Bounds for the Query Complexity of Approximate Equilibria

PAUL W. GOLDBERG, University of Oxford
AARON ROTH, University of Pennsylvania

We analyze the number of payoff queries needed to compute approximate equilibria of multi-player games. We find that query complexity is an effective tool for distinguishing the computational difficulty of alternative solution concepts, and we develop new techniques for upper- and lower bounding the query complexity. For binary-choice games, we show logarithmic upper and lower bounds on the query complexity of approximate correlated equilibrium. For *well-supported* approximate correlated equilibrium (a restriction where a player's behavior must always be approximately optimal, in the worst case over draws from the distribution) we show a linear lower bound, thus separating the query complexity of well supported approximate correlated equilibrium from the standard notion of approximate correlated equilibrium.

Finally, we give a query-efficient reduction from the problem of *computing* an approximate well-supported Nash equilibrium to the problem of verifying a well supported Nash equilibrium, where the additional query overhead is proportional to the description length of the game. This gives a polynomial-query algorithm for computing well supported approximate Nash equilibria (and hence correlated equilibria) in concisely represented games. We identify a class of games (which includes congestion games) in which the reduction can be made not only query efficient, but also computationally efficient.

Categories and Subject Descriptors: F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms: Algorithms, Economics, Theory

Additional Key Words and Phrases: Payoff queries, correlated equilibrium

ACM Reference Format:
Paul W. Goldberg and Aaron Roth. 2016. Bounds for the query complexity of approximate equilibria. ACM Trans. Econom. Comput. 4, 4, Article 24 (August 2016), 25 pages.
DOI: http://dx.doi.org/10.1145/2956582

1. PRELIMINARIES

This article compares the query complexity of alternative game-theoretic solution concepts. Instead of a game G being presented in its entirety as input to an algorithm $\mathcal{A}$, we assume that $\mathcal{A}$ may submit queries consisting of strategy profiles and get told the resulting payoffs to the players in G. This model is appealing when G has many players, in which case a naive representation of G would be exponentially large. Assuming that G belongs to a known class of games $\mathcal{G}$, this gives rise to the question of how many queries are needed to find a solution, such as exact/approximate Nash/correlated equilibrium. One can study this question is conjunction with other notions of cost, such as runtime of the algorithm. An appealing aspect of query complexity is that it allows

This work was supported in part by the EPSRC under grant EP/K01000X/1, an NSF CAREER award under NSF grants CCF-1101389 and CNS-1065060, and a Google Faculty Research award.
Authors' addresses: P. W. Goldberg, Department of Computer Science, Wolfson Building, Parks Road, Oxford OX1 3QD, U.K.; email: Paul.Goldberg@cs.ox.ac.uk; A. Roth, Department of Computer and Information Science, Levine Hall, 3330 Walnut Street, Philadelphia, PA 19104-6389; email: aaroth@cis.upenn.edu.

DOI: http://dx.doi.org/10.1145/2956582

new upper and lower bounds to be obtained, providing a mathematical criterion to distinguish the difficulty of alternative solution concepts, as discussed in more detail in the following.

We consider queries that consist of pure-strategy profiles, and in which the answer to any query is the payoffs that all players receive from that profile. In this work, we mostly focus on n-player binary-action games, which have 2^n pure profiles. In the context of approximate equilibria, we use ϵ to denote the bound on a player's incentive to deviate, and we make the standard assumption that all payoffs lie in the range $[0, 1]$. We are interested in algorithms with query complexity at most polynomial in n, meaning of course that only a very small fraction of a game's profiles may be queried.

Notation. We have n players denoted by the numbers $\{1, 2, \ldots, n\}$. Let A^i be the set of possible actions, or pure strategies, of player i, and let $A = A^1 \times \ldots \times A^n$ be the set of pure profiles. In this article, we assume that all players have the same number m of pure strategies (i.e., $|A^i| = m$ for all i). $u^i : A \longrightarrow [0, 1]$ denotes player i's utility function. Generally, $\mathcal{G}$ will denote a class of games, and G denotes a specific game. $\mathcal{G}_n$ denotes n-player binary-choice games, in which $m = 2$.

1.1. Alternative Definitions of Approximate Equilibrium

We review the notions of exact and approximate correlated equilibrium (CE) and introduce the definition of well-supported approximate CE.A probability distribution ψ on A is a *correlated equilibrium* if for every player i, all pure strategies $j, k \in A^i$, we have (letting (k, a^{-i}) denote the profile a with i's strategy replaced with k)

$$\sum_{a \in A: a^i = j} \psi(a)[u^i(k, a^{-i}) - u^i(a)] \leq 0. \tag{1}$$

An *ϵ-approximate correlated equilibrium* (ϵ-CE) is a distribution ψ where for every player i, every function $f : A^i \longrightarrow A^i$, we have

$$\sum_{a \in A} \psi(a)[u^i(f(a^i), a^{-i}) - u^i(a)] \leq \epsilon. \tag{2}$$

The preceding definition is based on swap regret from Blum and Mansour [2007], although other definitions (e.g., based on internal regret) are possible. An alternative definition from Hart and Mas-Colell [2000] of *correlated ϵ-equilibrium* replaces the RHS of (1) with $\epsilon > 0$. The definitions are not quite the same: an ϵ-CE is a correlated ϵ-equilibrium, whereas a correlated ϵ-equilibrium need only be an $m\epsilon$-CE.

An *ϵ-approximate coarse correlated equilibrium* (ϵ-CCE) is a distribution ψ in which for all players i, strategies j,

$$\sum_{a \in A} \psi(a)[u^i(j, a^{-i}) - u^i(a)] \leq \epsilon.$$

In general an ϵ-CE is an ϵ-CCE, but the converse does not hold.[1] However, in the case of binary-choice games (the class of games that we mainly consider here), an ϵ-CE is an ϵ-CCE, whereas an ϵ-CCE is a 2ϵ-CE, and hence the two notions are basically the same from the perspective of our interest in asymptotic query complexity in terms of n and ϵ.

An *ϵ-well-supported approximate correlated equilibrium* (ϵ-WSCE) imposes the more demanding requirement that after a player observes his own action, his expected gain

[1]For example, in the game of rock-paper-scissors, the uniform distribution over the three strategy profiles in which both players play the same strategy is a CCE but not a CE.

from switching to any other action is at most ϵ. It can be precisely defined by saying that for any player i, strategies $j, k \in A^i$, letting $p^i_j = \Pr_{a\sim\psi}[a^i = j]$,

$$\sum_{a\in A:a^i=j} \psi(a)u^i(k, a^{-i}) - \sum_{a\in A:a^i=j} \psi(a)u^i(j, a^{-i}) \le p^i_j\epsilon,$$

which is equivalent to $\mathbb{E}[u^i(k, a^{-i})] - \mathbb{E}[u^i(j, a^{-i})] \le \epsilon$, where the expectations with respect to ψ are restricted to profiles where i plays j.

It may be helpful to note the following comparison with the earlier definition of correlated ϵ-equilibrium. Observe that if $\sum_{a\in A:a^i=j} \psi(a)$ is small (meaning that it is unlikely that in a random profile, player i plays j), then putting some given $\epsilon > 0$ into the RHS of (1) introduces more slackness than would be the case if $\sum_{a\in A:a^i=j} \psi(a)$ were large. The definition of ϵ-WSCE corresponds to an attempt to redress this variable slackness.

OBSERVATION 1. *Let G be a game and fix $\epsilon > 0$. The set of ϵ-WSCE of G is convex.*

PROOF. Let ψ and ψ' be two ϵ-WSCE of G. We show that $\psi'' = \lambda\psi + (1 - \lambda)\psi'$ is also an ϵ-WSCE (for $\lambda \in (0, 1)$). Suppose that strategy profile s is sampled from ψ'' and some player i observes his action a—that is, his marginal observation of s on his own behavior. If there were some action a' that would pay i more than ϵ more (in expectation), then one (or both) of ψ or ψ' would have to have that property. □

For completeness, recall that ψ is a *Nash equilibrium* (NE) if it obeys the further constraint that ψ is a product distribution of its restriction to the individual players.

Example 1.1. In the *directed path graphical game* G_n, each player $i \in \{1, 2, \ldots, n\}$ has two actions, 0 and 1. Player 1 gets paid 1 for playing 1 and 0 for playing 0. For $i > 1$, player i gets paid 1 for copying player $i - 1$ and 0 for playing the opposite action.

Observations about Example 1.1. In an exact NE of G_n, all players play 1 with probability 1. Furthermore it is not hard to see that for $\epsilon < 1/2$, the only ϵ-WSCE requires all players to play 1 with probability 1. In an ϵ-NE of G_n, for small ϵ all players play 1 with high probability. For example, putting $\epsilon = \frac{1}{100}$, it can be proved by induction on i that player i plays 1 with probability $> \frac{9}{10}$. By contrast, for any $\epsilon > 0$, there exist ϵ-CE where the probability that i plays 1 can decrease toward 0 as i increases. Specifically, let $z \sim U[0, 1]$; if $z \in [r\epsilon, (r + 1)\epsilon]$, let players $\{1, \ldots, r\}$ play 1 and let the other players play 0. It can be checked that the resulting distribution over pure-strategy profiles is an ϵ-approximate CE.

These observations indicate that for some games, there are many approximate CEs that are ruled out when the "well-supported" requirement is imposed.

1.2. Related Work

Various game-theoretic settings have been studied in which an agent's type may be exponentially complex (raising the specter of exponential communication complexity), but a solution may be found via a feasibly small sequence of queries to an agent about his type. Some examples are the following. In the setting of combinatorial auctions where buyers have valuations for bundles (subsets) of a set of items, Conen and Sandholm [2001] present algorithms that exploit properties of valuation functions to find a good allocation without querying the functions exhaustively. Blum et al. [2004] study (from the learning-theory perspective) welfare-optimal allocation of items among buyers obtained via value queries in which the query elicits the value a buyer places on a bundle. Conitzer [2009] studies rank aggregation in a setting where a voter may be asked which of two alternatives he prefers; that work makes the point that to find a winner, it is not

always necessary to learn all voters' complete rankings (although the query complexity of finding an aggregate ranking is similar to that of learning all voters' rankings).

We next discuss the more closely-related work on the search for solutions, such as Nash/correlated equilibrium of games, via queries to their payoff function. (In the following, we discuss the relationship with earlier work on the topic of the communication complexity of such solutions.) Hart and Nisan [2013], Babichenko and Barman [2013], and Babichenko [2013] consider general n-player games, where the payoff function is of size exponential in n, and they consider what solutions require just polynomially many queries. This article mainly follows that line of work until Section 3.2, where we consider games that have concise descriptions (but are otherwise unrestricted). Fearnley et al. [2013], Fearnley and Savani [2013], and Goldberg and Turchetta [2014] study various classes of concisely represented games; again, the aim is to find game-theoretic solutions via few queries to the payoff function. Fearnley et al. [2013] were the first to study such classes, and their motivation for studying payoff queries arises from empirical game-theoretic analysis. They give query-efficient algorithms for approximate NEs of bimatrix games, graphical games, and exact equilibria for a class of congestion games. In Section 3.3, we obtain an efficient algorithm for approximate equilibria of a slightly more general class of congestion games. Fearnley and Savani [2014] study bimatrix games and give lower bounds for the query complexity of approximate well-supported NEs, separately for deterministic and randomized algorithms. Goldberg and Turchetta [2014] present bounds on the query complexity of various classes of anonymous games.

Returning to general n-player games, the main focus of recent works is on exponential (in n) lower bounds for solutions of n-player games having a constant number m of pure strategies per player. For deterministic algorithms, Hart and Nisan [2013] show that approximate Nash/correlated equilibria need exponentially many queries, which motivates a focus on randomized algorithms (in this work, all algorithms are randomized). Hart and Nisan [2013] also show that *exact* Nash/correlated equilibria have exponential query complexity even with randomness, which motivates a focus on approximate solution concepts. The main open question (noted in Hart and Nisan [2013]) is the query complexity of ϵ-NE; in this article, we give a positive result[2] for the special case of concisely represented games (Theorem 3.3). Hart and Nisan [2013] and Babichenko and Barman [2013] note that (with randomization) one can obtain query-efficient algorithms for approximate CE by simulating regret-based algorithms. Here, we analyze this query complexity in more detail and use it as a criterion to distinguish the difficulty of approximate CE from well-supported approximate CE. Babichenko [2013] shows that ϵ-well-supported approximate NE needs exponentially many queries for $m = 10^4$ and $\epsilon = 10^{-8}$; it is an open question whether these constants can be improved.

ϵ-WSCE is a refinement of approximate CE that is analogous to well-supported approximate NE, studied in Kontogiannis and Spirakis [2010], Fearnley et al. [2012], Goldberg and Pastink [2014], and Babichenko [2013]. In an ϵ-NE, a player's payoff is allowed to be up to ϵ worse than his best response. The motivation behind the "well-supported" refinement is that in an ϵ-NE, it may still be the case that a player allocates positive probability to some strategy whose payoff is more than ϵ worse than his best response. Such behavior is disallowed in a well-supported ϵ-NE. Under this more demanding definition of approximate NE, the values of ϵ known to be achievable in polynomial time (in the context of bimatrix games) are accordingly higher; ϵ-NE can be computed for ϵ slightly above $\frac{1}{3}$ [Tsaknakis and Spirakis 2008], whereas for

[2]In other words, a query-efficient algorithm, but one that need not be computationally efficient.

ϵ-WSNE, the best value known is slightly less than $\frac{2}{3}$ [Fearnley et al. 2012]. For the games studied in this article, we will see that the payoff query complexity of ϵ-WSCE is strictly higher than that of ϵ-CE. The technique of Daskalakis et al. [2009, Lemma 3.2] (see also Lemma 3.2 in Chen et al. [2009]) for converting approximate NE into approximate WSNE works for NEs but not for CEs, and so we need new upper-bounding techniques.

In the context of searching for game-theoretic equilibria, communication complexity is the issue of how much information needs to be passed between the players, assuming that each player initially knows only his own payoff function. Very roughly, communication complexity is an lower bound for query complexity in the sense that interaction via payoff queries is a restricted form of communication. Hart and Nisan [2013] note that payoff-query bounded algorithms can be efficiently converted into communication-bounded algorithms. Communication complexity was analyzed for n-player binary-choice games by Hart and Mansour [2010] and for bimatrix games in Goldberg and Pastink [2014]. Lower bounds are easier to obtain in the payoff-query setting. In the communication-bounded setting, Hart and Mansour [2010] show efficient upper bounds for ϵ-CE and exponential lower bounds just for exact (pure or mixed) NEs. The communication complexity of finding an exact CE is polynomial in the number of players for a wide class of concisely represented multiplayer games [Papadimitriou and Roughgarden 2008]. The approach of Papadimitriou and Roughgarden [2008] uses the ellipsoid algorithm to compute a mixture of product distributions that constitutes a CE. The algorithm interacts with the game using mixed-strategy payoff queries and receiving exact answers. Note that mixed-strategy payoff queries can be approximately simulated by randomly sampled pure-strategy payoff queries; however, we believe that the algorithm of Papadimitriou and Roughgarden [2008] cannot be straightforwardly adapted to yield an approximate CE from approximate payoffs to mixed-strategy profiles.

The connection between no-regret algorithms and equilibrium notions, including CEs, were first noted by Freund and Schapire [1999] and Hart and Mas-Colell [2000]. For an excellent survey of this connection, see Blum and Mansour [2007]. Appendix B.3 of Hart and Mansour [2010] makes the observation that regret-minimization techniques yield simple polynomial upper bounds for the communication complexity of approximate CE. We note that this straightforward approach also gives a polynomial upper bound on the number of payoff queries needed to compute approximate CE (but not well-supported CE). We improve this straightforward polynomial dependence on n to an $O(\log n)$ dependence and give a matching lower bound. To our knowledge, we are the first to study bounds on computing well-supported CEs, which do not follow from no-regret guarantees.

1.3. Our Results and Techniques

Section 2 gives bounds on the query complexity of ϵ-CE in terms of n and ϵ, whereas Section 3 gives bounds on the query complexity of ϵ-WSCE in terms of n and ϵ. As functions of the number of players n, the bounds for ϵ-CE are logarithmic. For ϵ-WSCE, we have a linear lower bound, thus showing a separation between the two solution concepts. The following observation is a useful starting-point.

Observation 2. *For binary-choice games, the uniform distribution is a $\frac{1}{2}$-approximate NE and thus also an ϵ-approximate CE for any $\epsilon \geq \frac{1}{2}$.*

In Section 2, we show that for any $\epsilon < \frac{1}{2}$, the query complexity is $\Theta(\log n)$, showing that the constant $\frac{1}{2}$ in Observation 2 represents a crisp threshold at which the query complexity becomes nontrivial.

For (non–well-supported) CEs, our algorithms for query-efficient upper bounds use the multiplicative weights algorithm, applying it in two alternative ways. To obtain an

upper bound that works well for the case of many strategies per player (Section 2.1), a polynomial (in the number of players) query upper bound follows straightforwardly from an application of multiplicative weights: the standard proof is included in the Appendix. We note that this also yields an $O(m \log m)$ upper bound for the problem of computing an approximate NE in a two-player $m \times m$ zero-sum game, which is substantially smaller than the representation size of the game matrix.

The case of many players having just two strategies (Section 2.2) is more subtle: a straightforward application of multiplicative weights would yield a superlinear query complexity upper bound (in the number of players n), but as we show, this linear dependence on n is not necessary: we are instead able to achieve a logarithmic upper bound. The reason the naive application of multiplicative weights requires a linear number of queries is because at every round of the algorithm, it is necessary to estimate the payoff for every action of each of the n players. We observe that if we had a guarantee that at each stage of the algorithm every player was playing a mixed strategy that placed at least $\Omega(\epsilon)$ probability mass on each action, then it would be possible to estimate the payoffs of every player simultaneously from a single sample consisting of $\approx \log(n)/\epsilon^3$ randomly chosen value queries. Moreover, we can enforce this condition by having each player play according to multiplicative weights coupled with a Bregman projection into an appropriately defined convex polytope. We show that even with this projection, play still converges to an approximate CE. Using this technique, we obtain quite an efficient upper bound on the query complexity of computing an ϵ-CE using $O(\log n/\epsilon^5)$ queries. The lower bound of Section 2.3 shows that $\Omega(\log n)$ queries are indeed required for any $\epsilon < \frac{1}{2}$. Theorem 2.4 contrasts the exponential lower bound for deterministic algorithms [Babichenko and Barman 2013; Hart and Nisan 2013].

The lower bounds of Theorems 2.8 and 3.1 work by identifying distributions over the payoff functions of the target game so that Yao's minimax principle can be applied to algorithms having lower query complexity. Finally, we apply Theorem 3.1 to show that small-support approximate CEs are harder (in the query complexity sense) to find than are larger-support approximate CEs.

Finally, we give query-efficient reductions from the problem of computing an ϵ-approximate well-supported NE to the problem of verifying an ϵ-approximate well-supported NE (and hence CE), where the query overhead is proportional to the description length of the game. Since verifying equilibria can be done query efficiently, this gives query-efficient algorithms for computing approximate well-supported equilibria in concisely represented games. In special classes of games (including some congestion games) in which the payoff function can be represented as a linear function over polynomially many dimensions, this reduction can be made computationally efficient as well. The main technique is to use no-regret algorithms as mistake bounded learning algorithms for the underlying game, which is similar to how no-regret algorithms have been recently used in data privacy [Roth and Roughgarden 2010; Hardt and Rothblum 2010; Gupta et al. 2012]. We use the algorithms to learn a hypothesis game representation: at each stage, we compute an equilibrium of the hypothesis game and then make polynomially many queries to check if our computed equilibrium is an equilibrium of the real game. If it is, we are done; otherwise, we have forced the learning algorithm to make a mistake, which we charge to its mistake bound.

2. BOUNDS FOR THE QUERY COMPLEXITY OF ϵ-CE

We use the multiplicative weights algorithm to get upper bounds on the query complexity of coarse CE. The first one is applied to zero-sum bimatrix games, and the second one is applied to n-player binary-action games.

APPROX COARSE CE
Parameters: learning rate $\eta = \frac{\epsilon}{3}$; number of iterations $T = \frac{9 \ln m}{\epsilon^2}$.
$w_j^i(t)$ is the weight of the j-th strategy of player i at iteration t;
initially $w_j^i(1) = 1$ for all i, j. Do the following for iterations $t = 1, \ldots, T$:

(1) For each player i, let $\mathbf{p}^i(t) = (p_1^i(t), \ldots, p_m^i(t))$ be the probability distribution over i's strategies corresponding to the weights; $p_j^i(t) = w_j^i(t) / \sum_j w_j^i(t)$.
(2) For each player i, sample a pure strategy $s^i(t)$ from $\mathbf{p}^i(t)$. Let $\mathbf{s}(t) = \times_i s^i(t)$ be the resulting strategy profile.
(3) $\forall i$, query the payoffs of all responses to $\mathbf{s}(t)$. Let $m_j^i(t)$ be the cost (negated payoff) of strategy j.
(4) Update weights: for each i, j, set $w_j^i(t+1) = w_j^i(t)(1 - \eta m_j^i(t))$.

Output: The uniform distribution over the set $\{s(1), \ldots, s(T)\}$.

Fig. 1. Using (the naive) application of multiplicative weights to compute an approximate coarse CE.

2.1. Upper Bound for ϵ-Approximate Coarse Correlated Equilibrium; Few Players, Many Strategies

As a warmup, we consider a straightforward upper bound on the query complexity of computing approximate coarse CEs that follows from using no-regret algorithms. In the special case of two player zero-sum games, Freund and Schapire [1999] showed that two no-regret algorithms played against each other converge to an approximate NE. We here observe that this approach yields an upper bound for the query complexity of these equilibrium concepts. The proof is standard, and we include the algorithm and the proof in the Appendix for completeness.

Theorem 2.1 identifies useful bounds in the case of $m \gg n$; in particular, it has an interesting application to the case of two-player zero-sum games in Corollary 2.2.

THEOREM 2.1. *Let G be a game with n players, each with m pure strategies where $m \geq n$; payoffs lie in* $[0, 1]$. *With probability $1 - nm^{-\frac{1}{8}}$, the algorithm* APPROX COARSE CE *(Figure 1) finds an ϵ-approximate coarse CE of G using $O(\frac{nm \log m}{\epsilon^2})$ payoff queries.*

COROLLARY 2.2. *Let G be a $m \times m$ zero-sum bimatrix game. It is possible to efficiently compute (with high probability using a randomized algorithm) an ϵ-NE of G using $O(\frac{m \log m}{\epsilon^2})$ payoff queries.*

This follows from the observation that for zero-sum bimatrix games, an ϵ-approximate coarse CE ψ can be converted to a 2ϵ-NE by taking the product of the marginal distributions of ψ for each player. This result was recently extended to well-supported ϵ-NE [Fearnley and Savani 2014].

2.2. Upper Bound for n Players, Binary Actions

We noted at the end of Section 1.2 that no-regret algorithms can be applied directly to yield a polynomial upper bound on the query complexity of ϵ-CE of binary-choice games (more generally for ϵ-CCE when players have more than two actions). In this section, we take a more sophisticated approach to show a logarithmic upper bound. We will give a matching lower bound to show that this is optimal.

First, we give the intuition for the approach. The naive application of no-regret algorithms gives a query complexity that is linear in the number of players n because we must take samples to estimate the payoffs of the two actions for each of the n players. The improved algorithm in this section is based on the following two optimizations:

APPROX COARSE CE (2)
Parameters: learning rate $\eta = \frac{\epsilon}{6}$; number of iterations $T = \frac{36 \ln 2}{\epsilon^2}$; sample size $N = \frac{216}{\epsilon^3} \log\left(\frac{8nT}{\delta}\right)$, where $\delta = 1/(2 \cdot n)$.

(a) Let $\mathbf{p}_j^i(t)$ denote the probability mass on strategy $j \in \{0,1\}$ of player i at iteration t; initially $\mathbf{p}_j^i(1) = 1/2$ for all i, $j \in \{0,1\}$.
(b) $K_\epsilon = \{(p_0, p_1) \ : \ p_0 + p_1 = 1 \text{ and } \min(p_0, p_1) \geq \epsilon\}$ is the convex body defined to be the 2-dimensional probability simplex intersected with the constraints that p_0, p_1 must both be greater than or equal to ϵ.

Do the following for iterations $t = 1, \ldots, T$:

(1) For each player i, let $\hat{\mathbf{p}}^i(t) = \arg\min_{p \in K_{\epsilon/6}} \mathrm{RE}(p || \mathbf{p}^i(t))$. Here, RE is the relative entropy function.
(2) Let $\mathbf{p}(t)$ be the product distribution over mixed-strategy profiles that results from each player i playing $\hat{\mathbf{p}}^i(t)$.
(3) Sample N pure-strategy profiles $S(t) = \{\mathsf{s}_1(t), \ldots, \mathsf{s}_N(t)\}$ from $\mathbf{p}(t)$.
(4) Query the payoffs for all the profiles in $S(t)$. Let $\hat{m}_j^i(t)$ be the empirical cost of strategy j when used by player i. (In other words, letting $S_j^i(t) = \{s \in S \ : \ i \text{ plays } j \text{ in } S\}$, $m^i(s)$ be 1 minus the payoff to i for $s \in S_j^i(t)$, then $\hat{m}_j^i(t) = |S_j^i(t)|^{-1} \sum_{s \in S_j^i(t)} m^i(s)$. If i never plays j in the profiles $S(t)$, the algorithm fails.)
(5) Update weights: for each i, j, set $\mathbf{p}_j^i(t+1) = \hat{\mathbf{p}}_j^i(t)(1 - \eta \hat{m}_j^i(t))/\phi$, where ϕ is the normalization constant needed to make $\mathbf{p}^i$ a probability distribution.

Output: The distribution $\mathbf{p} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{p}(t)$.

Fig. 2. Using multiplicative weights to compute an approximate coarse CE.

(1) If the distribution over game states induced by the mixed strategies of all n players were such that each player had probability mass at least ϵ on each of his two actions, then it would be enough to take $O(\log(n)/\epsilon^3)$ samples from this joint distribution to simultaneously estimate the payoffs of all n players. We formalize this in Proposition 2.3.
(2) We can simulate play of the game in such a way so as to force each agent to play a mixed strategy that puts weight at least $\epsilon/4$ on each of their two actions. One way to do this is to have each player play multiplicative weights, coupled with a Bregman projection into the polytope $K_{\epsilon/4}$, where $K_{\epsilon/4}$ is the two-dimensional probability simplex over distributions p, intersected with the linear inequalities constraining $||p||_\infty \geq \epsilon/4$. The result is that the final empirical average of player strategies will always place minimum weight $\epsilon/4$ on any action, and each player will have no regret to the best mixed strategy in $K_{\epsilon/4}$ (e.g., see Arora et al. [2012]). However, because the total variation distance between *any* probability distribution and $K_{\epsilon/4}$ is at most $\epsilon/2$, this means that no player will have regret greater than $\epsilon/2$ to *any* strategy. Hence, the resulting play will be an $\epsilon/2$-approximate coarse CE, and therefore an ϵ-approximate CE.

The algorithm is given in Figure 2. Using standard concentration bounds, we can establish the following fact, which is proven in the Appendix.

PROPOSITION 2.3. *Let* s *be a mixed-strategy profile of an n-player two-action game having the property that for any player i and strategy j, i plays j with probability at least* γ. *Let* $\mathsf{s}_i(a)$ *be the expected payoff to i when i plays a and the other players play* s_{-i}.

With probability $1-\delta$, *we can find, with additive error* $\beta \le \gamma/2$, *all* $\mathsf{s}_i(a)$ *values using* N *payoff queries randomly sampled from* s, *whenever* $N \ge \max\{\frac{1}{\gamma\beta^2}\log(\frac{8n}{\delta}), \frac{8}{\gamma}\log(\frac{4n}{\delta})\}$.

THEOREM 2.4. *Let* G *be a game with* n *players, each with two actions; payoffs lie in* $[0,1]$. *For any* ϵ, *with probability* $1-\frac{2}{n}$, *the algorithm* APPROX COARSE CE (2) *(Figure 2) finds an* ϵ*-approximate coarse CE of* G *using* $\tilde{O}(\frac{\log n}{\epsilon^5})$ *payoff queries.*

PROOF. The algorithm APPROX COARSE CE (2) is simulating play of the n-player game where each player would be using the multiplicative weights update algorithm with restricted distributions of Arora et al. [2012], where each player is restricted to playing a distribution in $K_{\epsilon/6}$—except each player is receiving noisy losses $\hat{m}^i_j(t)$ computed from a sample of actions rather than the true losses $m^i_j(t)$. The guarantee of the multiplicative weights update algorithm with restricted distributions (e.g., see Theorem 2.4 in Arora et al. [2012]) gives us (where $\hat{m}^i(t)$ denotes the loss vector $(\hat{m}^i_j(t))_j$):

$$\frac{1}{T}\sum_{t=1}^{T}\hat{m}^i(t)\cdot\mathbf{p}^i(t) \le \min_{p\in K_{\epsilon/6}}\frac{1}{T}\sum_{t=1}^{T}\hat{m}^i(t)\cdot p + \eta + \frac{\ln 2}{\eta T}.$$

We now make several observations. First, note that for every probability distribution p, there exists a distribution $\hat{p}\in K_{\epsilon/6}$ such that $d_{TV}(p,\hat{p})\le\epsilon/3$, where d_{TV} represents the total variation distance. Therefore, since the losses $\hat{m}^i(t)\in[0,1]$, we can deduce

$$\frac{1}{T}\sum_{t=1}^{T}\hat{m}^i(t)\cdot\mathbf{p}^i(t) \le \min_{p}\frac{1}{T}\sum_{t=1}^{T}\hat{m}^i(t)\cdot p + \eta + \frac{\ln 2}{\eta T} + \epsilon/3,$$

which follows from Holder's inequality.

Similarly, as observed by Kearns et al. [2013], if we have that $\max_{i,j,t}|m^i_j(t)-\hat{m}^i_j(t)|\le\beta$, then we can deduce

$$\frac{1}{T}\sum_{t=1}^{T}m^i(t)\cdot\mathbf{p}^i(t) \le \min_{p}\frac{1}{T}\sum_{t=1}^{T}m^i(t)\cdot p + \eta + \frac{\ln 2}{\eta T} + \epsilon/3 + 2\beta.$$

Proposition 2.3 gave conditions for the observed losses to be within β of true losses.

Applying Proposition 2.3 with $\beta=\epsilon/6$ and $\gamma=\epsilon/6$, we have that with probability at least $1-\delta/T$, the following value of N is sufficient for observed losses to be within β of true losses at any iteration:

$$N \ge \frac{216}{\epsilon^3}\log\left(\frac{8nT}{\delta}\right). \tag{3}$$

We therefore have

$$\frac{1}{T}\sum_{t=1}^{T}m^i(t)\cdot\mathbf{p}^i(t) \le \min_{p}\frac{1}{T}\sum_{t=1}^{T}m^i(t)\cdot p + \eta + \frac{\ln 2}{\eta T} + 2\epsilon/3.$$

Plugging in our values for T and η bounds the regret of each player by at most

$$\frac{1}{T}\sum_{t=1}^{T}m^i(t)\cdot\mathbf{p}^i(t) - \min_{p}\frac{1}{T}\sum_{t=1}^{T}m^i(t)\cdot p \le \epsilon,$$

which proves the theorem. □

Theorem 2.4 generalizes to m actions at the cost of a factor-m increase in sample complexity. However, we focus on the two-action case, because in this case, an ϵ-approximate coarse CE is a 2ϵ-approximate CE. This yields the following result.

COROLLARY 2.5. *For binary-choice games (where each player has two pure strategies), for any ϵ, with probability $1 - \frac{1}{n}$, the algorithm* APPROX COARSE CE (2) *finds an ϵ-CE using $\tilde{O}(\frac{\log n}{\epsilon^5})$ payoff queries.*

2.3. Lower Bound for n Players, Binary Actions

For positive integer $k > 2$, we lower bound the number of queries needed to find a $(\frac{1}{2} - \frac{1}{k})$-approximate CE by fixing the number of queries to be $Q = \lfloor \log_{k(k-1)} n \rfloor$ and applying Yao's minimax principle. The cost of the output of algorithm $\mathcal{A}$ is the smallest value of ϵ for which $\mathcal{A}$'s output is an ϵ-approximate CE. We identify a distribution over payoff functions of n-player binary-action games such that the expected cost of the solution output by any algorithm $\mathcal{A}$ that uses Q queries (or fewer) is approximately $\frac{1}{2}$.

A distribution over payoff functions. Define the following probability distribution over payoffs of n-player binary-choice games. Each player $i \in [n]$ shall be a "type-0" player or a "type-1" player, with a player's type being obtained by flipping a fair coin. Let t_i be the type of player i. After types have been obtained, for each player i, construct i's payoff function as follows. For each strategy profile s_{-i} of players other than i, with probability $\frac{k-1}{k}$, player i gets paid 1 to play t_i and 0 to play $1 - \mathsf{t}_i$. With probability $\frac{1}{k}$, i gets paid 0 to play t_i and 1 to play $1 - \mathsf{t}_i$. Hence, for a (expected) fraction $\frac{k-1}{k}$ of profiles s_{-i}, player i has t_i as best response, with $1 - \mathsf{t}_i$ as the best response for the remaining profiles s_{-i}.

Notation. Let $\mathcal{A}$ be an algorithm that makes Q pure profile queries for $Q = \log_{k(k-1)} n$. Let $\mathsf{s}^1, \ldots, \mathsf{s}^Q$ be the queried profiles, where s^j is the j-th query in the sequence made by $\mathcal{A}$.

An important observation is that, conditional on a choice of player types, all payoff vectors are generated independently of each other. Consider the process of generating the payoff function as earlier and then querying it. That process is equivalent to one where the type vector is initially generated, then the algorithm selects various pure-strategy profiles to query, and then each time a pure-strategy profile is queried, we flip the biased coins that produce its payoff vector. This shows a useful limitation on how the answers to a sequence of queries can indicate what the answer will be to any subsequent query.

PROPOSITION 2.6. *For $0 \le j \le Q$, let p^j be the probability distribution over type vectors, conditioned on the answers to the first j queries (i.e., the players' payoffs for $\{\mathsf{s}^1, \ldots, \mathsf{s}^j\}$). Then p^j is a product distribution, $\mathsf{p}^j = \times_i \mathsf{p}_i^j$, where p_i^j is the probability that player i has type 1.*

PROOF. The claim follows by induction on j. Initially, p^0 is the uniform distribution. Subsequently, the payoffs for each queried profile s^j are obtained by making a (biased) coin flip independently for each player. (After the type vector has been selected, we can assume that the payoffs for a queried pure profile are generated by making biased coin flips independently of the results of previous queries.)

In particular, when $\mathcal{A}$ queries strategy profile s^j, $\mathcal{A}$ observes a payoff for each player i consisting of the result of acoin flip that is

—equal to 1 with probability $\frac{k-1}{k}$ and 0 with probability $\frac{1}{k}$ if $\mathsf{t}_i = \mathsf{s}_i^j$, and
—equal to 1 with probability $\frac{1}{k}$ and 0 with probability $\frac{k-1}{k}$ if $\mathsf{t}_i = 1 - \mathsf{s}_i^j$. □

OBSERVATION 3. *We say that the payoffs for* s^j *indicate that player* i *is type* $t \in \{0,1\}$ *if* i *plays 1 and gets paid* t*, or* i *plays 0 and gets paid* $1-t$*. Let* Q_t^i *be the number of queries that indicate that player* i *is type* t*. It is not hard to check that* $\Pr[\mathsf{t}_i = 1]/\Pr[\mathsf{t}_i = 0] = (k-1)^{(2Q_1^i - Q)}$*, where* $\Pr[\mathsf{t}_i = t]$ *is the probability that* i *has type* t*, conditioned on the data.*

Definition 2.7. $\mathcal{A}$ outputs a distribution ψ over pure-strategy profiles. Let ψ^u be ψ restricted to the unqueried profiles. We will say that $\mathcal{A}$ *has bias* b *for player* i if on profiles sampled from ψ^u, i plays 1 with probability b—that is, $\mathbb{E}_{x\sim\psi^u}[x_i] = b$, where x_i is the action played by i in profile x.

THEOREM 2.8. *Let* $k > 2$ *be an integer. Let* $\mathcal{A}$ *be a payoff-query algorithm that uses at most* $\log_{k(k-1)} n$ *queries, where* n *is the number of players, and outputs distribution* ψ*.*

With probability more than $\frac{1}{2}$*, there will exist a player* i *who would improve his payoff by* $\frac{1}{2} - \frac{1}{k}$ *by playing some fixed strategy in* $\{0,1\}$ *relative to the payoff he gets in* ψ*.*

PROOF. Assume that the payoffs for game G are generated by the distribution defined earlier.

We identify a lower bound on the probability that any individual player i is paid 0 in every profile in $\{\mathsf{s}^1, \ldots, \mathsf{s}^Q\}$ and has a type t_i that is in a sense "bad" for ψ.

We start by lower bounding the probability that a given player gets paid 0 on every query. To this end, suppose that $\mathcal{A}$ tries to maximize the probability that each player gets paid at least 1. This is done by selecting s_j that allocates to each player i the action that is more likely to pay 1 to i, conditioned on the answers to $\mathsf{s}_1, \ldots, \mathsf{s}_{j-1}$. However, since i's payoff is obtained by a coin flip that pays i zero with probability either $\frac{1}{k}$ or $\frac{k-1}{k}$, at each query there is probability at least $\frac{1}{k}$ that i will get paid 0. Hence, with probability at least $\frac{1}{k}^Q$, i is paid 0 on all queries.

Conditioned on the answers to all Q queries, we have (noting Observation 3) for any player i that $\Pr[\mathsf{t}_i = 1]/\Pr[\mathsf{t}_i = 0] \le (k-1)^Q$ and similarly $\Pr[\mathsf{t}_i = 0]/\Pr[\mathsf{t}_i = 1] \le (k-1)^Q$, where $\Pr[\mathsf{t}_i = t]$ is the conditional probability that t_i is equal to t. Therefore, any prediction of a player's type has probability at least $(k-1)^{-Q}$ of being incorrect, regardless of how much that player was paid during the queries.

Consider some player i who gets paid 0 on all Q queried profiles. Let b be the bias (Definition 2.7) for player i, and let λ be the probability that $x \sim \psi$ happens to be a queried profile. Thus, $\psi = \lambda\psi^q + (1-\lambda)\psi^u$, where ψ^q is a distribution over queried profiles and ψ^u is a distribution over unqueried profiles.

Let p_0 (respectively, p_1) be the probability that $x \sim \psi$ is a queried profile where i plays 0 (respectively, 1). Let p_0' (respectively, p_1') be the probability that $x \sim \psi$ is an unqueried profile where i plays 0 (respectively, 1). Thus, $p_0 + p_1 + p_0' + p_1' = 1$; $p_0 + p_1 = \lambda$; $p_0' = (1-\lambda)(1-b)$; $p_1' = (1-\lambda)b$.

Suppose that player i is paid 0 in all queried profiles:

—If $\mathsf{t}_i = 0$, then i's expected payoff under ψ is $p_0'\frac{k-1}{k} + p_1'\frac{1}{k}$.
—If $\mathsf{t}_i = 1$, then i's expected payoff under ψ is $p_1'\frac{k-1}{k} + p_0'\frac{1}{k}$.

Furthermore,

—If $\mathsf{t}_i = 0$ and i plays 0 against all profiles generated by ψ, i's expected payoff is

$$p_1 + (1-\lambda)\frac{k-1}{k} = p_1 + \frac{k-1}{k}(p_0' + p_1').$$

—If $t_i = 1$ and i plays 1 against all profiles generated by ψ, i's expected payoff is

$$p_0 + (1-\lambda)\frac{k-1}{k} = p_0 + \frac{k-1}{k}(p_0' + p_1').$$

Suppose that $t_i = 0$. i can improve his expected payoff by $p_1 + \frac{k-1}{k}(p_0' + p_1') - (p_0'\frac{k-1}{k} + p_1'\frac{1}{k})$ by playing 0 always. Suppose that $t_i = 1$. i can improve his expected payoff by $p_0 + \frac{k-1}{k}(p_0' + p_1') - (p_1'\frac{k-1}{k} + p_0'\frac{1}{k})$ by playing 1 always. Thus, there exists a value for t_i such that i's regret is at least

$$\max\left\{p_1 + \frac{k-1}{k}(p_0' + p_1') - \left(p_0'\frac{k-1}{k} + p_1'\frac{1}{k}\right),\ p_0 + \frac{k-1}{k}(p_0' + p_1') - \left(p_1'\frac{k-1}{k} + p_0'\frac{1}{k}\right)\right\},$$

which simplifies to

$$\max\left\{p_1 + \frac{k-2}{k}p_1',\ p_0 + \frac{k-2}{k}p_0'\right\}.$$

The sum of these two terms is $p_0 + p_1 + \frac{k-2}{k}(p_0' + p_1')$, and since $p_0 + p_1 + p_0' + p_1' = 1$, the sum of the terms is at least $\frac{k-2}{k}$, so at least one of the terms is at least $\frac{k-2}{2k}$, and hence the maximum of them is at least $\frac{k-2}{2k}$.

For a player i to have regret at least $\frac{k-2}{2k}$, it is sufficient for the following two events to occur: player i is paid 0 on all queried profiles, and player i turns out to have type t_i that leads to larger regret than the alternative type $1 - t_i$. The first of these events occurs with probability at least $\frac{1}{k}^Q$, and (given the first) the second occurs with probability at least $(\frac{1}{k-1})^Q$. Therefore, for any player i, with probability at least $(\frac{1}{k(k-1)})^Q$, i has regret at least $\frac{k-2}{2k}$. Thus, for $n \geq (k(k-1))^Q$ (i.e., $Q \leq \log_{k(k-1)} n$), a bad player exists with probability more than $\frac{1}{2}$. This uses the fact that for any two separate players, the events that both conditions hold for the two players are independent. □

3. BOUNDS FOR THE QUERY COMPLEXITY OF ϵ-WSCE

In Section 3.1, we present a lower bound that is linear in n. We then present an upper bound in Section 3.2 that applies to the class of concisely representable games and finds an ϵ-WSNE. In Section 3.3, we show that this upper bound takes the form of a generic reduction from the problem of computing ϵ-WSNE to the problem of verifying ϵ-WSNE and can be implemented computationally efficiently in certain games.

3.1. Lower Bound

To obtain a lower bound that applies to randomized algorithms, in a similar way to Theorem 2.8 we define an adversarial distribution over games in $\mathcal{G}_n$ and argue that given a deterministic algorithm $\mathcal{A}$ that uses $o(n)$ queries, $\mathcal{A}$ is likely to fail to find an ϵ-WSCE.

A distribution over payoff functions. Let $\mathcal{D}_n$ be the following distribution over $\mathcal{G}_n$. For each player i, and each bit vector $\mathbf{b}$ of length $i-1$, let $b_{i,\mathbf{b}} \in \{0,1\}$ be obtained by flipping a fair coin. If players $1,\ldots,i-1$ play $\mathbf{b}$, then i is paid $b_{i,\mathbf{b}} \in \{0,1\}$ to play 0 and $1 - b_{i,\mathbf{b}} \in \{0,1\}$ to play 1. Note that every game generated by $\mathcal{D}$ has a unique (pure) NE.

THEOREM 3.1. *For $\epsilon < 1$, the payoff query complexity of computing ϵ-WSCE of $\mathcal{G}_n$ (i.e., n-player two-action games with payoffs in $[0,1]$) is $\Omega(n)$.*

PROOF. For any game G in the support of $\mathcal{D}_n$, observe that G has a unique (pure) NE $\mathcal{N}$. $\mathcal{N}$ is found by considering each player i in ascending order, and noting that each

player is incentivized to select one of his actions based on the behavior of $1, \ldots, i-1$, and i's payoffs are not a function of the behavior of $i+1, \ldots, n$. Moreover, $\mathcal{N}$ can be seen to be the unique ϵ-WSCE of G.

For $G \sim \mathcal{D}_n$, let $\mathsf{s}_1, \ldots, \mathsf{s}_j$ be a sequence of query profiles for G and consider the conditional distribution $\mathcal{D}'$ on $\mathcal{G}_n$ that results from the answers to those queries. We claim that $\mathcal{D}'$ has the following property. Let m_j be the length of the longest prefix of players that all get paid 1 in some query in sequence $\mathsf{s}_1, \ldots, \mathsf{s}_j$,

$$m_j = \arg\max_m \{\exists q \in 1 \ldots j \,:\, \mathsf{s}_q \text{ pays 1 to } 1, \ldots, m\}.$$

Let s be the queried profile that pays 1 to players $1, \ldots, m_j$ and 0 to player $m_j + 1$. Consider $\mathcal{D}'$ restricted to strategy profiles where players $1, \ldots, m_j$ play as in s, and player $m_j + 1$ plays the opposite strategy to the one he plays in s. We claim that the resulting distribution on payoff functions for players $m_j + 2, \ldots, n$ consists of $\mathcal{D}_{n-(m_j+1)}$ (renumbering the players $1, \ldots, n-(m_j+1)$ referred to in $\mathcal{D}_{n-(m_j+1)}$, to $m_j + 2, \ldots, n$).

The claim can be proved by induction on j. Define m_j in terms of j as previously, suppose that $\mathcal{D}'$ has the claimed property, and consider query s_{j+1}, and consider the resulting value m_{j+1}. If $m_{j+1} = m_j$, then query s_{j+1} provides no additional information on the payoffs to players $m+2, \ldots, n$, conditioned on players $1, \ldots, m+1$ playing as per the restriction of $\mathcal{D}'$. Alternatively, if $m_{j+1} > m_j$, then the payoffs of s_{j+1} shows how players $1, \ldots, m_{j+1}+1$ should play in $\mathcal{N}$ but yields no further information about the remaining players. The claim follows.

Define the *progress* of query s_{j+1} to be equal to $m_{j+1} - m_j$, the increase in the length of the prefix of players whose behavior in $\mathcal{N}$ is identified due to s_{j+1}. Then the expected progress of each query is less than $1 + \sum_{r \geq 0} r/2^{r+1} = 2$. (A query can make progress by flipping the choice of the first player who is paid 0, and further progress may be made if subsequent players in the prefix are also paid 1. But a query will be incorrect with probability $\frac{1}{2}$ for each subsequent player.) Meanwhile, the total progress of all queries required to find an ϵ-WSCE is n. □

Babichenko et al. [2013, 2014] have recently studied the topic of computing approximate Nash/correlated equilibria that are simple in the sense of being probability distributions having small support (e.g., logarithmic in the number of players or strategies). For approximate CEs, the main open question is whether n-player two-action games have ϵ-CEs whose support depends only on ϵ but not n. We conclude this section by noting that Theorem 3.1 can be used to show (Corollary 3.2) that finding a small-support ϵ-CE may sometimes require more queries than an unrestricted one. Without loss of generality, we can view approximate CEs as uniform distributions over (multi)sets of action profiles S: note that multiplicative weights explicitly finds CEs of this form, and any CE can be put in this form with arbitrarily small loss in the approximation factor by sampling. We note that multiplicative weights finds a CE with a support size $|S|$ that depends on n: $|S| \geq \log(n)/\epsilon^2$. We show that no algorithm with polylogarithmic query complexity can find an approximate CE with support size $O(1/\epsilon)$ (independent of n). Of course, such a separation is vacuous in games in which there are no ϵ-CEs with support $O(1/\epsilon)$, but in any game that has a pure-strategy (ϵ)-NE, there is always a small support CE—in particular, one with support just 1.

Corollary 3.2. *The query complexity of computing ϵ-CE with support size $|S| \leq 1/\epsilon$ is $\Omega(n)$: strictly greater than the query complexity of computing ϵ-CE with support size $|S| > \log(n)/\epsilon^2$ for any constant ϵ.*

APPROX WSNE
Let V be the version space, initially $V = \mathcal{G}_n$, where $|\mathcal{G}_n| \leq 2^{p(n)}$.
Repeat the following until an output is obtained:

(1) Construct game G' as follows. For each profile x, each player i is paid the median payoff that i obtains in x for elements of V.
(2) Compute an $\frac{\epsilon}{2}$-WSNE $\mathcal{N}$ of G';
(3) For every player i, strategy j, let $\mathcal{N}_j^i$ be a distribution over strategy profiles obtained by sampling from $\mathcal{N}$ and setting i's strategy to j; let S_j^i be a set of pure profiles sampled uniformly at random from $\mathcal{N}_j^i$, where $|S_j^i| = (\frac{4}{\epsilon}) \log(2p(n))$; query all $x \in S_j^i$.
(4) If any x queried above is inconsistent with G' (in terms of the payoffs resulting from the query), then update V, else halt and output $\mathcal{N}$.

Fig. 3. Query-efficient algorithm for ϵ-WSNE.

PROOF. If ψ is an ϵ-CE that is supported over $|S|$ strategy profiles a, then ψ is also a $|S|.\epsilon$-WSCE. This is because the probability p of any action a with $a^i = j$ with positive probability in ψ must be at least $1/|S|$.

Since Theorem 3.1 gives an $\Omega(n)$ lower bound for computing $\Omega(1)$-WSCE, this in particular also implies an $\Omega(n)$ lower bound for computing a CE with support $O(1/\epsilon)$. This is in contrast to the $O(\log(n)/\epsilon^5)$ upper bound of Theorem 2.4 for computing CE with larger support (in particular, support $\log(n)/\epsilon^2$). □

3.2. Upper Bound

We give an upper bound on the query complexity of ϵ-WSNE (and hence ϵ-WSCE) that is polynomial in the number of players n and strategies m, together with the description length of the target game. We leave the query complexity of ϵ-WSCE for unrestricted n-player games (i.e., those that have no polynomial length description) open. Recall that for unrestricted games, the query complexity of ϵ-WSNE is exponential in n [Babichenko 2013] for constant ϵ and m. The class of games used by Babichenko [2013] is based on random walks on the n-dimensional hypercube; notice that to write down a description of a generic member of this class would require a string of length exponential in n. Thus, any polynomial query algorithm for ϵ-WSCE for general games would have to (unlike our algorithm) avoid also computing ϵ-WSNE. Note that in contrast, the algorithms APPROX COARSE CE and APPROX COARSE CE (2) do not require games to come from a concisely represented class.

Our upper bound applies just to the query complexity, and it remains an open problem whether a computationally efficient approach exists in general. It yields a positive result for a special case of the question of query complexity of ϵ-NE using randomized algorithms [Hart and Nisan 2013].

The algorithm APPROX WSNE of Figure 3 can be viewed as a query-efficient reduction from the problem of computing an ϵ-CE to the problem of verifying one using the following "halving algorithm" approach. The "concisely representable" constraint means that the number of n-player games is upper bounded by an exponential function of n. We maintain a "version space," the set of all games that are consistent with queries that have been made so far. If we can find a query that is inconsistent with some constant fraction of the games in the version space, then we reduce the size of the version space by a constant fraction, and hence only polynomially many such queries are sufficient to identify the target game. In each iteration of the algorithm, we construct a proposed solution $\mathcal{N}$ (a probability distribution over strategy profiles), and we sample from it.

We query the payoffs associated with each sample. With high probability, if $\mathcal{N}$ is not a valid ϵ-WSNE, it will generate a sample that is inconsistent with at least half the elements of the version space.

THEOREM 3.3. *Let $\mathcal{G}_n$ be a class of n-player, m-strategy games whose elements can be represented using bit strings of length at most $p(n)$. With probability at least $\frac{1}{2}$, the algorithm* APPROX WSNE *(Figure 3) identifies an ϵ-WSNE using $O(nmp(n)(\log p(n))/\epsilon)$ payoff queries.*

PROOF. We prove that at each iteration, with high probability, either a profile is queried that is inconsistent with at least half of the elements of V, or the algorithm halts and outputs an ϵ-WSNE of the target game. Hence, the number of iterations is at most $p(n)$.

We consider two cases. Let $G(x)$ denote the payoff vector for game G on profile x. Let G^* be the target game.

Case 1: For all i, j, $\Pr_{x\sim\mathcal{N}_j^i}[G^*(x) \neq G'(x)] < \frac{\epsilon}{4}$.

It follows from the condition of case 1 that for any i, j, the payoff that i gets for j in response to $\mathcal{N}$ in game G^*, is within $\frac{\epsilon}{4}$ of the payoff that i gets for j in response to $\mathcal{N}$ in game G'.

Since $\mathcal{N}$ is an $\frac{\epsilon}{2}$-WSNE of G', if i plays j with positive probability in $\mathcal{N}$, then the payoff i gets for j in game G' in response to $\mathcal{N}$ is at most $\frac{\epsilon}{2}$ less than the payoff i gets for any j' in game G' in response to $\mathcal{N}$.

It follows that if i plays j with positive probability in $\mathcal{N}$, then the payoff that i gets for j in $\mathcal{N}$ in game G^* is at most ϵ less than the payoff that i get for any j' in $\mathcal{N}$ in game G^*. Hence, $\mathcal{N}$ is an ϵ-WSNE of G^*, so $\mathcal{N}$ constitutes an acceptable output. There is also a small probability that some x is found for which $G^*(x) \neq G'(x)$. By construction of G', the payoffs resulting from the query of x are inconsistent with at least half of the elements of V.

Case 2: For some i, j, $\Pr_{x\sim\mathcal{N}_j^i}[G^*(x) \neq G'(x)] \geq \frac{\epsilon}{4}$.

In this case, it is not hard to check that $|S_j^i|$ is large enough that with probability more than $1 - \frac{1}{2p(n)}$, S_j^i contains a pure profile x whose payoffs under G^* differ from the payoffs under G'. By construction of G', the query of x is inconsistent with at least half of the elements of V.

Since there are at most $p(n)$ iterations, the overall failure probability (an iteration where Case 2 arises and the sample does not contain x for which $G^*(x) \neq G'(x)$) is at most $\frac{1}{2}$. The number of queries at each iteration is $nm(\frac{4}{\epsilon})\log(2p(n))$. □

3.3. A Class of Efficient Algorithms

In this section, we build on the ideas behind our query-efficient algorithm for finding ϵ-WSNE and instantiate an efficient version of it for classes of games that have payoff functions with concise linear representations. The motivating scenario here is that we have convenient black-box access to a game's payoff function: as discussed in Fearnley et al. [2013], such a function may arise as a software program P simulating some game of interest, and we do not understand the details of P. However, we suspect that the function computed by P can in fact be expressed as a linear function of certain unknown features. For example, P may represent a congestion game of a kind that we analyze in the following. Consider how our algorithm APPROX WSNE worked:

(1) We had a mechanism to generate some hypothesis game G' given a collection of play profiles x queried so far.

(2) We compute a WSNE $\mathcal{N}$ of G' and query the unknown game G^* to check if $\mathcal{N}$ is an ϵ-WSNE of G^*. If yes, we output $\mathcal{N}$. If no, we update our hypothesis G' with the new queries we have made and repeat.

The algorithm works because every time we compute a distribution $\mathcal{N}$ that is a WSNE of G' but not of G^*, we find a profile x that has payoff differing between G^* and G' by at least $\epsilon/2$: in other words, a query that witnesses that our algorithm for predicting a hypothesis G' made a significant mistake. Moreover, we have a polynomial upper bound on how many mistakes our prediction algorithm can make. The running time of the algorithm is dominated by two computations: generating the hypothesis G' and computing the WSNE of G'. In our general reduction from the previous section, both are computationally expensive.

This framework suggests a more general approach. We can instantiate it with any mistake bounded learning algorithm for player payoff functions. First, let us define the mistake bounded model of learning for real valued functions.

Definition 3.4. A sequence of *labeled examples* is a collection of pairs (x^i, y^i) where $x^i \in \mathbb{R}^d$ is the example and $y^i \in \mathbb{R}$ is the label.

A mistake bounded learning algorithm takes as input an arbitrary sequence of labeled examples $(x^1, y^1), \ldots, (x^t, y^t)$ generated adaptively. Its goal is to predict the label of each example before seeing any subsequent example. Formally, a mistake bounded learning algorithm produces a guess $\hat{y}^i$ about the label of example i as a function of x^i and of all previously seen labeled examples: $\hat{y}_i = g(x^1, y^1, \ldots, x^{i-1}, y^{i-1}, x^i)$ for some arbitrary function g. (Crucially, g cannot depend on either the true label y^i or on any future example (x^j, y^j) for $j > i$.) Given a prediction $\hat{y}^i$ of label y^i, we say that the learning algorithm made an ϵ-mistake if $|y^i - \hat{y}^i| \geq \epsilon$.

It is of course not possible to accurately predict labels unless we assume some functional relationship between the examples x^i and their labels y^i.

Definition 3.5. A set of labeled examples $(x^1, y^1), \ldots, (x^t, y^t)$ is consistent with a function class C if there exists some function $f \in C$ such that $y_i = f(x^i)$ for all i.

A function $f : \mathbb{R}^d \to \mathbb{R}$ is *linear* if there exists a vector $y := y^f \in \mathbb{R}^d$ such that $f(x) = \langle y^f, x \rangle$ for all $x \in \mathbb{R}^d$. We say that a linear function f has ℓ_1 norm B, written $||f||_1 \leq B$ if $||y^f||_1 \leq B$. Let $L_B = \{f : \mathbb{R}^d \to \mathbb{R} : \exists y : f(x) = \langle y, x \rangle \wedge ||y||_1 \leq B\}$ denote the set of linear functions with norm bounded by B.

We say that there exists a mistake bounded learner for a class of functions C, with mistake bound $m(\epsilon, C)$ if for any fixed ϵ, on any sequence of examples consistent with C, the learning algorithm makes at most $m(\epsilon, C)$ ϵ-mistakes.

In this section, we will make use of the following fact (which follows from application of standard techniques and is proven in the Appendix for completeness).

COROLLARY 3.6. *There is an algorithm called* MWMBOUND *that is a mistake bounded learning algorithm for the set of d-dimensional linear functions with ℓ_1 norm bounded by B (denoted L_B) with mistake bound*

$$m(\epsilon, L_B) \leq \frac{4B^2 \log d}{\epsilon^2}.$$

This corollary is useful whenever the payoff function for each player can be represented as some known linear function of the action profile.

Definition 3.7. An n-player game is efficiently linearizable in d dimensions with norm B if there exists a linear function $f : \mathbb{R}^d \to \mathbb{R}$ defined by $f(x) = \langle y^f, x \rangle$ such that

(1) $y^f \in \mathbb{R}^d$ satisfies $||y^f||_1 \leq B$.
(2) There is a polynomial-time computable function $\phi : [n] \times A \to \mathbb{R}^d$ such that $u^i(a) = f(\phi(i, a))$ for all $a \in A$ and $i \in [n]$.
(3) For every $a \in A$ and $i \in [n]$, $||\phi(i, a)||_\infty \leq 1$.

Note that although we have defined ϕ in terms of pure action profiles, it naturally extends to mixed action profiles: given a distribution s over action profiles a, we can define $\phi(i, s) := E_{a\sim s}[\phi(i, a)]$ to be the expected value of ϕ when an action profile a is drawn from s. Since f is linear, this leads to a correct representation of the expected payoff of a.

Whenever we have a game that is efficiently linearizable in polynomially many dimensions, we can use MWMBound to learn the function f as part of a query-efficient algorithm for computing an ϵ-WSNE. Moreover, whenever an ϵ-WSNE can be efficiently computed for every hypothesis linearization of the game, the entire algorithm will be computationally efficient.

Specifically, suppose that we have a learning algorithm A that for any sequence of payoff-query/answer pairs $(a^1, y^1), \ldots, (a^t, y^t)$ produces a set of hypothesis utility functions $u^1, \ldots, u^n$ that map play profiles a to payoff values $u^i(a)$, one for each player. (In other words, the payoff functions represent a hypothesis game G'.) Say that the algorithm A makes a mistake with respect to a game G if it errs on its prediction $u^i(x)$ of the payoff of some queried profile a and player i by more than $\epsilon/2$. Suppose furthermore that for any game G in a restricted class, it is guaranteed that A can never make more than m mistakes. We then have an algorithm for computing ϵ-WSNE using only $m \cdot Q$ queries, where Q is the number of queries needed to check if a distribution $\mathcal{N}$ is an ϵ-WSNE. Moreover, if algorithm A is efficient, and WSNE can be computed efficiently for every hypothesis game G' generated in this manner, then the reduction is computationally efficient.

In Lemma 3.8, we identify conditions under which MWMBound can be used to obtain new query bounds for certain classes of games. The approach is motivated by the algorithm Approx WSNE, in which at each step t, a game G^t is constructed, an equilibrium $\mathcal{N}^t$ is obtained for G^t, and if $\mathcal{N}^t$ does not solve the target game, we find an informative strategy profile obtained by querying responses to $\mathcal{N}^t$.

Lemma 3.8. *Suppose that a target game G^* belongs to a class $\mathcal{G}$ of games where*

(1) *Any $G \in \mathcal{G}$ is efficiently linearizable with norm B, and*
(2) *Given any strategy profile, Q queries are sufficient to find an ϵ-better response for some player, assuming one exists.*

Then for target game G^*, an ϵ-WSNE of G^* can be found using $O(Q \cdot B^2 \log(d)/\epsilon^2)$ queries. Furthermore, if ϵ-WSNE can be computed efficiently, we also have that the search is efficient.

Proof. Consider the algorithm Approx NE of EL Games (Figure 4). Let f^* denote the linear function corresponding to the efficient linearization of G^*. To find an ϵ-NE, it is sufficient to find a game whose payoff function is given by f' that approximates f^* in the sense that $|f'(\phi(i, a)) - f^*((\phi(i, a))| \leq \epsilon$ for all $i \in [n]$ and $a \in A$. To do this, it is sufficient that $|f'(x) - f^*(x)| \leq \epsilon$ for all $x \in [0, 1]^d$. This reduces the problem to approximately learning a linear function.

We utilize MWMBound to learn f^* in the mistake bounded model. At each step t, we define the game G^t to be the game whose linearization is given by the current MWMBound hypothesis h^t: $f'(x) = \langle x, h^t \rangle$. Since this game is fully specified, we can compute an $\epsilon/2$-WSNE $\mathcal{N}^t$ of G^t. By hypothesis, we can verify whether $\mathcal{N}^t$ is indeed an ϵ-WSNE of the true game G^* by making at most Q queries to G^*. At any stage t, if $\mathcal{N}^t$ is an

APPROX NE OF EL GAMES
Input: d-dimensional linearization function ϕ for the efficiently linearizable game G^*.
Initialize MWMBOUND, and obtain the initial MWMBOUND hypothesis $h^1 \in \mathbb{R}^d$. Define G^1 to be the game with utility functions $\hat{u}^i : A \to \mathbb{R}$ such that $\hat{u}^i(a) = \langle \phi(i,a), h^1 \rangle$.
For $t = 1, 2, \ldots,$ do the following until a solution is found:

(1) Compute an $\frac{\epsilon}{2}$-NE $\mathcal{N}^t$ of G^t.
(2) Query Q action profiles to G^* to check whether any player can improve his payoff by $> \epsilon$ (in G^*); If so, let a be the strategy profile resulting from this deviation.
(3) If not, halt and output $\mathcal{N}^t$.
(4) Otherwise, For each player i, feed $(\phi(i,a), u^i(a))$ and $(\phi(i,\mathcal{N}^t), u^i(\mathcal{N}^t))$ to MWMBOUND and let $h^{t+1} \in \mathbb{R}^d$ be the MWMBOUND hypothesis that results.
(5) Let game G^{t+1} have payoff function $\hat{u}^i : A \to \mathbb{R}$ such that $\hat{u}^i(a) = \langle \phi(i,a), h^{t+1} \rangle$.

Fig. 4. Query-efficient algorithm to find ϵ-NE of efficiently linearizable games.

ϵ-WSNE, we are done. Otherwise, since $\mathcal{N}^t$ is an $\epsilon/2$-WSNE of G^t, there must be some player i and some action profile a that results in player i playing a unilateral deviation from $\mathcal{N}^t$ such that either $|f'(\phi(i,a)) - f^*(\phi(i,a))| \geq \epsilon$ or $|f'(\phi(i,\mathcal{N}^t)) - f^*(\phi(i,\mathcal{N}^t))| \geq \epsilon$. This is an example on which MWMBOUND makes an $\epsilon/2$-mistake. Since at each round our algorithm gives each of these labeled examples to MWMBOUND, and by Corollary 3.6, MWMBOUND can make at most $O(B^2 \log(d)/\epsilon^2)\epsilon/2$-mistakes. This provides an upper bound on the number of rounds that the algorithm can run. Since it makes at most Q queries in each round, the theorem follows. □

We give an example of a class of games to which the preceding lemma can be usefully applied. These are network congestion games of the kind studied in Fearnley et al. [2013], where the costs of edges are unknown, increasing functions of the number of players using them, so that these costs need to be learned to compute an equilibrium. In the case of n players sharing a directed acyclic graph with m edges, Fearnley et al. [2013] show how an exact equilibrium can be found using mn queries. For approximate NE, we next give a query bound dependent only on ϵ and the number of facilities.

Definition 3.9. An n-player congestion game is defined by m facilities F. Each player has an action set $A_i \subseteq 2^F$ consisting of subsets of facilities. Given an action profile a, let $w(a) \in \{0, 1, \ldots, n\}^m$ denote the histogram of usage of each facility at action profile a (i.e., $w(a)_j = |\{i : j \in a_i\}|$). Each facility j has a cost function $\ell_j : \{0, 1, \ldots, n\} \to [0, 1]$ mapping the usage of that facility to a nonnegative cost. Given an action profile a, the cost of agent i is the sum of the costs of the facilities that he is using: $c^i(a) = \sum_{j \in a_i} \ell_j(w(a)_j)$. We write $\ell_j(0) = 0$ for all facilities j.

We observe that n-player congestion games with m facilities is efficiently linearizable with norm m in $n \cdot m$ dimensions. Define $y \in \mathbb{R}^{n \cdot m}$ as follows: for $1 \leq i \leq n, 1 \leq j \leq m$, $y_{i,j} = \ell_j(i) - \ell_j(i-1)$. For each player i and action profile a, define $\phi(i,a) \in \mathbb{R}^{n \times m}$ as follows: for $j \in a_i$, let $\phi(i,a)_{k,j} = 1$ if $w(a)_j \geq k$ and $\phi(i,a)_{k,j} = 0$ otherwise. For $j \notin a_i$, let $\phi(i,a)_{k,j} = 0$ for all $0 \leq k \leq n$. Note that for all i: $c^i(a) = \langle y, \phi(i,a) \rangle$, $||y||_1 \leq m$, and $||\phi(i,a)||_\infty = 1$ as required for an efficiently linearizable game. Hence, we have the following theorem.

THEOREM 3.10. *Consider n-player congestion games over m facilities, where we think of m as being a constant. Assume that each facility j has an increasing cost function ℓ_j that takes values in* $[0, 1]$. *The number of queries needed to find ϵ-WSNE is at most* $O(2^{2m} \cdot m^2 \cdot \log(m \cdot n)/\epsilon^2)$.

PROOF. As we saw earlier, such games are efficiently linearizable with norm m in $n \cdot m$ dimensions. We can therefore invoke Lemma 3.8. With regard to the value of Q, given a specific pure profile x, to find an ϵ-better response for some player if it exists, it suffices to query the cost of every action and deviation in $A_1 \cup \ldots \cup A_n$. Since each $A_i \subseteq 2^F$, there are at most 2^{2m} such pairs and hence $Q \leq 2^{2m}$ such queries that need to be made. Finally, efficient computation of ϵ-NE of these games can be done using ϵ-best response dynamics—this requires making only $O(n \cdot m/\epsilon)$ calls to a best-response oracle for each player, and best responses can be computed in time 2^m using a brute force search (or using a shortest path algorithm if the game is a network routing game). □

4. CONCLUSIONS AND FURTHER WORK

Query complexity provides a useful criterion for distinguishing the relative difficulty of alternative solution concepts in game theory. Here, we have used this criterion to formally separate the complexity of finding approximate CEs and finding approximate well-supported CEs. We have extended the analytical toolkit for query complexity and have proven a polynomial upper bound for computing well-supported NEs (Theorem 3.3) that contrasts with exponential lower bounds in recent work, provided that the game in question has a concise representation. We applied this idea in Section 3.3 to get a robust and generic method for efficiently finding ϵ-well-supported NEs of certain congestion games.

Our work leaves open the intriguing question of whether finding well-supported CEs is easier (from a query complexity standpoint) than finding well-supported NEs. Is there a polynomial upper bound for well-supported CEs for arbitrary (nonconcisely represented) games? Are there efficient game dynamics that converge to such equilibria?

APPENDIXES

A. PROOFS OF RESULTS FROM SECTION 2

THEOREM 2.1. *Let G be a game with n players, each with m pure strategies where $m \geq n$; payoffs lie in* $[0, 1]$. *With probability* $1 - nm^{-\frac{1}{8}}$, *the algorithm* APPROX COARSE CE *(Figure 1) finds an ϵ-approximate coarse correlated equilibrium of G using* $O(\frac{nm \log m}{\epsilon^2})$ *payoff queries.*

PROOF. APPROX COARSE CE applies, for each player, the multiplicative weights algorithm as presented in Arora et al. [2012]. For iteration $t = 1, \ldots, T$, $w_j^i(t)$ denotes the weight of player i's action j at step t; initially, $w_j^i(1) = 1$ for all i, j. $\mathbf{p}^i(t)$ denotes the probability distribution over i's pure strategies in which the probability of a strategy is proportional to its weight. At iteration t, each player i observes a cost vector $\mathbf{m}^i(t)$ (each strategy $j \in [m]$ has a cost (negated payoff) $m_j^i(t) \in [-1, 0]$) and weights are updated according to the rule $w_j^i(t+1) = w_j^i(t)(1 - \eta m_j^i(t))$, where $\eta \leq \frac{1}{2}$ is the learning rate parameter.

The performance guarantee of multiplicative weights, as given in Theorem 2.1 of Arora et al. [2012], is that after T rounds, for any strategy j of player i, we have

$$\sum_{t=1}^{T} \mathbf{m}^i(t) \cdot \mathbf{p}^i(t) \leq \sum_{t=1}^{T} m_j^i(t) + \eta \sum_{t=1}^{T} |m_j^i(t)| + \frac{\ln m}{\eta}. \tag{4}$$

Dividing by T and noting that $|m_j^i(t)| \leq 1$, we have

$$\frac{1}{T}\sum_{t=1}^{T} \mathbf{m}^i(t) \,.\, \mathbf{p}^i(t) \leq \frac{1}{T}\sum_{t=1}^{T} m_j^i(t) + \eta + \frac{\ln m}{T\eta}. \tag{5}$$

Thus, the expected cost incurred using the $\mathbf{p}^i(t)$ distributions (the LHS) is upper bounded by the cost that would be incurred by using any fixed strategy j (the first term of the RHS) plus the remaining terms of the RHS.

Approx Coarse CE obtains the costs $\mathbf{m}^i(t)$ by sampling, for each player i, a single pure strategy from $\mathbf{p}^i(t)$. This gives us a pure-strategy profile $\mathsf{s}(t)$. nm payoff queries are used at Step 3 to obtain the payoffs (or costs) of all responses to $\mathsf{s}(t)$. These values are used for each player to update his weights. Finally, the output of the algorithm is the uniform distribution over the elements of multiset $\{\mathsf{s}(t) \;:\; 1 \leq t \leq T\}$. We show that with high probability, the payoffs under this distribution approximate the payoffs of $\sum_{t=1}^{T} \mathbf{m}^i(t) \,.\, \mathbf{p}^i(t)$.

Let c^i be the average cost per iteration for player i using Approx Coarse CE, and thus $c^i = \frac{1}{T}\sum_{t=1}^{T} \mathbf{m}^i(t) \,.\, \hat{\mathbf{p}}^i(t)$, where $\hat{\mathbf{p}}^i(t)$ is a unit vector with a 1 at the entry corresponding to $s^i(t)$ (defined in Step 2), sampled from $\mathbf{p}(t)$. Note that

$$\mathbb{E}[c^i] = \frac{1}{T}\sum_{t=1}^{T} \mathbf{m}^i(t) \,.\, \mathbf{p}^i(t), \tag{6}$$

where expectation is over random choice of $s^i(t) \sim \mathbf{p}^i(t)$ at Step 2. Azuma's inequality tells us that if $\{X_t : t = 0, 1, 2, \ldots\}$ is a martingale and $|X_t - X_{t-1}| < c_t$, then for all positive integers N and all positive reals d,

$$\Pr[X_N - X_0 \geq d] \leq \exp\left(\frac{-d^2}{2\sum_{t=1}^{N} c_t^2}\right). \tag{7}$$

Here we let X_t be the cost that player i pays at iteration t minus its expected value; $X_t = \mathbf{m}^i(t) \,.\, \hat{\mathbf{p}}^i(t) - \mathbf{m}^i(t) \,.\, \mathbf{p}^i(t)$. Thus, $\mathbb{E}[X_t] = 0$, and since $X_t \in [-1, 1]$, we can use $c_t = 2$ for all t.

We upper bound c^i by using (7) to show that c^i is (usually) not too much higher than its expected value given in (6), which itself has the upper bound of (5). We can write

$$\Pr[c^i - \mathbb{E}[c^i] > \beta] = \Pr[X_N - X_0 > N\beta] \leq \exp\left(-\frac{1}{8}N\beta^2\right).$$

With $N = T$, we get the following bound for c^i:

$$\Pr\left[c^i > \frac{1}{T}\sum_{t=1}^{T} m_j^i(t) + \eta + \frac{\ln m}{T\eta} + \beta\right] < \exp\left(-\frac{1}{8}\beta^2 T\right).$$

Putting $T = \frac{\ln m}{\min\{\eta\beta, \beta^2\}}$, we have

$$\Pr\left[c^i > \frac{1}{T}\sum_{t=1}^{T} m_j^i(t) + \eta + \beta + \beta\right] < \exp\left(-\frac{1}{8}\ln m\right).$$

The RHS is upper bounded by $m^{-\frac{1}{8}}$; by a union bound, with probability at least $1-nm^{-\frac{1}{8}}$, the costs of every player i obey

$$c^i \leq \frac{1}{T}\sum_{t=1}^{T}\mathbf{m}_j^{(t)} + \eta + \beta + \beta.$$

Thus, to get an ϵ-approximate coarse CE, we need η and β to satisfy $\eta + \beta + \beta < \epsilon$. Using $\eta = \beta = \epsilon/3$, we get $T = O(\frac{\ln m}{\epsilon^2})$ as required (noting that nm queries are used at each iteration). □

In the following result, we use a standard bound on the probability that a sum X of N independent random variables taking values in $[0, 1]$ is less than its expectation μ by some factor:

$$\Pr[X \leq (1-\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2}{2}\mu\right). \tag{8}$$

Putting $\epsilon = \frac{1}{2}$ we get:

$$\Pr[X \leq \mu/2] \leq \exp\left(-\frac{1}{8}\mu\right). \tag{9}$$

We will also use Hoeffding's inequality, in the context of estimating an expected value from samples, where values are known to lie in $[0, 1]$. Let μ be the expected payoff, and let $\hat{\mu}$ be the empirical payoff based on N samples:

$$\Pr[|\mu - \hat{\mu}| \geq \beta] \leq 2\exp(-2\beta^2 N). \tag{10}$$

PROPOSITION 2.3. *Let* s *be a mixed-strategy profile of an n-player two-action game, having the property that for any player i and strategy j, i plays j with probability at least* γ. *Let* $\mathsf{s}_i(a)$ *be the expected payoff to i when i plays a and the other players play* s_{-i}.

With probability $1-\delta$, *we can find, with additive error* $\beta \leq \gamma/2$, *all* $\mathsf{s}_i(a)$ *values, using N payoff queries randomly sampled from* s, *whenever*

$$N \geq \max\left\{\frac{1}{\gamma\beta^2}\log\left(\frac{8n}{\delta}\right), \frac{8}{\gamma}\log\left(\frac{4n}{\delta}\right)\right\}.$$

PROOF. N payoff queries are obtained by sampling repeatedly from s. N is chosen to be large enough to ensure that for each player $i \in [n]$, each $a \in \{0, 1\}$, with probability $1 - \delta/4n$, we make at least N' queries in which i plays a. In turn, N' is large enough to ensure that with probability $1 - \delta/4n$, the average payoff that player i gets for action a is within β of the true expected payoff $\mathsf{s}_i(a)$. Using (10), it is sufficient for N' to satisfy $2\exp(-2\beta^2 N') \leq \delta/4n$, equivalently:

$$N' \geq \frac{1}{2\beta^2}\log\left(\frac{8n}{\delta}\right).$$

Fix $i \in [n]$, $a \in \{0, 1\}$. Let $X = \sum_{j=1}^{N} X_j$, where $X_j = 1$ if player i plays a and 0 if i plays $1 - a$. We want $X \geq N'$ with probability $1 - \frac{\delta}{4n}$.

We know that $\mathbb{E}[X_j] \geq \gamma$, so $\mathbb{E}[X] \geq N\gamma$. Choose N large enough such that $\mathbb{E}[X] \geq 2N'$ (so we can use (9) to get a lower bound on the probability that we get N' observations of i playing s_i); it suffices to use

$$N \geq 2N'/\gamma.$$

We also want from (9) that $\exp(-\frac{1}{8}N\gamma) \leq \frac{\delta}{4n}$, equivalently,

$$N \geq \frac{8}{\gamma}\log\left(\frac{4n}{\delta}\right).$$

Taken in conjunction with $N \geq 2N'/\gamma$, we have

$$N \geq \max\left\{\frac{2N'}{\gamma}, \frac{8}{\gamma}\log\left(\frac{4n}{\delta}\right)\right\}.$$

Plugging in the expression that we obtained for N', we get the expression for N in the statement. □

B. USING MULTIPLICATIVE WEIGHTS TO LEARN LINEAR FUNCTIONS

Here we show how the multiplicative weights learning algorithm can be used as a mistake bounded learning algorithm to learn a linear function f over a d-dimensional space, which we refer to as MWMBound. The analysis follows from the standard regret bound of multiplicative weights (e.g., see Arora et al. [2012]).

First, we formally define the mistake bounded model of learning for linear functions $f : \mathbb{R}^d \to \mathbb{R}$ over a d-dimensional real valued domain.

Definition B.1. A sequence of *labeled examples* is a collection of pairs (x^i, y^i), where $x^i \in \mathbb{R}^d$ is the example and $y^i \in \mathbb{R}$ is the label.

The mistake bounded learning algorithm takes as input an arbitrary sequence of labeled examples $(x^1, y^1), \ldots, (x^t, y^t)$ generated adaptively. Its goal is to predict the label of each example before seeing any subsequent example. Formally, a mistake bounded learning algorithm produces a guess $\hat{y}^i$ about the label of example i as a function of x^i and of all previously seen labeled examples: $\hat{y}_i = g(x^1, y^1, \ldots, x^{i-1}, y^{i-1}, x^i)$ for some arbitrary function g. (Crucially, g cannot depend on either the true label y^i or on any future example (x^j, y^j) for $j > i$.) Given a prediction $\hat{y}^i$ of label y^i, we say that the learning algorithm made an ϵ-mistake if $|y^i - \hat{y}^i| \geq \epsilon$.

It is of course not possible to accurately predict labels unless we assume some functional relationship between the examples x^i and their labels y^i.

Definition B.2. A set of labeled examples $(x^1, y^1), \ldots, (x^t, y^t)$ is consistent with a function class C if there exists some function $f \in C$ such that $y_i = f(x^i)$ for all i.

A function $f : \mathbb{R}^d \to \mathbb{R}$ is *linear* if there exists a vector $y := y^f \in \mathbb{R}^d$ such that $f(x) = \langle y^f, x\rangle$ for all $x \in \mathbb{R}^d$. We say that a linear function f has ℓ_1 norm B, written $||f||_1 \leq B$ if $||y^f||_1 \leq B$. Let $L_B = \{f : \mathbb{R}^d \to \mathbb{R} : \exists y : f(x) = \langle y, x\rangle \wedge ||y||_1 \leq B\}$ denote the set of linear functions with norm bounded by B.

We say that there exists a mistake bounded learner for a class of functions C, with mistake bound $m(\epsilon, C)$, if for any fixed ϵ, on any sequence of examples consistent with C, the learning algorithm makes at most $m(\epsilon, C)$ ϵ-mistakes.

In the following, we show that L_1, the set of linear functions with ℓ_1-norm 1, is efficiently mistake bound learnable. A scaling argument generalizes this to L_B for any finite B.

For now, let us assume that the coefficients of f are nonnegative and have L_1 norm 1. (If this is not the case, we can simply reduce to this case by (a) doubling the feature space to introduce "negative" versions of each of the variables and (b) rescaling the function to have L_1 norm 1, then scaling back afterward.) Let us also assume that for every example x, $||x||_\infty \leq 1$. (This can also be achieved by rescaling.)

MWMBound
Initialize $h^1 = (1/d, \ldots, 1/d) \in \mathbb{R}^d$, $\eta(t) = \sqrt{\log(d)/t}$.
For each example (x^t, y^t), $t = 1, 2, \ldots$ do the following:
(1) Predict $\hat{y}^t = \langle x^t, h^t \rangle$.
(2) If $\hat{y}^t > y^t + \epsilon$ (Mistake—prediction too big.)
    (a) For each i, set $\hat{h}_i^{t+1} = h_i^t \cdot \exp(-\eta(t) x_i^t)$.
    (b) For each i, set $h_i^{t+1} = \hat{h}_i^{t+1} / \sum_{j=1}^d \hat{h}_j^{t+1}$.
(3) If $\hat{y}^t < y^t - \epsilon$ (Mistake—prediction too small)
    (a) For each i, set $\hat{h}_i^{t+1} = h_i^t \cdot \exp(\eta(t) x_i^t)$.
    (b) For each i, set $h_i^{t+1} = \hat{h}_i^{t+1} / \sum_{j=1}^d \hat{h}_j^{t+1}$.
(4) Else, continue. (No mistake.)

Fig. 5. Mistake bounded algorithm for learning linear functions.

Theorem B.3. MWMBound *is a mistake bounded learning algorithm for the set of all unit* ℓ_1 *bounded linear functions* L_1 *with mistake bound*

$$m(\epsilon, L_1) \leq \frac{4 \log d}{\epsilon^2}.$$

Proof. The algorithm is simply an invocation of the multiplicative weights update rule with an update only on those days in which the algorithm makes an ϵ-mistake. It invokes the multiplicative weights update rule with the following loss functions: when the predicted value $\hat{y}^t$ is too large, it uses the loss vector $L^t = x^t$, and when the predicted value $\hat{y}^t$ is too small, it uses the loss vector $L^t = -x^t$. The analysis is based on the fact that the multiplicative weights update rule guarantees the following for any point y^* in the simplex [Arora et al. 2012]:

$$\sum_{t=1}^{T} \langle h^t, L^t \rangle - \sum_{t=1}^{T} \langle y^*, L^t \rangle \leq 2\sqrt{\log(d) T}.$$

Combining the summations on the LHS, we can rewrite this as

$$\sum_{t=1}^{T} (\langle y^t, L^t \rangle - \langle y^*, L^t \rangle) \leq 2\sqrt{\log(d) T},$$

We now note two things. First, because the labeled examples are consistent with L_1, there exists some y^* in the simplex such that for all t, $y^t = \langle y^*, x^t \rangle$. Second, note that we have chosen our loss functions so that the LHS $> \epsilon T$. Thus, we have $\epsilon T \leq 2\sqrt{\log(d) T}$, and solving, we must have

$$T \leq 4 \log(d)/\epsilon^2. \quad \square$$

We note that it is easy to extend this algorithm to learning linear functions with ℓ_1 norm $B > 1$. If initially our vector y^* has ℓ_1 norm B, then we can scale down all examples to have ℓ_1 norm 1 and learn $\hat{y}^* = 1/B y^*$ so that we are in the unit vector case. In this case, if we want error ϵ with respect to the true vector y^*, we must require error ϵ/B for the scaled-down version $\hat{y}^*$. Thus, we obtain the following corollary.

COROLLARY B.4. MWMBOUND *is a mistake bounded learning algorithm for the set* L_B *of all linear functions with* ℓ_1 *norm bounded by* B *with mistake bound*

$$m(\epsilon, L_B) \leq \frac{4B^2 \log d}{\epsilon^2}.$$

REFERENCES

S. Arora, E. Hazan, and S. Kale. 2012. The multiplicative weights update method: A meta-algorithm and applications. *Theory of Computing* 8, 121–164.

Y. Babichenko. 2013. Query complexity of approximate Nash equilibria. arXiv:1306.6686.

Y. Babichenko and S. Barman. 2013. Query complexity of correlated equilibrium. arXiv:1306.2437.

Y. Babichenko, S. Barman, and R. Peretz. 2013. Small-support approximate correlated equilibria. arXiv:1308.6025.

Y. Babichenko, S. Barman, and R. Peretz. 2014. Simple approximate equilibria in large games. In *Proceedings of the 15th ACM Conference on Economics and Computation (EC'14)*. 753–770.

A. Blum, J. Jackson, T. Sandholm, and M. Zinkevich. 2004. Preference elicitation and query learning. *Journal of Machine Learning Research* 5, 649–667.

A. Blum and Y. Mansour. 2007. Learning, regret minimization, and equilibria. In *Algorithmic Game Theory*, N. Nisan, T. Roughgarden, E. Tardos, and Vijay V. Vazirani (Eds.). Cambridge University Press, New York, NY, 79–102.

X. Chen, X. Deng, and S.-H. Teng. 2009. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM* 56, 3, 14:1–14:57.

W. Conen and T. Sandholm. 2001. Preference elicitation in combinatorial auctions. In *Proceedings of the 3rd ACM Conference on Electronic Commerce (EC'01)*. 256–259.

V. Conitzer. 2009. Eliciting single-peaked preferences using comparison queries. *Journal of Artificial Intelligence Research* 35, 161–191.

C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. 2009. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing* 39, 1, 195–259.

J. Fearnley, M. Gairing, P. W. Goldberg, and R. Savani. 2013. Learning equilibria of games via payoff queries. In *Proceedings of the 14th ACM Conference on Electronic Commerce (EC'13)*. 397–414.

J. Fearnley, P. W. Goldberg, R. Savani, and T. B. Sørensen. 2012. Approximate well-supported Nash equilibria below two-thirds. In *Algorithmic Game Theory*. Lecture Notes in Computer Science, Vol. 7615. Springer, 108–119.

J. Fearnley and R. Savani. 2014. Finding approximate Nash equilibria of bimatrix games via payoff queries. In *Proceedings of the 15th ACM Conference on Economics and Computation (EC'14)*. 657–674.

Y. Freund and R. Schapire. 1999. Adaptive game playing using multiplicative weights. *Games and Economic Behavior* 29, 79–103.

P. W. Goldberg and A. Pastink. 2014. On the communication complexity of approximate Nash equilibria. *Games and Economic Behavior* 85, 19–31.

P. W. Goldberg and S. Turchetta. 2014. Query complexity of approximate equilibria in anonymous games. arXiv:1412.6455.

A. Gupta, A. Roth, and J. Ullman. 2012. Iterative constructions and private data release. In *Proceedings of the 9th International Conference on Theory of Cryptography (TCC'12)*. 339–356.

M. Hardt and G. Rothblum. 2010. A multiplicative weights mechanism for privacy-preserving data analysis. In *Proceedings of the 2010 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*. 61–70.

S. Hart and Y. Mansour. 2010. How long to equilibrium? The communication complexity of uncoupled equilibrium procedures. *Games and Economic Behavior* 69, 1, 107–126.

S. Hart and A. Mas-Colell. 2000. A simple adaptive procedure leading to correlated equilibrium. *Econometrica* 68, 5, 1127–1150.

S. Hart and N. Nisan. 2013. The query complexity of correlated equilibria. arXiv:1305.4874.

M. Kearns, M. M. Pai, A. Roth, and J. Ullman. 2013. Mechanism design in large games: Incentives and privacy. arXiv:1207.4084.

S. C. Kontogiannis and P. G. Spirakis. 2010. Well supported approximate equilibria in bimatrix games. *Algorithmica* 57, 4, 653–667.

C. H. Papadimitriou and T. Roughgarden. 2008. Computing correlated equilibria in multi-player games. *Journal of the ACM* 55, 3, Article No. 14.

A. Roth and T. Roughgarden. 2010. Interactive privacy via the median mechanism. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC'10)*. ACM, New York, NY, 765–774.

H. Tsaknakis and P. G. Spirakis. 2008. An optimization approach for approximate Nash equilibria. *Internet Mathematics* 5, 4, 365–382.

Received January 2015; revised July 2015; accepted November 2015