Pareto-Optimal Learning Algorithms for Repeated Games

Penn Theory Seminar

Eshwar Ram Arunachaleswaran, Natalie Collina, Jon Schneider
February 20, 2024

University of Pennsylvania, Google Research
1. Introduction

2. Model

3. Menus

4. No-Swap-Regret Algorithms are Pareto-Optimal

5. Multiplicative Weights (and friends) are Pareto-Dominated
Intro
What is a good algorithm to commit to in a repeated 2-player game? (Bimatrix game, linear payoff functions)
What is a good algorithm to commit to in a repeated 2-player game?

**Assumption**
The other player, called an optimizer, knows your algorithm and will best-respond (non-myopically).
What is a good algorithm to commit to in a repeated 2-player game?

**Full Information**

Knowing the optimizer’s payoff means we can design optimal algorithms to play with (Stackelberg).
What is a good algorithm to commit to in a repeated 2-player game?

**Assumption**
You do not know the optimizer’s payoffs.
What is a good algorithm to commit to in a repeated 2-player game?

Our Setting
Starting with no information with the other player, what is a reasonable guarantee to ask for?
What is a good algorithm to commit to in a repeated 2-player game?

Our Setting
Starting with no information with the other player, what is a reasonable guarantee to ask for?

Optimistic
Pointwise (over all optimizers) optimality
What is a good algorithm to commit to in a repeated 2-player game?

Our Setting
Starting with no information with the other player, what is a reasonable guarantee to ask for?

Pessimistic
The maximin value, on average.
What is a good algorithm to commit to in a repeated 2-player game?

**Our Setting**
Starting with no information with the other player, what is a reasonable guarantee to ask for?

**A Little Less Pessimistic**
Low Regret on every transcript.
What is a good algorithm to commit to in a repeated 2-player game?

Our Setting
Starting with no information with the other player, what is a reasonable guarantee to ask for?

Our answer
**Pareto-Optimality** (based on a Partial Ordering over Algorithms) and No-Regret.
Pareto Optimality

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

Figure 1: Space of Optimizer Payoffs: Three Scenarios
Main Results

- All No-Swap-Regret Algorithms are Pareto-optimal.
Main Results

• All No-Swap-Regret Algorithms are Pareto-optimal.
• Not all No-Regret algorithms are Pareto-optimal.
Main Results

- All No-Swap-Regret Algorithms are Pareto-optimal.
- 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.
Other Results/ Questions:

- A Geometric View of Algorithms
- A characterization of best-responses to a no-regret algorithm
- A characterization of Pareto-optimal No-Regret Algorithms
Model
Two players - Learner and Optimizer
In Each round

- The Learner has an action set $\Delta_n$
- The Optimizer has an action set $\Delta_m$
- They play actions $x_t, y_t$ in the $t$-th round
- Linear utility functions $u_L, u_O$
The Learner Perspective

Without seeing $u_0$, the Learner commits to an algorithm $\mathcal{A}$ mapping (deterministic) from histories of play of length $t - 1$ to distributions over actions $y_t$ in round $t$. 
The Learner Perspective

Without seeing $u_0$, the Learner commits to an algorithm $A$ mapping (deterministic) from histories of play of length $t - 1$ to distributions over actions $y_t$ in round $t$.

The resulting transcript of play is $(x_1, y_1), (x_2, y_2) \cdots (x_t, y_t)$.
No-Regret

Without seeing $u_o$, the Learner commits to an algorithm $\mathcal{A}$ mapping (deterministic) from histories of play of length $t - 1$ to distributions over actions $y_t$ in round $t$.

$$
\sum_{t=1}^{T} u_L(x_t, y_t) \geq \left( \max_{y^* \in [n]} \sum_{t=1}^{T} u_L(x_t, y^*) \right) - o(T).
$$
A learning algorithm $\mathcal{A}$ is a no-swap-regret algorithm if it is the case that, regardless of the sequence of actions $(x_1, x_2, \ldots, x_T)$ taken by the optimizer, the learner’s utility satisfies

$$\sum_{t=1}^{T} u_L(x_t, y_t) \geq \max_{\pi: [n] \rightarrow [n]} \sum_{t=1}^{T} u_L(x_t, \pi(y_t)) - o(T).$$
No-Regret and No-Swap-Regret algorithms are known to exist.
Only moves within $o(T)$ being the historical best-response action get non-trivial, i.e., $\Omega_T(1)$ mass.

**Figure 2:** Space of Cumulative Payoff Vectors
Model: Mean-Based Algorithms

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

Examples of Mean-Based Algorithms
MWU, FTPL, OGD are all mean-based.

Figure 3: Space of Cumulative Payoff Vectors
Given that $R$ is continuous and strongly-convex, and $\eta T = \frac{1}{o(T)}$:

$$y_t = \arg\max_{y \in \Delta^n} \left( \sum_{s=1}^{t-1} u_L(x_s, y) - \frac{R(y)}{\eta T} \right)$$
Model: Follow-the-Regularized-Leader (FTRL)

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

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

Examples of FTRL Algorithms

MWU, FTPL, OGD.
The Optimizer Perspective

With full information (payoffs, learner algorithm), the optimizer plays a best-response sequence
The Optimizer Perspective

With full information (payoffs, learner algorithm), the optimizer plays a best-response sequence \(^1\).

\[
x_1, x_2 \cdots x_T \in \arg \max_{(x_1, x_2 \cdots x_T) \in \Delta_m^T} \frac{1}{T} \sum_{t=1}^{T} u_O(x_t, y_t)
\]

where \(y_t = \mathcal{A}(x_1, x_2 \cdots x_{t-1})\)

\(^1\)Tie-breaking in favor of the learner.
The Optimizer Perspective

With full information (payoffs, learner algorithm), the optimizer plays a best-response sequence of actions\(^2\), \(^3\).

\[
x_1, x_2 \cdots x_T \in \arg \max_{(x_1, x_2 \cdots x_T) \in \Delta_m^T} \frac{1}{T} \sum_{t=1}^{T} u_O(x_t, y_t)
\]

where \(y_t = \mathcal{A}(x_1, x_2 \cdots x_{t-1})\)

\(^2\)Tie-breaking in favor of the learner.

\(^3\)Cheating slightly here!
The Optimizer Perspective

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

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

The learner gets payoff \( V_L(A, u_O, T) = \frac{1}{T} \sum_{t=1}^{T} u_O(x_t, y_t) \)
Limit Payoffs

- The learner’s limit payoff is \( V_L(A, u_0) = \lim_{T \to \infty} V_L(A, u_0, T) \).
Limit Payoffs

• The learner’s limit payoff is $V_L(A, u_O) = \lim_{T \to \infty} V_L(A, u_O, T)$.
• Motivation: Do not care about $o_T(1)$ differences in average payoff.
Algorithm $\mathcal{A}$ dominates algorithm $\mathcal{B}$ for some payoff $u_L$ if:

1. For all $\mu_0 : V_L(\mathcal{A}, u_0) \geq V_L(\mathcal{B}, u_0)$.
2. $\exists \mu_0$ s.t. $V_L(\mathcal{A}, u_0) > V_L(\mathcal{B}, u_0)$.

---

The algorithms do equally well
Algorithm $\mathcal{A}$ does better
Algorithm $\mathcal{B}$ does better

---

\(^4\)In fact equivalent to a positive measure set
Algorithm \( A \) dominates algorithm \( B \) for some payoff \( u_L \) if:

- For all \( \mu_O : V_L(A, u_O) \geq V_L(B, u_O) \).
- \( \exists \mu_O \) s.t. \( V_L(A, u_O) > V_L(B, u_O) \) \(^5\).

All our Pareto-domination results are for a positive-measure set of learner payoffs.

\(^5\)In fact equivalent to a positive measure set
### Pareto-optimality of Algorithms

Algorithm \( A \) is Pareto-optimal if it is not Pareto-dominated by any other algorithm \( B \).
Related Work

- Learning in Games - [BSV24], [DSS19], [MMSS22]
- Stackelberg Equilibria in Repeated Games - [CAK23], [HLNW22]


Nika Haghtalab, Thodoris Lykouris, Sloan Nietert, and Alexander Wei.

**Learning in stackelberg games with non-myopic agents.**

Yishay Mansour, Mehryar Mohri, Jon Schneider, and Balasubramanian Sivan.

**Strategizing against learners in bayesian games.**
Menus
Correlated Strategy Pairs (CSPs)
Consider all possible distribution of action pairs generated over sequences over optimizers.
Correlated Strategy Pairs (CSPs)
Consider all possible distribution of action pairs generated over sequences over optimizers.

\[ \{ \varphi \in \Delta_{mn} : \exists x_1, x_2 \cdots x_T \text{ s.t. } \varphi = \frac{1}{T} \sum_{t=1}^{T} x_t \otimes y_t \} \]
Correlated Strategy Pairs (CSPs)

Consider all possible distribution of action pairs generated over sequences over optimizers.

\[
\left\{ \phi \in \Delta_{mn} : \exists x_1, x_2 \cdots x_T \text{ s.t. } \phi = \frac{1}{T} \sum_{t=1}^{T} x_t \otimes y_t \right\}
\]

Take their convex hull and call this set the menu \( \mathcal{M}(A_T) \).
Correlated Strategy Pairs (CSPs)

Consider all possible distribution of action pairs generated over sequences over optimizers.

Take their convex hull and call this set the menu $\mathcal{M}(A_T)$.

Figure 4: A Simple Menu
Correlated Strategy Pairs (CSPs)

- Consider all possible distribution of action pairs generated over sequences over optimizers.
- Take their convex hull and call this set the menu $\mathcal{M}(A_T)$.

**Figure 5:** An Optimizer’s Choice on a Simple Menu
Recall that the learner’s limit payoff is

$$V_L(A, u_0) = \lim_{T \to \infty} V_L(A, u_0, T).$$
• Recall that the learner’s limit payoff is 
  \[ V_L(A, u_O) = \lim_{T \to \infty} V_L(A, u_O, T). \]
• So, we would have to optimize over an infinite sequence of menus and take the limit.
• Recall that the learner’s limit payoff is
\[ V_L(\mathcal{A}, u_0) = \lim_{T \to \infty} V_L(\mathcal{A}, u_0, T). \]
• So, we would have to optimize over an infinite sequence of menus and take the limit.
• Instead, take the limit menu and optimize over it!
• So, we would have to optimize over an infinite sequence of menus and take the limit.
• Instead, take the limit menu and optimize over it!
• The limit menu is defined as \( \mathcal{M}(\mathcal{A}) = \lim_{T \to \infty} \mathcal{M}(\mathcal{A}_T) \).

Figure 6: An Optimizer’s Choice on the limit Menu
Comparing two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ for a given $u_0$:

**Key Idea**

The learner (and optimizer) payoffs can be entirely inferred from the limit menus.
Comparing two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ for a given $u_0$:

**Key Idea**

Algorithms can be replaced by their limit menus while discussing Pareto-domination (and optimality).
Menus: Examples

\[
\begin{bmatrix}
A & B \\
P & X & X \\
Q & X & X \\
\end{bmatrix}
\]
Menus: Examples

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Learning Algorithm $\mathcal{A}_1$: Always play P
Learning Algorithm $\mathcal{A}_1$: Always play $P$

$$(A \otimes P) \quad (B \otimes P)$$
Learning Algorithm $\mathcal{A}_2$: Play Q as long as the Optimizer has always played A. Otherwise, play P.
Learning Algorithm $\mathcal{A}_2$: Play Q as long as the Optimizer has always played A. Otherwise, play P.
What do menus look like in general?
What do menus look like in general?

**Approachable Sets**

A set $S$ is approachable if, for every $x \in \Delta_m$, there exists a $y \in \Delta_n$ such that $x \otimes y \in S$. 
## Approachable Sets

A set $S$ is approachable if, for every $x \in \Delta_m$, there exists a $y \in \Delta_n$ such that $x \otimes y \in S$.

## Theorem

A closed, convex subset $\mathcal{M} \subseteq \Delta_{mn}$ is an limit menu iff it is approachable.
Approachable Sets

A set $S$ is approachable if, for every $x \in \Delta_m$, there exists a $y \in \Delta_n$ such that $x \otimes y \in S$.

- For every convex approachable set $S$, there is some $\mathcal{M} \subseteq S$ which is a valid menu.
Approachable Sets
A set $S$ is approachable if, for every $x \in \Delta_m$, there exists a $y \in \Delta_n$ such that $x \otimes y \in S$.

- For every convex approachable set $S$, there is some $\mathcal{M} \subseteq S$ which is a valid menu
- Menus are Upwards-Closed
Upwards Closedness
Menu Properties

Approachable Sets
A set $S$ is approachable if, for every $x \in \Delta_m$, there exists a $y \in \Delta_n$ such that $x \otimes y \in S$.

- For every convex approachable set $S$, there is some $\mathcal{M} \subseteq S$ which is a valid menu
- Menus are Upwards-Closed

Putting these together:
Every approachable set $S$ is a valid menu
Approachable Sets
A set $S$ is approachable if, for every $x \in \Delta_m$, there exists a $y \in \Delta_n$ such that $x \otimes y \in S$.

Theorem
A closed, convex subset $\mathcal{M} \subseteq \Delta_{mn}$ is an limit menu iff it is approachable.
No(-Swap)-Regret Redux

No(-Swap)-Regret is a property of just the CSPs.
No(-Swap)-Regret Redux

No(-Swap)-Regret is a property of just the CSPs.

\[
\sum_{t=1}^{T} u_L(x_t, y_t) \geq \max_{\pi: [n] \rightarrow [n]} \sum_{t=1}^{T} u_L(x_t, \pi(y_t)).
\]
No(-Swap)-Regret Redux

No(-Swap)-Regret is a property of just the CSPs.

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

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

where $\varphi = \frac{1}{T} \sum_{t=1}^{T} x_t \otimes y_t$. 
A natural set of CSPs vis-a-vis no-regret:

$\mathcal{M}_{NSR}$ is the set of all CSPs that are no-swap-regret.
A natural set of CSPs vis-a-vis no-regret:

$\mathcal{M}_{NSR}$ is the set of all CSPs that are no-swap-regret.

**Observation**

$\mathcal{M}_{NSR}$ is a polytope.
A natural set of CSPs vis-a-vis no-regret:

\( \mathcal{M}_{NR} \) is the set of all CSPs that are no-regret.

**Observation**

The limit menu \( \mathcal{M} \) of any no-swap-regret algorithm is contained in \( \mathcal{M}_{NSR} \).
Theorem
All no-swap-regret algorithms $\mathcal{A}$ have the same limit menu, which is $\mathcal{M}_{NSR}$.
Third Main Result

**Theorem**

All no-swap-regret algorithms $\mathcal{A}$ have the same limit menu, which is $\mathcal{M}_{NSR}$.

Particularly interesting in the context of multiple, seemingly different, approaches to NSR algorithms.
No-Swap-Regret Algorithms are Pareto-Optimal
**Theorem**

All no-swap-regret algorithms $A$ have the same limit menu, which is $M_{NSR}$. 
No Swap Regret Algorithms are Pareto-Optimal

**Theorem**
All no-swap-regret algorithms $\mathcal{A}$ have the same limit menu, which is $\mathcal{M}_{NSR}$.

**Theorem: $\mathcal{M}_{NSR}$ Characterization**
$\mathcal{M}_{NSR}$ is the convex hull of all CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in BR(x)$.

![Diagram](attachment:image_url)
Theorem
All no-swap-regret algorithms \( \mathcal{A} \) have the same limit menu, which is \( M_{NSR} \).

Theorem: \( M_{NSR} \) Characterization
\( M_{NSR} \) is the convex hull of all CSPs of the form \( x \otimes y \), with \( x \in \Delta_m \) and \( y \in BR_L(x) \).

Theorem: \( M_{NSR} \) Minimality
\( M_{NSR} \) is inclusion-minimal and includes \( \varphi^+ \).
Theorem
All no-swap-regret algorithms $\mathcal{A}$ have the same limit menu, which is $\mathcal{M}_\text{NSR}$.

Theorem: $\mathcal{M}_\text{NSR}$ Characterization
$\mathcal{M}_\text{NSR}$ is the convex hull of all CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in \text{BR}_L(x)$.

Theorem: $\mathcal{M}_\text{NSR}$ Minimality
$\mathcal{M}_\text{NSR}$ is inclusion-minimal and includes $\varphi^+$.

Theorem: $\varphi^+$-minimality implies optimality
Every inclusion-minimal menu that contains $u_L^+$ is pareto-optimal.
No Swap Regret Algorithms are Pareto-Optimal

Theorem
All no-swap-regret algorithms $\mathcal{A}$ have the same limit menu, which is $\mathcal{M}_{NSR}$.

Theorem: $\mathcal{M}_{NSR}$ Characterization
$\mathcal{M}_{NSR}$ is the convex hull of all CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in BR_L(x)$.

Theorem: $\mathcal{M}_{NSR}$ Minimality
$\mathcal{M}_{NSR}$ is inclusion-minimal and includes $\varphi^+$. 

Theorem: $\varphi^+$-minimality implies optimality
Every inclusion-minimal menu that contains $u_L^+$ is pareto-optimal.
Definition: Inclusion-Minimality
A menu $\mathcal{M}_1$ is *inclusion-minimal* if there is no menu $\mathcal{M}_2$ such that $\mathcal{M}_2 \subsetneq \mathcal{M}_1$.

Definition: $\varphi^+$
$u_L^+ = x^* \otimes y^*$, where $(x^*, y^*) = \arg \max_{(x, y)} u_L(x, y)$. 
Recall: $\mathcal{M}_{NSR}$ is the convex hull of all CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in BR_L(x)$. 
Recall: $\mathcal{M}_{NSR}$ is the convex hull of all CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in BR_L(x)$. 

$\mathcal{M}_{NSR}$ includes $\varphi^+$
No Swap Regret Algorithms are Pareto-Optimal

Theorem
All no-swap-regret algorithms $\mathcal{A}$ have the same limit menu, which is $\mathcal{M}_{NSR}$.

Theorem: $\mathcal{M}_{NSR}$ Characterization
$\mathcal{M}_{NSR}$ is the convex hull of all CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in BR_L(x)$.

Theorem: $\mathcal{M}_{NSR}$ Minimality
$\mathcal{M}_{NSR}$ is inclusion-minimal and includes $\varphi^+$.

Theorem: $\varphi^+$-minimality implies optimality
Every inclusion-minimal menu that contains $u^+_L$ is pareto-optimal.
Theorem
All no-swap-regret algorithms $\mathcal{A}$ have the same limit menu, which is $\mathcal{M}_{NSR}$.

**Theorem: $\mathcal{M}_{NSR}$ Characterization**
$\mathcal{M}_{NSR}$ is the convex hull of all CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in BR_L(x)$.

**Theorem: $\mathcal{M}_{NSR}$ Minimality**
$\mathcal{M}_{NSR}$ is inclusion-minimal and includes $\varphi^+$. 

**Theorem: $\varphi^+$-minimality implies optimality**
Every inclusion-minimal menu that contains $u_L^+$ is pareto-optimal.
\(\varphi^+\)-minimality implies pareto-optimality

Sufficient to prove:

**Lemma**
If \(\mathcal{M}_1\) contains \(\varphi^+\) and \(\mathcal{M}_2 \setminus \mathcal{M}_1 \neq \emptyset\), then there is an Optimizer payoff \(u_0\) such that

\[
V_L(\mathcal{M}_1, u_0) > V_L(\mathcal{M}_2, u_0)
\]
Lemma
If $\mathcal{M}_1$ contains $\varphi^+$ and $\mathcal{M}_2 \setminus \mathcal{M}_1 \neq \emptyset$, then there is an Optimizer payoff $u_0$ such that

$$V_L(\mathcal{M}_1, u_0) > V_L(\mathcal{M}_2, u_0)$$

Proof:
Two cases:

- $\mathcal{M}_2$ does not contain $\varphi^+$
Lemma
If $M_1$ contains $\varphi^+$ and $M_2 \setminus M_1 \neq \emptyset$, then there is an Optimizer payoff $u_0$ such that

$$V_L(M_1, u_0) > V_L(M_2, u_0)$$

Proof:
Two cases:

- $M_2$ does not contain $\varphi^+$ (easy)
Lemma
If $\mathcal{M}_1$ contains $\varphi^+$ and $\mathcal{M}_2 \setminus \mathcal{M}_1 \neq \emptyset$, then there is an Optimizer payoff $u_0$ such that

$$V_L(\mathcal{M}_1, u_0) > V_L(\mathcal{M}_2, u_0)$$

Proof:
Two cases:

- $\mathcal{M}_2$ does not contain $\varphi^+$ (easy)
- $\mathcal{M}_2$ does contain $\varphi^+$
Lemma

If \( M_1 \) contains \( \varphi^+ \) and \( M_2 \setminus M_1 \neq \emptyset \), then there is an Optimizer payoff \( u_0 \) such that

\[
V_L(M_1, u_0) > V_L(M_2, u_0)
\]

Proof:

Two cases:

- \( M_2 \) does not contain \( \varphi^+ \) (easy)
- \( M_2 \) does contain \( \varphi^+ \) (a little trickier)
Lemma
If $\mathcal{M}_1$ contains $\varphi^+$ and $\mathcal{M}_2 \setminus \mathcal{M}_1 \neq \emptyset$, then there is an Optimizer payoff $u_0$ such that

$$V_L(\mathcal{M}_1, u_0) > V_L(\mathcal{M}_2, u_0)$$

Special Case
Both Menus are Polytopes.
\( \varphi^+ \)-minimality implies pareto-optimality

**Lemma**

If \( \mathcal{M}_1 \) contains \( \varphi^+ \) and \( \mathcal{M}_2 \setminus \mathcal{M}_1 \neq \emptyset \), then there is an Optimizer payoff \( u_0 \) such that

\[
V_L(\mathcal{M}_1, u_0) > V_L(\mathcal{M}_2, u_0)
\]
Lemma
If $M_1$ contains $\varphi^+$ and $M_2 \setminus M_1 \neq \emptyset$, then there is an Optimizer payoff $u_0$ such that

$$V_L(M_1, u_0) > V_L(M_2, u_0)$$
Lemma
If $\mathcal{M}_1$ contains $\varphi^+$ and $\mathcal{M}_2 \setminus \mathcal{M}_1 \neq \emptyset$, then there is an Optimizer payoff $u_0$ such that

$$V_L(\mathcal{M}_1, u_0) > V_L(\mathcal{M}_2, u_0)$$
\( \varphi^+ \)-minimality implies pareto-optimality

Take the convex hull of the union.
$\varphi^+$-minimality implies pareto-optimality

Take the convex hull of the union.
\( \varphi^+-\text{minimality implies pareto-optimality} \)

- Start with an “extra” vertex in \( \mathcal{M}(A_2) \).
\( \phi^+ \)-minimality implies pareto-optimality

- Start with an “extra” vertex in \( M(A_2) \).
- Construct a path of strictly increasing \( u_L \) value.

**Key**

- \( M(A_2) \setminus M(A_1) \)
- \( M(A_1) \)
\( \varphi^+ \)-minimality implies pareto-optimality

- Start with an “extra” vertex in \( M(A_2) \).
- Construct a path of strictly increasing \( u_L \) value.
- Find a crossover edge.
\( \varphi^+ \)-minimality implies pareto-optimality

- Start with an “extra” vertex in \( M(A_2) \).
- Construct a path of strictly increasing \( u_L \) value.
- Find a crossover edge.

Key

- \( M(A_2) \setminus M(A_1) \)
- \( M(A_1) \)
\(\varphi^+\)-minimality implies pareto-optimality

- Start with an “extra” vertex in \(\mathcal{M}(A_2)\).
- Construct a path of strictly increasing \(u_L\) value.
- Find a crossover edge.
Multiplicative Weights (and friends) are Pareto-Dominated
**Theorem**

*All FTRL algorithms are Pareto-dominated.*
Theorem
All FTRL algorithms are Pareto-dominated.

- What’s the smallest-size game in which we can hope to prove this?
Theorem
All FTRL algorithms are Pareto-dominated.

- What’s the smallest-size game in which we can hope to prove this?
- The optimizer must have more than one action.
Theorem
All FTRL algorithms are Pareto-dominated.

• What’s the smallest-size game in which we can hope to prove this?
• The optimizer must have more than one action.
• The Learner must have more than 2 actions.
Theorem
All FTRL algorithms are Pareto-dominated.

- What’s the smallest-size game in which we can hope to 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.
Theorem
All FTRL algorithms are Pareto-dominated.

- What’s the smallest-size game in which we can hope to 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 $3 \times 2$ games.
**Theorem**

All FTRL algorithms are Pareto-dominated.

**Proof Sketch**

- All FTRL algorithms induce the same menu.
- And the menu is a polytope (with a succinct description) \(^6\)

\[^6\text{implicitly gives the optimizer their exact best response information}\]
All FTRL algorithms induce the same menu

Figure 7: Space of Cumulative Payoffs
All FTRL algorithms induce the same menu

Figure 8: Space of Cumulative Payoffs
All FTRL algorithms induce the same menu

"Mean-Based" Trajectory

Trajectory has a “clear” leader for all but $o(T)$ time steps.
All FTRL algorithms induce the same menu

"Mean-Based" Trajectory
Convert arbitrary trajectories to mean-based trajectories.

Figure 10: Space of Cumulative Payoffs
Oh No I Stopped Listening!!!

• hi
Oh No I Stopped Listening!!!

• hi
• it’s not too late
Oh No I Stopped Listening!!!

• hi
• it’s not too late
• here’s what we want you to know
Takeaways

• Pareto-Optimality
• Menus
Takeaways

- Pareto-Optimality
  - Incomparable with No-Regret
- Menus
Takeaways

- Pareto-Optimality
  - Incomparable with No-Regret
  - No-Swap-Regret Algorithms are Pareto-Optimal
- Menus
Takeaways

• Pareto-Optimality
  • Incomparable with No-Regret
  • No-Swap-Regret Algorithms are Pareto-Optimal

• Menus
  • Progress towards understanding FTRL
Takeaways

- Pareto-Optimality
  - Incomparable with No-Regret
  - No-Swap-Regret Algorithms are Pareto-Optimal
- Menus
  - Progress towards understanding FTRL
  - A new paradigm for algorithm design
Thank you!

Figure 11: Us, being happy you listened to our talk