Electronic Colloquium on Computational Complexity, Report No. 136 (2013)

Bounds for the Query Complexity of Approximate Equilibria

Paul W. Goldberg
Department of Computer Science,
University of Oxford
email: Paul.Goldberg@cs.ox.ac.uk

Aaron Roth
Department of Computer and Information Science,
University of Pennsylvania
email: aaroth@cis.upenn.edu

September 27, 2013

Abstract

We analyze the number of payoff queries needed to compute approximate correlated equilibria. For multi-player, 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 which separates 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 *verifying* an approximate well-supported Nash equilibrium to the problem of computing 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.

1 Preliminaries

This paper 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 G is known to belong to a given 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 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 below.

We consider queries that consist of pure-strategy profiles, and in which the answer to any query is the payoffs that all the players receive from that profile. In this paper we mostly focus on n-player binary-action games, which have 2^n pure profiles. In the context of approximate equilibria, we use ϵ

ISSN 1433-8092

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 whose query complexity is at most polynomial in n, which of course means that only a very small fraction of a game's profiles may be queried. The solution concepts we consider are ϵ-approximate correlated equilibrium (ϵ-CE), and ϵ-approximate well-supported correlated equilibrium (ϵ-WSCE), defined below.

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 paper we assume all players have the same number m of pure strategies, i.e. $|A^i| = m$ for all i. $u^i : A \longrightarrow \mathbb{R}$ 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, where binary-choice means $m = 2$.

1.1 Alternative definitions of approximate correlated equilibrium

We review the notions of exact and approximate correlated equilibrium, 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

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

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 above definition is based on swap regret, from [6], although other definitions (e.g. based on internal regret) are possible. An alternative definition from [19] 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, while a correlated ϵ-equilibrium need only be a $m\epsilon$-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 we mainly consider here) an ϵ-CE is an ϵ-CCE, while an ϵ-CCE is a 2ϵ-CE, hence the two notions are basically the same from the perspective of our interest in asympototic 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 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 $p = \Pr_{a \sim \psi}[a^i = j]$,

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

[1]For example, in the game of rock-paper-scissors, the uniform distribution over the 3 strategy profiles in which both players play the same strategy is a CCE but not a CE.

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

Thus, an ϵ-WSCE is a correlated $p\epsilon$-equilibrium. One way to understand this new definition is to 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) constitutes 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.

ϵ-WSCE is a refinement of approximate CE that is analogous to well-supported approximate Nash equilibrium, studied in [17, 10, 14, 3]. In an ϵ-Nash equilibrium (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 Nash equilibrium, 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 $\frac{1}{3}$ [26] while for ϵ-WSNE, the best value known is slightly less than $\frac{2}{3}$ [10]. (On the other hand, it is also known from [7] that a PTAS for ϵ-NE would imply a PTAS for ϵ-WSNE.) For the games studied in this paper, we will see that the payoff query complexity of ϵ-WSCE is strictly higher than than the query complexity of ϵ-CE.

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 a ϵ-WSCE (for $\lambda \in (0, 1)$). Suppose strategy profile s is sampled from ψ'' and some player i observes his action a, i.e. his marginal observation of s on his own behavior. If there was some action a' that would pay i more then ϵ more (in expectation) then one (or both) of ψ or ψ' would have to have that property. $\square$

Example 1 *In the* directed path graphical game *G_n, each player $i \in \{1, 2, \ldots, n\}$ has 2 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: In an exact Nash equilibrium of G_n, all players play 1 with probability 1. Furthermore it is not hard to see that for $\epsilon < 1$, the only ϵ-WSCE requires all players to play 1 with probability 1. In an ϵ-Nash equilibrium 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 towards 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 in an ϵ-approximate CE.

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

1.2 Related work

The papers of Fearnley et al. [9], Hart and Nisan [20], Barman and Babichenko [4], and Babichenko [3] also study query complexity of equilibria of multiplayer games. The main focus of [20, 4, 3] is on exponential (in n) lower bounds for solutions of n-player games having a constant number m of pure strategies per player. [20, 4] note that with randomization, one can obtain query-efficient algorithms for approximate correlated equilibrium by simulating regret-based algorithms — thus showing query complexity serves as a criterion to distinguish the difficulty of Nash equilibrium from Correlated equilibrium. In this paper we study in more detail the query complexity of approximate correlated equilibria as a function of n and ϵ, and we find that it serves as a criterion to separate the difficulty of approximate CE from well-supported approximate CE. [20] show that deterministic algorithms for ϵ-approximate correlated equilibria require exponentially-many payoff queries, hence the same holds for ϵ-NE. [20] also shows that exact CE needs exponentially-many queries. [3] shows that ϵ-well-supported approximate NE needs exponentially-many queries, for $m = 10^4$ and $\epsilon = 10^{-8}$; it is currently an open question whether these constants can be improved.

Fearnley et al. [9] study of payoff query complexity of Nash and approximate Nash equilibrium of classes of games where more structure is known about the payoffs; the main focus is on congestion games on networks. For the classes of games studied there, the properties of the payoff function can be exploited to obtain algorithms whose query complexity is polynomial. For the classes of games studied in this paper (n players, binary actions, no restriction on the structure of the payoff function), it has already been observed [4, 20] that Nash equilibrium requires exponentially-many queries. Indeed, as noted in [20], even approximate Nash equilibrium requires exponentially-many queries for deterministic algorithms (for randomized algorithms, the query complexity is noted as an open problem). Amin et al. [1] can be regarded as a study of the one-player version of the problem. A "player" has exponentially-many pure strategies, and wants to choose the one that gives the highest payoff. The payoff function is assumed to come from a known class of functions, and the *haystack dimension* is shown to characterize the number of queries needed. (This issue relates slightly to uncoupledness, in which players know their own utility functions but not other players'. Here players are in a sense are ignorant of their own payoff function, not just their opponents'.)

The *communication complexity* model was introduced to the game-theory context by Hart and Mansour [18] (n-player binary choice games), and studied in [14] for bimatrix games. [20] notes that payoff-query bounded algorithms can be efficiently converted into communication-bounded algorithms. Lower bounds seem to be easier to obtain in the payoff-query setting. In the communication-bounded setting, [18] show efficient upper bounds for ϵ-CE, and exponential lower bounds just for exact (pure or mixed) Nash equilibria. The communication complexity of finding an exact correlated equilibrium is polynomial in the number of players, in contrast with the exponential requirement of NE. It uses Papadimitriou's approach [23] (using 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; this suggests that pure payoff queries can be used by a randomized algorithm to find approximate correlated equilibria.

The connection between no-regret algorithms and equilibrium notions, including correlated equilibria, were first noted by [13, 19]. For an excellent survey of this connection, see Blum and Mansour [6]. Appendix B.3 of [18] 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 correlated equilibrium (but not well-supported correlated equilibrium). 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 correlated equilibria, 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, while 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 $\epsilon \geq \frac{1}{2}$, the computation of an ϵ-approximate CE for binary-choice games is trivial, since the uniform distribution is a $\frac{1}{2}$-approximate Nash equilibrium.*

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 non-trivial.

For (non-well-supported) correlated equilibria, 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: we include the standard proof in the appendix for completeness. We note that this also yields an $O(m \log m)$ upper bound for the problem of computing an approximate Nash equilibrium in an m-action two player zero sum game, which is substantially smaller than the representation size of the game matrix.

The case of many players having just 2 strategies (Section 2.2) is more subtle: a straightforward application of multiplicative weights would again yield a polynomial (in the number of players) upper bound on value queries, but we are instead able to achieve a logarithmic upper bound. As before, we use the Multiplicative Weights algorithm, but this time, at each iteration we draw a sample S of pure-strategy profiles from the current mixed-strategy profile, and we estimate the costs of individual pure-strategies of a player by restricting S to each individual strategy, and using that as an estimate of the "true" cost of using that strategy in response to S. We use a "noise-tolerance" result of Kearns et al. [22] that if reported losses are within b of true losses, this adds at most b to the approximation error of the resulting CE. We obtain quite an efficient upper bound on the query complexity of computing an ϵ-CE, using $O(\log n/\epsilon^2)$ 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 contrasts with the exponential lower bound for deterministic algorithms [4, 20].

The lower bounds of Theorems 3 and 4 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 4 to show that small-support approximate CE are harder (in the query complexity sense) to find than are larger-support approximate CE.

Finally, we give query efficient reductions from the problem of verifying an ϵ-approximate well supported Nash equilibrium to the problem of computing an ϵ-approximate well supported Nash equilibrium (and hence correlated equilibrium), 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 [25, 16, 15]. We use the algorithms to learn a hypothesis game representation: at each stage, we compute an equilibrium of the hypothesis game,

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

    1. For each player $i$, let $\mathbf{p}^i(t) = (p^i_1(t), \dots, p^i_m(t))$ be the probability
       distribution over $i$'s strategies corresponding to the weights;
       $p^i_j(t) = w^i_j(t) / \sum_j w^i_j(t)$.

    2. For each player $i$, sample a pure strategy $s^i(t)$ from $\mathbf{p}^i(t)$. Let
       $\mathsf{s}(t) = \times_i s^i(t)$ be the resulting strategy profile.

    3. $\forall i$, query the payoffs of all responses to $\mathsf{s}(t)$. Let $m^i_j(t)$ be the
       cost (negated payoff) of strategy $j$.

    4. update weights: for each $i$, $j$, set $w^i_j(t+1) = w^i_j(t)(1 - \eta m^i_j(t))$.

Output: uniform distribution over $\{\mathsf{s}(t) : 1 \leq t \leq T\}$.
```

Figure 1: Using Multiplicative Weights to compute an approximate coarse CE

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 correlated equilibrium. The first one is applied to zero-sum bimatrix games, and the second one is applied to n-player binary-action games.

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-correlated equilibria that follows from using no-regret algorithms. In the special case of two player zero-sum games, Freund and Schapire [13] showed that this algorithm converges to an approximate Nash equilibrium. 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 it in Appendix A for completeness.

Theorem 1 identifies useful bounds in the case of $m \gg n$; in particular it has an interesting application to the case of 2-player zero-sum games in Corollary 1.

Theorem 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 $1 - nm^{-\frac{1}{8}}$, 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.*

Corollary 1 *Let G be a $m \times m$ zero-sum bimatrix game. It is possible to efficiently compute (using a randomized algorithm having small failure probability) an ϵ-Nash equilibrium 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 correlated equilibrium ψ can be converted to a 2ϵ-NE by taking the product of the marginal distributions of ψ for each player.

2.2 Upper bound for n players, binary actions

In binary action games, coarse correlated equilibria coincide with correlated equilibria, and so the result in the previous section immediately yields a polynomial upper bound (in terms of the number of players n) on the query complexity of approximate correlated equilibria. 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.

We use the following 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:

$$\Pr[X \leq (1-\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2}{2}\mu\right) \tag{3}$$

putting $\epsilon = \frac{1}{2}$ we get

$$\Pr[X \leq \mu/2] \leq \exp\left(-\frac{1}{8}\mu\right) \tag{4}$$

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.

$$\Pr[|\mu - \hat{\mu}| \geq \beta] \leq 2\exp(-2\beta^2 N) \tag{5}$$

Proposition 1 *Let s be a mixed-strategy profile of an n-player 2-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 the $\mathsf{s}_i(a)$ 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 (5), it is sufficient for N' to satisfy $2\exp(-2\beta^2 N') \leq \delta/4n$, equivalently:

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

Let $X = \sum_{j=1}^{N} X_j$ where $X_i = 1$ if player i plays s_i and 0 if i plays $1 - \mathsf{s}_i$. 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 (4) 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 (4) 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 we obtained for N', we get the expression for N in the statement. □

Proposition 2 *Suppose each player of $G \in \mathcal{G}_n$ runs T iterations of Multiplicative Weights with learning rate parameter $\eta \leq \frac{1}{2}$. Assume that initially all weights are equal. At the end, all weights correspond to probabilities in the range $[\gamma, 1-\gamma]$, provided that $T \leq \log_{1-\eta}(2\gamma)$.*

Proof. The initial weights correspond to probabilities of $\frac{1}{2}$. Each iteration reduces a weight by a factor at most $1-\eta$. To reduce a weight to γ requires $\log_{1-\eta}(2\gamma)$ iterations, i.e. $T = \log_{1-\eta}(2\gamma)$. □

Theorem 2 *Let G be a game with n players, each with 2 actions; payoffs lie in $[0,1]$. With probability $1-\frac{2}{n}$, Algorithm* Approx Coarse CE (2) *(Figure 2) finds an ϵ-approximate coarse correlated equilibrium of G, using $\tilde{O}(\log n/\epsilon^5)$ payoff queries.*

Proof. At each iteration $t = 1, \ldots, T$, let $w_j^i(t)$ be the weight of player i's strategy 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 actions in which the probability of a strategy is proportional to its weight. At iteration t, each player i observes costs $\hat{m}_0^i(t)$ and $\hat{m}_1^i(t)$, the costs of actions 0 and 1 respectively. Weights are updated according to the rule $w_j^i(t+1) = w_j^i(t)(1-\eta\hat{m}_j^i(t))$, where $\eta \leq \frac{1}{2}$ is the learning rate parameter.

Now, these observed costs differ from the costs that derive from profiles sampled uniformly from $S(t)$ (the distribution used by Approx Coarse CE (2) at iteration t), but we can bound the difference and use Lemma 2 of [22], which says that when a regret-minimizing algorithm is run with losses altered by additive noise b, the regret with respect to the noise-free loss matrix is higher by at most $2b$ than the regret with respect to the noisy matrix. Proposition 1 gave conditions for the observed losses to be within β of true losses.

Applying Proposition 1 with $\gamma = \epsilon/2$, $\beta = \epsilon/4$, 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:

$$N \geq \max\left\{\frac{32}{\epsilon^3}\log\left(\frac{8nT}{\delta}\right), \frac{16}{\epsilon}\log\left(\frac{4nT}{\delta}\right)\right\}. \tag{6}$$

The following is a modified version of equation (10), taking into account the extra additive β due to the sampling error and putting $m = 2$. $\mathbf{m}^i(t)$ is costs w.r.t. $S(t)$; $\hat{m}_j^i(t)$ is the cost of strategy j w.r.t. elements of $S(t)$ where i plays j.

$$\frac{1}{T}\sum_{t=1}^{T}\mathbf{m}^i(t).\mathbf{p}^i(t) \leq \frac{1}{T}\sum_{t=1}^{T}\hat{m}_j^i(t) + \eta + \frac{\ln 2}{T\eta} + 2\beta. \tag{7}$$

The expected cost incurred by the player (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 (2)
Parameters: learning rate $\eta = \frac{\epsilon}{4}$; number of iterations $T = \frac{4\ln 2}{\epsilon\eta}$;
sample size $N$ (defined in Equation 6).
$w^i_j(t)$ is the weight of strategy $j \in \{0,1\}$ of player $i$ at iteration $t$;
initially $w^i_j(1) = 1$ for all $i$, $j$. Do the following for iterations $t =
1,\dots,T$:

  1. Let $\mathbf{p}^i(t) = (p^i_0(t), p^i_1(t))$ be the probability distribution over $i$'s
     strategies corresponding to the weights; $p^i_j(t) = w^i_j(t)/\sum_j w^i_j(t)$.
     Let $\mathbf{p}(t)$ be the corresponding mixed-strategy profile.
  2. Sample $N$ pure-strategy profiles $S(t) = \{\mathsf{s}_1(t),\dots,\mathsf{s}_N(t)\}$ from $\mathbf{p}(t)$.
  3. query the payoffs for all the profiles in $S(t)$. Let $\hat{m}^i_j(t)$ be
     the empirical cost of strategy $j$ when used by player $i$. (If $i$
     never plays $j$ in the profiles $S(t)$, the algorithm fails.)
  4. update weights: for each $i$, $j$, set $w^i_j(t+1) = w^i_j(t)(1 - \eta\hat{m}^i_j(t))$.

Output: uniform distribution over the multiset $\cup_t S(t)$.
```

Figure 2: Using Multiplicative Weights to compute an approximate coarse CE

Based on (7), a player's regret can be upper bounded by

$$\eta + \frac{\ln 2}{T\eta} + 2\beta. \tag{8}$$

So, to make this expression at most ϵ, put $\eta = \frac{\epsilon}{4}$ and $T = \frac{4\ln 2}{\epsilon\eta}$.
The total number of queries is $NT = \tilde{O}(\log n/\epsilon^5)$. □

When players have just 2 pure strategies, an ϵ-approximate coarse correlated equilibrium is a 2ϵ-approximate correlated equilibrium, so we have the following result:

Corollary 2 *For binary-choice games (where each player has 2 pure strategies), with probability $1 - \frac{1}{n}$, Algorithm* APPROX COARSE CE (2) *finds an ϵ-approximate correlated equilibrium, using $\tilde{O}(\log n/\epsilon^5)$ 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, 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 above, 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 3 *For $0 \leq j \leq Q$, 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, $\mathsf{p}^j = \times_i \mathsf{p}^j_i$, where p^j_i 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 a coin-flip which is

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

$\square$

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^i_t be the number of queries that indicate that player i is type t. It is not hard to check that $\Pr[\mathsf{t}_i = 1]/\Pr[\mathsf{t}_i = 0] = (k-1)^{(2Q^i_1 - Q)}$, where $\Pr[\mathsf{t}_i = t]$ is the probability that i has type t, conditioned on the data.*

Definition 1 *$\mathcal{A}$ outputs a distribution ψ over pure-strategy profiles. Let ψ^u be ψ restricted to the un-queried 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, i.e.*

$$\mathop{\mathbb{E}}_{x \sim \psi^u}[x_i] = b$$

where x_i is the action played by i in profile x.

Theorem 3 *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 above.

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 zero 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 zero on all queries.

Conditioned on the answers to all Q queries, we have (noting Observation 3) for any player i that $\Pr[\mathsf{t}_i = 1]/\Pr[\mathsf{t}_i = 0] \leq (k-1)^Q$ and similarly $\Pr[\mathsf{t}_i = 0]/\Pr[\mathsf{t}_i = 1] \leq (k-1)^Q$, where $\Pr[\mathsf{t}_i = t]$ is the conditional probability that t_i is equal to t. Hence, 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 zero on all Q queried profiles. Let b be the bias (Definition 1) 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 (resp. p_1) be the probability that $x \sim \psi$ is a queried profile where i plays 0 (resp. 1). Let p'_0 (resp. p'_1) be the probability that $x \sim \psi$ is an unqueried profile where i plays 0 (resp. 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 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 $\mathsf{t}_i = 1$ 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 $\mathsf{t}_i = 0$. 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 $\mathsf{t}_i = 1$. 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. So, there exists a value for t_i such that i's regret is at least
$$\max\Big\{p_1 + \frac{k-1}{k}(p'_0 + p'_1) - \Big(p'_0\frac{k-1}{k} + p'_1\frac{1}{k}\Big),\ p_0 + \frac{k-1}{k}(p'_0 + p'_1) - \Big(p'_1\frac{k-1}{k} + p'_0\frac{1}{k}\Big)\Big\}$$
which simplifies to
$$\max\Big\{p_1 + \frac{k-2}{k}p'_1,\ p_0 + \frac{k-2}{k}p'_0\Big\}.$$
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 $\frac{k-2}{2k}$, 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 2 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 $1-\mathsf{t}_i$. 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 $(\frac{1}{k-1})^Q$. So 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 $\frac{1}{2}$. $\square$

3 Bounds for the query complexity of ϵ-WSCE

The approach of [7] for converting a PTAS for ϵ-NE into a PTAS for ϵ-WSNE works for Nash equilibria but fails for correlated equilibria, and so we need new upper bound techniques. We first present a lower bound that separates the query complexity of ϵ-WSCE from ϵ-CE. We then present an upper bound in Section 3.2 that applies to the subclass of concisely-representable games. Finally, we show that this upper bound takes the form of a generic reduction between the problem of testing ϵ-WSCE to the problem of finding ϵ-WSCE, 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 3 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, that $\mathcal{A}$ is likely to fail to find a ϵ-WSNE.

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 4 *For $\epsilon < 1$, the payoff query complexity of computing ϵ-WSCE of $\mathcal{G}_n$ (i.e. n-player 2-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) Nash equilibrium x. x 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, x can be seen to be the unique ϵ-WSCE of G.

For $G \sim \mathcal{D}_n$, let $\mathsf{s}_1,\ldots,s_j$ 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 form. Let m be the length of the longest prefix of players that all get paid 1 in some query,

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

Then $\mathcal{D}'$ is uniform over all x in the support of $\mathcal{D}_n$ that agree with s_q on players $1\ldots,m$ and disagree with s_q on player $m+1$.

The claim can be proved by induction on j. Let $\mathcal{D}^j$ be the conditional distribution that results from the first j queries, and consider query s_{j+1}. Define m as above. $\mathcal{D}^j$ is uniform over elements of $\mathcal{G}_n$ in its support. $\mathcal{D}^{j+1}$ is obtained by taking the uniform distribution over any such game that is consistent with s_{j+1}.

Define the *progress* of query s_{j+1} to be the increase in the length of the prefix of players whose behavior is determined by elements of $\mathcal{D}^j$. Then the expected progress of each query is less than $1 + \sum_{r \geq 0} r/2^{r+1} = 2$. And, the total progress of all queries required to find an ϵ-WSCE is n. $\square$

Using the above result we can obtain a separation between the query complexity of computing small support ϵ-CE and arbitrary support ϵ-CE for any $\epsilon \geq (\log(n)/n)^{1/3}$. Without loss of generality, we can view approximate correlated equilibria as uniform distributions over (multi)sets of action profiles S: note that multiplicative weights explicitly finds correlated equilibria of this form, and any correlated equilibrium can be put in this form with arbitrarily small loss in the approximation factor by sampling. We note that multiplicative weights finds a correlated equilibrium 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 correlated equilibrium with support size $O(1/\epsilon)$ (independent of n). Of course such a separation is vacuous in games in which there are no ϵ-CE with support $O(1/\epsilon)$, but in any game that has a pure strategy (ϵ)-Nash equilibrium, there is always a small support CE – in particular, one with support just 1. On this topic, Babichenko et al. [5] show that multiplayer games have approximate CEs with polylogarithmic support size, and approximate CCEs with logarithmic support size. They note as an open problem the existence of CEs whose support size depends only on ϵ and not n.

Corollary 3 *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.*

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 4 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 for computing CE with larger support (in particular, support $\log(n)/\epsilon^2$). $\square$

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 and actions in the game, 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. The query complexity of approximate well-supported Nash equilibrium (ϵ-WSNE) for unrestricted n-player games is exponential in n [3] for $\epsilon = 10^{-8}$ and $m = 10^4$ (where m is the number of pure strategies of each player). The class of games used by [3] are 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. An assumption that the unknown target game can be written down using $poly(n)$ bits appears to be benign in practice. Thus, any polynomial query algorithm for ϵ-WSCE for general games would have to (unlike our algorithm) not also compute ϵ-WSNE. Note that in contrast, 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 also yields a positive result for a special case of the question of query complexity of ϵ-NE using randomized algorithms [20].

```
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 $\mathcal{N}^i_j$ be a distribution over
     strategy profiles obtained by sampling from $\mathcal{N}$ and setting $i$'s
     strategy to $j$; let $S^i_j$ be a set of pure profiles sampled u.a.r.
     from $\mathcal{N}^i_j$, where $|S^i_j| = (\frac{4}{\epsilon})\log(2p(n))$; query all $x \in S^i_j$;

  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 $\mathcal{N}$.
```

Figure 3: Query-efficient algorithm for ϵ-WSNE

Algorithm APPROX WSNE of Figure 3 can be viewed as a query efficient reduction between the problem of testing an ϵ-CE to the problem of finding 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 5 *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 $\frac{1}{2}$, 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, $\Pr_{x \sim \mathcal{N}^i_j}[G^*(x) \neq G'(x)] < \frac{\epsilon}{4}$.

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 generalize 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. Consider how our algorithm 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 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 $\mathcal{N}$ which is a WSNE of G' but not of G, we find a profile x which 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 last 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 in G. Specifically, suppose we have a learning algorithm A such that for any sequence of payoff-query/answer pairs $(x_1, a_1), \ldots, (x_m, a_m)$ produces a hypothesis $f = A((x_1, a_1), \ldots, (x_m, a_m))$ which maps play profiles x to payoff values $f(x)$, one for each player. (that is, f represents a hypothesis game G'). Say that the algorithm A makes a mistake with respect to a game G if it errs on its prediction $f(x)$ of the payoff of some queried profile x 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 B mistakes. We then have an algorithm for computing ϵ-WSCE using only $B \cdot Q$ 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 1 below, we identify conditions under which Multiplicative Weights can be used to obtain new query bounds for certain classes of games. The approach is motivated by 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 $\mathcal{N}^t$.

We consider games where the payoff function can be expressed as a linear function of a limited number of attributes (or features) of strategy profiles. For example consider congestion games with

n players and k facilities. The cost incurred by a player is the sum of the costs of the facilities he uses, and those costs are determined by the number of users of each facility. Thus, given a strategy profile, we extract the following nk features: for each facility j and $i \in [n]$, an associated feature is set to 1 if i players use j, otherwise it is set to 0. A payoff query gives us an observation of the input/output behavior of this linear function, and we aim to learn the linear function. Notice that if we assumed that facility costs were linear in the number of their users, then only k features would be needed for the payoff function. With quadratic costs we could use $2k$ features (the coefficents of the loads, and the squares of the loads).

The problem of finding an ϵ-NE of G^* is reducible to learning the coefficients of payoff function f with accuracy ϵ. (A solution to a game whose payoff function approximates target game G^* will be an approximate solution to G^*.) As described in the Appendix, Multiplicative Weights can be used to learn approximations of linear functions via queries, with a mistake bound proportional to the L_1 norm of the target function and logarithmic in its dimensionality. If the target function has a low L_1 norm this leads to a good query bound on the equilibrium learning problem.

Let f^* be the linear function associated with the target game G^* and f corresponds to the game G being tested. Let $\mathcal{N}$ be a pure Nash equilibrium of G. Check whether $\mathcal{N}$ is an ϵ-NE of G^* by querying all pure-strategy deviations of every player. If $\mathcal{N}$ is not an ϵ-NE of G^*, we should be able to find an alternative pure strategy profile $\mathcal{N}'$ for which in G^* some player i's payoff is $> \epsilon$ higher in $\mathcal{N}'$ than in $\mathcal{N}$, but in G i's payoff in $\mathcal{N}'$ is at most what it is in $\mathcal{N}$. This means that either $\mathcal{N}$ or $\mathcal{N}'$ corresponds to a point x in feature space where $|f^t(x) - f(x)| > \epsilon$.

Lemma 1 *Suppose a target game G^* belongs to a class $\mathcal{G}$ of potential games such that*

1. *members of $\mathcal{G}$ has a payoff function that can be expressed as a linear function $f(x) = \langle x, y\rangle$, where $x \in [0,1]^d$ is a d-dimensional attribute vector of strategy profiles,*
2. *given any strategy profile, Q queries are sufficient to search for an ϵ-better response.*

Then for target game G^* with payoff function $f^*(x) = \langle x, y^*\rangle$, letting $\|y^*\|$ denote the L_1 norm of y^*, an ϵ-NE of G^* can be found using $O(Q\|y^*\|^2 \log(d)/\epsilon^2)$ queries. Furthermore, if pure ϵ-NE can be computed efficiently we also have that the search is efficient.

Proof. Consider Algorithm Approx NE of potential games (Figure 4). Let f^* denote the linear function corresponding to 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 $|f'(x) - f^*(x)| \leq \epsilon$ for all $x \in [0,1]^d$. This reduces the problem to approximately learning a linear function.

We learn an approximation to y^* as follows. Let y^1 be the d-dimensional vector $(1/d, \dots, 1/d)$. Consider the game G^1 whose payoff function is given by $f^1(x) = \langle x, y^1\rangle$. Compute an ϵ-NE $\mathcal{N}^1$ of G^1 and check (using Q queries) whether N^1 is an ϵ-NE of G^*. If it is, we are done, if not we find a value of x such that $|f^1(x) - f^*(x)| > \epsilon$.

Each weights-update operation (incrementing t) is based on having found a point x^t in the domain where $|f^*(x^t) - f^t(x^t)| > \frac{\epsilon}{4}$. (This uses $f^t(\mathcal{N}^t) \geq f^t(x) - \frac{\epsilon}{2}$ and $f^*(x) \leq f^*(x) - \epsilon$, from which it follows that a suitable x^t is found at Step 4.

Plugging in $\frac{\epsilon}{4}$ as the discrepancy between the value of f^* and the current f^t, into the mistake bound in the appendix, we get a mistake bound of $64\log(d)\|y^*\|^2/\epsilon^2$, which upper bounds the number of iterations of Algorithm Approx NE of potential games . □

We give an example of a class of games to which the above lemma can be usefully applied. These are network congestion games of the kind studied in Fearnley et al. [9], where the costs of edges are unknown increasing functions of the number of players using them, so that these costs

```
APPROX NE OF POTENTIAL GAMES
Let game $G^1$ have payoff function $f^1(x) = \langle x, y^1 \rangle$ where $y^1 \in \mathbb{R}_{\geq}0^d$ be the
initial hypothesis of the multiplicative weights learning algorithm
for linear functions.  (See Appendix B)
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. Use $Q$ queries to $G^*$ to check whether a player can improve his
     payoff by $> \epsilon$ (in $G^*$);
  3. If not, halt and output $\mathcal{N}^t$;
  4. If yes, let $x$ be the strategy profile resulting from the
     deviation; if $|f^*(x) - f^t(x)| > \frac{\epsilon}{4}$ let $x^t = x$, else if
     $|f^*(\mathcal{N}^t) - f^t(\mathcal{N}^t)| > \frac{\epsilon}{4}$ let $x^t = -x$;
  5. Feed $x^t$ as the loss vector to the multiplicative weights
     algorithm for learning linear functions, and receive the new
     hypothesis vector $y^{t+1}$.  Let game $G^{t+1}$ have payoff function
     $f^{t+1}(x) = \langle x, y^{t+1} \rangle$.
```

Figure 4: Query-efficient algorithm for ϵ-NE

need to be learned in order to compute an equilibrium. In the case of n players sharing a directed acyclic graph with m edges, [9] shows how an exact equilibrium can be found using mn queries. In the case of parallel-link networks, they obtain an algorithm using $O(\log(n).\frac{\log^2(m)}{\log\log(n)})$ queries. In the parallel-links case, the logarithmic dependence on n shows that the cost functions do not have to be completely learned in order to find an equilibrium, and raises the question of whether for the more general networks, one should be able to select queries so that only a relatively small number of observations of each cost function is sufficient. We make progress on this question in the special case of networks having a small number of edges, and for approximate (not exact) equilibria.

Theorem 6 *Consider n-player congestion games over d facilities, where we think of d as being a constant. Assume that each faciity j has an increasing cost function c_j that takes values in $[0, 1]$. The number of queries needed to find ϵ-NE is at most $4.2^{2d}.d^2/\epsilon^2$.*

Proof. Let $\alpha_{i,j} = c_j(i) - c_j(i-1)$ be the extra cost incurred by raising the load on j from $i-1$ players to i players.

Notice that the cost function is linear in the following attributes x_{ij} of a strategy profile: $x_{ij} = 1$ if at least i players use j, otherwise $x_{ij} = 0$. Any observed cost is of the form $\sum_{j \in S} \sum_{i=1}^{N_j} \alpha_{ij} x_{ij}$, where S is the set of facilities used by a player, and N_j is the number of players using facility j. Also, the L_1 norm of any target game cannot exceed d since for any facility j we have $\sum_i \alpha_{ij} \leq 1$.

With regard to the value of Q, given a specific pure profile x (for which we seek a ϵ-better response), there are (at most) 2^{2d} queries that need to be made, since the cost of an alternative strategy for a player will depend on which of the 2^d subsets of facilities he is using in x, and which subset of the d facilities he may move to.

Finally, efficient computation of ϵ-NE of these games can be done using ϵ-best response dynamics. □

The above analysis has used a crude upper bound of Q that is exponential in the number of facilities. In the case of network congestion games it may well be possible to show that ϵ-better responses can be searched for using a number of queries that is polynomial in the number of edges.

4 Further work

The bound of Theorem 4 could plausibly be improved, and regarding the query complexity of ϵ-WSCE, our result leaves open the question of question of efficient computability. It is also limited to concisely-represented games, but arguably that is a benign restriction.

Hart and Nisan [20] note the open problem of query complexity of approximate NE for randomized algorithms; the exponential LB is for deterministic ones. Algorithm Approx WSNE makes progress on this question. An extension of that question is to ask whether ϵ-WSCE are easier to learn than ϵ-NE. Similarly, Section 8.3 of [18] asks whether ϵ-NE requires exponentially-many (as a function of the number of players) rounds in the communication complexity setting.

In the context of centralized computation of solutions, ϵ-well-supported CE may be easier to *compute* than ϵ-NE. For example Papadimitriou and Roughgarden and Jiang and Leyton-Brown [24, 21] show that exact (and hence, well-supported approximate) CE can be computed for concisely represented multiplayer games, including graphical games, where it is PPAD-complete to compute ϵ-NE even for constant ϵ. Indeed, the algorithm of [21] interacts with a game via mixed-strategy payoff queries in which exact answers are returned.

References

[1] K. Amin, M. Kearns and U. Syed. Bandits, Query Learning, and the Haystack Dimension. *Journal of Machine Learning Research - Proceedings Track 19* pp. 87–106 (2011).

[2] S. Arora, E. Hazan, and S. Kale. The Multiplicative Weights Update Method: a Meta-Algorithm and Applications *Theory of Computing* Volume 8 Article 6 pp. 121-164 (2012).

[3] Y. Babichenko. Query Complexity of Approximate Nash Equilibrium. *ArXiv tech rept.* 1306.6686 (2013).

[4] Y. Babichenko and S. Barman. Query complexity of correlated equilibrium. *ArXiv tech rept.* 1306.2437 (2013).

[5] Y. Babichenko, S. Barman and R. Peretz. Small-Support Approximate Correlated Equilibria. *ArXiv tech rept.* 1308.6025 (2013).

[6] A. Blum and Y. Mansour. Learning, Regret Minimization and Equilibria. In *Algorithmic Game Theory*, Cambridge University Press (2007).

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

[8] C. Daskalakis, A. Mehta and C.H. Papadimitriou. A Note on Approximate Nash Equilibria. *Theoretical Computer Science* 410(17), pp. 1581–1588 (2009).

[9] J. Fearnley, M. Gairing, P.W. Goldberg and R. Savani. Learning Equilibria of Games via Payoff Queries. *Arxiv rept.* `http://arxiv.org/abs/1302.3116` (to appear in ACM-EC) (2013).

[10] J. Fearnley, P.W Goldberg, R. Savani and T.B. Sørensen. Approximate Well-supported Nash Equilibria Below Two-thirds. *Procs of the 5th International Symposium on Algorithmic Game Theory* LNCS 7615 pp.108-119 (Oct 2012).

[11] T. Feder, H. Nazerzadeh and A. Saberi. Approximating Nash equilibria using small-support strategies. In *Proc. of 8th ACM EC.* 352–354 (2007).

[12] Dean P. Foster and H. Peyton Young. Regret testing: learning to play Nash equilibrium without knowing you have an opponent. *Theoretical Economics* 1, pp. 341–367 (2006).

[13] Y. Freund and R. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79-103, (1999).

[14] P.W. Goldberg and A. Pastink. On the Communication Complexity of Approximate Nash Equilibria. *Procs of the 5th SAGT*, LNCS 7615, pp. 192–203 (Oct 2012). *Arxiv rept.* `http://arxiv.org/abs/1302.3793` (1012).

[15] A. Gupta, A. Roth, and J. Ullman. Iterative Constructions and Private Data Release. *Procs of the 9th TCC* pp. 339–356 (2012).

[16] M. Hardt and G. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. *Procs of the 51st FOCS* pp. 61–70 (2010).

[17] S.C. Kontogiannis and P.G. Spirakis. Well Supported Approximate Equilibria in Bimatrix Games. *Algorithmica* 57(4), pp. 653–667 (2010).

[18] S. Hart and Y. Mansour. How long to equilibrium? The communication complexity of uncoupled equilibrium procedures. *Games and Economic Behavior* 69(1) pp. 107–126 (2010).

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

[20] S. Hart and N. Nisan. The Query Complexity of Correlated Equilibria. *ArXiv tech rept.* 1305.4874 (2013).

[21] A.X. Jiang and K. Leyton-Brown. Polynomial-time Computation of Exact Correlated Equilibrium in Compact Games. *Procs. of ACM-EC*, 119–126 (2011).

[22] M. Kearns, M.M. Pai, A. Roth and J. Ullman. Mechanism Design in Large Games: Incentives and Privacy. *Arxiv rept.* `http://arxiv.org/abs/1207.4084` (2013).

[23] C.H. Papadimitriou. Computing correlated equilibria in multi-player games. *Procs. of the 37th STOC.* ACM, pp. 4956 (2005).

[24] C.H. Papadimitriou and T. Roughgarden. Computing Correlated Equilibria in Multi-Player Games. *Journal of the ACM* 55(3), article 14 (2009).

[25] A. Roth and T. Roughgarden Interactive Privacy via the Median Mechanism. *Procs. of the 42nd STOC.* ACM, pp. 765-774 (2010).

[26] H. Tsaknakis and P.G. Spirakis. An Optimization Approach for Approximate Nash Equilibria. *Internet Mathematics* 5(4), pp. 365–382 (2008).

A Upper Bounds on the Query Complexity of CCE Using MW

Theorem 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 $1 - nm^{-\frac{1}{8}}$, 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 [2]. For iteration $t = 1, \ldots, T$, $w^i_j(t)$ denotes the weight of player i's action j at step t; initially $w^i_j(1) = 1$ 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) $m^i_j(t) \in [-1,1]$) and weights are updated according to the rule $w^i_j(t+1) = w^i_j(t)(1 - \eta m^i_j(t))$, where $\eta \leq \frac{1}{2}$ is the learning rate parameter.

The performance guarantee of Multiplicative Weights, as given in Theorem 2.1 of [2], is that after T rounds, for any strategy j of player i we have

$$\sum_{t=1}^{T} \mathbf{m}^i(t).\mathbf{p}^i(t) \leq \sum_{t=1}^{T} m^i_j(t) + \eta \sum_{t=1}^{T} |m^i_j(t)| + \frac{\ln m}{\eta}. \tag{9}$$

Dividing by T and noting that $|m^i_j(t)| \leq 1$ we have

$$\frac{1}{T}\sum_{t=1}^{T} \mathbf{m}^i(t).\mathbf{p}^i(t) \leq \frac{1}{T}\sum_{t=1}^{T} m^i_j(t) + \eta + \frac{\ln m}{T\eta}. \tag{10}$$

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 the 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 , 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

$$\mathbb{E}[c^i] = \frac{1}{T}\sum_{t=1}^{T} \mathbf{m}^i(t).\mathbf{p}^i(t) \tag{11}$$

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,

$$\Pr[X_N - X_0 \geq d] \leq \exp\left(\frac{-d^2}{2\sum_{t=1}^{N} c_t^2}\right). \tag{12}$$

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 (12) to show that c^i is (usually) not too much higher than its expected value given in (11), which itself has the upper bound of (10). We can write

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

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

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

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

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

The right-hand side 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

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

So 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). □

B Using MW 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. The analysis follows from the standard regret bound of Multiplicative weights (see, e.g. [2]). For now, let us assume that the coefficients of f are non-negative and have L_1 norm 1. (Consequently they form a probability distribution over the feature space, which is a convenient form for multiplicative weights. 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, and then scaling back afterwards.) In other words, for every $x \in [0,1]^d$, $f(x) = \langle y^*, x\rangle$ for some target y^* such that $\|y^*\|_1 = 1$ and $y^* \geq 0$.

We will run Multiplicative weights to try to learn a vector y_t and make predictions using the function $g_t(x) = \langle y_t, x\rangle$. Multiplicative weights naturally maintains a distribution y^t over d feature vectors, which is what we will use as our hypothesis at step t. All that remains is to specify what loss functions we feed to multiplicative weights to let it perform updates.

We initialize multiplicative weights such that y_1 is the uniform distribution: $y_1 = (1/d, \ldots, 1/d)$. Let $t = 1$. For each example x, predict using $g(x) = \langle y_t, x\rangle$. We say that multiplicative weights *makes a mistake* at time t if on example x, $|f(x) - g(x)| \geq \epsilon$. There are two cases:

1. $|\langle y_t, x\rangle - \langle y^*, x\rangle| \leq \epsilon$ – i.e. we did not make a mistake. In this case we do nothing.

2. $|\langle y_t, x\rangle - \langle y^*, x\rangle| > \epsilon$, in which case we need to update multiplicative weights.

    - If the prediction was too big, i.e. $\langle y_t, x\rangle - \langle y^*, x\rangle > \epsilon$, then feed $L_t = x$ to multiplicative weights as a loss vector.

- If the prediction was too small, i.e. $\langle y^*, x\rangle - \langle y_t, x\rangle > \epsilon$, then feed $L_t = -x$ to multiplicative weights as a loss vector.

In either case, generate the updated multiplicative weights hypothesis y_{t+1} and increment t.

This algorithm can make at most $T \leq 4\log(d)/\epsilon^2$ mistakes. We will use the fact that for any point y^* in the simplex, multiplicative weights guarantees regret:

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

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

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

But note that we have chosen our loss functions so that the LHS $> \epsilon T$. So we have $\epsilon T \leq 2\sqrt{\log(d)T}$ and solving, we must have:

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

We note that if initially our vector y^* had L_1 norm N, then we can scale it down to have L_1 norm 1 and learn $\hat{y}^* = 1/Ny^*$, 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/N for the scaled down version $\hat{y}^*$. Thus, the number of mistakes is now $4N^2\log(d)/\epsilon^2$ – i.e. we pay quadratically for the L_1 norm of the target.

The same reduction works for any no regret algorithm. It is possible to get a slightly different bound with gradient descent, that pays for the L_2 norm of the target instead of the L_1 norm.