Near-Optimal No-Regret Algorithms for Zero-Sum Games

Constantinos Daskalakis[1], Alan Deckelbaum[2], Anthony Kim[3]

Abstract

We propose a new no-regret learning algorithm. When used against an adversary, our algorithm achieves average regret that scales optimally as $O\left(\frac{1}{\sqrt{T}}\right)$ with the number T of rounds. However, when our algorithm is used by both players of a zero-sum game, their average regret scales as $O\left(\frac{\ln T}{T}\right)$, guaranteeing a near-linear rate of convergence to the value of the game. This represents an almost-quadratic improvement on the rate of convergence to the value of a zero-sum game known to be achievable by any no-regret learning algorithm. Moreover, it is essentially optimal as we also show a lower bound of $\Omega\left(\frac{1}{T}\right)$ for all distributed dynamics, as long as the players do not know their payoff matrices in the beginning of the dynamics. (If they did, they could privately compute minimax strategies and play them ad infinitum.)
JEL classification: C72, C73

1. Introduction

Von Neumann's minimax theorem [24] lies at the origins of the fields of both algorithms and game theory. Indeed, it was the first example of a static game-theoretic solution concept: If the players of a zero-sum game arrive at a min-max pair of strategies, then no player can improve his payoff by unilaterally deviating, resulting in an equilibrium state of the game. The min-max equilibrium played a central role in von Neumann and Morgenstern's foundations of Game Theory [25], and inspired the discovery of the Nash equilibrium [21] and the foundations of modern economic thought [20].

At the same time, the minimax theorem is tightly connected to the development of mathematical programming, as linear programming itself reduces to the computation of a min-max equilibrium, while strong linear programming duality is equivalent to the minimax theorem. [4] Given the further developments in linear programming in the past century [16, 17],

[1]EECS, MIT. Email: costis@csail.mit.edu. Supported by a Sloan Foundation Fellowship, a Microsoft Research Fellowship, and NSF Award CCF-0953960 (CAREER) and CCF-1101491.

[2]Department of Mathematics, MIT. Email: deckel@mit.edu. Supported by Fannie and John Hertz Foundation Daniel Stroock Fellowship.

[3]Department of Computer Science, Stanford University. Email: tonyekim@stanford.edu. Work done while the author was a student at MIT. Supported in part by an NSF Graduate Research Fellowship.

[4]This equivalence was apparently felt by Dantzig and von Neumann at the inception of linear programming, but no rigorous proof was given until very recently [1].

we now have efficient algorithms for computing equilibria in zero-sum games, even in very large ones such as poker [10, 11].

On the other hand, the min-max equilibrium is a static notion of stability, leaving open the possibility that there are no simple distributed dynamics via which stability comes about. This turns out not to be the case, as many distributed protocols for this purpose have been discovered. One of the first protocols suggested for this purpose is *ficticious play*, whereby players switch rounds playing the pure strategy that optimizes their payoff against the historical play of their opponent (viewed as a distribution over strategies). This simple scheme, suggested by Brown in 1949 [5], was shown to converge to the min-max value of the game by Robinson [26]. However, its convergence rate has recently been shown to be exponentially slow in the number of strategies [3]. [5] Such poor convergence guarantees do not offer much by way of justifying the plausibility of the min-max equilibrium in a distributed setting, making the following questions rather important: *Are there efficient and natural distributed dynamics converging to min-max equilibrium/value?* And *what is the optimal rate of convergence?*

The answer to the first question is, by now, very well understood. A typical source of efficient dynamics converging to min-max equilibria is online optimization. The results here are very general: If both players of a game use a no-regret learning algorithm to adapt their strategies to their opponent's strategies, then the average payoffs of the players converge to their min-max value, and their average strategies constitute an approximate min-max equilibrium, with the approximation converging to 0 [6]. In particular, if a no-regret learning algorithm guarantees average external regret $g(T, n, u)$, as a function of the number T of rounds, the number n of "experts", and the magnitude u of the maximum in absolute value payoff of an expert at each round, we can readily use this algorithm in a game setting to approximate the min-max value of the game to within an additive $O(g(T, n, u))$ in T rounds, where u is now the magnitude of the maximum in absolute value payoff in the game, and n an upper bound on the players' strategies.

For instance, if we use the *multiplicative weights update algorithm* [9, 19], we would achieve approximation $O\left(\frac{u\sqrt{\log n}}{\sqrt{T}}\right)$ to the value of the game in T rounds. Given that the dependence of $O(\frac{\sqrt{\log n}}{\sqrt{T}})$ in the number n of experts and the number T of rounds is optimal for the regret bound of any no-regret learning algorithm [6], the convergence rate to the value of the game achieved by the multiplicative weights update algorithm is the optimal rate that can be achieved by a black-box reduction of a regret bound to a convergence rate in a zero-sum game.

Nevertheless, a black-box reduction from the learning-with-expert-advice setting to the game-theoretic setting may be lossy in terms of approximation. Indeed, no-regret bounds apply even when playing against an adversary; it may be that, when two players of a zero-sum game update their strategies following a no-regret learning algorithm, faster convergence

[5] Harris [13] has studied the convergence rate of a continuous-time analog of fictitious play. However, convergence in the discrete-time setting is difficult to compare to the continuous-time setting.

to the min-max value of the game is possible. As concrete evidence of this possibility, take fictitious play (a.k.a. the "follow-the-leader" algorithm in online optimization): against an adversary, it may be forced not to converge to zero average regret; but if both players of a zero-sum game use fictitious play, their average payoffs *do* converge to the min-max value of the game, given Robinson's proof.

Motivated by this observation, we investigate the following: *Is there a no-regret learning algorithm that, when used by both players of a zero-sum game, converges to the min-max value of the game at a rate faster than $O\left(\frac{1}{\sqrt{T}}\right)$ with the number T of rounds?* We answer this question in the affirmative, by providing a no-regret learning algorithm, called NoRegretEgt, with asymptotically optimal regret behavior of $O\left(\frac{u\cdot\sqrt{\log n}}{\sqrt{T}}\right)$, and convergence rate of $O\left(\frac{u\cdot \log n\cdot(\log T+(\log n)^{3/2})}{T}\right)$ to the min-max value of a game, where n is an upper bound on the number of the players' strategies. In particular,

Theorem 1. *Let $x_1, x_2, \dots, x_t, \dots$ be a sequence of randomized strategies over a set of experts $[n] := \{1, 2, \dots, n\}$ produced by the* NoRegretEgt *algorithm under a sequence of payoffs $\ell_1, \ell_2, \dots, \ell_t, \dots \in [-u, u]^n$ observed for these experts, where ℓ_t is observed after x_t is chosen. Then for all T:*

$$\frac{1}{T}\sum_{t=1}^{T}(x_t)^{\mathrm{T}}\ell_t \geq \max_{i\in[n]}\frac{1}{T}\sum_{t=1}^{T}(e_i)^{\mathrm{T}}\ell_t - O\left(\frac{u\cdot\sqrt{\log n}}{\sqrt{T}}\right),$$

where e_i is the i^{th} unit basis vector.

Moreover, let $x_1, x_2, \dots, x_t, \dots$ be a sequence of randomized strategies over $[n]$ and $y_1, y_2, \dots, y_t, \dots$ a sequence of randomized strategies over $[m]$, and suppose that these sequences are produced when both players of a zero-sum game $(-A, A)$, $A \in [-u, u]^{n\times m}$, use the NoRegretEgt *algorithm to update their strategies under observation of the sequence of payoff vectors $(-Ay_t)_t$ and $(A^{\mathrm{T}}x_t)_t$, respectively. Then for all T:*

$$\left|\frac{1}{T}\sum_{t=1}^{T}(x_t)^{\mathrm{T}}(-A)y_t - v\right| \leq O\left(\frac{u\cdot\log k\cdot(\log T+(\log k)^{3/2})}{T}\right),$$

where v is the row player's value in the game and $k = \max\{m, n\}$. Moreover, for all T, the pair $\left(\frac{1}{T}\sum_{t=1}^{T}x_t, \frac{1}{T}\sum_{t=1}^{T}y_t\right)$ is an (additive) $O\left(\frac{u\cdot\log k\cdot(\log T+(\log k)^{3/2})}{T}\right)$-approximate min-max equilibrium of the game.

In addition, our algorithm provides the first (to the best of our knowledge) example of a *strongly-uncoupled distributed protocol* converging to the value of a zero-sum game at a rate faster than $O(\frac{1}{\sqrt{T}})$. Strong-uncoupledness is the property of a distributed game-playing protocol under which the players can observe the payoff vectors of their own strategies at every round ($(-Ay_t)_t$ and $(A^{\mathrm{T}}x_t)_t$ for the row and column players respectively), but:

- they do not know the payoff tables of the game, or even the number of strategies available to the other player; [6]

- they can only use private storage to keep track of a constant number of observed payoff vectors (or cumulative payoff vectors), a constant number of mixed strategies (or possibly cumulative information thereof), and a constant number of state variables such as the round number.

The precise details of our model and comparison to other models in the literature are given in Section 2.2. Notice that, without the assumption of strong-uncoupledness, there can be trivial solutions to the problem. Indeed, if the payoff tables of the game were known to the players in advance, they could just privately compute their min-max strategies and use these strategies ad infinitum. If the payoff tables were unknown but the type of information the players could privately store were unconstrained, they could engage in a protocol for recovering their payoff tables, followed by the computation of their min-max strategies. Even if they also didn't know each other's number of strategies, they could interleave phases in which they either recover pieces of their payoff matrices, or they compute min-max solutions of recovered square submatrices of the game until convergence to an exact equilibrium is detected. Arguably, such protocols are of limited interest in highly distributed game-playing settings.

And what could be the optimal convergence rate of distributed protocols for zero-sum games? We show that, insofar as convergence of the average payoffs of the players to their values in the game is concerned, the convergence rate achieved by our protocol is essentially optimal. Namely, we show the following: [7]

Theorem 2. *Assuming that the players of a zero-sum game $(-A, A)$ do not know their payoff matrices at the beginning of time, any distributed protocol producing sequences of strategies $(x_t)_t$ and $(y_t)_t$ such that the average payoffs of the players, $\frac{1}{T}\sum_t (x_t)^{\mathrm{T}}(-A)y_t$ and $\frac{1}{T}\sum_t (x_t)^{\mathrm{T}}Ay_t$, converge to their corresponding value in the game, cannot do so at a convergence rate faster than an additive $\Omega(1/T)$ in the number T of rounds of the protocol. The same is true of any distributed protocol whose average strategies converge to a min-max equilibrium.*

Future work. Our no-regret learning algorithm provides, to the best of our knowledge, the first example of a strongly-uncoupled distributed protocol converging to the min-max equilibrium of a zero-sum game at a rate faster than $\frac{1}{\sqrt{T}}$, and in fact at a nearly-optimal rate. The strong-uncoupledness arguably adds to the naturalness of our protocol, since no funny bit arithmetic, private computation of the min-max equilibrium, or anything of the similar flavor

[6] In view of this requirement, our notion of uncoupled dynamics is stronger than that of Hart and Mas-Colell [15]. In particular, we do not allow a player to initially have full knowledge of his utility function, since knowledge of one's own utility function in a zero-sum game reveals the entire game matrix.

[7] In this paper, we are concerned with bounds on *average* regret and the corresponding convergence of *average* strategy profiles. If we are concerned only with how close the *final* strategy profile is to an equilibrium, then we suspect that similar techniques to those of our paper can be used to devise a distributed protocol with even faster convergence of final strategy profiles, possibly by using techniques in [10].

is allowed. Moreover, the strategies that the players use along the course of the dynamics are fairly natural in that they constitute smoothened best responses to their opponent's previous strategies. Nevertheless, there is a certain degree of careful choreography and interleaving of these strategies, turning our protocol less simple than, say, the multiplicative weights update algorithm. So we view our contribution mostly as an *existence proof*, leaving the following as an interesting future research direction: Is there a simple variant of the multiplicative weights update method or Zinkevich's algorithm [27] which, when used by the players of a zero-sum game, converges to the min-max equilibrium of the game at the optimal rate of $\frac{1}{T}$? Another direction worth exploring is to shift away from our model, which allows players to play mixed strategies x_t and y_t and observe whole payoff vectors $(-A)y_t$ and $A^{\mathrm{T}}x_t$ in every round, and prove analogous results for the more restrictive multi-armed bandit setting that only allows players to play pure strategies and observe realized payoffs in every round. Finally, it would be interesting to prove formal lower bounds on the convergence rate of standard learning algorithms, such as the multiplicative weights update method, when both players use the same algorithm.

Structure. In Section 2 we provide more detail on the settings of online learning from expert advice and uncoupled dynamics in games, and proceed to the outline of our approach. Sections 3 through 5 present the high-level proof of Theorem 1, while Sections 6 through 9 present the technical details of the proof. Finally, Section 10 presents the proof of Theorem 2.

2. Online Learning, Game Dynamics, and Outline of the Approach

2.1. Learning from Expert Advice.

In the setting of learning from expert advice, a learner has a set $[n] := \{1, \ldots, n\}$ of experts to choose from at each round $t = 1, 2, \ldots$. After committing to a distribution $x_t \in \Delta_n$ over the experts,[8] a vector $\ell_t \in [-u, u]^n$ is revealed to the learner with the payoff achieved by each expert at round t. He can then update his distribution over the experts for the next round, and so forth. The goal of the learner is to minimize his *average (external) regret*, measured by the following quantity at round T:

$$\max_i \frac{1}{T}\sum_{t=1}^{T}(e_i)^{\mathrm{T}}\ell_t - \frac{1}{T}\sum_{t=1}^{T}(x_t)^{\mathrm{T}}\ell_t,$$

where e_i is the standard unit vector along dimension i (representing the deterministic strategy of choosing the i-th expert). A learning algorithm is called *no-regret* if the average regret can be bounded by a function $g(T)$ which is $o(T)$, where the function $g(T)$ may depend on the number of experts n and the maximum absolute payoff u.[9]

[8]We use the notation Δ_n to represent the n-dimensional simplex.

[9]We are concerned only with minimizing *external* regret, as opposed to the Foster and Vohra's stronger concept of internal regret [7]. While Blum and Mansour [4] have a reduction from external regret minimizing algorithms to internal regret minimizing algorithms, it is unclear whether their transformation preserves the

The *multiplicative weights update (MWU) algorithm* is a simple no-regret learning algorithm, whereby the learner maintains a "weight" for every expert, continually updates this weight by a multiplicative factor based on how the expert would have performed in the most recent round, and chooses a distribution proportionally to this weight vector at each round. The performance of the algorithm is characterized by the following:

Lemma 3 ([6])**.** *Let $(x_t)_t$ be the sequence of distributions generated by the MWU algorithm in response to the sequence of payoff vectors $(\ell_t)_t$ for the n experts, where $\ell_t \in [-u, u]^n$. Then for all T:*

$$\max_{i\in[n]} \frac{1}{T}\sum_{t=1}^{T}(e_i)^{\mathrm{T}}\ell_t - \frac{1}{T}\sum_{t=1}^{T}(x_t)^{\mathrm{T}}\ell_t \le \frac{2u}{\sqrt{2}-1}\sqrt{\frac{\ln n}{T}}.$$

Remark 4. *In this paper we do not consider online learning settings where the learner is restricted to use a single expert in every round, i.e. to use a deterministic x_t for every t. Instead we assume that the learner can use any $x_t \in \Delta_n$, realizing a payoff of $(x_t)^{\mathrm{T}}\ell_t$. Moreover, we assume that the learner can observe the whole payoff vector ℓ_t, as opposed to his realized payoff only. In particular, our model is weaker than the* partial monitoring *and the* multi-armed bandit *models. See [6] for more discussion on these models.*

2.2. Strongly-Uncoupled Dynamics in Zero-Sum Games.

A zero-sum game is described by a pair $(-A, A)$, where A is an $n\times m$ payoff matrix, whose rows are indexed by the pure strategies of the "row" player and whose columns are indexed by the pure strategies of the "column" player. If the row player chooses a randomized, or *mixed*, strategy $x \in \Delta_n$ and the column player a mixed strategy $y \in \Delta_m$, then the row player receives payoff of $-x^{\mathrm{T}}Ay$, and the column player payoff of $x^{\mathrm{T}}Ay$. (Thus, the row player aims to minimize the quantity $x^{\mathrm{T}}Ay$, while the column player aims to maximize this quantity.)[10] A *min-max or Nash equilibrium* of the game is then a pair of strategies (x, y) such that, for all $x' \in \Delta_n$, $x^{\mathrm{T}}Ay \le (x')^{\mathrm{T}}Ay$, and for all $y' \in \Delta_m$, $x^{\mathrm{T}}Ay \ge x^{\mathrm{T}}Ay'$. If these conditions are satisfied to within an additive ϵ, (x, y) is called an *ϵ-approximate equilibrium.* Von Neumann showed that a min-max equilibrium exists in any zero-sum game; moreover, that there exists a value v such that, for all Nash equilibria (x, y), $x^{\mathrm{T}}Ay = v$ [24]. Value v is called the *value of the column player in the game.* Similarly, $-v$ is called the *value of the row player in the game.*

Now let us consider the repeated interaction between two players of a zero-sum game in the framework of Robinson [26] and the basic framework of Freund and Schapire [9]. In each round $t = 1, 2, \ldots$, the players of the game (privately) choose mixed strategies x_t and y_t. After the players commit to these strategies, they realize payoffs of $(x_t)^{\mathrm{T}}(-A)y_t$ and $(x_t)^{\mathrm{T}}Ay_t$ respectively, and observe payoff vectors $-Ay_t$ and $A^{\mathrm{T}}x_t$ respectively, which correspond to the payoffs achieved by each of their pure strategies against the strategy of their opponent.

fast convergence when both players use the same learning algorithm in our setting.

[10]Throughout this paper, if we refer to "payoff" without specifying a player, we are referring to the $x^{\mathrm{T}}Ay$, the value received by the column player.

We are interested in *strongly-uncoupled efficient dynamics*, placing the following additional restrictions on the capability of players:

1. **Unknown Game Matrix.** We assume that the game matrix $A \in \mathbb{R}^{n\times m}$ is unknown to both players. In particular, the row player does not know the number of pure strategies (m) available to the column player, and vice versa. (We obviously assume that the row and column players know the numbers n and m respectively of their own pure strategies.) To avoid degenerate cases in our analysis, we will assume that both n and m are at least 2.

2. **Limited Private Storage.** The information that a player is allowed to record between rounds of the game is limited to a constant number of payoff vectors observed in the past, or cumulative information thereof, a constant number of mixed strategies played in the past, or cumulative information thereof, and a constant number of registers recording the round number and other state variables of the protocol. This is intended to preclude a player from recording the whole history of play and the whole history of observed payoff vectors, or using funny bit arithmetic that would allow him to keep all the history of play in one huge real number, etc.

   This restriction is quite natural and satisfied, e.g., by the multiplicative weights protocol, where the learner only needs to keep record of the previously used mixed strategy and update using the newly observed payoff vector at every round.

   As explained in the introduction, this restriction is important in disallowing obvious protocols where the players attempt to reconstruct the entire game matrix A to privately compute a min-max equilibrium and then use it ad infinitum.

3. **Efficient Computations.** In each round, a player can do polynomial-time computation on his private information and the observed payoff vector.[11]

We note that our protocols do not abuse the framework to "cheat" (for example, by abusing numerical precision to encode long messages in lower-order bits and locally reconstructing the entire game matrix). We do not attempt to formally define what it means for a learning algorithm to "cheat" in this manner, and we do not claim to have an information-theoretic proof that such cheating is impossible in the computational model proposed above. Rather, we point out that our computational assumptions are standard and are shared by other learning algorithms, and that, to the best of our knowledge, there is no obvious natural cheating algorithm in our setting.

We remark that we will only place the above computational restrictions to honest players. In the case of a dishonest player (an adversary who deviates from the prescribed protocol in

[11] We will not address issues of numerical precision in this paper, assuming that the players can do unit-time real arithmetic such as basic real operations, exponentiation, etc, as typically assumed in classical learning protocols such as multiplicative weights updates.

an attempt to gain additional payoff, for instance), we will make no assumptions about that player's computational abilities, private storage, or private information.

Finally, for our convenience, we make the following assumptions for all the game dynamics described in this paper. We assume that both players know a value $|A|_{max}$, which is an upper bound on the largest absolute-value payoff in the matrix A. (We assume that both the row and column player know the same value for $|A|_{max}$.) This assumption is similar to a typical bounded-payoff assumption made in the MWU protocol.[12] We assume without loss of generality that the players know the identity of the "row" player and of the "column" player. We make this assumption to allow for protocols that are asymmetric in the order of moves of the players.[13]

Comparison with dynamics in the literature: We have already explained that our model of strongly-uncoupled dynamics is stronger than the Hart and Mas-Colell model of *uncoupled dynamics* in that we do not allow players to initially have full knowledge of their own utility functions [15]. On the other hand, our model bears a strong similarity to the *unknown payoff* model of Hart and Mas-Colell [14], the *radically uncoupled dynamics* of Foster and Young [8], and the *completely uncoupled dynamics* of Babichenko [2]. These papers propose dynamics to be used by honest parties for convergence to different kinds of equilibria than our paper (pure Nash equilibria, Nash equilibria or correlated equilibria in general games). Despite our different goals, compared to those models our model is

- identical in the restriction that the players initially only know their own strategy set and are oblivious to the numbers of strategies of their opponents and (hence) also their own and their opponents' payoff functions;
- weaker in that these dynamics only allow players to use pure strategies in every round of the interaction and only observe their own realized payoffs, while we allow players to use mixed strategies and observe the payoff that each of their pure strategies would have achieved against the mixed strategies of their opponents (à la Robinson [26] and the basic framework of Freund and Schapire [9]); [14] and
- stronger in that we assume more restrictive private storage, limiting the players to remembering only a *constant* number of past played strategies and observed payoff

[12]We suspect that we can modify our protocol to work in the case where no upper bound is known, by repeatedly guessing values for $|A|_{max}$ and thereby slowing the protocol's convergence rate down by a factor logarithmic in $|A|_{max}$.

[13]We can always augment our protocols with initial rounds of interaction where both players select strategies at random, or according to a simple no-regret protocol such as the MWU algorithm. As soon as a round occurs with a non-zero payoff, the player who received the positive payoff designates himself the "row" player while the opponent designates himself the "column" player. Barring degenerate cases where the payoffs are always 0, we can show that this procedure is expected to terminate very quickly.

[14]In fact, most of the Hart and Mas-Colell paper [14] uses a model that allows players to observe non-realized payoffs as in our model, but they provide a modification of their protocol in page 1142 extending their result to the more stringent model.

vectors or cumulative information of strategies and payoffs; indeed, in our two-player zero-sum setting, players with unlimited storage capability (or able to do funny bit-arithmetic storing the whole history of play in a single register) could engage in a protocol that reconstructs sub-matrices of the game until a min-max equilibrium can be identified, resulting into trivial dynamics; our storage and computational constraints, along with the possibility that the game matrix may be highly skewed (say if $m \gg n$) seem to rule out most of these trivial protocols.

We note that our storage constraints are different than the “finite memory” model of Babichenko [2], which assumes that a player can select strategies using a finite automaton that changes states depending on the played strategy and received payoff. We do not assume such a stringent model of computation because we want to allow players to play mixed strategies and do real operations with received payoffs (such as the exponentiation needed for implementing the multiplicative-weights-update algorithm). But we only allow players to store a constant number of played strategies and observed payoffs or cumulative information thereof, in accordance with the informal definition of “simplicity” given by Foster and Young [8].

Dynamics from Experts: A typical source of strongly-uncoupled dynamics converging to min-max equilibria in zero-sum games are no-regret algorithms for learning from expert advice. For example, if both players of a zero-sum game use the Multiplicative-Weights-Update algorithm to choose strategies in repeated interaction, [15] we can bound their average payoffs in terms of the value of the game as follows:

Proposition 5 ([9])**.** *Let $(x_t)_t$ and $(y_t)_t$ be sequences of mixed strategies chosen by the row and column players respectively of a zero sum game $(-A, A)_{n\times m}$ when using the MWU algorithm under observation of the sequence of payoff vectors $(-Ay_t)_t$ and $(A^{\mathrm{T}}x_t)_t$. Then*

$$v - C\sqrt{\frac{\ln m}{T}} \le \frac{1}{T}\sum_{t=1}^{T}(x_t)^{\mathrm{T}}Ay_t \le v + C\sqrt{\frac{\ln n}{T}},$$

where v is the value of the column player in the game and $C = \frac{2u}{\sqrt{2}-1}$. Moreover, for all T, $(\frac{1}{T}\sum_t x_t, \frac{1}{T}\sum_t y_t)$ is a $\left(\frac{2u}{\sqrt{2}-1}\frac{\sqrt{\ln m}+\sqrt{\ln n}}{\sqrt{T}}\right)$-approximate Nash equilibrium of the game.

2.3. Outline of our Approach.

Our no-regret learning algorithm is based on a gradient-descent algorithm for computing a Nash equilibrium in a zero-sum game. Our construction for converting this algorithm into a no-regret protocol has several stages as outlined below. We start with the centralized algorithm for computing Nash equilibria in zero-sum games, disentangle the algorithm into strongly-uncoupled game-dynamics, and proceed to make them robust to adversaries, obtaining our general purpose no-regret algorithm.

[15]We have already argued that implementing MWU fits the constraints of strongly-uncoupled dynamics.

To provide a unified description of the game-dynamics and no-regret learning algorithms in this paper, we describe both in terms of the interaction of two players. Indeed, we can reduce the learning-with-expert advice setting to the setting where a row (or a column) player interacts with an adversarial (also called *dishonest*) column (respectively row) player in a zero-sum game, viewing the payoff vectors that the row (resp. column) player receives at every round as new columns (rows) of the payoff matrix of the game. The regret of the row (respectively column) player is the difference between the round-average payoff that he received and the best payoff he could have received against the round-average strategy of the adversary.

In more detail, our approach is the following:

- In Section 3, we present Nesterov's Excessive Gap Techinique (EGT) algorithm, a gradient-based algorithm for computing an ϵ-approximate Nash equilibrium in $O(\frac{1}{\epsilon})$ number of rounds.

- In Section 4, we "decouple" the EGT algorithm to construct the HONESTEGTDYNAMICS protocol. This protocol has the property that, if both players honestly follow their instructions, their actions will exactly simulate the EGT algorithm.

- In Section 5.2, we modify the HONESTEGTDYNAMICS protocol to have the property that, in an honest execution, both players' average payoffs are nearly best-possible against the opponent's historical average strategy.

- In Section 5.3, we construct BOUNDEDEGTDYNAMICS(b), a no-regret protocol. The input b is a presumed upper bound on a game parameter (unknown by the players) which dictates the convergence rate of the EGT algorithm. If b indeed upper bounds the unknown parameter and if both players are honest, then an execution of this protocol will be the same as an honest execution of HONESTEGTDYNAMICS, and the player will detect low regret. If the player measures higher regret than expected, he detects a "failure", which may correspond to either b not upper bounding the game parameter, or the other player significantly deviating from the protocol. However, the player is unable to distinguish what went wrong, and this creates important challenges in using this protocol as a building block for our no-regret protocol.

- In Section 5.4, we construct NOREGRETEGT, a no-regret protocol. In this protocol, the players repeatedly guess values of b and run BOUNDEDEGTDYNAMICS(b) until a player detects a failure. Every time the players need to guess a new value of b, they interlace a large number of rounds of the MWU algorithm. Note that detecting a deviating player here can be very difficult, if not impossible, given that neither player knows the details of the game (payoff matrix and dimensions) which come into the right value of b to guarantee convergence. While we cannot always detect deviations, we can still manage to obtain no-regret guarantees, via a careful design of the dynamics. The NOREGRETEGT protocol has the regret guarantees of Theorem 1.

- Finally, Sections 6 through 9 contain the precise technical details of the aforementioned steps, which are postponed to the end of the paper to help the flow of the high-level description of our construction.

3. Nesterov's Minimization Scheme

In this section, we introduce Nesterov's Excessive Gap Technique (EGT) algorithm and state the necessary convergence result. The EGT algorithm is a gradient-descent approach for approximating the minimum of a convex function. In this paper, we apply the EGT algorithm to appropriate best-response functions of a zero-sum game. For a more detailed description of this algorithm, see Section 6. Let us define the functions $f : \Delta_n \to \mathbb{R}$ and $\phi : \Delta_m \to \mathbb{R}$ by

$$f(x) = \max_{v \in \Delta_m} x^{\mathrm{T}} A v \quad \text{and} \quad \phi(y) = \min_{u \in \Delta_n} u^{\mathrm{T}} A y.$$

In the above definitions, $f(x)$ is the payoff arising from the column player's best response to $x \in \Delta_n$, while $\phi(y)$ is the payoff arising from the row player's best response to $y \in \Delta_m$. Note that $f(x) \geq \phi(y)$ for all x and y, and that $f(x) - \phi(y) \leq \epsilon$ implies that (x, y) is an ϵ-approximate Nash equilibrium.

Nesterov's algorithm constructs sequences of points $x^1, x^2, \ldots$ and $y^1, y^2, \ldots$ such that $f(x^k) - \phi(y^k)$ becomes small, and therefore (x^k, y^k) becomes an approximate Nash equilibrium. In the EGT scheme, we will approximate f and ϕ by smooth functions, and then simulate a gradient-based optimization algorithm on these smooth approximations. This approach for minimization of non-smooth functions was introduced by Nesterov in [23], and was further developed in [22]. Nesterov's excessive gap technique (EGT) is a gradient algorithm based on this idea. The EGT algorithm from [22] in the context of zero-sum games (see [11], [12]) is presented in its entirety in Section 6.

The main result concerning this algorithm is the following theorem from [22]:

Theorem 6. *The x^k and y^k generated by the EGT algorithm satisfy*

$$f(x^k) - \phi(y^k) \leq \frac{4||A||_{n,m}}{k+1}\sqrt{\frac{D_n D_m}{\sigma_n \sigma_m}},$$

where D_n, D_m, σ_n and σ_m are parameters which depend on the choice of norm and prox function used for smoothing f and ϕ.

In our application of the above theorem, we will have $||A||_{n,m} = |A|_{max}$ and $\frac{D_n D_m}{\sigma_n \sigma_m} = \ln n \ln m$. Our first goal is to construct a protocol such that, if both players follow the protocol, their moves simulate the EGT algorithm.

4. Honest Game Dynamics

In this section we use game dynamics to simulate the EGT algorithm, by "decoupling" the operations of the algorithm, obtaining the HonestEgtDynamics protocol. Basically, the players help each other perform computations necessary in the EGT algorithm by playing appropriate strategies at appropriate times. In this section, we assume that both players are "honest," meaning that they do not deviate from their prescribed protocols.

We recall that when the row and column players play x and y respectively, the row player observes $-Ay$ and the column player observes $x^{\mathrm{T}}A$. This enables the row and column players to solve minimization problems involving Ay and $x^{\mathrm{T}}A$, respectively. The HonestEgtDynamics protocol is a direct decoupling of the EGT algorithm.

We illustrate this decoupling idea by an example. The EGT algorithm requires solving the following optimization problem:

$$\breve{x} := \arg\max_{x \in \Delta_n}(-x^{\mathrm{T}}Ay^k - \mu_n^k d_n(x)),$$

where $d_n(\cdot)$ is a function, μ_n^k is a constant known by the row player, and y^k is a strategy known by the column player. We can implement this maximization distributedly by instructing the row player to play x^k (a strategy computed earlier) and the column player to play y^k. The row player observes the loss vector $-Ay^k$, and he can then use local computation to compute $\breve{x}$.

The HonestEgtDynamics protocol decouples the EGT algorithm exploiting this idea. We present the entire protocol in Section 7. In that section, we also prove that the average payoffs of this protocol converge to the Nash equilibrium value with rate $O(\frac{\log T}{T})$.[16]

5. No-Regret Game Dynamics

We use the HonestEgtDynamics protocol as a starting block to design a no-regret protocol.

5.1. The No-Regret Property in Game Dynamics.

We restate the *no-regret property* from Section 2.1 in the context of repeated zero-sum player interactions and define the *honest no-regret* property, a restriction of the no-regret property to the case where neither player is allowed to deviate from a prescribed protocol.

Definition 7. *Fix a zero-sum game $(-A, A)_{n\times m}$ and a distributed protocol, specifying directions for the strategy that each player should choose at every time step given his observed payoff vectors. We call the protocol* honest no-regret *if it satisfies the following property: For all $\delta > 0$, there exists a T such that for all $T' > T$ and infinite sequences of strategies $(x_1, x_2, \ldots)$ and $(y_1, y_2, \ldots)$ resulting when the row and column players both follow the*

[16]The proof of this convergence is not necessary for the remainder of the paper, since our later protocols will be simpler to analyze directly. We only provide it for completeness.

protocol:

$$\frac{1}{T'}\sum_{t=1}^{T'}(-x_t^{\mathrm{T}}Ay_t) \geq \max_{i\in[n]}\frac{1}{T'}\sum_{t=1}^{T'}-(e_i)^{\mathrm{T}}Ay_t - \delta \tag{1}$$

$$\frac{1}{T'}\sum_{t=1}^{T'}(x_t^{\mathrm{T}}Ay_t) \geq \max_{i\in[m]}\frac{1}{T'}\sum_{t=1}^{T'}x_t^{\mathrm{T}}Ae_i - \delta. \tag{2}$$

We call the protocol no-regret for the column player *if it satisfies the following property: For all $\delta > 0$, there exists a T such that for all $T' > T$ and infinite sequences of moves $(x_1, x_2, \ldots)$ and $(y_1, y_2, \ldots)$ resulting when the column player follows the protocol and the row player behaves arbitrarily,* (2) *is satisfied. We define similarly what it means for a protocol to be* no-regret for the row player. *We say that a protocol is* no-regret *if it is no-regret for both players.*

The no-regret properties state that by following the protocol, a player's payoffs will not be significantly worse than the payoff that any single deterministic strategy would have achieved against the opponent's sequence of strategies.

We already argued that the average payoffs in the HonestEgtDynamics converge to the value of the game. However, this is not tantamount to the protocol being honest no-regret.[17] To exemplify what goes wrong in our setting, in Lines 18-19 of the protocol (Algorithm 2), the column player plays the strategy obtained by solving the following program, given the observed payoff vector $\hat{x}^{\mathrm{T}}A$ induced by the strategy $\hat{x}$ of the other player.

$$\hat{y} := \arg\max_y(\hat{x}^{\mathrm{T}}Ay - \mu_m^k d_m(y)).$$

It is possible that the vector $\hat{y}$ computed above differs significantly from an equilibrium strategy y^* of the column player, even if the row player has converged to an equilibrium strategy $\hat{x} = x^*$. For example, suppose that $\hat{x} = x^*$, where x^* is an equilibrium strategy for the row player, and suppose that y^* is an equilibrium strategy for the column player that involves mixing between two pure strategies in a 99%-1% ratio. We know that any combination of the two pure strategies supported by y^* will be a "best response" to x^*. Therefore, the minimizer of the above expression may involve mixing in, for example, a 50%-50% ratio of these strategies (given the canonization term $-\mu_m^k d_m(y)$ in the objective function). Since $\hat{y}$ differs significantly from y^*, there might be some best response x' to $\hat{y}$ which performs significantly better than x^* performs against $\hat{y}$, and thus the protocol may end up not being honest no-regret for the row player. A similar argument shows that the protocol is not necessarily honest no-regret for the column player.

[17]For an easy example of why these two are not equivalent, consider the rock-paper-scissors game. Let the row player continuously play the uniform strategy over rock, paper, and scissors, and let the column player continuously play rock. The average payoff of the players is 0, which is the value of the game, but the row player always has average regret bounded away from 0.

5.2. "Honest No-Regret" Protocols.

We perform a simple modification to the HonestEgtDynamics protocol to make it honest no-regret. The idea is for the players to only ever play strategies which are very close to the strategies x^k and y^k maintained by the EGT algorithm at round k, which—by Theorem 6—constitute an approximate Nash equilibrium with the approximation going to 0 with k. Thus, for example, instead of playing $\hat{y}$ in Line 19 of HonestEgtDynamics, the column player will play $(1-\delta_k)y^k + \delta_k\hat{y}$, where δ_k is a very small fraction (say, $\delta_k = \frac{1}{(k+1)^2}$). Since the row player has previously observed Ay^k, and since δ_k is known to both players, the row player can compute the value of $A\hat{y}$. Furthermore, we note that the payoff of the best response to $(1-\delta_k)y^k + \delta_k\hat{y}$ is within $2|A|_{max}\delta_k$ of the payoff of the best response to y^k. Hence, the extra regret introduced by the mixture goes down with the number of rounds k. Indeed, the honest no-regret property resulting from this modification follows from this observation and the fact that x^k and y^k converge to a Nash equilibrium in the EGT algorithm (Theorem 6). (We do not give an explicit description of the modified HonestEgtDynamics and the proof of its honest no-regret property, as we incorporate this modification to further modifications that follow.)

5.3. Presumed Bound on $\sqrt{\ln n \ln m}$.

We now begin work towards designing a no-regret protocol. Recall from Theorem 6 that the convergence rate of the EGT algorithm, and thus the rate of decrease of the average regret of the protocol from Section 5.2, depends on the value of $\sqrt{\ln n \ln m}$. However, without knowing the dimensions of the game (i.e. without knowledge of $\sqrt{\ln n \ln m}$), the players are incapable of measuring if their regret is decreasing as it should be, were they playing against an honest opponent. And if they have no ability to detect dishonest behavior and counteract, they could potentially be tricked by an adversary and incur high regret. In an effort to make our dynamics robust to adversaries and obtain the desired no-regret property, we design in this section a protocol, BoundedEgtDynamics(b), which takes a presumed upper bound b on $\sqrt{\ln n \ln m}$ as an input. This protocol will be our building block towards obtaining a no-regret protocol in the next section.

The idea for BoundedEgtDynamics(b) is straightforward: since a presumed upper bound b on $\sqrt{\ln n \ln m}$ is decided, the players can compute an upper bound on how much their regret ought to be in each round of the Section 5.2 protocol, assuming that b was a correct bound. If a player's regret in a round is ever greater than this computed upper bound, the player can conclude that either $b < \sqrt{\ln n \ln m}$, or that the opponent has not honestly followed the protocol. In the BoundedEgtDynamics protocol, a participant can detect two different types of failures, "YIELD" and "QUIT," described below. Both of these failures are internal state updates to a player's private computations and are not directly communicated to the other player. However, by the construction of our protocol, whenever one player detects a failure the other player will have the information necessary to detect the failure as well. The distinction between the types of detectable violations will be important in Section 5.4.

- YIELD(s)- A YIELD failure means that a violation of a convergence guarantee has been detected. (In an honest execution, this will be due to b being smaller than $\sqrt{\ln n \ln m}$.) Our protocol can be designed so that, whenever one player detects a YIELD failure, the other player detects the same YIELD failure. A YIELD failure has an associated value s, which is the smallest "presumed upper bound on $\sqrt{\ln n \ln m}$" which, had s been given as the input to BoundedEgtDynamics instead of b, the failure would not have been declared.[18]

- QUIT- A QUIT failure occurs when the opponent has been caught cheating. For example, a QUIT failure occurs if the row player is supposed to play the same strategy twice in a row but the column player observes different loss vectors. Unlike a YIELD failure, which could be due to the presumed upper bound being incorrect, a QUIT failure is a definitive proof that the opponent has deviated from the protocol.

For the moment, we can imagine a player switching to the MWU algorithm if he ever detects a failure. Clearly, this is not the right thing to do as a failure is not always due to a dishonest opponent, so this will jeopardize the fast convergence in the case of honest players. To avoid this, we will specify the appropriate behavior more precisely in Section 5.4.

We explicitly state and analyze the BoundedEgtDynamics(b) protocol in detail in Section 8. The main lemma that we show is the following regret bound:

Lemma 8. *Let $(x_1, x_2, \ldots)$ and $(y_1, y_2, \ldots)$ be sequences of strategies played by the row and column players respectively, where the column player used the* BoundedEgtDynamics(b) *protocol to determine his moves at each step. (The row player may or may not have followed the protocol.) If, after the first T rounds, the column player has not yet detected a YIELD or QUIT failure, then*

$$\max_{i \in [m]} \frac{1}{T} \sum_{t=1}^{T} x_t^{\mathrm{T}} A e_i \leq \frac{1}{T} \sum_{t=1}^{T} x_t^{\mathrm{T}} A y_t + \frac{37|A|_{max}}{T} + \frac{20|A|_{max} b \ln (T+2)}{T}.$$

The analogous result holds for the row player.

Note that the value of b does not affect the strategies played in an execution of the BoundedEgtDynamics(b) protocol where both players are honest, as long as $b > \sqrt{\ln n \ln m}$. In this case, no failures will ever be detected.

5.4. The NoRegretEGT *Protocol.*

In this section, we design our final no-regret protocol, NoRegretEgt. The idea is to use the BoundedEgtDynamics(b) protocol with successively larger values of b, which we will guess as upper bounds on $\sqrt{\ln n \ln m}$. Notice that if we ever have a QUIT failure in the BoundedEgtDynamics protocol, the failure is a definitive proof that one of the players

[18]The returned value s will not be important in this section, but will be used in Section 5.4.

is dishonest. In this case, we instruct the player detecting the failure to simply perform the MWU algorithm forever, obtaining low regret.

The main difficulty is how to deal with the YIELD failures. The naive approach of running the BoundedEgtDynamics algorithm and doubling the value of b at every YIELD failure is not sufficient; intuitively, because this approach is not taking extra care to account for the possibility that either the guess on b is too low, or that the opponent is dishonest in a way preventing the dynamics from converging. Our solution is this: every time we would double the value of b, we first perform a number of rounds of the multiplicative weights update method for a carefully chosen period length. In particular, we ensure that b is never greater than $\sqrt[4]{T}$ (for reasons which become clear in the analysis).

Now we have the following: If both players are honest, then after finitely many YIELD failures (at most $\lfloor \log_2 (2\sqrt{\ln n \ln m}) \rfloor$), b becomes larger than $\sqrt{\ln n \ln m}$. From that point on, we observe a failure-free run of the BoundedEgtDynamics protocol. Since this execution is failure-free, we argue that after the original finite prefix of rounds the regret can be bounded by Lemma 8. The crucial observation is that, if one of the players is dishonest and repeatedly causes YIELD failures of the BoundedEgtDynamics protocol, then the number of rounds of the MWU algorithm will be overwhelmingly larger than the number of rounds of the BoundedEgtDynamics (given our careful choice of the MWU period lengths), and the no-regret guarantee will follow from the MWU algorithm's no-regret guarantees.

We present the NoRegretEgt protocol in detail in Section 9. The key results are the following two theorems, proved in Section 9. Together they imply Theorem 1.

Theorem 9. *If the column player follows the* NoRegretEgt *protocol, his average regret over the first T rounds is at most $O\left(\frac{|A|_{max}\sqrt{\ln m}}{\sqrt{T}}\right)$, regardless of the row player's actions. Similarly, if the row player follows the* NoRegretEgt *protocol, his average regret over the first T rounds is at most $O\left(\frac{|A|_{max}\sqrt{\ln n}}{\sqrt{T}}\right)$, regardless of the column player's actions.*

Theorem 10. *If both players honestly follow the* NoRegretEgt *protocol, then the column player's average regret over the first T rounds is at most*

$$O\left(\frac{|A|_{max}\sqrt{\ln n \ln m}\ln T}{T} + \frac{|A|_{max}(\ln m)^{3/2}\ln n}{T}\right)$$

and the row player's average regret over the first T rounds is at most

$$O\left(\frac{|A|_{max}\sqrt{\ln n \ln m}\ln T}{T} + \frac{|A|_{max}(\ln n)^{3/2}\ln m}{T}\right).$$

6. Detailed Description of Nesterov's EGT Algorithm

In this section, we explain the ideas behind the Excessive Gap Technique (EGT) algorithm and we show how this algorithm can be used to compute approximate Nash equilibria in two-player zero-sum games. Before we discuss the algorithm itself, we introduce some necessary background terminology.

6.1. Choice of Norm.

When we perform Nesterov's algorithm, we will use norms $||\cdot||_n$ and $||\cdot||_m$ on the spaces Δ_n and Δ_m, respectively.[19] With respect to the norms $||\cdot||_n$ and $||\cdot||_m$ chosen above, we define the *norm* of A to be

$$||A||_{n,m} = \max_{x,y} \left\{ x^{\mathrm{T}} A y : ||x||_n = 1, ||y||_m = 1 \right\}.$$

In this paper, we will choose to use ℓ_1 norms on Δ_n and Δ_m, in which case $||A||_{n,m} = |A|_{max}$, the largest absolute value of an entry of A.

6.2. Choice of Prox Function.

In addition to choosing norms on Δ_n and Δ_m, we also choose smooth *prox-functions*, $d_n : \Delta_n \to \mathbb{R}$ and $d_m : \Delta_m \to \mathbb{R}$ which are strongly convex with convexity parameters $\sigma_n > 0$ and $\sigma_m > 0$, respectively.[20] These prox functions will be used to construct the smooth approximations of f and ϕ. Notice that the strong convexity of our prox functions depends on our choice of norms $||\cdot||_n$ and $||\cdot||_m$. Without loss of generality, we will assume that d_n and d_m have minimum value 0.

Furthermore, we assume that the prox functions d_n and d_m are bounded on the simplex. Thus, there exist D_n and D_m such that

$$\max_{x \in \Delta_n} d_n(x) \leq D_n$$

and

$$\max_{y \in \Delta_m} d_m(y) \leq D_m.$$

6.3. Approximating f and ϕ by Smooth Functions.

We will approximate f and ϕ by smooth functions f_{μ_m} and ϕ_{μ_n}, where μ_m and μ_n are smoothing parameters. (These parameters will change during the execution of the algorithm.) Given our choice of norms and prox functions above, we define

$$f_{\mu_m}(x) = \max_{v \in \Delta_m} x^{\mathrm{T}} A v - \mu_m d_m(v)$$

$$\phi_{\mu_n}(y) = \min_{u \in \Delta_n} u^{\mathrm{T}} A y + \mu_n d_n(u).$$

We see that for small values of μ, the functions will be a very close approximation to their non-smooth counterparts. We observe that since d_n and d_m are strongly convex functions, the optimizers of the above expressions are unique.

[19]We use the notation Δ_n to represent the n-dimensional simplex.

[20]Recall that d_m is strongly convex with parameter σ_m if, for all v and $w \in \Delta_m$,

$$(\nabla d_m(v) - \nabla d_m(w))^{\mathrm{T}}(v - w) \geq \sigma_m ||v - w||_m^2.$$

As discussed above, for all $x \in \Delta_n$ and $y \in \Delta_m$ it is the case that $\phi(y) \leq f(x)$. Since $f_{\mu_m}(x) \leq f(x)$ and $\phi_{\mu_n}(y) \geq \phi(y)$ for all x and y, it is possible that some choice of values μ_n, μ_m, x and y may satisfy the *excessive gap condition* of $f_{\mu_m}(x) \leq \phi_{\mu_n}(y)$. The key point behind the excessive gap condition is the following simple lemma from [22]:

Lemma 11. *Suppose that*

$$f_{\mu_m}(x) \leq \phi_{\mu_n}(y).$$

Then

$$f(x) - \phi(y) \leq \mu_n D_n + \mu_m D_m.$$

Proof. For any $x \in \Delta_n$ and $y \in \Delta_m$, we have $f_{\mu_m}(x) \geq f(x) - \mu_m D_m$ and $\phi_{\mu_n}(y) \leq \phi(y) + \mu_n D_n$. Therefore

$$f(x) - \phi(y) \leq f_{\mu_m}(x) + \mu_m D_m - \phi_{\mu_n}(y) + \mu_n D_n$$

and the lemma follows immediately.

□

In the algorithms which follow, we will attempt to find x and y such that $f_{\mu_n}(x) \leq \phi_{\mu_n}(y)$ for μ_n, μ_m small.

6.4. Excessive Gap Technique (EGT) Algorithm.

We now present the gradient-based excessive gap technique from [22] in the context of zero-sum games (see [11], [12]). The main idea behind the excessive gap technique is to gradually lower μ_m and μ_n while updating values of x and y such that the invariant $f_{\mu_m}(x) \leq \phi_{\mu_n}(y)$ holds. Algorithm 1 uses the techniques of [22], and is presented here in the form from [12]. In Section 7 (Algorithm 2), we show how to implement this algorithm by game dynamics.

In Algorithm 1, we frequently encounter terms of the form

$$d_n(x) - x^{\mathrm{T}} \nabla d_n(\hat{x}).$$

We intuitively interpret these terms by noting that

$$\xi_n(\hat{x}, x) = d_n(x) - d_n(\hat{x}) - (x - \hat{x})^{\mathrm{T}} \nabla d_n(\hat{x})$$

is the *Bregman distance* between $\hat{x}$ and x. Thus, when $\hat{x}$ is fixed, looking at an expression such as

$$\arg\max_{x \in \Delta_n} -x^{\mathrm{T}} A y^0 + \mu_n^0 (x^{\mathrm{T}} \nabla d_n(\hat{x}) - d_n(x))$$

should be interpreted as looking for x with small Bregman distance from $\hat{x}$ which makes $-x^{\mathrm{T}} A y^0$ large. Loosely speaking, we may colloquially refer to the optimal x above as a "smoothed best response" to Ay^0.

The key point to this algorithm is the following theorem, from [22]:

Algorithm 1 Nesterov's Excessive Gap Algorithm

```
function EGT
    $\mu_n^0 := \mu_m^0 := \frac{||A||_{n,m}}{\sqrt{\sigma_n \sigma_m}}$
    $\hat{x} := \arg\min_{x \in \Delta_n} d_n(x)$
    $y^0 := \arg\max_{y \in \Delta_m} \hat{x}^{\mathrm{T}} A y - \mu_m^0 d_m(y)$
    $x^0 := \arg\max_{x \in \Delta_n} -x^{\mathrm{T}} A y^0 + \mu_n^0 (x^{\mathrm{T}} \nabla d_n(\hat{x}) - d_n(x))$

    for $k = 0, 1, 2, \ldots$ do
        $\tau := \frac{2}{k+3}$

        if $k$ is even then /* Shrink $\mu_n$ */
            $\breve{x} := \arg\max_{x \in \Delta_n} -x^{\mathrm{T}} A y^k - \mu_n^k d_n(x)$
            $\hat{x} := (1-\tau) x^k + \tau \breve{x}$
            $\hat{y} := \arg\max_{y \in \Delta_m} \hat{x}^{\mathrm{T}} A y - \mu_m^k d_m(y)$
            $\tilde{x} := \arg\max_{x \in \Delta_n} -\frac{\tau}{1-\tau} x^{\mathrm{T}} A \hat{y} + \mu_n^k x^{\mathrm{T}} \nabla d_n(\breve{x}) - \mu_n^k d_n(x)$
            $y^{k+1} := (1-\tau) y^k + \tau \hat{y}$
            $x^{k+1} := (1-\tau) x^k + \tau \tilde{x}$
            $\mu_n^{k+1} := (1-\tau) \mu_n^k$
            $\mu_m^{k+1} := \mu_m^k$
        end if

        if $k$ is odd then /* Shrink $\mu_m$ */
            $\breve{y} := \arg\max_{y \in \Delta_m} y^{\mathrm{T}} A^{\mathrm{T}} x^k - \mu_m^k d_m(y)$
            $\hat{y} := (1-\tau) y^k + \tau \breve{y}$
            $\hat{x} := \arg\max_{x \in \Delta_n} -x^{\mathrm{T}} A \hat{y} - \mu_n^k d(x)$
            $\tilde{y} := \arg\max_{y \in \Delta_m} \frac{\tau}{1-\tau} y^{\mathrm{T}} A^{\mathrm{T}} \hat{x} + \mu_m^k y^{\mathrm{T}} \nabla d_m(\breve{y}) - \mu_m^k d_m(y)$
            $x^{k+1} := (1-\tau) x^k + \tau \hat{x}$
            $y^{k+1} := (1-\tau) y^k + \tau \tilde{y}$
            $\mu_m^{k+1} := (1-\tau) \mu_m^k$
            $\mu_n^{k+1} := \mu_n^k$
        end if
    end for
end function
```

Theorem 6. *The x^k and y^k generated by the EGT algorithm satisfy*

$$f(x^k) - \phi(y^k) \leq \frac{4||A||_{n,m}}{k+1}\sqrt{\frac{D_n D_m}{\sigma_n \sigma_m}},$$

where D_n, D_m, σ_n and σ_m are parameters which depend on the choice of norm and prox function used for smoothing f and ϕ.

6.5. Entropy Prox Function and the ℓ_1 Norm.

When we simulate the EGT algorithm with game dynamics, we will choose to use the ℓ_1 norm and the entropy prox function, as defined below. (This choice of norm and prox function was mentioned in [23].)

$$\begin{aligned} d_n(x) &= \ln n + \sum_{i=1}^{n} x_i \ln x_i \\ d_m(y) &= \ln m + \sum_{j=1}^{m} y_j \ln y_j \\ ||x||_n &= \sum_{i=1}^{n} |x_i| \\ ||y||_m &= \sum_{j=1}^{m} |y_j| \end{aligned}$$

From Lemma 4.3 of [23], we know that the above choice of norms and prox functions satisfy:

$$\begin{aligned} \sigma_n &= \sigma_m = 1 \\ D_n &= \ln n \\ D_m &= \ln m \\ ||A||_{n,m} &= |A|, \end{aligned}$$

where $|A|$ is the largest absolute value entry of A. (In the EGT algorithm, it suffices to replace $||A||_{n,m}$ with $|A|_{max}$, an upper bound of $|A|$. When we make this change, we will simply replace $||A||_{n,m}$ with $|A|_{max}$ in the above theorem.)

There are three main benefits of choosing these prox functions. The first reason is that this choice will make our convergence bounds depend on the same parameters as the MWU convergence bounds, and thus it will be easy to compare the convergence rates of these techniques.

The second reason is that in the first step of the EGT algorithm, we set $\mu_n^0 := \mu_m^0 := \frac{||A||_{n,m}}{\sqrt{\sigma_n \sigma_m}}$. Since $\sigma_n = \sigma_m = 1$ under our choice of prox functions and ℓ_1 norm, this step of the algorithm simply becomes

$$\mu_n^0 := \mu_m^0 := |A|_{max},$$

which is a known constant.

The third reason is that all of the required optimizations have simple closed-form solutions. In particular, our algorithm requires us to solve optimization problems of the form

$$\arg\max_{x \in \Delta_n} x^{\mathrm{T}} s - \mu_n d_n(x),$$

where $s \in \mathbb{R}^n$ is some fixed vector. In this case, the solution has a closed form (see [23]). The solution is the vector x, with j^{th} component

$$x_j = \frac{e^{s_j/\mu_n}}{\sum_{i=1}^{n} e^{s_i/\mu_n}}.$$

The analogous result holds for optimizations over $y \in \Delta_m$.

7. The Honest EGT Dynamics Protocol

In this section, we present the entirety of the HonestEGTDynamics protocol, introduced in Section 4, and compute convergence bounds for the average payoffs. Note that throughout the paper, we present the HonestEgtDynamics protocol, and protocols which follow, as a single block of pseudocode containing instructions for both row and column players. However, this presentation is purely for notational convenience, and our pseudocode can clearly be written as a protocol for the row player and a separate protocol for the column player.

For notational purposes, most lines of our pseudocode begin with either a "R" or a "C" marker. These symbols refer to instructions to the row and column players, respectively. A line which begins with the "R, C" marker is a computation performed independently by both players. An instruction such as "PLAY $x^{\mathrm{T}}Ay$" is shorthand for an instruction of "PLAY x" in the row player's protocol, and "PLAY y" in the column player's protocol.

We compute convergence bounds for the average payoff in the HonestEgtDynamics protocol, assuming that both players honestly follow the protocol. These bounds are slightly more difficult to compute than the bounds for the BoundedEgtDynamics(∞) protocol (Algorithm 3), which also converges quickly when both players follow the protocol. We analyze the less efficient HonestEgtDynamics protocol for the sake of completeness.

We will use Theorem 6 to bound the payoffs every time the players play a round of the game. Our goal is to prove that the average payoffs in HonestEgtDynamics converge to the Nash Equilibrium value quickly (with convergence rate $O(\frac{\ln T}{T})$).

In what follows, we let P be the Nash equilibrium payoff (for the column player) of the game. For ease of notation, in the analysis that follows we let

$$\epsilon_k = \frac{4|A|_{max}\sqrt{\ln n \ln m}}{k+1}.$$

Algorithm 2

```
function HONEST EGT DYNAMICS
    $R : \mu_n^0 := |A|_{max}$     $C$: $\mu_m^0 := |A|_{max}$
    $R : \hat{x} := \arg\min_{x \in \Delta_n} d_n(x)$
    $C$ : Pick $\bar{y} \in \Delta_m$ arbitrary
    PLAY: $\hat{x}^{\mathrm{T}} A \bar{y}$
    $C : y^0 := \arg\max_{y \in \Delta_m} \hat{x}^{\mathrm{T}} A y - \mu_m^0 d_m(y)$
    PLAY: $\hat{x}^{\mathrm{T}} A y^0$
    $R : x^0 := \arg\max_{x \in \Delta_n} -x^{\mathrm{T}} A y^0 + \mu_n^0 (x^{\mathrm{T}} \nabla d_n(\hat{x}) - d_n(x))$

    for $k = 0, 1, 2, \ldots$ do
        $R$, $C$: $\tau := \frac{2}{k+3}$
        PLAY: $(x^k)^{\mathrm{T}} A y^k$

        if $k$ is even then /* Shrink $\mu_n$ */
            $R$: $\breve{x} := \arg\max_{x \in \Delta_n} -x^{\mathrm{T}} A y^k - \mu_n^k d_n(x)$
            $R$: $\hat{x} := (1-\tau) x^k + \tau \breve{x}$
            PLAY: $\hat{x}^{\mathrm{T}} A y^k$
            $C$: $\hat{y} := \arg\max_{y \in \Delta_m} \hat{x}^{\mathrm{T}} A y - \mu_m^k d_m(y)$
            PLAY: $\hat{x}^{\mathrm{T}} A \hat{y}$
            $R$: $x^{k+1} := (1-\tau) x^k + \tau (\arg\max_{x \in \Delta_n} \{-\frac{\tau}{1-\tau} x^{\mathrm{T}} A \hat{y} + \mu_n^k (x^{\mathrm{T}} \nabla d_n(\breve{x}) - d_n(x))\})$
            $C$: $y^{k+1} := (1-\tau) y^k + \tau \hat{y}$
            $R$: $\mu_n^{k+1} := (1-\tau) \mu_n^k$
            $C$: $\mu_m^{k+1} := \mu_m^k$
        end if

        if $k$ is odd then /* Shrink $\mu_m$ */
            $C$: $\breve{y} := \arg\max_{y \in \Delta_m} y^{\mathrm{T}} A^{\mathrm{T}} x^k - \mu_m^k d_m(y)$
            $C$: $\hat{y} := (1-\tau) y^k + \tau \breve{y}$
            PLAY: $(x^k)^{\mathrm{T}} A \hat{y}$
            $R$: $\hat{x} := \arg\max_{x \in \Delta_n} -x^{\mathrm{T}} A \hat{y} - \mu_n^k d_n(x)$
            PLAY: $\hat{x}^{\mathrm{T}} A \hat{y}$
            $C$: $y^{k+1} := (1-\tau) y^k + \tau (\arg\max_{y \in \Delta_m} \{\frac{\tau}{1-\tau} y^{\mathrm{T}} A^{\mathrm{T}} \hat{x} + \mu_m^k (y^{\mathrm{T}} \nabla d_m(\breve{y}) - d_m(y))\})$
            $R$: $x^{k+1} := (1-\tau) x^k + \tau \hat{x}$
            $C$: $\mu_m^{k+1} := (1-\tau) \mu_m^k$
            $R$: $\mu_n^{k+1} := \mu_n^k$
        end if
    end for
end function
```

We now have the following bounds on the payoffs, where we analyze each line of HonestEgtDynamics separately:

- Lines 5 and 7- We simply bound each of these payoffs by

$$-|A|_{max} \le \hat{x}^{\mathrm{T}} A \bar{y} \le |A|_{max}, \text{ and}$$
$$-|A|_{max} \le \hat{x}^{\mathrm{T}} A y^0 \le |A|_{max}.$$

- Line 12-

  The row and column players are approximately best-responding to each other, by Theorem 6. Therefore, the payoff received must be close to the Nash equilibrium value of the game, and thus
$$P - \epsilon_k \le (x^k)^{\mathrm{T}} A y^k \le P + \epsilon_k.$$

- Line 17-

  We notice that $\hat{x}^{\mathrm{T}} A y^k \le (1-\tau)(x^k)^{\mathrm{T}} A y^k + \tau |A|_{max}$. This will enable us to bound $\hat{x}^{\mathrm{T}} A y^k$ using Theorem 6. In particular,

$$\begin{aligned}\hat{x}^{\mathrm{T}} A y^k &\le (1-\tau)(P+\epsilon_k) + \tau|A|_{max} \\ &\le P + \tau|A|_{max} + (1-\tau)\epsilon_k + \tau|A|_{max} \\ &\le P + \epsilon_k + \frac{4|A|_{max}}{k+3}.\end{aligned}$$

  Therefore, we have the bounds

$$P - \epsilon_k \le \hat{x}^{\mathrm{T}} A y^k \le P + \epsilon_k + \frac{4|A|_{max}}{k+3},$$

  where for the first inequality we used that $\min_x x^{\mathrm{T}} A y^k \ge P - \epsilon_k$, which follows from Theorem 6.

- Line 19- We notice that, since $\hat{y} := \arg\max_{y\in\Delta_m} \hat{x}^{\mathrm{T}} A y - \mu_m^k d_m(y)$, we have

$$\hat{x}^{\mathrm{T}} A \hat{y} \ge \max_{y\in\Delta_m}\{\hat{x}^{\mathrm{T}} A y\} - \mu_m^k D_m.$$

  Furthermore, since $\max_{y\in\Delta_m}\{\hat{x}^{\mathrm{T}} A y\} \ge P$, this gives us the bound

$$P - \mu_m^k \ln m \le \hat{x}^{\mathrm{T}} A \hat{y}.$$

Now we determine the value of μ_m^k. Notice that k is even at Line 19, and therefore

$$\begin{aligned}\mu_m^k &= \mu_m^0 \cdot \prod_{\substack{1\le i\le k-1\\ i \text{ odd}}} \left(1 - \frac{2}{i+3}\right)\\ &= \mu_m^0 \cdot \prod_{\substack{1\le i\le k-1\\ i \text{ odd}}} \frac{i+1}{i+3}\\ &= \frac{2|A|_{max}}{k+2}.\end{aligned}$$

To obtain an upper bound, we notice that

$$\hat{x} := (1-\tau)x^k + \tau(\arg\max_{x\in\Delta_n}\{-x^{\mathrm{T}}Ay^k - \mu_n^k d_n(x)\}).$$

Therefore,

$$\begin{aligned}\hat{x}^{\mathrm{T}}A\hat{y} &\le (1-\tau)(x^k)^{\mathrm{T}}A\hat{y} + \tau|A|_{max}\\ &\le P + \epsilon_k + 2\tau|A|_{max}\\ &= P + \epsilon_k + \frac{4|A|_{max}}{k+3},\end{aligned}$$

where in the second inequality above we used the fact that $\max_y (x^k)^{\mathrm{T}}Ay \le P + \epsilon_k$, which is guaranteed by Theorem 6. Putting these bounds together, we have

$$P - \frac{2|A|_{max}\ln m}{k+2} \le \hat{x}^{\mathrm{T}}A\hat{y} \le P + \epsilon_k + \frac{4|A|_{max}}{k+3}.$$

- Line 29-

  By the same analysis as Line 17, we have

$$P - \epsilon_k - \frac{4|A|_{max}}{k+3} \le (x^k)^{\mathrm{T}}A\hat{y} \le P + \epsilon_k.$$

- Line 31-

  The analysis is nearly identical to the analysis from Line 19. The only difference is that, since k is odd, we have

$$\mu_n^k = \mu_n^0 \cdot \prod_{\substack{0\le i\le k-1\\ i \text{ even}}} \frac{i+1}{i+3} = \frac{|A|_{max}}{k+2}.$$

Therefore, we have the bound

$$P - \epsilon_k - \frac{4|A|_{max}}{k+3} \leq \hat{x}^{\mathrm{T}} A \hat{y} \leq P + \frac{|A|_{max} \ln n}{k+2}.$$

Using the above bounds, we obtain the following lemma, which we prove immediately below:

Lemma 12. *For all $K \geq 1$, the average payoff (of the column player) resulting from the* HonestEgtDynamics *for a total of $3K+2$ rounds is bounded by*

$$\begin{aligned}
&P - \frac{4|A|_{max}}{3K+2} - \frac{24|A|_{max} \ln (K+1)}{3K+2}(\sqrt{\ln n \ln m} + \ln m + 1) \\
&\leq \textit{Average Payoff} \leq \\
&P + \frac{4|A|_{max}}{3K+2} + \frac{24|A|_{max} \ln (K+1)}{3K+2}(\sqrt{\ln n \ln m} + \ln n + 1),
\end{aligned}$$

where P is the value of the column player.

Comparing this lemma to Proposition 5, we observe that the average payoffs of HonestEgtDynamics have better asymptotic convergence (in the number of rounds played) to a Nash equilibrium than the MWU algorithm.

Proof. We see that we can lower bound the sum of the three payoffs obtained for any fixed value of k (the payoffs received in Lines 12, 17, and 19 if k is even, and in Lines 12, 29, and 31 if k is odd) by

$$3P - 3\epsilon_k - \frac{8|A|_{max}}{k+3} - \frac{2|A|_{max} \ln m}{k+2}.$$

Therefore, we can lower bound the average payoff by

$$\begin{aligned}
&\frac{1}{3K+2}\left(-2|A|_{max} + \sum_{k=0}^{K-1}\left(3P - 3\epsilon_k - \frac{8|A|_{max}}{k+3} - \frac{2|A|_{max} \ln m}{k+2}\right)\right) \\
&\geq \frac{1}{3K+2}\left(-2|A|_{max} + 3KP - 3\sum_{k=0}^{K-1}\epsilon_k - |A|_{max}(8 + 2\ln m)\sum_{k=0}^{K-1}\frac{1}{k+2}\right) \\
&\geq \frac{1}{3K+2}\left(-2|A|_{max} + 3KP - |A|_{max}\big(12\sqrt{\ln n \ln m} + 8 + 2\ln m\big)\sum_{k=0}^{K-1}\frac{1}{k+1}\right) \\
&\geq \frac{1}{3K+2}\Big((-4|A|_{max} + (3K+2)P) - |A|_{max}(1 + \ln K)(12\sqrt{\ln n \ln m} + 8 + 2\ln m)\Big) \\
&\geq \frac{1}{3K+2}\Big(-4|A|_{max} + (3K+2)P - |A|_{max}(2\ln(K+1))(12\sqrt{\ln n \ln m} + 8 + 2\ln m)\Big) \\
&= P - \frac{1}{3K+2}\Big\{4|A|_{max} + \Big(24|A|_{max}\sqrt{\ln n \ln m} + 4|A|_{max}\ln m + 16|A|_{max}\Big)\ln(K+1)\Big\}.
\end{aligned}$$

Similarly, we can upper bound the sum of the three payoffs obtained for any fixed value of k by

$$3P + 3\epsilon_k + \frac{8|A|_{max}}{k+3} + \frac{|A|_{max}\ln n}{k+2}.$$

Therefore, by similar calculations as to those above, we can upper bound the average payoff received over the first $3K+2$ rounds by

$$P + \frac{1}{3K+2}\Big\{4|A|_{max} + \Big(24|A|_{max}\sqrt{\ln n \ln m} + 2|A|_{max}\ln n + 16|A|_{max}\Big)\ln(K+1)\Big\}.$$

The statement of the lemma follows.

□

8. The BoundedEgtDynamics(b) Protocol

In this section, we describe and analyze the BoundedEgtDynamics protocol in detail. For clarity, we break the algorithm apart into subroutines. The overall structure is very similar to the HonestEgtDynamics protocol, but the players continually check for evidence that the opponent might have deviated from his instructions. We emphasize that if a YIELD failure occurs during an honest execution of BoundedEgtDynamics, both players detect the YIELD failure in the same step.

8.1. The Initialization *Routine*

We first describe the Initialization routine. This routine sets the values of x^0, y^0, μ_n^0, and μ_m^0. It is identical to Lines 2 through 8 of the HonestEgtDynamics protocol.

```
function Initialization /* $R$ sets $x^0$ and $\mu_n^0$. $C$ sets $y^0$ and $\mu_m^0$ */
    $R : \mu_n^0 := |A|_{max}$    $C$: $\mu_m^0 := |A|_{max}$
    $R : \hat{x} := \arg\min_{x\in\Delta_n} d_n(x)$
    $C$ : Pick $\bar{y} \in \Delta_m$ arbitrary
    PLAY: $\hat{x}^{\mathrm{T}} A\bar{y}$
    $C : y^0 := \arg\max_{y\in\Delta_m} \hat{x}^{\mathrm{T}} Ay - \mu_m^0 d_m(y)$
    PLAY: $\hat{x}^{\mathrm{T}} Ay^0$
    $R : x^0 := \arg\max_{x\in\Delta_n} -x^{\mathrm{T}} Ay^0 + \mu_n^0(x^{\mathrm{T}}\nabla d_n(\hat{x}) - d_n(x))$
end function
```

8.2. The CheckConv *Routine*

We now describe the CheckConv routine. The goal of this routine is to verify if (x^k, y^k) is indeed an (additive) ϵ_k-approximate Nash equilibrium.[21] In this routine, the row player is

[21]More precisely, the row player verifies that his best response to y^k gives him no more than ϵ_k additional payoff over $-(x^k)^{\mathrm{T}}Ay^k$. The column player then checks an analogous property.

given an opportunity to play a move which gives him more than ϵ_k payoff against y^k than he would obtain by playing x^k. If he cannot find such a move, then the column player is given a chance. If (x^k, y^k) is indeed an ϵ_k-approximate Nash equilibrium and both players are honest, then this routine will simply consist of three rounds of $(x^k)^{\mathrm{T}} A y^k$. Otherwise a YIELD or QUIT failure may be declared, whose semantics are as explained in Section 5.3.

```
function CheckConv($x^k$, $y^k$, $\epsilon_k$) /* Check if $(x^k, y^k)$ is $\epsilon_k$-Nash.*/
    /* If no failures occur, returns the value of $(x^k)^{\mathrm{T}} A y^k$ */
    PLAY: $(x^k)^{\mathrm{T}} A y^k$

    $R$: $\dot{x} := \arg\max_{x \in \Delta_n} -x^{\mathrm{T}} A y^k$
    $R$: If $(-\dot{x}^{\mathrm{T}} A y^k) > (-(x^k)^{\mathrm{T}} A y^k) + \epsilon_k$, $\ddot{x} := \dot{x}$. Else $\ddot{x} := x^k$

    PLAY: $\ddot{x}^{\mathrm{T}} A y^k$

    $R$: If the observed loss vectors $Ay^k$ in Lines 3 and 8 differ, QUIT.
    $R$, $C$: If $\ddot{x}^{\mathrm{T}} A y^k < ((x^k)^{\mathrm{T}} A y^k) - \epsilon_k$, YIELD($\frac{((x^k)^{\mathrm{T}} A y^k - \ddot{x}^{\mathrm{T}} A y^k)(k+1)}{4|A|_{max}}$).
    $C$: If $\ddot{x}^{\mathrm{T}} A y^k \neq ((x^k)^{\mathrm{T}} A y^k)$ and $\ddot{x}^{\mathrm{T}} A y^k \geq ((x^k)^{\mathrm{T}} A y^k) - \epsilon_k$, QUIT.

    $C$: $\dot{y} := \arg\max_{y \in \Delta_y} ((x^k)^{\mathrm{T}} A y)$
    $C$: If $(x^k)^{\mathrm{T}} A \dot{y} > ((x^k)^{\mathrm{T}} A y^k) + \epsilon_k$, $\ddot{y} := \dot{y}$. Else $\ddot{y} := y^k$

    PLAY: $(x^k)^{\mathrm{T}} A \ddot{y}$

    $C$: If the observed loss vectors $(x^k)^{\mathrm{T}} A$ in Lines 3 and 17 differ, QUIT.
    $R$, $C$: If $(x^k)^{\mathrm{T}} A \ddot{y} > ((x^k)^{\mathrm{T}} A y^k) + \epsilon_k$, YIELD($\frac{((x^k)^{\mathrm{T}} A \ddot{y} - (x^k)^{\mathrm{T}} A y^k)(k+1)}{4|A|_{max}}$).
    $R$: If $(x^k)^{\mathrm{T}} A \ddot{y} \neq ((x^k)^{\mathrm{T}} A y^k)$ and $(x^k)^{\mathrm{T}} A \ddot{y} \leq ((x^k)^{\mathrm{T}} A y^k) + \epsilon_k$, QUIT.
    return $(x^k)^{\mathrm{T}} A y^k$
end function
```

8.3. The SafePlay *Routines*

The key to making our protocol no-regret is the SafePlay routines. These routines are used to replace instructions such as "PLAY $\hat{x}^{\mathrm{T}} A y^k$" from the HonestEgtDynamics protocol to ensure that the average regret of the players goes down. (See Section 5.1 for a discussion of why vanilla HonestEgtDynamics may fail to be no-regret even if both players are honest.) Besides playing conservatively, in the SafePlay routines the players also verify several properties of their observed payoff vectors.

Before running the SafePlay routines, we assume that $(x^k)^{\mathrm{T}} A y^k$ has already been played at some point in the protocol, so that the row player knows the loss vector Ay^k and the column player knows the loss vector $(x^k)^{\mathrm{T}} A$. Both players know a value ϵ_k, and they currently believe (x^k, y^k) to be an ϵ_k-approximate Nash equilibrium.[22] In particular, they use $\hat{P}$, the value of $(x^k)^{\mathrm{T}}Ay^k$, as an estimate of the Nash equilibrium value.

The idea of the routines is that instead of playing $\hat{x}$, the row player will play $\delta_k\hat{x} + (1 - \delta_k)x^k$, where δ_k is some (small) value known to both players. Since the column player will have already observed the loss vector $(x^k)^{\mathrm{T}}A$, he will be able to determine the vector $\hat{x}^{\mathrm{T}}A$.

We now define the SafePlayRow routine (for the row player to convey a loss vector to the column player) and the SafePlayCol routine (for the column player to convey a loss vector to the row player).

```
1: function SafePlayRow($x$, $\hat{P}$, $\epsilon_k$, $\delta_k$, $(x^k)^{\mathrm{T}}A$, $Ay^k$)
2:     /* Protocol for the row player to convey $x^{\mathrm{T}}A$ to the column player */
3:     /* $(x^k)^{\mathrm{T}}A$ is a loss vector previously observed by the column player */
4:     /* $Ay^k$ is a loss vector previously observed by the row player */
5:     /* $\hat{P} := (x^k)^{\mathrm{T}}Ay^k$, $\epsilon_k$, $\delta_k$ known by both players */
6:     PLAY: $(\delta_k x + (1-\delta_k)x^k)^{\mathrm{T}}Ay^k$. Call this value $p$. Let $u^{\mathrm{T}}$ be the loss vector observed
       by the column player, and let $v$ be the loss vector observed by the row player.
7:     $C$: Set $ans = \frac{u^{\mathrm{T}}-(1-\delta_k)(x^k)^{\mathrm{T}}A}{\delta_k}$
8:     $C$: If any entry of $ans$ has absolute value greater than $|A|_{max}$, QUIT.
9:     $R$: If $v \neq Ay^k$, QUIT.
10:    $R$, $C$: If $|\hat{P} - p| > \epsilon_k + 2|A|_{max}\delta_k$, YIELD($\frac{(|\hat{P}-p|-2|A|_{max}\delta_k)(k+1)}{4|A|_{max}}$).
11:    $C$: Conclude that $x^{\mathrm{T}}A = ans$
12: end function
```

Notice that the check on Line 8 of the SafePlayRow routine ensures that the loss vector u^{T} is very close to $(x^k)^{\mathrm{T}}A$. This is a key property for showing that the protocol is no-regret (since it ensures that the payoff of a best response to the loss vector u^{T} and the payoff of a best response to the loss vector $(x^k)^{\mathrm{T}}A$ differ by no more than $2\delta_k|A|_{max}$.) In particular, it means that y^k is very close to a best response to the loss vector u^{T}.

8.4. The BoundedEgtDynamics(b) *Protocol*

We now describe the BoundedEgtDynamics protocol using the above subroutines. This protocol is nearly identical in structure to the HonestEgtDynamics protocol. If a YIELD or QUIT failure is detected by a player, the player immediately switches to the MWU algorithm.

8.5. Bounding the Regret

We will show Lemma 8, i.e. that the protocol is no-regret for the column player as long as no YIELD or QUIT failures have been declared. We will also establish that the average regret of the column player goes down as $O(\frac{\ln T}{T})$. The analysis for the row player will be nearly identical.

[22]These beliefs have been established by the CheckConv routine.

```
1: function SafePlayCol($y$, $\hat{P}$, $\epsilon_k$, $\delta_k$, $(x^k)^{\mathrm{T}}A$, $Ay^k$)
2:     /* Protocol for the column player to convey $Ay$ to the row player */
3:     /* $(x^k)^{\mathrm{T}}A$ is a loss vector previously observed by the column player */
4:     /* $Ay^k$ is a loss vector previously observed by the row player */
5:     /* $\hat{P} := (x^k)^{\mathrm{T}}Ay^k$, $\epsilon_k$, $\delta_k$ known by both players */
6:     PLAY: $(x^k)^{\mathrm{T}}A(\delta_k y+(1-\delta_k)y^k)$. Call this value $p$. Let $u^{\mathrm{T}}$ be the loss vector observed
       by the column player, and let $v$ be the loss vector observed by the row player.
7:     $R$: Set $ans = \frac{v-(1-\delta_k)A(y^k)}{\delta_k}$
8:     $R$: If any entry of $ans$ has absolute value greater than $|A|_{max}$, QUIT.
9:     $C$: If $u^{\mathrm{T}} \neq (x^k)^{\mathrm{T}}A$, QUIT.
10:    $R$, $C$: If $|\hat{P}-p| > \epsilon_k + 2|A|_{max}\delta_k$, YIELD($\frac{(|\hat{P}-p|-2|A|_{max}\delta_k)(k+1)}{4|A|_{max}}$).
11:    $R$: Conclude that $Ay = ans$
12: end function
```

Let us consider some execution of the protocol in which the column player never detects a failure and look at all sections of the BoundedEgtDynamics protocol where the game was played. In this analysis, we will prove a stronger claim than the no-regret property. Instead of showing that the column player has no single strategy which would have performed significantly better against the opponent's historical average, we show that the column player has no sequence of strategies which would have performed significantly better against the opponent's history. (Thus, if we were to tell the column player in advance all opponent's strategies in order, and allowed the column player to change his strategy from round to round, he would still not be able to perform significantly better.)

- Line 2 -The Initialization routine is only played once during the entire execution. We can lower bound the total payoff received in the two plays of this round by $-2|A|_{max}$. By deviating in both plays, it is possible that the column player could have changed his payoff to no more than $2|A|_{max}$, and therefore the column player could have gained at most $4|A|_{max}$ by deviating.

- Line 11 - Since the protocol never failed, it must be the case that every time Line 11 of BoundedEgtDynamics is reached, the moves $(x^k)^{\mathrm{T}}Ay^k$ are played three times in succession. Furthermore, since the column player always sets $\ddot{y} := y^k$ in Line 15 of the CheckConv protocol, it must be the case that, by deviating, the column player could have improved his payoff by no more than ϵ_k in each of the three rounds.

- Line 16 - This is a SafePlayRow routine. Notice that, in Line 8 of the SafePlayRow routine, the column player ensures that each entry in the vector $|u^{\mathrm{T}} - (1-\delta_k)(x^k)^{\mathrm{T}}A|$ has absolute value no more than $\delta_k|A|_{max}$. In particular, for all $j \in \{1, 2, \ldots, m\}$, we have

$$\begin{aligned} |u^{\mathrm{T}} - (x^k)^{\mathrm{T}}A|_j &\leq |u^{\mathrm{T}} - (1-\delta_k)(x^k)^{\mathrm{T}}A|_j + |\delta_k(x^k)^{\mathrm{T}}A|_j \\ &\leq 2\delta_k|A|_{max}. \end{aligned}$$

Algorithm 3

```
function BoundedEgtDynamics($b$) /* $b$ is a presumed upper bound on $\sqrt{\ln n \ln m}$ */
    Run Initialization

    while No YIELD or QUIT failures have occurred do
        /* Remark: It is meant that if a YIELD or QUIT failure occurs anywhere within
        the while-loop, the while-loop breaks.*/

        for $k = 0, 1, 2, \ldots$ do
            $R$, $C$: $\tau_k := \frac{2}{k+3}$
            $R$, $C$: $\epsilon_k := \frac{4|A|_{max} b}{k+1}$
            $R$, $C$: $\delta_k := \frac{1}{(k+1)^2}$
            Run CheckConv($x^k, y^k, \epsilon_k$). $R$ and $C$ set $\hat{P} := (x^k)^{\mathrm{T}} A y^k$

            if $k$ is even then /* Shrink $\mu_n$ */
                $R$: $\breve{x} := \arg\max_{x \in \Delta_n} -x^{\mathrm{T}} A y^k - \mu_n^k d_n(x)$
                $R$: $\hat{x} := (1 - \tau_k) x^k + \tau_k \breve{x}$
                SafePlayRow($\hat{x}, \hat{P}, \epsilon_k, \delta_k, (x^k)^{\mathrm{T}} A, A y^k$)
                $C$: $\hat{y} := \arg\max_{y \in \Delta_m} \hat{x}^{\mathrm{T}} A y - \mu_m^k d_m(y)$
                SafePlayCol($\hat{y}, \hat{P}, \epsilon_k, \delta_k, (x^k)^{\mathrm{T}} A, A y^k$)
                $R$: $x^{k+1} := (1 - \tau_k) x^k + \tau_k (\arg\max_{x \in \Delta_n} \{-\frac{\tau_k}{1-\tau_k} x^{\mathrm{T}} A \hat{y} + \mu_n^k (x^{\mathrm{T}} \nabla d_n(\breve{x}) - d_n(x))\})$
                $C$: $y^{k+1} := (1 - \tau_k) y^k + \tau_k \hat{y}$
                $R$: $\mu_n^{k+1} := (1 - \tau_k) \mu_n^k$
                $C$: $\mu_m^{k+1} := \mu_m^k$
            end if

            if $k$ is odd then /* Shrink $\mu_m$ */
                $C$: $\breve{y} := \arg\max_{y \in \Delta_m} y^{\mathrm{T}} A^{\mathrm{T}} x^k - \mu_m^k d_m(y)$
                $C$: $\hat{y} := (1 - \tau_k) y^k + \tau_k \breve{y}$
                SafePlayCol($\hat{y}, \hat{P}, \epsilon_k, \delta_k, (x^k)^{\mathrm{T}} A, A y^k$)
                $R$: $\hat{x} := \arg\max_{x \in \Delta_n} -x^{\mathrm{T}} A \hat{y} - \mu_n^k d_n(x)$
                SafePlayRow($\hat{x}, \hat{P}, \epsilon_k, \delta_k, (x^k)^{\mathrm{T}} A, A y^k$)
                $C$: $y^{k+1} := (1 - \tau_k) y^k + \tau_k (\arg\max_{y \in \Delta_m} \{\frac{\tau_k}{1-\tau_k} y^{\mathrm{T}} A^{\mathrm{T}} \hat{x} + \mu_m^k (y^{\mathrm{T}} \nabla d_m(\breve{y}) - d_m(y))\})$
                $R$: $x^{k+1} := (1 - \tau_k) x^k + \tau_k \hat{x}$
                $C$: $\mu_m^{k+1} := (1 - \tau_k) \mu_m^k$
                $R$: $\mu_n^{k+1} := \mu_n^k$
            end if
        end for
    end while
    Use the multiplicative weights update algorithm in all subsequent rounds.
end function
```

Therefore, we know that the payoff of the column player's best response against the loss vector u^{T} he observes in this round differs from the payoff of the best response to the loss vector $(x^k)^{\mathrm{T}}A$ (observed previously) by no more than $2\delta_k|A|_{max}$. Since the column player has already verified that his strategy y^k is within ϵ_k of a best response to the loss vector $(x^k)^{\mathrm{T}}A$, and since the payoff of playing any strategy against $(x^k)^{\mathrm{T}}A$ differs from the payoff of playing the same strategy against u^{T} by at most $2\delta_k A_{max}$, we conclude that by deviating in this round the column player could have improved his payoff by at most $4\delta_k|A|_{max} + \epsilon_k$.

- Line 18- In this line, the players perform a SafePlayCol routine. In this routine, the column player plays the strategy $\delta_k y + (1-\delta_k)y^k$. We know that the payoff of playing $\delta_k y + (1-\delta_k)y^k$ against any move x is within $2\delta_k|A|_{max}$ of playing y^k against x. Since the payoff of y^k is within ϵ_k of the maximum possible payoff against x^k, we conclude that the payoff received by the column player in this round is within $2\delta_k|A|_{max} + \epsilon_k$ of his best response to the opponent's move.

- Line 28- The analysis of this round is identical to the analysis of Line 18. Therefore, we conclude that by deviating the column player could have improved his payoff in this round by no more than $2\delta_k|A|_{max} + \epsilon_k$.

- Line 30- The analysis is identical to Line 16. By deviating, the column player could have improved his payoff by no more than $4\delta_k|A|_{max} + \epsilon_k$.

This analysis gives us Lemma 8, which we formally prove below.

Proof of Lemma 8. We obviously have the inequality

$$\max_{y\in\Delta_m}\sum_{t=1}^{T}(x^t)^{\mathrm{T}}Ay \le \sum_{t=1}^{T}\max_{y\in\Delta_m}(x^t)^{\mathrm{T}}Ay.$$

We now upper bound the right hand side of the above expression.[23] From above, we know that the column player could gain at most $4|A|_{max}$ total utility by deviating in the two plays in the initialization subroutine. For all $t \ge 3$, we have

$$\max_{y\in\Delta_m}(x^t)^{\mathrm{T}}Ay \le (x^t)^{\mathrm{T}}A(y^t) + 4\delta_k|A|_{max} + \epsilon_k,$$

where $k = \lfloor \frac{t-3}{5} \rfloor$. (Note that there are 5 rounds played for each value of k.) Therefore, we

[23] In general, $\max_y \sum x_t^{\mathrm{T}}Ay$ may be significantly smaller than $\sum \max_y x_t^{\mathrm{T}}Ay$. In our case, however, the expressions will be very close to each other. The reason is that, since no failures have been detected, the x_t will be very close to Nash equilibrium strategies, for large enough t.

can bound

$$\sum_{t=1}^{T} \max_{y\in\Delta_m}(x^t)^{\mathrm{T}}Ay \leq \sum_{t=1}^{T}(x^t)^{\mathrm{T}}Ay_t + 4|A|_{max} + 5\sum_{k=0}^{\lfloor\frac{T-3}{5}\rfloor}(4\delta_k|A|_{max}+\epsilon_k)$$
$$= \sum_{t=1}^{T}(x^t)^{\mathrm{T}}Ay_t + 4|A|_{max} + 20|A|_{max}\sum_{k=0}^{\lfloor\frac{T-3}{5}\rfloor}\frac{1}{(k+1)^2} + 5\sum_{k=0}^{\lfloor\frac{T-3}{5}\rfloor}\frac{4|A|_{max}b}{k+1}.$$

We now use the bound

$$20|A|_{max}\sum_{k=0}^{\lfloor\frac{T-3}{5}\rfloor}\frac{1}{(k+1)^2} \leq 20|A|_{max}\sum_{k=0}^{\infty}\frac{1}{(k+1)^2} = \frac{20|A|_{max}\pi^2}{6} < 33|A|_{max}$$

and the bound

$$5\sum_{k=0}^{\lfloor\frac{T-3}{5}\rfloor}\frac{4|A|_{max}b}{k+1} = 20|A|_{max}b\sum_{s=1}^{\lfloor\frac{T-3}{5}\rfloor+1}\frac{1}{s} \leq 20|A|_{max}b\left(1+\ln\left(\frac{T+2}{5}\right)\right).$$

Note that

$$1+\ln((T+2)/5) = \ln e + \ln((T+2)/5) \leq \ln(T+2).$$

The result of the lemma now follows.

□

9. The NoRegretEgt Protocol

Our final NoRegretEgt protocol is presented as Algorithm 4. Note that the state variables for MWU are completely separate from the BoundedEgtDynamics state variables of k, x^k, and y^k. Whenever instructed to run additional rounds of the MWU algorithm, the players work with these MWU-only state variables. We proceed to analyze its performance establishing Theorems 9 and 10.

9.1. Bounding the Regret

Let us look at some execution of the NoRegretEgt algorithm where the column player plays honestly (we make no assumptions about the row player at the moment), and suppose that T total rounds of the game have been played thus far. We will now formally bound the column player's total regret. (His total regret is the difference between the payoff of his optimal single strategy against the opponent's history and his payoff actually received.) We can write $T = T_{EGT} + T_{MW}$, where T_{EGT} is the number of rounds of the game which have been played when the column player was in Line 10 of the NoRegretEgt protocol, and T_{MW} is the number of rounds which have been played during Lines 11 or 14 of the protocol.

Let b be the largest (most recent) value which has been used as the input to BoundedEgtDynamics(b) on Line 10 of NoRegretEgt. Notice that, if we ignore the rounds

Algorithm 4

```
1:  function NoRegretEGT
2:      Run Initialization
3:      $R$,$C$: $b := 1, k := 0$
4:
5:      Initialize MWU-only state variables for the MWU algorithm. In particular, we use
        a separate round counter for the MWU executions, initialized to 0.
6:
7:      while no QUIT errors have occurred do
8:          /* Remark: It is meant that if a QUIT failure occurs anywhere within the while-loop
            or within a subroutine that has been called from within the while-loop, the while-loop
            breaks. */
9:
10:         Run BoundedEgtDynamics($b$), starting from Line 8 of that protocol, using the
            most recent values of $k$, $x^k$, $y^k$. If a YIELD($s$) failure occurs while running Bound-
            edEgtDynamics($b$), go immediately to Line 11.
11:         $R$, $C$: Run an additional $\lceil (\max(2b, s))^4 \rceil$ rounds of the MWU algorithm.
12:         $R$, $C$: Set $b := \max(2b, s)$.
13:     end while
14:     $R$, $C$: Run the MWU algorithm forever
15: end function
```

of the game which occurred closely before YIELD failures, then the remaining rounds from Line 10 constitute a failure-free execution of BoundedEgtDynamics(b).[24]

There have been at most $\lfloor \log_2 b + 1 \rfloor$ total YIELD failures thus far, since we at least double the value of b in Line 12 of NoRegretEgt. Since we restart from Line 8 of the BoundedEgtDynamics protocol every time there is a YIELD failure (regardless of the particular line of BoundedEgtDynamics on which the failure occurred), it is possible that at most the 5 rounds prior to the failure will be "redone" when we restart after the failure.[25] For simplicity, we will (unnecessarily loosely) upper bound the regret during each of these "redone" rounds as $2|A|_{max}$. Let the total number of "redone" rounds be T_{redone}.

From Lemma 8, we can upper bound the column player's total regret during the T_{EGT} rounds from Line 10 by

$$37|A|_{max} + 20|A|_{max} b \ln\big((T_{EGT} - T_{redone}) + 2\big) + 2|A|_{max} T_{redone}.$$

[24]The key point is that these rounds constitute a failure-free execution of BoundedEgtDynamics(b) even if, when they were played, the "presumed upper bound" input to BoundedEgtDynamics was something other than b. This is because the value of b only impacts the execution of BoundedEgtDynamics in the case that a YIELD error occurs.

[25]Since we restart from Line 8 of BoundedEgtDynamics, it is possible that we will redo at most 5 rounds of the game after we readjust the b value. Also, note that, for example, the row player's move on Line 8 of the CheckConv routine differs depending on whether or not a YIELD failure will be declared after the next round. This is one of the lines which will be "redone."

Notice that $T_{redone} \leq 5\log_2 b$, since at most 5 rounds are "redone" after a YIELD failure, but these happen after b is increased. Hence, we can upper bound the total regret by

$$37|A|_{max} + 20|A|_{max} b \ln(T_{EGT} + 2) + 10|A|_{max} \log_2 b.$$

During the T_{MW} rounds of the game for which the column player was on Lines 11 or 14 of NoRegretEgt, we can upper bound the total regret using Lemma 3 by

$$\frac{2|A|_{max}\sqrt{T_{MW} \ln m}}{\sqrt{2} - 1} \leq 5|A|_{max}\sqrt{T_{MW} \ln m}.$$

Therefore, the column player's average regret over the T rounds is upper bounded by

$$\frac{1}{T}\Big(37|A|_{max} + 20|A|_{max} b \ln(T_{EGT} + 2) + 10|A|_{max} \log_2 b + 5|A|_{max}\sqrt{T_{MW} \ln m}\Big)$$

and hence is upper bounded by

$$\frac{1}{T}\Big(37|A|_{max} + 20|A|_{max} b \ln(T + 2) + 10|A|_{max} \log_2 b + 5|A|_{max}\sqrt{T \ln m}\Big).$$

The key observation is that we will always have $b \leq \sqrt[4]{T}$. Indeed, immediately before setting the current value of b in Line 12, the algorithm runs at least $(2 * (b/2))^4 = b^4$ rounds of MWU in Line 11. Therefore the total number of rounds played thus far must be at least b^4, and hence $b \leq \sqrt[4]{T}$. Therefore, we can upper bound the average regret by

$$\begin{aligned}&\frac{37|A|_{max}}{T} + \frac{20|A|_{max}\sqrt[4]{T}\ln(T+2)}{T} + \frac{10|A|_{max}\log_2(\sqrt[4]{T})}{T} + \frac{5|A|_{max}\sqrt{\ln m}}{\sqrt{T}} \\ &\leq \frac{37|A|_{max}}{T} + \frac{20|A|_{max}\sqrt[4]{T}\ln(T+2)}{T} + \frac{4|A|_{max}\ln(T)}{T} + \frac{5|A|_{max}\sqrt{\ln m}}{\sqrt{T}}.\end{aligned}$$

We can use a nearly identical argument to upper bound the row player's average regret in the case that the row player is honest, regardless of the actions of the column player.

This yields Theorem 9.

9.2. Convergence with Honest Players

We now consider an execution of the NoRegretEgt protocol in which both the row and column players honestly follow the prescribed protocol. The key observation is that once b becomes greater than $\sqrt{\ln n \ln m}$, there will never be any YIELD failures. Therefore, the total number of YIELD failures will be at most $\lfloor \log_2(2\sqrt{\ln n \ln m}) \rfloor$. Furthermore, the total number of rounds with the players in the MWU phase (Line 11 of NoRegretEgt) is at most

$$\sum_{l=1}^{\lfloor \log_2 (2\sqrt{\ln n \ln m}) \rfloor + 1} (2^l)^4 \leq 2(4\sqrt{\ln n \ln m})^4 = 512(\ln n)^2(\ln m)^2.$$

Furthermore, in this honest execution of the NoRegretEgt protocol, at most $5\lfloor \log_2 (2\sqrt{\ln n \ln m}) \rfloor$ rounds of BoundedEgtDynamics will be "redone" following YIELD errors (see Section 9.1). Therefore, using Lemmas 3 and 8, (and bounding the regret by $2|A|_{max}$ during each "redone" round) we can upper bound the column player's total regret over T rounds by

$$\begin{aligned} &10|A|_{max} \log_2 (2\sqrt{\ln n \ln m}) + 5|A|_{max}\sqrt{512(\ln n)^2(\ln m)^2}\sqrt{\ln m} \\ &\quad + 37|A|_{max} + 20|A|_{max}\sqrt{\ln n \ln m} \ln (T+2). \end{aligned}$$

This yields the final theorem of the paper, Theorem 10.

10. Lower Bounds on Optimal Convergence Rate

In this section, we prove Theorem 2. The main idea is that since the players do not know the payoff matrix A of the zero-sum game, it is unlikely that their historical average strategies will converge to a Nash equilibrium very fast. In particular, the players are unlikely to play a Nash equilibrium in the first round and the error from that round can only be eliminated at a rate of $\Omega(1/T)$, forcing the $\Omega(1/T)$ convergence rate for the average payoffs and average strategies to the min-max solution.

Proof of Theorem 2. We show that there exists a set of zero-sum games such that when a zero-sum game is selected randomly from the set, any distributed protocol's convergence to the corresponding value of the game is $\Omega(1/T)$ with high probability. We assume that n and m are at least 2 to avoid degenerate cases. For $i = 1, \ldots, n$, let A_i be the all-ones matrix except for its i-th row which is the all-zero vector. Note that the Nash equilibrium value of the game $(-A_i, A_i)$ is 0 for both players, and that all Nash equilibria are of the form (e_i, y), where e_i is the deterministic strategy of the row player choosing the i-th row and $y \in \Delta_m$. Given any distributed protocol, consider choosing a game $(-A, A)$ uniformly at random from the set $\mathcal{A} = \{(-A_1, A_1), \ldots, (-A_n, A_n)\}$. Since the players do not know A before the protocol begins, the strategies x_1 and y_1 played in the first round of the protocol will have expected payoff of $E[(x_1)^T(-A)y_1] = -1 + 1/n$ for the row player. Thus, with probability at least $1 - \frac{3}{2n} \geq 1/4$, the first-round payoff is at most $-1/3$. Suppose that this event happens. Since the row player's payoffs are never strictly positive, the average payoff $\frac{1}{T}\sum_t x_t^{\mathrm{T}}(-A)y_t$ may not converge to 0 (the value of the row player) at an expected rate faster than $\Omega(1/T)$ in the number T of rounds. A similar argument can be applied to bound the rate at which the average strategies converge to min-max equilibrium strategies. $\square$

References

[1] I. Adler. The Equivalence of Linear Programs and Zero-Sum Games. *International Journal of Game Theory*, 42(1):165–177, 2013.

[2] Y. Babichenko. Completely Uncoupled Dynamics and Nash Equilibria. Discussion Paper Series, *Center for Rationality and Interactive Decision Theory*, Hebrew University, Jerusalem, 2010.

[3] F. Brandt, F. Fischer, and P. Harrenstein. On the Rate of Convergence of Fictitious Play. In *Proceedings of the 3rd International Symposium on Algorithmic Game Theory (SAGT)*, 102–113, 2010.

[4] A. Blum and Y. Mansour. From External to Internal Regret. *Journal of Machine Learning Research*, 8:1307–1324, 2007.

[5] G. W. Brown. Iterative Solution of Games by Fictitious Play. *Activity Analysis of Production and Allocation,* 13(1):374–376, 1951.

[6] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games.* Cambridge University Press, 2006.

[7] D. P. Foster and R. Vohra. Calibrated Learning and Correlated Equilibrium. *Biometrika,* 85:379–390, 1998.

[8] D. Foster and H. P. Young. Regret Testing: Learning to Play Nash Equilibrium without Knowing You Have an Opponent. *Theoretical Economics,* 1:341–367, 2006.

[9] Y. Freund and R. Schapire. Adaptive Game Playing Using Multiplicative Weights. *Games and Economic Behavior*, 29:79–103, 1999.

[10] A. Gilpin, J. Peña, and T. Sandholm. First-Order Algorithm With $O(ln(1/\epsilon))$ Convergence for ϵ-Equilibrium in Two-Person Zero-Sum games. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI),* 75–82, 2008.

[11] A. Gilpin, S. Hoda, J. Peña, and T. Sandholm. Gradient-based Algorithms for finding Nash Equilibria in Extensive Form Games. In *Proceedings of the 3rd International Workshop on Internet and Network Economics (WINE),* 57–69, 2007.

[12] A. Gilpin, S. Hoda, J. Peña, and T. Sandholm. Smoothing Techniques for Computing Nash Equilibria of Sequential Games. *Mathematics of Operation Research*, 35(2):494–512, 2010.

[13] C. Harris. On the Rate of Convergence of Continuous-Time Fictitious Play. *Games and Economic Behavior*, 22:238–259, 1998.

[14] S. Hart and A. Mas-Colell. A Simple Adaptive Procedure Leading to Correlated Equilibrium. *Econometrica,* 68(5): 1127–1150, 2000.

[15] S. Hart and A. Mas-Colell. Uncoupled Dynamics Do Not Lead to Nash Equilibrium. *American Economic Review*, 93:1830–1836, 2003.

[16] N. Karmarkar. A New Polynomial-Time Algorithm for Linear Programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing (STOC),* 302–311, 1984.

[17] L. G. Khachiyan. A Polynomial Algorithm in Linear Programming. *Soviet Math. Dokl.*, 20(1):191–194, 1979.

[18] G. Lan, Z. Lu, and R. Monteiro. Primal-Dual First-Order Methods with $O(1/\epsilon)$ Iteration-Complexity for Cone Programming. *Mathematical Programming,* 126(1):1–29, 2009.

[19] N. Littlestone and M. Warmuth. The Weighted Majority Algorithm. *Information and Computation*, 108:212–261, 1994.

[20] R. B. Myerson. Nash Equilibrium and the History of Economic Theory. *Journal of Economic Literature,* 37:1067-1082, 1999.

[21] J. Nash. Noncooperative Games. *Annals of Mathematics,* 54:289–295, 1951.

[22] Y. Nesterov. Excessive Gap Technique in Nonsmooth Convex Minimization. *SIAM J. on Optimization* 16(1):235–249, 2005.

[23] Y. Nesterov. Smooth Minimization of Non-Smooth Functions. *Mathematical Programming,* 103(1):127–152, 2005.

[24] J. von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen,* 100:295–320, 1928.

[25] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[26] J. Robinson. An Iterative Method of Solving a Game. *Annals of Mathematics,* 54:296–301, 1951.

[27] M. Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML),* 928–936, 2003.