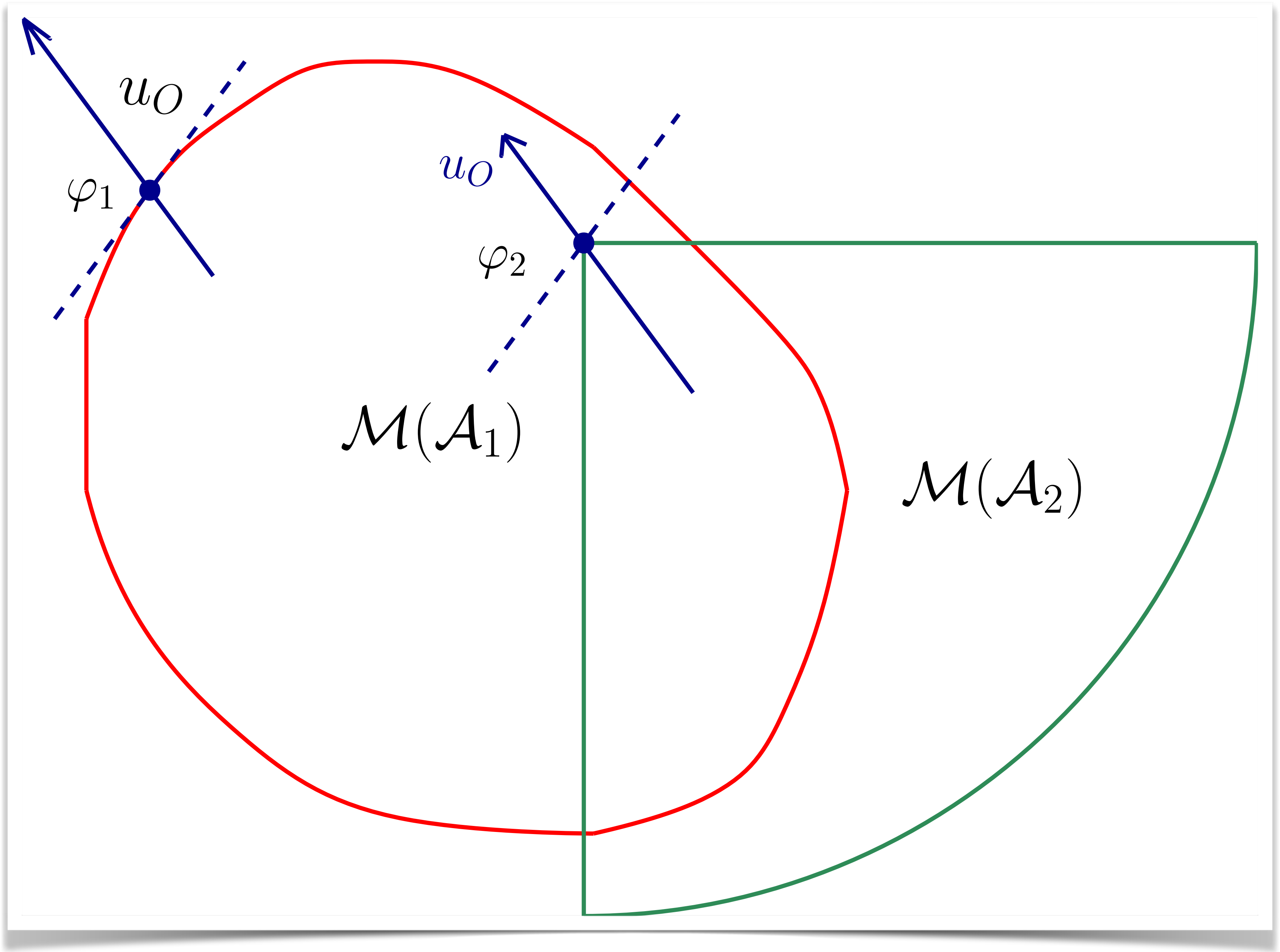
Learner and Optimizer Payoffs



The optimizer "picks" their favorite extreme point

Menus are all you need

Pareto-Optimality



Menus are all you need

No-Regret : Property of the CSPs

A CSP ϕ is no-regret if, for each $j \in [n]$, it satisfies

$$\sum_{i \in [m]} \phi_{ij} u_L(i,j) \geq \max_{j^* \in [n]} \sum_{i \in [m]} \phi_{ij} u_L(i,j^*) \,.$$

Menus are all you need

Non-Manipulability

All Extreme points of the algorithm's menu are product distributions

Menus are all you need

Non-Manipulability : A Negative Example

All Extreme points of the algorithm's menu are product distributions

Algorithm : Follow the Leader

Learner Sequence

Optimizer Sequence

......

......

Corresponding CSP $1/3(P \otimes R) + 1/3(S \otimes P) + 1/3(R \otimes S)$

Menus: Proving Pareto-Optimality

Inclusion-Minimality implies Pareto-Optimality

Every inclusion-minimal menu that contains ϕ^+ is Pareto-Optimal.

Menus: Proving Pareto-Optimality

Inclusion-Minimality implies Pareto-Optimality

Definition: Inclusion-Minimality

A menu M_1 is inclusion-minimal if there is no menu M_2 such that $M_2 \subsetneq M_1$.

Definition: φ^+

$$\phi^+ = x^* \otimes y^*, \text{ where } (x^*, y^*) = \arg\max_{(x,y)} u_L(x, y)$$

Menus: Proving Pareto-Optimality

ϕ_+-Inclusion-Minimality implies Pareto-Optimality

Lemma : If M_1 contains φ^+ and $M_2 \backslash M_1 \neq \emptyset$, then there is an Optimizer payoff u_O such that

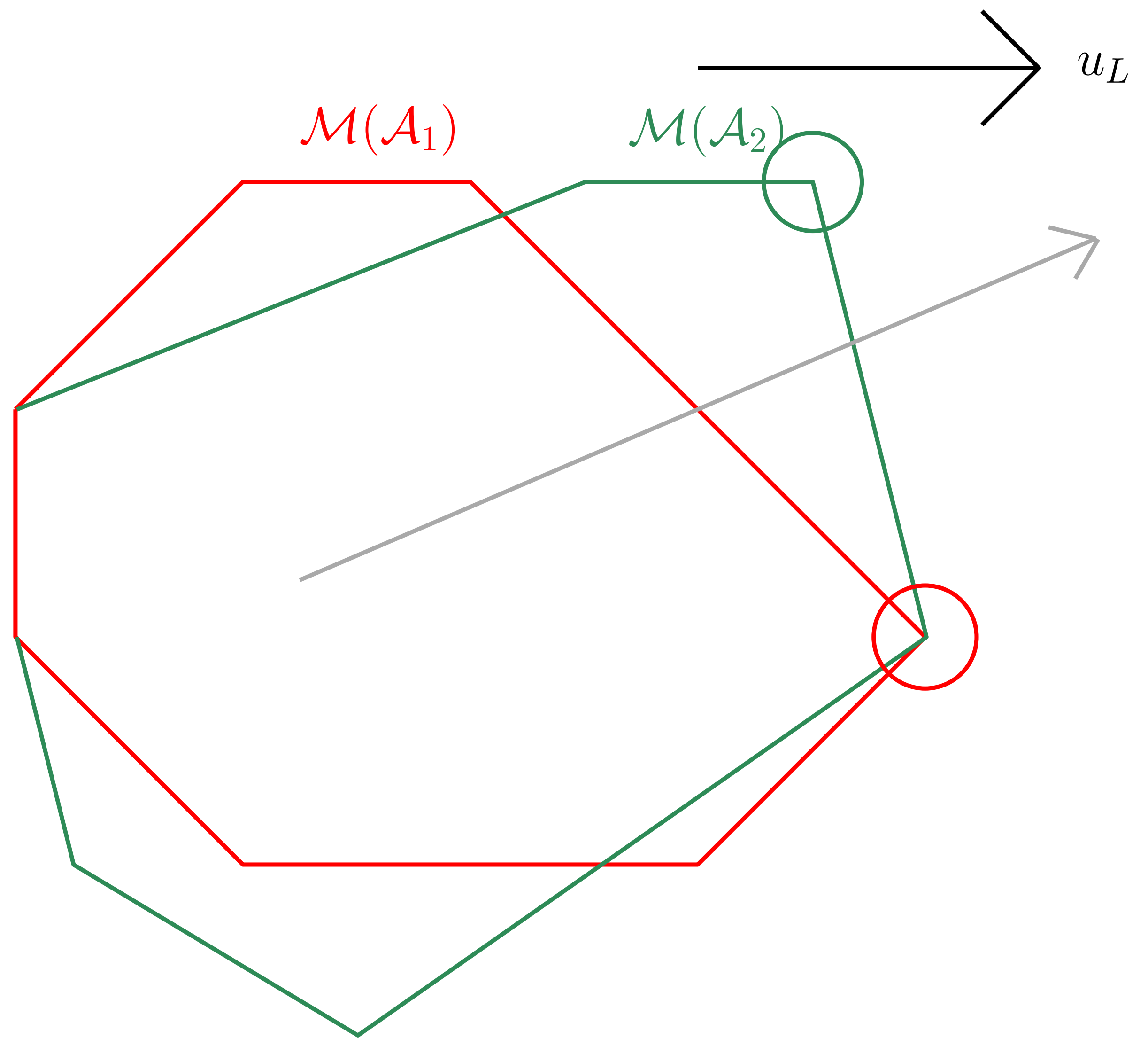
$$V_L(M_1, u_O) > V_L(M_2, u_O)$$

Negating Pareto-domination of one algorithm by another requires only a single certificate

Menus: Proving Pareto-Optimality

Lemma : If M_1 contains φ^+ and $M_2 \backslash M_1 \neq \emptyset$, then there is an Optimizer payoff u_O such that

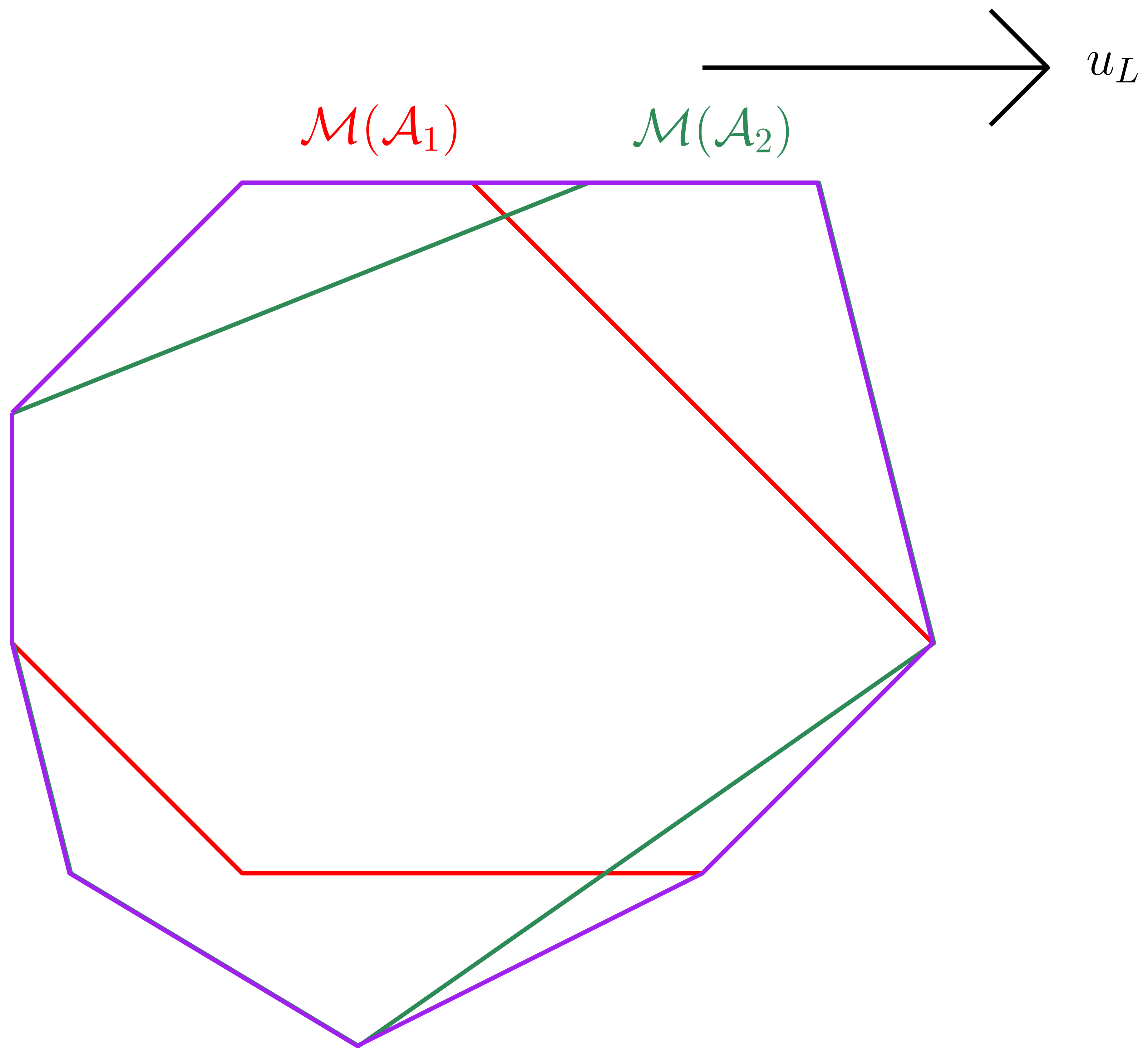
$$V_L(M_1, u_O) > V_L(M_2, u_O)$$



Special case: Both menus are polytopes

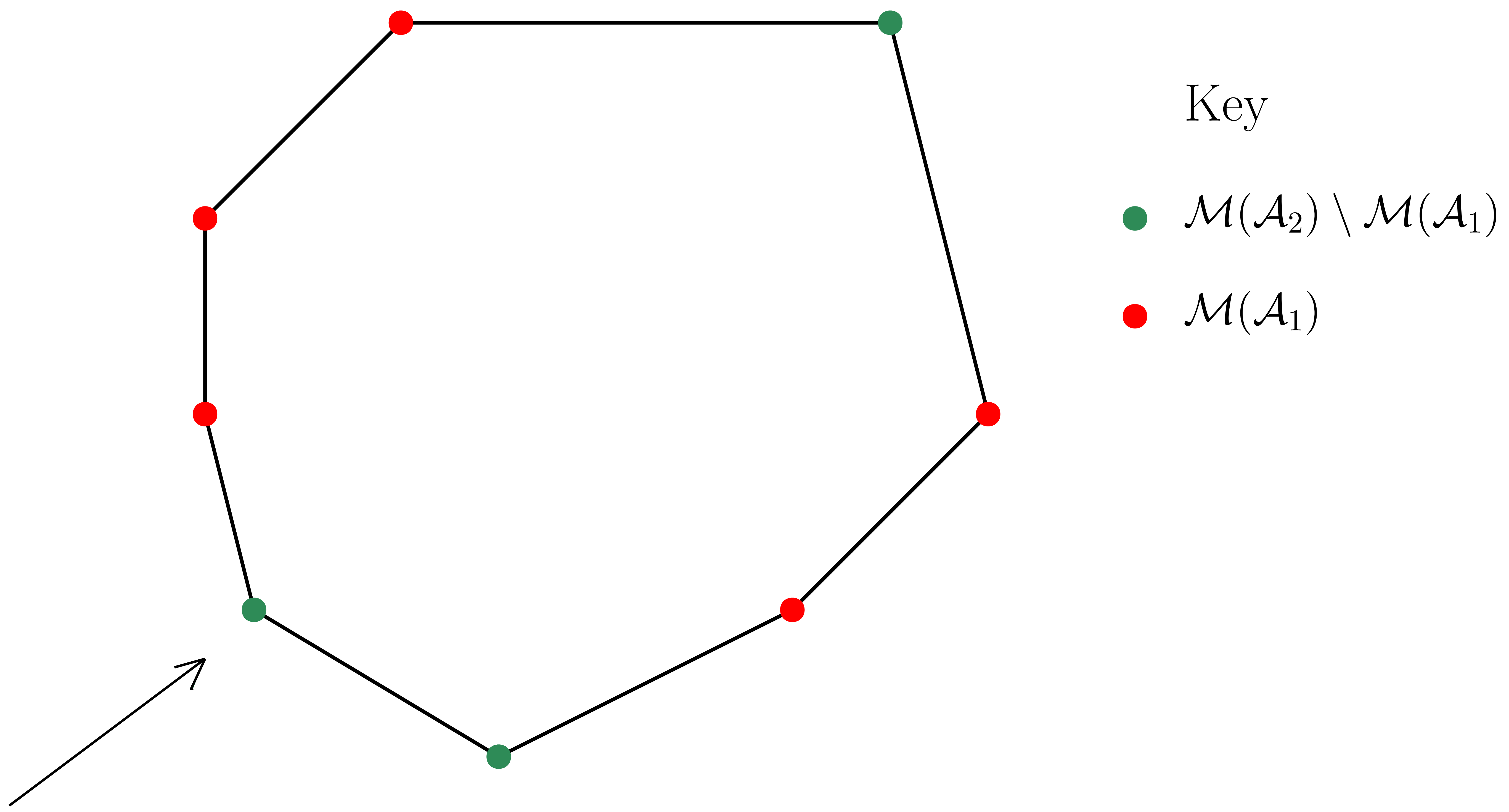
Menus: Proving Pareto-Optimality

Take the convex hull of the union



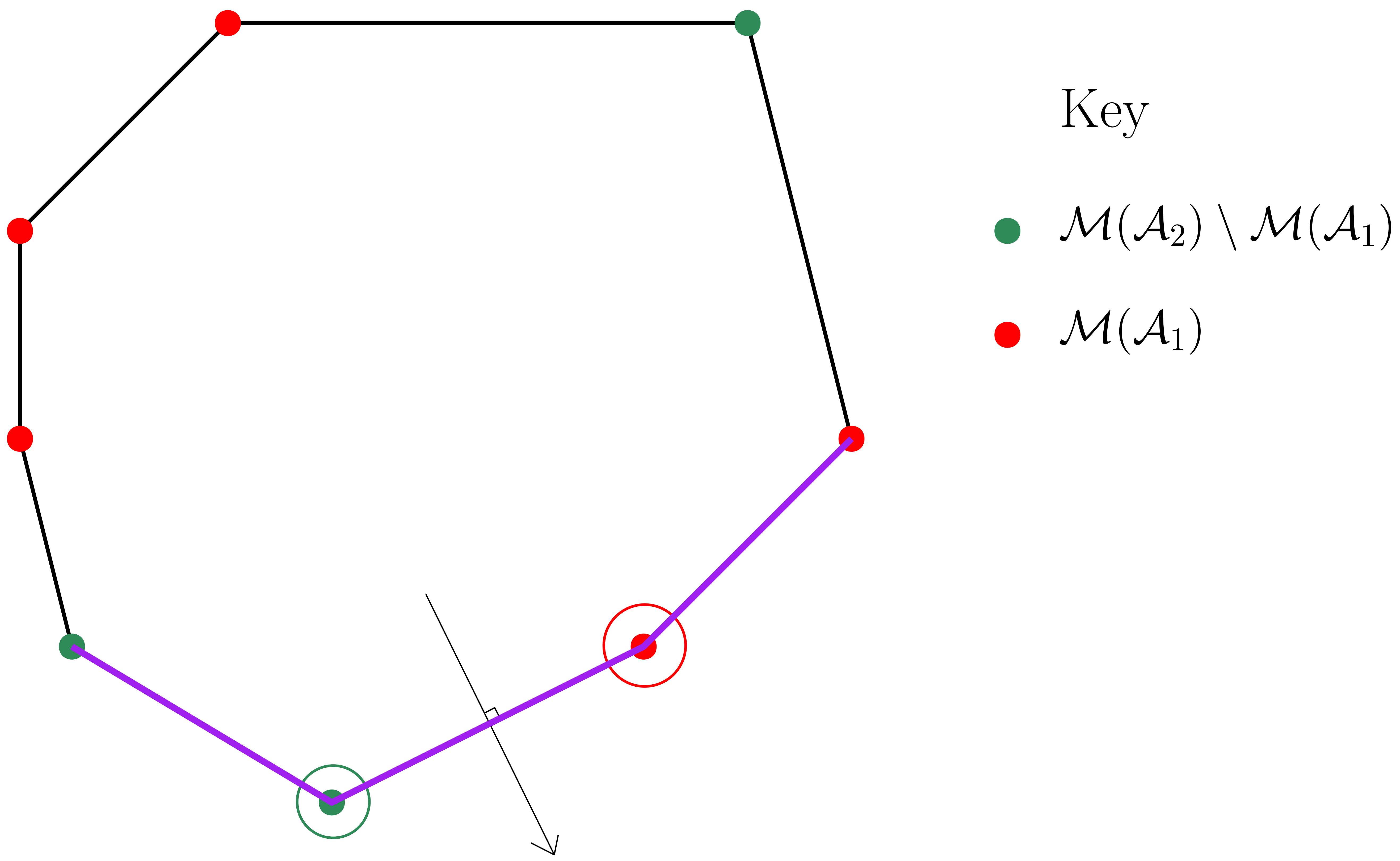
Menus: Proving Pareto-Optimality

Start with an “extra” vertex in M_2



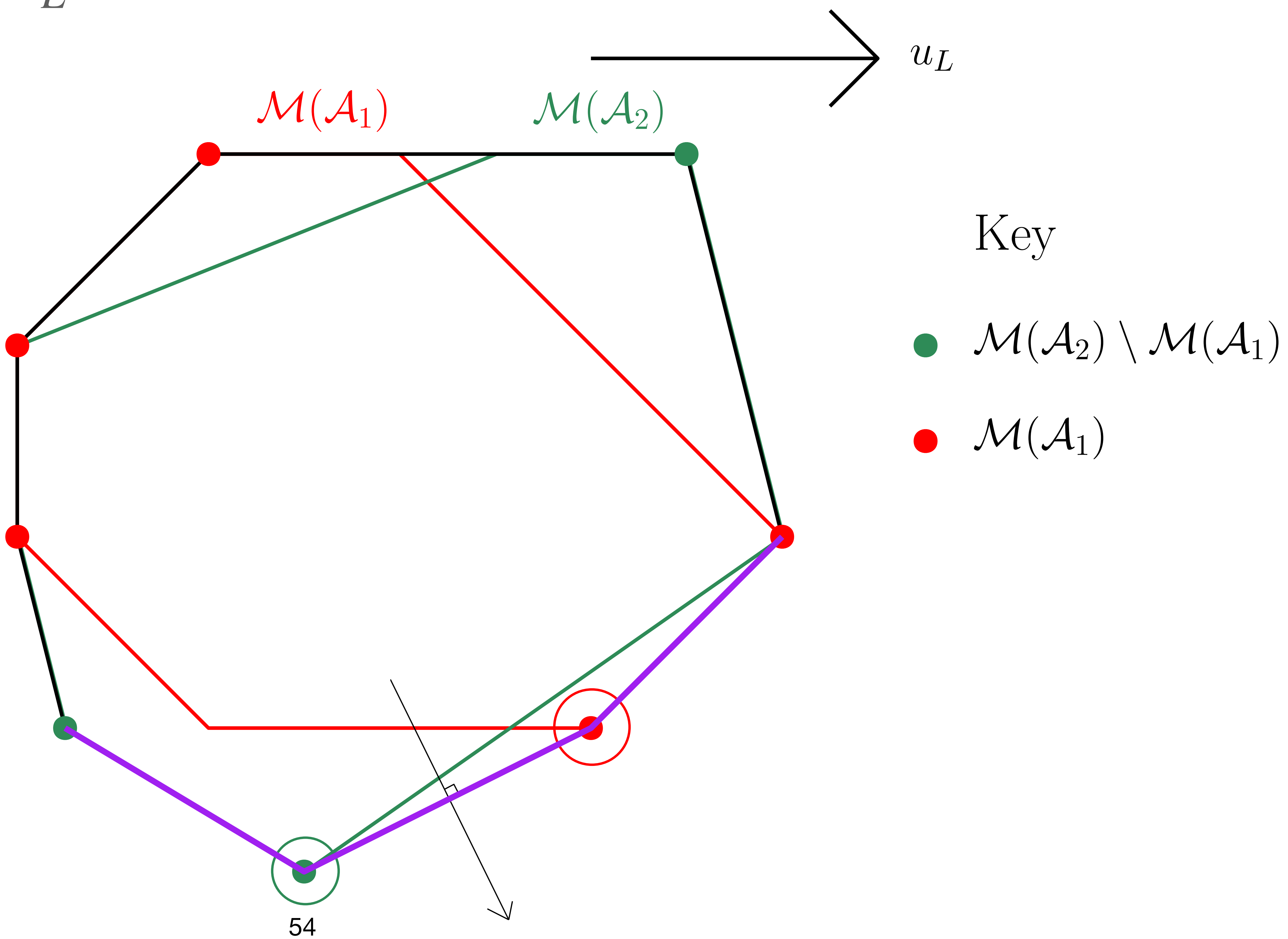
Menus: Proving Pareto-Optimality

1. Start with an “extra” vertex in M_2
2. Construct a path of strictly increasing u_L value
3. Find a “crossover” edge



Menus: Proving Pareto-Optimality

1. Start with an "extra" vertex in M_2
2. Construct a path of strictly increasing u_L value
3. Find a "crossover" edge



No-Swap-Regret

No-Swap-Regret

Characterization (implying Non-Manipulability)

Theorem : The menu of every NSR algorithm is the convex hull of all CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in \mathrm{BR}_L(x)$

Proof : Via showing that the optimizer always has a static best-response

No-Swap-Regret

Characterization (implying Non-Manipulability)

Proof : Via showing that the optimizer always has a static best-response

Consider the optimal transcript, and color based on learner actions

No-Swap-Regret

Characterization (implying Non-Manipulability)

Proof : Via showing that the optimizer always has a static best-response

Consider the optimal transcript, and color based on learner actions

Collect all the time steps, by action played (dividing fractionally on steps with mixed strategies)

No-Swap-Regret

Characterization (implying Non-Manipulability)

Proof : Via showing that the optimizer always has a static best-response

Collect all the time steps, by action played (dividing fractionally on steps with mixed strategies);

Record the optimizer marginals for each color

x_{blue}

x_{black}

x_{pink}

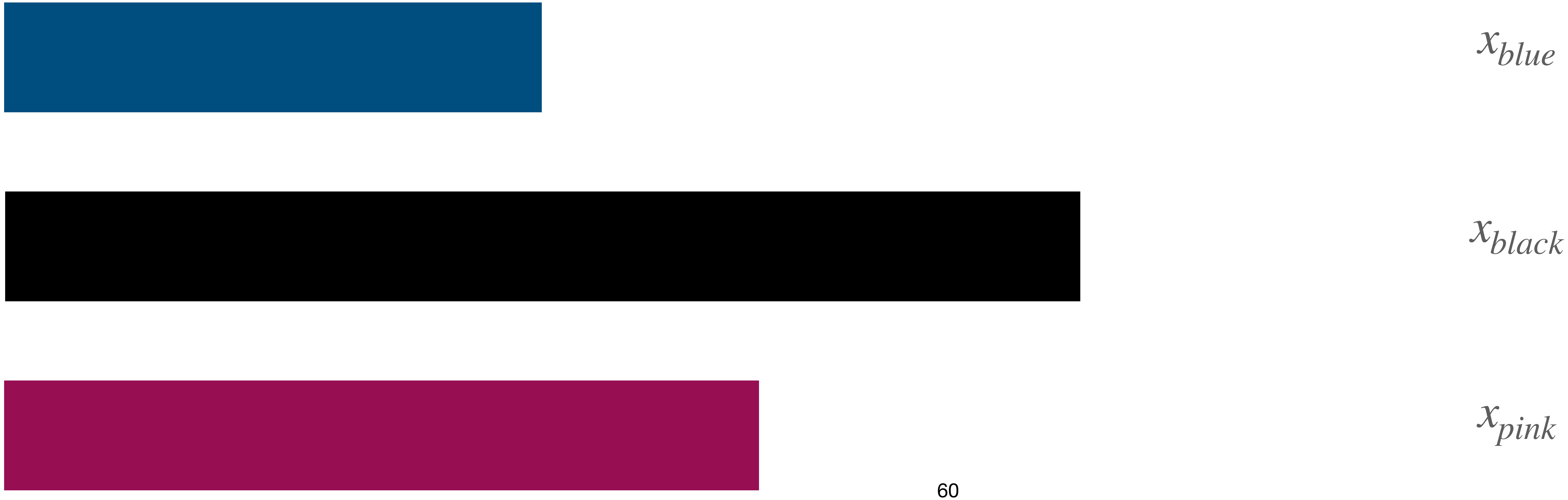
No-Swap-Regret

Characterization (implying Non-Manipulability)

Collect all the time steps, by action played (dividing fractionally on steps with mixed strategies);

Record the optimizer marginals for each color

No-Swap-Regret: Blue, black and pink are respectively best-responses to x_{blue}, x_{black} and x_{pink}



No-Swap-Regret

Characterization (implying Non-Manipulability)

No-Swap-Regret: Blue, black and pink are respectively best-responses to x_{blue}, x_{black} and x_{pink}

The optimal CSP is now a convex combination of CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in \mathrm{BR}_L(x)$

No-Swap-Regret

Characterization (implying Non-Manipulability)

No-Swap-Regret: Blue, black and pink are respectively best-responses to x_{blue}, x_{black} and x_{pink}

The optimal CSP is now a convex combination of CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in \mathrm{BR}_L(x)$

Might as well play a single distribution x and let the NSR learner learn a best-response to x

Reduces to the Stackelberg Equilibrium problem, solvable using m linear programs

No-Swap-Regret

Characterization (proving Pareto-Optimality)

Theorem : The menu of every NSR algorithm is the convex hull of all CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in \mathrm{BR}_L(x)$

With a little more effort, we can show that this menu is inclusion-minimal, with some additional characterization of valid menus, i.e., convex sets that can be realized by some learning algorithm.

FTRL is Pareto-dominated

Recall : FTRL

Only moves within $o(T)$ of being the historical best-response action get non-trivial, i.e., $\Omega_T(1)$ mass.

Given that R is continuous and strongly-convex, and $\eta_T = \dfrac{1}{o(T)}$:

$$y_t = \arg\max_{y \in \Delta^n} \left(\sum_{s=1}^{t-1} u_L(x_s, y) - \frac{R(y)}{\eta_T} \right)$$

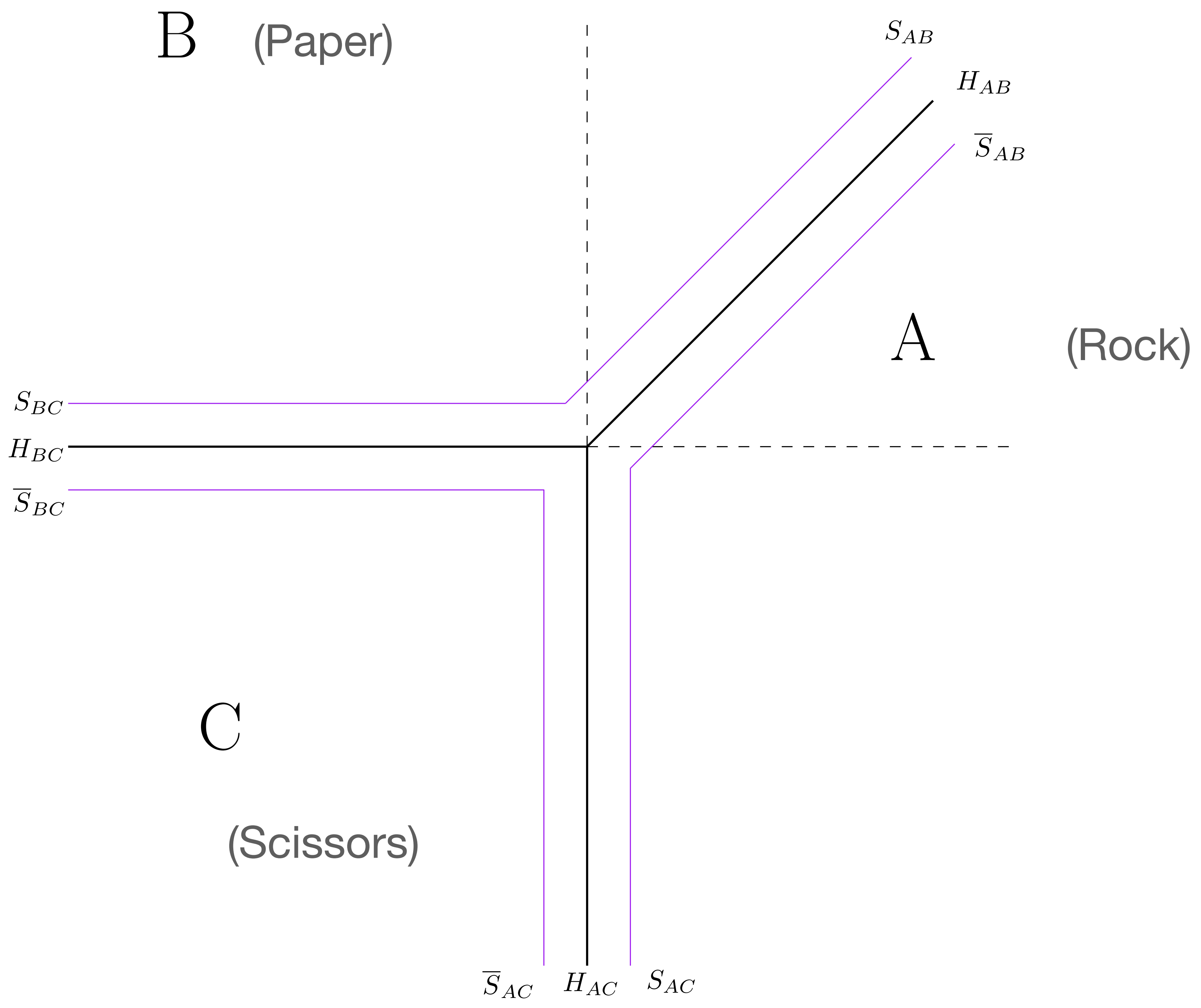
All Follow-the-Regularized Leader type algorithms, including Multiplicative Weights (Hedge), Online Gradient Descent are Mean-Based No-Regret Algorithms

Mean-Based Algorithms (FTRL)

Only moves within $o(T)$ of being the historical best-response action get non-trivial, i.e., $\Omega_T(1)$ mass.

Optimizer Sequence

……



Space of Cumulative Payoffs

FTRL is Pareto-dominated

What’s the smallest size-game in which we can prove this?

- The optimizer must have more than one action.
- The Learner must have more than 2 actions. Since No-Regret with two actions implies no-swap-regret.

We prove this for a non-degenerate set of 3×2 games.

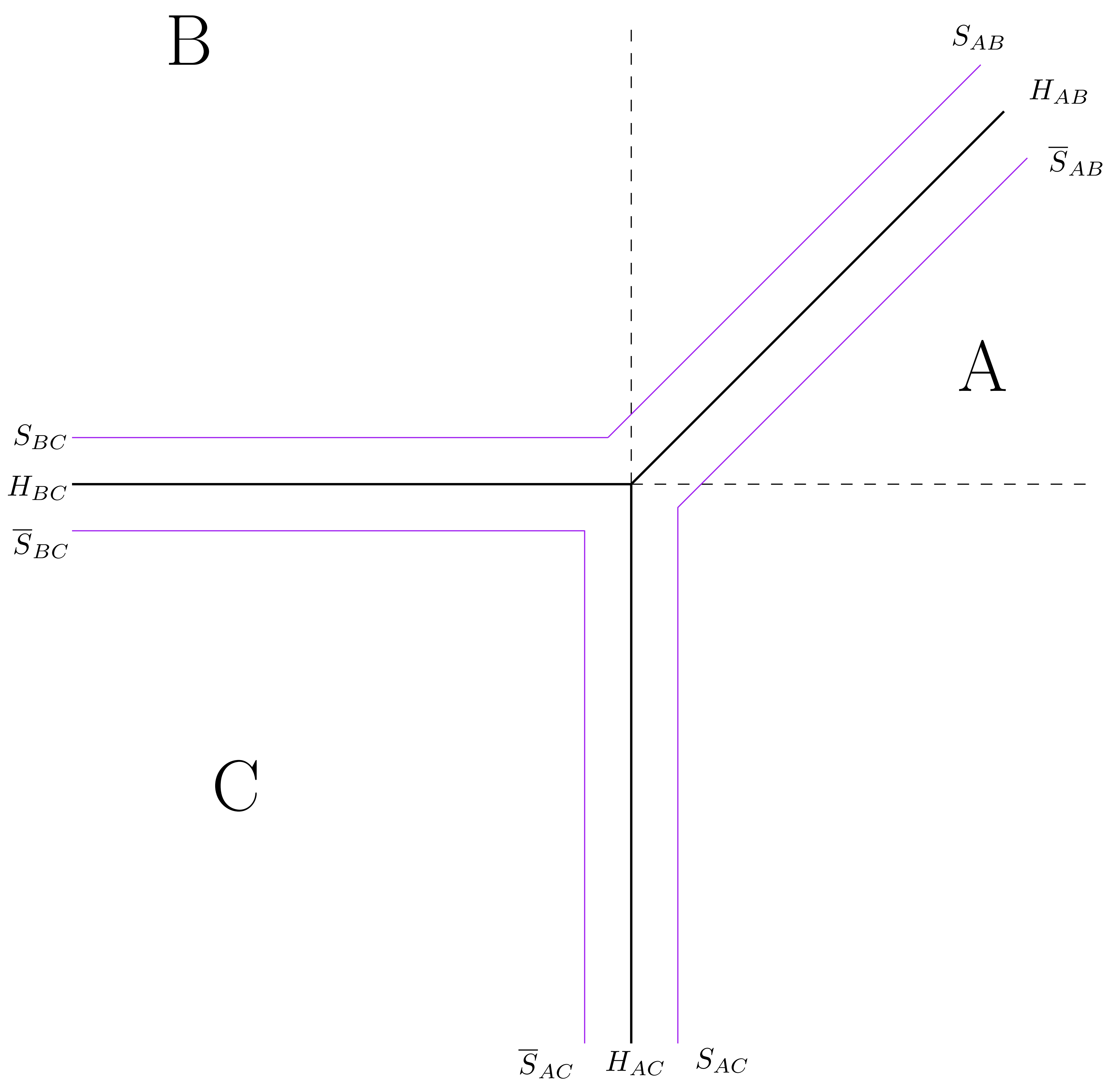
FTRL is Pareto-dominated

Theorem: All FTRL algorithms are Pareto-dominated.

Proof Sketch:

1. All FTRL algorithms induce the same menu
2. And the menu is a polytope with a succinct description (implicitly gives the optimizer their exact best response information).

All FTRL algorithms induce the same menu

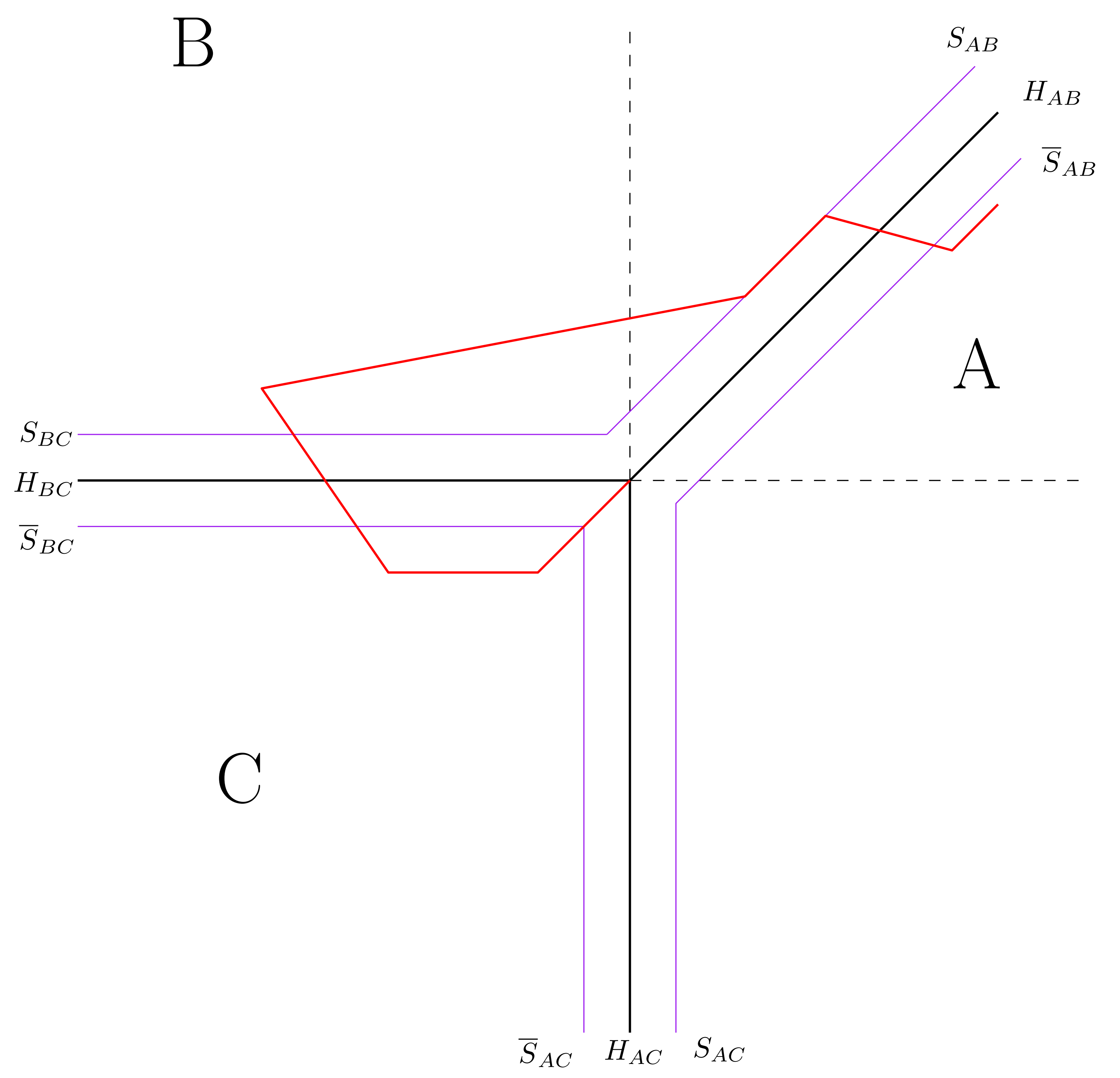


Cumulative Payoffs over time

All FTRL algorithms induce the same menu

Mean-Based Trajectories

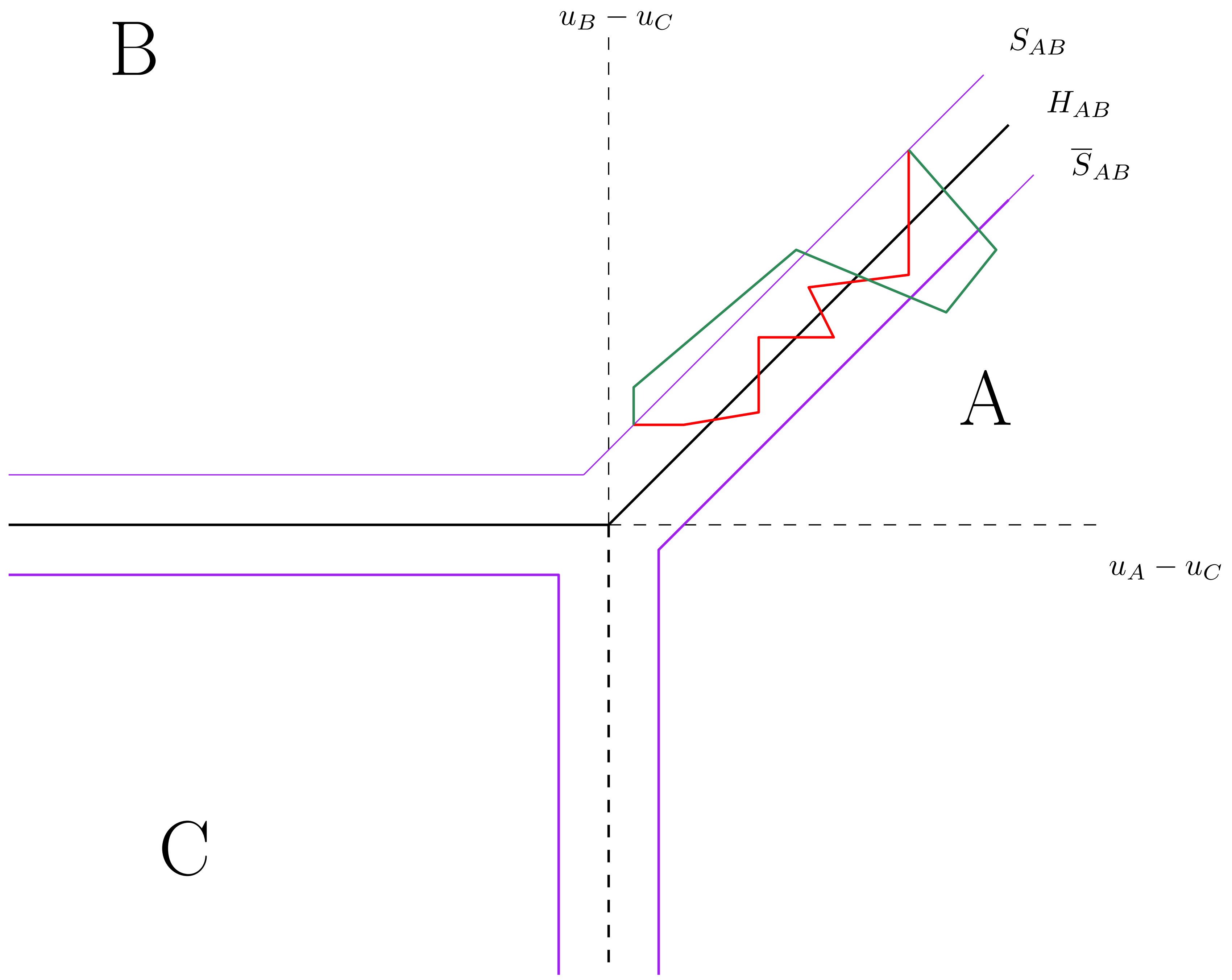
Trajectory has a ``clear" leader for all but o(T) time steps.



All FTRL algorithms induce the same menu

Mean-Based Trajectories

Convert arbitrary trajectories to mean-based trajectories.



Future Directions

Future Directions/ Related Work

Direction 1 : Auctions as repeated Bayesian Games

The learner receives a private context in each round drawn from a prior distribution, for eg., a click through rate prediction for an ad slot

- Mansour et al. (2022) show non-manipulability results via a notion of regret against policies mapping contexts to actions
- Kumar et al. [2024] show similar properties for Online Mirror Descent when used in repeated first price auctions

The Pareto-Optimality question remains open in this setting

Future Directions/ Related Work

Direction 2 : Repeated Auctions with a Budget

The learner has a total budget that they can spend, and must optimize spending, possibly based on private contexts


Meta

Research Publications Programs Datasets Careers

Pacing Equilibrium in First Price Auction Markets

Management Science

How do learning algorithms for the budget pacing problem fare against each other?

Thanks for Listening. Questions?