Menus: A Framework for Learning Against Strategic Opponents

ESHWAR RAM ARUNACHALESWARAN
SESCO Enterprises
and
NATALIE COLLINA
University of Pennsylvania
and
YISHAY MANSOUR
Google Research
and
MEHRYAR MOHRI
Google Research
and
BALASUBRAMANIAN SIVAN
Google Research
and
JON SCHNEIDER
Google Research

Algorithms increasingly mediate repeated strategic interactions in marketplaces, from automated pricing to auction bidding. When one party commits to a learning algorithm, the other party can respond strategically over time by steering the algorithm's internal state toward a favorable long-run outcome. This note surveys a line of work that studies this "learning-as-commitment" perspective via a geometric object we call a *menu*: the convex set of long-run outcomes an opponent can induce against a fixed learning rule. Menus provide a common language for (i) comparing learning algorithms against strategic opponents, (ii) optimizing over learning rules under uncertainty about opponent objectives, and (iii) characterizing when an opponent can manipulate learning dynamics beyond what they could achieve with a static strategy. Using this machinery, we converge upon no-swap-regret algorithms as an "optimal" commitment strategy for robust learning against a strategic opponent. We also identify principled generalizations of no-swap-regret beyond normal-form games that preserve the same strategic guarantees while remaining computationally tractable.

Categories and Subject Descriptors: [**Theory of Computation**]: Algorithmic Game Theory and Mechanism Design

General Terms: Algorithms, Economics, Theory

Additional Key Words and Phrases: Online Learning, Swap Regret, Strategizing with Algorithms

Authors' addresses: eshwarram.arunachaleswaran@gmail.com, ncollina@seas.upenn.edu

1. INTRODUCTION

Algorithms are increasingly used to make repeated decisions in strategic environments. A canonical example is automated pricing: two sellers may repeatedly compete to sell an identical item, with the lower posted price winning the sale (a Bertrand-style competition). In principle, sellers could adjust prices manually, checking in periodically to respond to market conditions. In practice, many rely on automated pricing tools that update based on observed competitor prices. Related examples include automated bidders in auction markets and algorithmic agents in platform-mediated interactions.

When a firm deploys a learning algorithm, it is not merely choosing a sequence of actions. It is committing to a *policy* that maps the evolving interaction history to future behavior. This changes the strategic problem faced by the opposing party: a sophisticated opponent can choose time-varying actions designed to exploit the learning rule itself. For instance, if a pricing algorithm undercuts competitors but "resets" when prices dip too low (a common heuristic), an informed competitor can deliberately trigger the reset and then exploit the resulting price increase. In such interactions, the opponent can steer the learner's dynamics to ensure their own long-term success. This raises a basic design question:

Design problem. If you are deploying a learning algorithm in a repeated, general-sum game, how should you design it to be robust to an opponent who responds strategically over time?

Setup We consider a repeated two-player normal-form game, where in each round t, the first player, called the learner, select a distribution $x_t \in \Delta^m$ over m pure actions, and the second player, called the optimizer, selects a distribution $y_t \in \Delta^n$ over n pure actions. The players receive bilinear utilities $u_L(x_t, y_t)$ and $u_O(x_t, y_t)$–in other words, their utilities are only a function of the probability that each move pair occurs, a linear function applied to $x_t \otimes y_t$. The learner's strategy is an algorithm $\mathcal{A}$ that maps the history of play so far $H_t = (x_1, y_1), (x_2, y_2) \cdots (x_{t-1}, y_{t-1})$ to an action x_t for the next round. The optimizer selects a sequence of actions[1] $y_{1:T}$ strategically in order to maximize their own long-term utility. We let $\mathcal{A}(y_{1:T})$ denote the move pair distribution induced by the optimizer playing $y_{1:T}$ against $\mathcal{A}$.

Definition 1.1 *Optimizer best-response.* We consider an optimizer who, when faced with the algorithm $\mathcal{A}$, plays the sequence

$$\mathsf{BR}(\mathcal{A}, u_O) = \arg\max_{y_{1:T} \in \mathcal{Y}^T} u_O(\mathcal{A}(y_{1:T}))$$

Here we omit details on tie-breaking[2].

What is a good algorithm for a learner to deploy in such an environment? A natural starting point is the standard learning-theoretic solution concept of regret

[1] In fact, the optimizer may choose to play their own adaptive strategy instead of a fixed sequence; however, in this note we primarily discuss learner algorithms which do not have correlated randomness, against which there is no need for the optimizer to play adaptively. Thus, for simplicitly of notation, we will write the optimizer's strategy as a sequence rather than an algorithm.

[2] A careful formulation of tie-breaking is needed to ensure continuity of the *learner's* outcomes, see [Arunachaleswaran et al. 2024] for a rigorous treatment

minimization. The (external) regret of the learner's realized play is

$$\text{Regret}_T := \max_{x \in \mathcal{X}} \sum_{t=1}^{T} u_L(x, y_t) - \sum_{t=1}^{T} u_L(x_t, y_t),$$

and we say the algorithm is *no-regret* if $\text{Regret}_T = o(T)$ (equivalently, average regret vanishes).

A stronger notion is *swap regret*, which compares to the best *action remapping* $\phi : \mathcal{X} \to \mathcal{X}$ applied to the learner's realized actions:

$$\text{SwapRegret}_T := \max_{\phi: \mathcal{X} \to \mathcal{X}} \sum_{t=1}^{T} u_L(\phi(x_t), y_t) - \sum_{t=1}^{T} u_L(x_t, y_t),$$

and we say the algorithm has *no swap regret* if $\text{SwapRegret}_T = o(T)$.

These guarantees may seem strong, but they are achievable: there are online algorithms with vanishing regret against *any* (even adaptive) sequence of opponent actions. In a repeated *zero-sum* game, playing a no-regret algorithm is essentially the right strategy: a strategic opponent can always hold the learner to its minimax value.

However, many market interactions are *general-sum*. Here the opponent is not trying to minimize the learner's utility, they are trying to maximize their own. In this setting, regret guarantees can become a poor proxy for strategic robustness. A vivid illustration appears in [Braverman et al. 2018], which shows that in a repeated auction with a mean-based, no-regret buyer, a strategic seller can extract essentially full surplus. The message is that the strategic guarantees of many no-regret algorithms are too weak when an opponent optimizes *against the algorithm*.

So what should replace regret as the organizing principle? Across a series of papers [Arunachaleswaran et al. 2024; 2025; Arunachaleswaran et al. 2025], we study this question under varying information regimes. A common technical tool is a geometric abstraction that makes the commitment aspect explicit: *menus of algorithms*.

2. MENUS OF ALGORITHMS.

The central theme of this letter is a view of algorithms in terms of the geometric set of all outcomes they can induce, an object which we call an algorithm's "menu".

Definition 2.1 *Finite-time menu.* For a horizon dependent algorithm $\mathcal{A}^T$, we define its menu $\mathcal{M}(\mathcal{A}^T)$ as the convex hull of all vectors of the form $\frac{1}{T}\sum_t x_t \otimes y_t$ for all possible optimizer sequences $y_1, y_2 \dots y_T$ and the corresponding induced sequences $x_1, x_2..x_T$ of the algorithm, i.e. $x_t = \mathcal{A}^T(H_t)$.

We refer to such vectors as Correlated Strategy Profiles or CSPs. CSPS have a clean interpretation, at least in normal-form games: they are a distribution over the learner's and optimizer's joint actions.

Definition 2.2 *Correlated Strategy Profile (CSP).* A CSP is a vector of the form $\frac{1}{T}\sum_t x_t \otimes y_t$.

If a learning algorithm $\mathcal{A}$, which we define as a composition of horizon dependent algorithms $\mathcal{A}^1, \mathcal{A}^2 \dots$, satisfies the property that its menus $\mathcal{M}(\mathcal{A}^1), \mathcal{M}(\mathcal{A}^2) \dots$

converge under the Hausdorff metric to some set $\mathcal{M}$, we define its menu $\mathcal{M}(\mathcal{A})$ to be the limit set[3] $\mathcal{M}$ (see [Arunachaleswaran et al. 2024] for a rigorous treatment).

Definition 2.3 *Menu.* For a consistent algorithm $\mathcal{A} = \mathcal{A}^1, \mathcal{A}^2 \ldots$, we define its menu $\mathcal{M}(\mathcal{A})$ as $\lim_{T\to\infty} \mathcal{M}(\mathcal{A}^T)$

A set $\mathcal{M} \subseteq \Delta^{mn}$ is said to be a menu if and only if there exists a consistent algorithm $\mathcal{A}$ such that $\mathcal{M}$ is the menu of $\mathcal{A}$. To elucidate this idea, we now provide a few examples of algorithms and their corresponding menus.

2.1 Example Menus

Warm-up example 1. Consider the learning algorithm $\mathcal{A}^*$ that ignores the interaction history and plays x^* on every round:

$$\mathcal{A}^*_t(h_{t-1}) := x^* \quad \text{for all } t \geq 1 \text{ and all histories } h_{t-1}.$$

What outcomes can be induced against $\mathcal{A}^*$? As the learning algorithm plays x^* each day, and as the optimizer can play any sequence of their actions, the menu of $\mathcal{A}^*$ precisely **the convex set of CSPs of the form $x^* \otimes y$ for $y \in \mathcal{Y}$.**

Warm-up example 2. Next, let us consider an adaptive example: the learning algorithm $\hat{\mathcal{A}}$, which plays x^* as long as the opponent has never played $\hat{y}$, and plays $\hat{x}$ otherwise:

$$\mathcal{A}_t(\hat{h}_{t-1}) := \begin{cases} \hat{x} & \text{if } \hat{y} \notin \{h_{t-1}\}, \\ x^* & \text{otherwise,} \end{cases}$$

As the optimizer could play $\hat{y}$ each day, the menu of $\hat{\mathcal{A}}$ includes the CSP $\hat{x} \otimes \hat{y}$. Furthermore, at any point the optimizer may play some action $y \neq \hat{y}$; at this point, $\hat{\mathcal{A}}$ begins behaving exactly as $\mathcal{A}^*$. Thus, its menu is **the convex hull of $\hat{x} \otimes \hat{y}$ and CSPs of the form $x^* \otimes y$ for $y \in \mathcal{Y}$.** [4]

2.2 Characterization of Menus

Given these examples, a natural question to ask is what subsets of Δ^{mn} are true menus–i.e., induced as the menu of some algorithm.

Theorem 2.4 [Arunachaleswaran et al. 2024]. *A closed, convex subset $\mathcal{M} \subseteq \Delta_{mn}$ is an asymptotic menu iff for every $y \in \Delta_n$, there exists a $x \in \Delta_m$ such that $x \otimes y \in \mathcal{M}$.*

The necessity of the condition is easy to argue, since every menu must contain a point corresponding to when the optimizer picks the same action in each round. The sufficiency of the condition is proved via Blackwell Approachability.

[3]If this property is satisfied, we call $\mathcal{A}$ a *consistent* algorithm; in [Arunachaleswaran et al. 2024] we show that many focal learning algorithms, such as no-swap regret algorithms, are consistent.
[4]It is possible for an optimizer to induce a move pair distribution of the form $\hat{x} \otimes y \neq \hat{y}$ in the single day after they defect from $\hat{y}$ and before the learner can react. However, the contribution of this day to the CSP will disappear in the limit, and thus this CSP is not contained in the menu.

2.3 Menus as a tool for learning in strategic settings

The concept of a menu allows us to recast the game between the learner and optimizer in geometric terms[5]:

—The learner picks an algorithm $\mathcal{A}$ and offers the menu $\mathcal{M}(\mathcal{A})$ to the optimizer.

—The optimizer picks an outcome CSP $\varphi \in \mathcal{M}(\mathcal{A})$ that maximizes its utility with ties broken in favor of the learner. The utilties of both the learner and optimizer are linear functions of CSPs in $\mathcal{M}(\mathcal{A})$. Specifically, the learner gets utility:

$$V_L(\mathcal{M}(\mathcal{A}), u_O) = \max\{u_L(\varphi) | \varphi \in \arg\max_{\phi \in \mathcal{M}(\mathcal{A})} u_O(\phi)\}$$

Importantly, given the menu $\mathcal{M}$ of an algorithm, it is easy to see what outcome a strategic optimizer with utility u_O would induce: it will be the CSP of $\mathcal{M}$ in the extreme direction of u_O.

It is also possible to recast the no-regret and no-swap-regret properties purely in terms of menus - to aid with this we define two special menus. We say that the CSP φ is *no-regret* if it satisfies the no-regret constraint

$$\sum_{i \in [m]} \sum_{j \in [n]} \varphi_{ij} u_L(i,j) \geq \max_{j^* \in [n]} \sum_{i \in [m]} \sum_{j \in [n]} \varphi_{ij} u_L(i,j^*). \tag{1}$$

Similarly, say that the CSP φ is *no-swap-regret* if, for each $j \in [n]$, it satisfies

$$\sum_{i \in [m]} \varphi_{ij} u_L(i,j) \geq \max_{j^* \in [n]} \sum_{i \in [m]} \varphi_{ij} u_L(i,j^*). \tag{2}$$

For a fixed u_L, we will define the *no-regret menu* $\mathcal{M}_{\text{NR}}$ to be the convex hull of all no-regret CSPs, and the *no-swap-regret menu* $\mathcal{M}_{\text{NSR}}$ to be the convex hull of all no-swap-regret CSPs. Menus provide a clean geometric statement of the no-regret and no-swap-regret properties. These algorithmic properties (restrictions on transcripts) translate to geometric containment of menus within corresponding polytopes (restrictions on outcomes).

THEOREM 2.5 [ARUNACHALESWARAN ET AL. 2024]. *A learning algorithm $\mathcal{A}$ is no-regret iff $\mathcal{M}(\mathcal{A}) \subseteq \mathcal{M}_{NR}$. Likewise, $\mathcal{A}$ is no-swap-regret iff $\mathcal{M}(\mathcal{A}) \subseteq \mathcal{M}_{NSR}$.*

We provide a visualization of these containment properties in Figure 1.

2.4 Results about Menus

Among no-regret algorithms, it is well known that different algorithms can induce distinctly different outcomes: for instance, multiplicative weights does not satisfy the no-swap-regret property, while other constructions (e.g., [Blum and Mansour 2007]) do satisfy this strictly stronger guarantee. However, it was not a priori

[5]While this game is a proxy for the actual finite horizon game between the learner and the optimizer, as long as the optimizer picks the best utility point in $\mathcal{M}(\mathcal{A}^T)$ and tie-breaks in favor of the learner, it is not hard to show that the sequence of utilities obtained by the learner converges in the limit to the utility $V_L(\mathcal{M}(\mathcal{A}), u_O)$ achieved by the learner in this platonic menu version of the game ([Arunachaleswaran et al. 2024]).

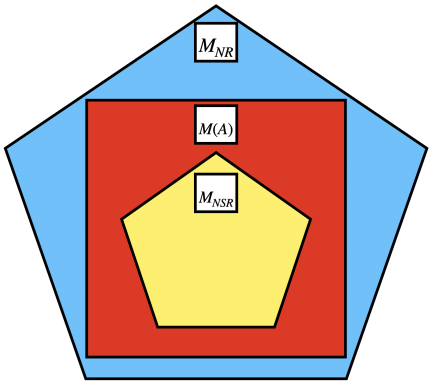

Fig. 1. Different no-regret menus in relation to each other. Here, A is any no-regret algorithm and $M(A)$ is its menu. $M(A)$ must be contained within M_{NR}, and must contain M_{NSR}.

clear whether algorithmically distinct NSR algorithms—such as [Blum and Mansour 2007] and [Dagan et al. 2024]—admit the same set of achievable outcomes, or whether their differing update rules lead to fundamentally different strategic guarantees. The menu framework resolves this question.

THEOREM 2.6 [ARUNACHALESWARAN ET AL. 2024]. *If $\mathcal{A}$ is a no-swap-regret algorithm, then $\mathcal{M}(\mathcal{A}) = \mathcal{M}_{NSR}$. Further, the no-swap-regret menu $\mathcal{M}_{NSR}$ is the convex hull of all CSPs of the form $x \otimes y$, with $x \in \Delta_m$ and $y \in BR_L(x)$.*

This result both collapses the menu of all no-swap-regret algorithms to the same menu and simultaneously provides a vertex definition of this polytope, to go along with its hyperplane based definition. We discuss the implications of this result -

(1) First, all no-swap-regret algorithms are *asymptotically equivalent*, in the sense that regardless of which no-swap-regret algorithm you run, any asymptotic strategy profile you converge to under one algorithm, you could also converge to under another algorithm (for appropriate play of the other player). This is true even when the no-swap-regret algorithms appear qualitatively quite different in terms of the strategies they choose (compare e.g. the fixed-point based algorithm of [Blum and Mansour 2007] with the more recent algorithms of [Dagan et al. 2024] and [Peng and Rubinstein 2024]).
(2) In particular, there is no notion of regret that is meaningfully *stronger* than no-swap-regret for learning in (standard, normal-form) games. That is, there is no regret-guarantee you can feasibly insist on that would rule out some points of the no-swap-regret menu while remaining no-regret in the standard sense. In other words, the no-swap-regret menu is *minimal* among all no-regret menus: every no-regret menu contains $\mathcal{M}_{\text{NSR}}$, and no asymptotic menu (whether it is no-regret or not) is a subset of $\mathcal{M}_{\text{NSR}}$.
(3) Finally, these claims are *not* generally true for external regret. There are different no-regret algorithms with very different asymptotic menus (as a concrete example, $\mathcal{M}_{\text{NR}}$ and $\mathcal{M}_{\text{NSR}}$ are often different, and they are both asymptotic

menus of some learning algorithm by Theorem 2.4).

3. APPLICATIONS OF THE MENU FRAMEWORK

The rest of this note is a tour of three papers that develop the menu framework in complementary directions. One direction asks what it means for an algorithm to be *globally good* against all possible optimizers, and whether standard learning algorithms are actually optimal in that sense. A second direction asks how to choose an algorithm when you have a known *distribution* over opponent types. And a third direction revisits the classical swap-regret and correlated equilibrium story, and asks: outside of normal-form games, what does it mean for a learning algorithm to be *non-manipulable*?

3.1 Pareto-optimal algorithms: when is one learning algorithm strictly better than another?

If you must commit to a learning algorithm in a general-sum repeated game, but you do not know the utility of the optimizer, what does it mean for this commitment to be "optimal"? In [Arunachaleswaran et al. 2024], we formalize a dominance relation between learning algorithms that is explicitly strategic.

We say an algorithm $\mathcal{A}$ is *Pareto-dominated* if there exists another algorithm $\mathcal{A}'$ that achieves at least as high long-run utility as $\mathcal{A}$ against *every* opponent payoff function, and strictly higher utility for at least one opponent type. An algorithm is *Pareto-optimal* if it is not Pareto-dominated. This definition is intentionally weak: it does not insist on optimality against each opponent type, only that the algorithm is not uniformly outperformed across all types. Nonetheless, we show in [Arunachaleswaran et al. 2024] that many standard no-regret algorithms, such as Multiplicative Weights and Follow-the-Perturbed-Leader, fail even this lenient benchmark. This leads to the question of whether there is any algorithm that is both no-regret and Pareto-optimal.

Our main positive result is that the stronger condition of *no-swap-regret* is sufficient for Pareto-optimality. We prove this result by redefining Pareto-optimality in terms of menus and characterizing it via geometric properties, which we then show that no-regret menus satisfy. One key property is inclusion-minimality; a menu $\mathcal{M}$ is *inclusion-minimal* if there is no valid menu whichx is strictly contained within $\mathcal{M}$.

THEOREM 3.1 INFORMAL. *Any algorithm whose menu is inclusion-minimal and contains the maximum learner utility CSP is Pareto-optimal.*

3.2 Unknown opponents: choosing an algorithm under a distribution over optimizer types

The Pareto-optimality benchmark is most natural when the optimizer's utility is completely unknown. In the other extreme, when the learner has full knowledge of the optimizer's utility, this becomes a Stackelberg game in the strategy space of algorithms, and approximately optimal solutions are known for this problem [Collina et al. 2023]. But in many applications, one has partial information: a model of competitor behavior in pricing, historical data about user tradeoffs on a platform, or a prior over agent "types." This motivates a Bayesian version of the algorithm-design

problem.

In [Arunachaleswaran et al. 2025], the learner commits to an algorithm, the opponent payoff function is drawn from a distribution $\mathcal{D}$, and the opponent then best-responds in algorithm space. The learner's objective is to maximize expected long-run utility given knowledge of $\mathcal{D}$. It is not clear how much the learner can infer about the optimizer's realized type; as the optimizer is non-myopic, it might be in their best interest to obfuscate. Additionally, the learner must balance their attempt to learn, to the extent possible, versus exploiting their information.

Naively, this problem may initially seem computationally prohibitive: the space of history-dependent algorithms is enormous, and computing a best response to a fixed algorithm can be NP-hard (making it tricky to systematically predict or evaluate the best response against a candidate commitment). Menus again make the problem tractable in natural regimes. Each opponent type induces a linear objective over the learner's menu, and a distribution over opponent types therefore induces a distribution over linear objectives. The learner's problem becomes: choose an algorithm whose menu scores well in expectation under these objectives.

We optimize implicitly over the set of all menus by focusing only on what outcomes the menu incentivizes for each optimizer. This yields efficient methods for computing approximately optimal commitment algorithms under $\mathcal{D}$ (namely, polynomial time under constant game size or bounded support assumptions). Importantly, our characterization of menus constrains this search so as to guarantee that any optimized menu can be implemented by an actual learning algorithm.

We also study a robustness-motivated restriction in which the learner insists on no-regret behavior while still optimizing expected performance under $\mathcal{D}$. Here, the menu viewpoint again clarifies the answer: the right algorithmic class is no swap regret, which both preserves the relevant strategic robustness and admits optimization of the induced menu.

3.3 Swap regret and correlated equilibria beyond normal-form games: starting from non-manipulability

In normal-form repeated games, no swap regret has a clean strategic interpretation: it is tightly connected to correlated equilibrium and captures a sense in which the opponent cannot benefit from dynamically steering the learner. This motivates taking "absence of dynamic manipulation" as a primitive desideratum.

Non-manipulability. Fix a learner algorithm $\mathcal{A}$ and an opponent utility function u_O. Let $V_T^{\mathrm{dyn}}(\mathcal{A}, u_O)$ be the maximum expected total utility the opponent can achieve over T rounds using any history-dependent strategy against $\mathcal{A}$. Let $V_T^{\mathrm{stat}}(\mathcal{A}, u_O)$ be the maximum expected total utility when the opponent is restricted to a *static* mixed strategy (the same distribution each round). We call $\mathcal{A}$ *non-manipulable* if for every u_O,

$$V_T^{\mathrm{dyn}}(\mathcal{A}, u_O) - V_T^{\mathrm{stat}}(\mathcal{A}, u_O) \;=\; o(T).$$

Informally: regardless of the opponent's objective, they cannot extract linear-in-T advantage by playing dynamically rather than statically.

Beyond Normal-Form Games Many strategic environments are not well-modeled as repeated normal-form games. Returning to pricing: payoffs may depend on per-period costs, demand states, or other exogenous factors, leading to repeated

Bayesian or more structured games. It was not clear what the "right" generalization of swap regret was in these richer settings; decompositions of action distributions are no longer unique, leading to many possible ways to "swap" the same mixed action. Thus, it was an open question whether there existed an efficient algorithm which was non-manipulable and no-regret. Prior algorithms that guaranteed non-manipulability and no-regret in these games were not known to be computationally tractable. Meanwhile, more tractable regret notions did not actually rule out manipulability. [Arunachaleswaran et al. 2025] studies a broad class called *Polytope games* (which generalizes both Bayesian games and Extensive-form games) and uses non-manipulability as the starting point for defining an appropriate notion of swap regret.

In menu language, non-manipulability has a particularly crisp geometric meaning: beneficial manipulation manifests as an extreme point that cannot be generated by any static product distribution. Equivalently, $\mathcal{A}$ is non-manipulable if and only if every extreme point of its menu is a product distribution. The no-regret condition further constraints what sort of product distributions the menu must contain. While the question of "what should be swapped" does not have a natural translation beyond normal-form games, the menu interpretation of non-manipulability operates identically.

We introduce a new notion of swap regret, *profile swap regret*, which is measured by the distance from a CSP to a specific "non-manipulable" menu composed of product distribution extreme points. Profile swap regret is efficiently achievable (our work gives algorithms with $O(\sqrt{T})$ profile swap regret) and guarantees non-manipulability.

4. CLOSING THOUGHTS

Against non-myopic opponents, a learning algorithm effectively serves as a commitment, shaping the set of long-run outcomes the opponent can induce. Menus provide an explicit geometric characterization of these outcomes and a unifying lens for comparing algorithms (via Pareto dominance), selecting algorithms under priors (Bayesian design), and identifying stability notions that prevent manipulation in richer game models (profile swap regret).

While this work focuses on the two-player setting, the dynamics of multiple interacting learning algorithms and strategic agents remains an open frontier, where menus may help define appropriate notions of equilibria. Moreover, in complex strategy spaces, such as high-dimensional auctions or pricing environments, understanding which menus admit computationally efficient optimization becomes a critical challenge. Ultimately, as algorithms increasingly govern economic interactions, viewing them not merely as adaptive procedures but as geometric commitments that induce constrained outcome sets will be essential for designing robust and transparent marketplaces.

REFERENCES

Arunachaleswaran, E. R., Collina, N., Mansour, Y., Mohri, M., Schneider, J., and Sivan, B. 2025. Swap regret and correlated equilibria beyond normal-form games. In *Proceedings of the 26th ACM Conference on Economics and Computation*. 130–157.

Arunachaleswaran, E. R., Collina, N., and Schneider, J. 2024. Pareto-optimal algorithms for

learning in games. In *Proceedings of the 25th ACM Conference on Economics and Computation*. 490–510.

Arunachaleswaran, E. R., Collina, N., and Schneider, J. 2025. Learning to play against unknown opponents. In *Proceedings of the 26th ACM Conference on Economics and Computation*. 478–504.

Blum, A. and Mansour, Y. 2007. From external to internal regret. *Journal of Machine Learning Research 8,* 6.

Braverman, M., Mao, J., Schneider, J., and Weinberg, M. 2018. Selling to a no-regret buyer. In *Proceedings of the 2018 ACM Conference on Economics and Computation*. 523–538.

Collina, N., Arunachaleswaran, E. R., and Kearns, M. 2023. Efficient stackelberg strategies for finitely repeated games. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 643–651.

Dagan, Y., Daskalakis, C., Fishelson, M., and Golowich, N. 2024. From external to swap regret 2.0: An efficient reduction for large action spaces. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*. 1216–1222.

Peng, B. and Rubinstein, A. 2024. Fast swap regret minimization and applications to approximate correlated equilibria. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*. 1223–1234.