Distributed Methods for Computing Approximate Equilibria.

Artur Czumaj · Argyrios Deligkas · Michail Fasoulakis · John Fearnley · Marcin Jurdziński · Rahul Savani

Abstract We present a new, distributed method to compute approximate Nash equilibria in bimatrix games. In contrast to previous approaches that analyze the two payoff matrices at the same time (for example, by solving a single LP that combines the two players' payoffs), our algorithm first solves two independent LPs, each of which is derived from one of the two payoff matrices, and then computes an approximate Nash equilibrium using only limited communication between the players. Our method gives improved bounds on the complexity of computing approximate Nash equilibria in a number of different settings. Firstly, it gives a polynomial-time algorithm for computing *approximate well supported Nash equilibria (WSNE)* that always finds a 0.6528-WSNE, beating the previous best guarantee of 0.6608. Secondly, since our algorithm solves the two LPs separately, it can be applied to give an improved bound in the limited communication setting, giving a randomized expected-polynomial-time algorithm that uses poly-logarithmic communication and finds a 0.6528-WSNE, which beats the previous best known guarantee of 0.732. It can also be applied to the case of *approximate Nash equilibria*, where we obtain a randomized expected-polynomial-time algorithm that uses poly-logarithmic communication and always finds a 0.382-approximate Nash

The first, third and fifth author were partially supported by the Centre for Discrete Mathematics and its Applications (DIMAP) and EPSRC award EP/D063191/1. The second, fourth and sixth author were supported by EPSRC award EP/L011018/1. The second author was also supported by ISF award #2021296.

A. Czumaj · M.Fasoulakis · M. Jurdziński
Department of Computer Science and DIMAP, University of Warwick, UK

A. Deligkas
Faculty of Industrial Engineering and Management, Technion, Israel

J. Fearnley · R. Savani
Department of Computer Science, University of Liverpool, UK

M.Fasoulakis
Institute of Computer Science, Foundation for Research and Technology-Hellas (ICS-FORTH), Greece

equilibrium, which improves the previous best guarantee of 0.438. Finally, the method can also be applied in the query complexity setting to give an algorithm that makes $O(n \log n)$ payoff queries and always finds a 0.6528-WSNE, which improves the previous best known guarantee of 2/3.

1 Introduction

The problem of finding equilibria in non-cooperative games is a central problem in game theory. Nash's seminal theorem proved that every finite normal-form game has at least one *Nash equilibrium* [18], and this raises the natural question of whether we can find one efficiently. After several years of extensive research, it was shown that finding a Nash equilibrium is PPAD-complete [7] even for two-player *bimatrix games* [3], which is considered to be strong evidence that there is no polynomial-time algorithm for this problem.

Approximate equilibria. The fact that computing an exact Nash equilibrium of a bimatrix game is unlikely to be tractable, has led to the study of *approximate* Nash equilibria. There are two natural notions of approximate equilibrium, both of which will be studied in this paper. An *ϵ-approximate Nash equilibrium* (ϵ-NE) is a pair of strategies in which neither player can increase their expected payoff by more than ϵ by unilaterally deviating from their assigned strategy. An *ϵ-well-supported Nash equilibrium* (ϵ-WSNE) is a pair of strategies in which both players only place probability on strategies whose payoff is within ϵ of the best response payoff. Every ϵ-WSNE is an ϵ-NE but the converse does not hold, so a WSNE is a more restrictive notion.

Approximate Nash equilibria are the more well studied of the two concepts. A line of work has studied the best guarantee that can be achieved in polynomial time [2, 6, 8]. The best algorithm known so far is the gradient descent method of Tsaknakis and Spirakis [20] that finds a 0.3393-NE in polynomial time, and examples upon which the algorithm finds no better than a 0.3385-NE have been found [12]. On the other hand, progress on computing approximate-well-supported Nash equilibria has been less forthcoming. The first correct algorithm was provided by Kontogiannis and Spirakis [16] (which shall henceforth be referred to as the KS algorithm), who gave a polynomial time algorithm for finding a $\frac{2}{3}$-WSNE. This was later slightly improved by Fearnley et al. [10] (whose algorithm we shall refer to as the FGSS-algorithm), who gave a new polynomial-time algorithm that extends the KS algorithm and finds a 0.6608-WSNE; prior to this work, this was the best known approximation guarantee for WSNEs. For the special case of symmetric games, there is a polynomial-time algorithm for finding a $\frac{1}{2} + \delta$-WSNE [5].

Previously, it was considered a strong possibility that there is a PTAS for this problem (either for finding an ϵ-NE or ϵ-WSNE, since their complexity is polynomially related). A very recent result of Rubinstein [19] sheds serious doubt on this possibility. EndOfTheLine is the canonical problem that defines the complexity class PPAD. The "Exponential Time Hypothesis" (ETH) for EndOfTheLine says that any algorithm that solves an EndOfTheLine

instance with n-bit circuits, requires $2^{\tilde{\Omega}(n)}$ time. Rubinstein's result says that, subject to the ETH for EndOfTheLine, there exists a constant, but so far undetermined, ϵ^*, such that for $\epsilon < \epsilon^*$, every algorithm for finding an ϵ-NE takes quasi-polynomial time, so the quasi-PTAS of Lipton et al. [17] is optimal.

Communication complexity. Approximate Nash equilibria can also be studied from the *communication complexity* point of view, which captures the amount of communication the players need to find a good approximate Nash equilibrium. It models a natural scenario where the two players each know their own payoff matrix, but do not know their opponent's payoff matrix. The players must then follow a communication protocol that eventually produces strategies for both players. The goal is to design a protocol that produces a sufficiently good ϵ-NE or ϵ-WSNE while also minimizing the amount of communication between the two players.

Communication complexity of equilibria in games has been studied in previous works [4, 15]. The recent paper of Goldberg and Pastink [13] initiated the study of communication complexity in the bimatrix game setting. There they showed $\Theta(n^2)$ communication is required to find an exact Nash equilibrium of an $n \times n$ bimatrix game. Since these games have $\Theta(n^2)$ payoffs in total, this implies that there is no communication-efficient protocol for finding exact Nash equilibria in bimatrix games. For approximate equilibria, they showed that one can find a $\frac{3}{4}$-NE *without any communication*, and that in the no-communication setting, finding a $\frac{1}{2}$-NE is impossible. Motivated by these positive and negative results, they focused on the most interesting setting, which allows only a polylogarithmic (in n) number of bits to be exchanged between the players. They showed that one can compute 0.438-NE and 0.732-WSNE in this context. Recently Babichenko and Rubinstein [1] proved the first lower bounds for the communication complexity of ϵ-NE. They proved that for bimatrix games there is constant $\epsilon > 0$ such that polynomial communication (in n) is needed in order to compute an ϵ-NE. Furthermore, they showed that in N-player binary-action games there exists a constant $\epsilon > 0$ such that $2^{\Omega(N)}$ communication is needed for computing an ϵ-NE.

Query complexity. The payoff query model is motivated by practical applications of game theory. It is often the case that we know that there is a game to be solved, but we do not know what the payoffs are, and in order to discover the payoffs, we would have to play the game. This may be costly, so it is natural to ask whether we can find an equilibrium while minimizing the number of experiments that we must perform.

Payoff queries model this situation. In the payoff query model we are told the structure of the game, i.e., the strategy space, but we are not told the payoffs. We can then make payoff queries, where we propose a pure strategy profile, and we are told the payoff of each player under that strategy profile. Our task is to compute an equilibrium of the game while minimizing the number of payoff queries that we make.

The study of query complexity in bimatrix games was initiated by Fearnley et al. [11], who gave a deterministic algorithm for finding a $\frac{1}{2}$-NE using

$2n-1$ payoff queries. A subsequent paper of Fearnley and Savani [9] showed a number of further results. Firstly, they showed an $\Omega(n^2)$ lower bound on the query complexity of finding an ϵ-NE with $\epsilon < \frac{1}{2}$, which combined with the result above, gives a complete view of the deterministic query complexity of approximate Nash equilibria in bimatrix games. They then give a randomized algorithm that finds a $(\frac{3-\sqrt{5}}{2}+\epsilon)$-NE using $O(\frac{n\cdot\log n}{\epsilon^2})$ queries, and a randomized algorithm that finds a $(\frac{2}{3}+\epsilon)$-WSNE using $O(\frac{n\cdot\log n}{\epsilon^4})$ queries.

Our contribution. In this paper we introduce a *distributed* technique that allows us to efficiently compute ϵ-NE and ϵ-WSNE using limited communication between the players.

Traditional methods for computing WSNEs have used an LP based approach that, when used on a bimatrix game (R, C), solves the zero-sum game $(R-C, C-R)$. The KS algorithm uses the fact that if there is no pure $\frac{2}{3}$-WSNE, then the solution to this zero-sum game is a $\frac{2}{3}$-WSNE. The slight improvement of the FGSS-algorithm [10] to 0.6608 was obtained by adding two further methods to the KS algorithm: if the KS algorithm does not produce a 0.6608-WSNE, then either there is a 2×2 *matching pennies* sub-game that is 0.6608-WSNE or the strategies from the zero-sum game can be improved by shifting the probabilities of both players within their supports in order to produce a 0.6608-WSNE.

In this paper, we take a different approach. We first show that the bound of $\frac{2}{3}$ can be matched using a pair of *distributed* LPs. Given a bimatrix game (R, C), we solve the two zero-sum games $(R, -R)$ and $(-C, C)$, and then give a simple procedure that we call the *base algorithm*, which uses the solutions to these games to produce a $\frac{2}{3}$-WSNE of (R, C). Goldberg and Pastink [13] also considered this pair of LPs, but their algorithm only produces a 0.732-WSNE. We then show that the base algorithm can be improved by applying the probability-shifting and matching-pennies ideas from the FGSS-algorithm. That is, if the base algorithm fails to find a 0.6528-WSNE, then a 0.6528-WSNE can be obtained either by shifting the probabilities of one of the two players, or by identifying a 2×2 sub-game. This gives a polynomial-time algorithm that computes a 0.6528-WSNE, which provides the best known approximation guarantees for WSNEs.

It is worth pointing out that, while these techniques are thematically similar to the ones used by the FGSS-algorithm, the actual implementation is significantly different. The FGSS-algorithm attempts to improve the strategies by shifting probabilities *within the supports* of the strategies returned by the two player game, with the goal of reducing the other player's payoff. In our algorithm, we shift probabilities *away from bad strategies* in order to improve that player's payoff. This type of analysis is possible because the base algorithm produces a strategy profile in which one of the two players plays a pure strategy, which simplifies the analysis that we need to carry out. On the other hand, the KS-algorithm can produce strategies in which both players play many strategies, and so the analysis used for the FGSS-algorithm is necessarily more complicated.

Complexity setting	Payoffs	Solution	Previous best approximation	This paper
Computational (polynomial)	$[0,1]$	ϵ-WSNE	0.6608 [10]	0.6528
Query ($n \cdot log(n)$ queries)	$[0,1]$	ϵ-WSNE	0.6667 [9]	$0.6528+\epsilon$
Communication (polylogarithmic)	$[0,1]$	ϵ-WSNE	0.7321 [13]	$0.6528+\epsilon$
Communication (polylogarithmic)	$\{0,1\}$	ϵ-WSNE	0.7321 [13]	$0.5+\epsilon$
Communication (polylogarithmic)	$[0,1]$	ϵ-NE	0.4384 [13]	$0.3820+\epsilon$

Table 1 Comparison of our approximation guarantees with the previous best-known guarantees.

Since our algorithm solves the two LPs separately, it can be used to improve upon the best known algorithms in the limited communication setting. This is because no communication is required for the row player to solve $(R, -R)$ and the column player to solve $(-C, C)$. The players can then carry out the rest of the algorithm using only poly-logarithmic communication. Hence, we obtain a randomized expected-polynomial-time algorithm that uses poly-logarithmic communication and finds a 0.6528-WSNE. Moreover, the base algorithm can be implemented as a communication efficient algorithm for finding a $(\frac{1}{2}+\epsilon)$-WSNE in a *win-lose* bimatrix game, where all payoffs are either 0 or 1.

The algorithm can also be used to beat the best known bound in the query complexity setting. It has already been shown by Goldberg and Roth [14] that an ϵ-NE of a *zero-sum game* can be found by a randomized algorithm that uses $O(\frac{n \log n}{\epsilon^2})$ payoff queries. Since the rest of the steps used by our algorithm can also be carried out using $O(n \log n)$ payoff queries, this gives us a query efficient algorithm for finding a 0.6528-WSNE.

We also show that the base algorithm can be adapted to find a $\frac{3-\sqrt{5}}{2}$-NE in a bimatrix game. Once again, this can be implemented in a communication efficient manner, and so we obtain an algorithm that computes a $(\frac{3-\sqrt{5}}{2}+\epsilon)$-NE (i.e., 0.382-NE) using only poly-logarithmic communication.

Finally, we provide a lower bound against the base algorithm that is essentially tight, namely within 0.0034 of the theoretical upper bound.

2 Preliminaries

Bimatrix games. Throughout, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. An $n \times n$ bimatrix game is a pair (R, C) of two $n \times n$ matrices: R gives payoffs for the *row* player and C gives the payoffs for the *column* player. We make the standard assumption that all payoffs lie in the range $[0, 1]$. For simplicity, as in [13], we assume that each payoff has constant bit-length[1]. A *win-lose* bimatrix game has all payoffs in $\{0, 1\}$.

Each player has n *pure* strategies. To play the game, both players simultaneously select a pure strategy: the row player selects a row $i \in [n]$, and the

[1] The statements of our results can easily be extended to the case where all payoffs can be represented using b bits by including a factor b in all our communication complexity bounds.

column player selects a column $j \in [n]$. The row player then receives payoff $R_{i,j}$, and the column player receives payoff $C_{i,j}$.

A *mixed strategy* is a probability distribution over $[n]$. We denote a mixed strategy for the row player as a vector $\mathbf{x}$ of length n, such that $\mathbf{x}_i$ is the probability that the row player assigns to pure strategy i. A mixed strategy of the column player is a vector $\mathbf{y}$ of length n, with the same interpretation. Given a mixed strategy $\mathbf{x}$ for either player, the *support* of $\mathbf{x}$ is the set of pure strategies i with $\mathbf{x}_i > 0$. If $\mathbf{x}$ and $\mathbf{y}$ are mixed strategies for the row and the column player, respectively, then we call $(\mathbf{x}, \mathbf{y})$ a *mixed strategy profile*. The expected payoff for the row player under strategy profile $(\mathbf{x}, \mathbf{y})$ is given by $\mathbf{x}^T R\mathbf{y}$ and for the column player by $\mathbf{x}^T C\mathbf{y}$. We denote the *support* of a strategy $\mathbf{x}$ as $\text{supp}(\mathbf{x})$, which gives the set of pure strategies i such that $\mathbf{x}_i > 0$.

Nash equilibria. Let $\mathbf{y}$ be a mixed strategy for the column player. The set of *pure best responses* against $\mathbf{y}$ for the row player is the set of pure strategies that maximize the payoff against $\mathbf{y}$. More formally, a pure strategy $i \in [n]$ is a best response against $\mathbf{y}$ if, for all pure strategies $i' \in [n]$ we have: $\sum_{j\in[n]} \mathbf{y}_j \cdot R_{i,j} \geq \sum_{j\in[n]} \mathbf{y}_j \cdot R_{i',j}$. Column player best responses are defined analogously.

A mixed strategy profile $(\mathbf{x}, \mathbf{y})$ is a *mixed Nash equilibrium* if every pure strategy in $\text{supp}(\mathbf{x})$ is a best response against $\mathbf{y}$, and every pure strategy in $\text{supp}(\mathbf{y})$ is a best response against $\mathbf{x}$. Nash [18] showed that all bimatrix games have a mixed Nash equilibrium.

Approximate equilibria. There are two commonly studied notions of approximate equilibrium, and we consider both of them in this paper. The first notion is of an *ϵ-approximate Nash equilibrium* (ϵ-NE), which weakens the requirement that a player's expected payoff should be equal to their best response payoff. Formally, given a strategy profile $(\mathbf{x}, \mathbf{y})$, we define the *regret* suffered by the row player to be the difference between the best response payoff and actual payoff, i.e.,

$$\max_{i\in[n]} \big((R \cdot \mathbf{y})_i\big) - \mathbf{x}^T \cdot R \cdot \mathbf{y}.$$

The term $R \cdot \mathbf{y}$ is a vector where the i-th entry of the vector, $(R \cdot \mathbf{y})_i$, corresponds the payoff the row player gets from playing his i-th pure strategy when the column player plays $\mathbf{y}$. Hence, the maximum over this vector is the best response payoff for the row player. The term $\mathbf{x}^T \cdot R \cdot \mathbf{y}$ encodes the expected payoff to the row player under the strategy profile $(\mathbf{x}, \mathbf{y})$.

Regret for the column player is defined analogously. We have that $(\mathbf{x}, \mathbf{y})$ is an ϵ-NE if and only if both players have regret less than or equal to ϵ.

The other notion is of an ϵ-approximate-well-supported equilibrium (ϵ-WSNE), which weakens the requirement that players only place probability on best response strategies. Given a strategy profile $(\mathbf{x}, \mathbf{y})$ and a pure strategy $j \in [n]$, we say that j is an ϵ-best-response for the row player if

$$\max_{i\in[n]} \big((R \cdot \mathbf{y})_i\big) - (R \cdot \mathbf{y})_j \leq \epsilon.$$

An ϵ-WSNE requires that both players only place probability on ϵ-best-responses. In an ϵ-WSNE both players place probability only on ϵ-best-responses. Formally, the row player's *pure strategy regret* under $(\mathbf{x}, \mathbf{y})$ is defined to be

$$\max_{i \in [n]} \big((R \cdot \mathbf{y})_i\big) - \min_{i \in \text{supp}(\mathbf{x})} \big((R \cdot \mathbf{y})_i\big).$$

Pure strategy regret for the column player is defined analogously. A strategy profile $(\mathbf{x}, \mathbf{y})$ is an ϵ-WSNE if both players have pure strategy regret less than or equal to ϵ.

Communication complexity. We consider the communication model for bimatrix games introduced by Goldberg and Pastink [13]. In this model, both players know the payoffs in their own payoff matrix, but do not know the payoffs in their opponent's matrix. The players then follow an algorithm that uses a number of communication rounds, where in each round they exchange a single bit of information. Between each communication round, the players are permitted to perform arbitrary randomized computations (although it should be noted that, in this paper, the players will only perform polynomial-time computations) using their payoff matrix, and the bits that they have received so far. At the end of the algorithm, the row player outputs a mixed strategy $\mathbf{x}$, and the column player outputs a mixed strategy $\mathbf{y}$. The goal is to produce a strategy profile $(\mathbf{x}, \mathbf{y})$ that is an ϵ-NE or ϵ-WSNE for a sufficiently small ϵ while limiting the number of communication rounds used by the algorithm. The algorithms given in this paper will use at most $O(\log^2 n)$ communication rounds. In order to achieve this, we use the following result of Goldberg and Pastink [13].

Lemma 1 ([13]) *Given a mixed strategy* $\mathbf{x}$ *for the row-player and an* $\epsilon > 0$*, there is a randomized expected-polynomial-time algorithm that uses* $O(\frac{\log^2 n}{\epsilon^2})$*-communication to transmit a strategy* $\mathbf{x}_s$ *to the column player where* $|\text{supp}(\mathbf{x}_s)| \in O(\frac{\log n}{\epsilon^2})$ *and for every strategy* $i \in [n]$ *we have:*

$$|(\mathbf{x}^T \cdot R)_i - (\mathbf{x}_s^T \cdot R)_i| \leq \epsilon.$$

The algorithm uses the well-known sampling technique of Lipton, Markakis, and Mehta to construct the strategy $\mathbf{x}_s$, so for this reason we will call the strategy $\mathbf{x}_s$ the *sampled strategy* from $\mathbf{x}$. Since this strategy has a logarithmically sized support, it can be transmitted by sending $O(\frac{\log n}{\epsilon^2})$ strategy indexes, each of which can be represented using $\log n$ bits. By symmetry, the algorithm can obviously also be used to transmit approximations of column player strategies to the row player.

Query complexity. In the query complexity setting, the algorithm knows that the players will play an $n \times n$ game (R, C), but it does not know any of the entries of R or C. These payoffs are obtained using *payoff queries* in which the algorithm proposes a pure strategy profile (i, j), and then it is told the value of R_{ij} and C_{ij}. After each payoff query, the algorithm can make arbitrary computations (although, again, in this paper the algorithms that

we consider take polynomial time) in order to decide the next pure strategy profile to query. After making a sequence of payoff queries, the algorithm then outputs a mixed strategy profile $(\mathbf{x}, \mathbf{y})$. Again, the goal is to ensure that this strategy profile is an ϵ-NE or ϵ-WSNE, while minimizing the number of queries made overall.

There are two results that we will use for this setting. Goldberg and Roth [14] have given a randomized algorithm that, with high probability, finds an ϵ-NE of a zero-sum game using $O(\frac{n \cdot \log n}{\epsilon^2})$ payoff queries. Given a mixed strategy profile $(\mathbf{x}, \mathbf{y})$, an *ϵ-approximate payoff vector* for the row player is a vector v such that, for all $i \in [n]$ we have $|v_i - (R \cdot \mathbf{y})_i| \leq \epsilon$. Approximate payoff vectors for the column player are defined symmetrically. Fearnley and Savani [9] observed that there is a randomized algorithm that when given the strategy profile $(\mathbf{x}, \mathbf{y})$, finds approximate payoff vectors for both players using $O(\frac{n \cdot \log n}{\epsilon^2})$ payoff queries and that succeeds with high probability. We summarise these two results in the following lemma.

Lemma 2 ([9, 14]) *Given an $n \times n$ zero-sum bimatrix game, with probability at least $(1 - n^{-\frac{1}{8}})(1 - \frac{2}{n})^2$, we can compute an ϵ-Nash equilibrium $(\mathbf{x}, \mathbf{y})$, and ϵ-approximate payoff vectors for both players under $(\mathbf{x}, \mathbf{y})$, using $O(\frac{n \cdot \log n}{\epsilon^2})$ payoff queries.*

3 The base algorithm

In this section, we introduce the *base algorithm*, which provides a simple way to find a $\frac{2}{3}$-WSNE. We present this algorithm separately for three reasons. Firstly, the algorithm is interesting in its own right, since it provides a relatively straightforward method for finding a $\frac{2}{3}$-WSNE that is quite different from the technique used in the KS-algorithm. Secondly, our algorithm for finding a 0.6528-WSNE will build on this algorithm and replace its final step with two more involved procedures. Thirdly, at the end of this section, we show how this algorithm can be adapted to provide a communication efficient way to find a $(0.5 + \epsilon)$-WSNE in win-lose games.

Algorithm 1

1. Solve the zero-sum games $(R, -R)$ and $(-C, C)$.
   - Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a NE of $(R, -R)$, and let $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ be a NE of $(-C, C)$.
   - Let v_r be the value secured by $\mathbf{x}^*$ in $(R, -R)$, and let v_c be the value secured by $\hat{\mathbf{y}}$ in $(-C, C)$. Without loss of generality assume that $v_c \leq v_r$.
2. If $v_r \leq 2/3$, then return $(\hat{\mathbf{x}}, \mathbf{y}^*)$.
3. If for all $j \in [n]$ it holds that $C_j^T \cdot \mathbf{x}^* \leq 2/3$, then return $(\mathbf{x}^*, \mathbf{y}^*)$.
4. Otherwise:
   - Let j^* be a pure best response to $\mathbf{x}^*$.
   - Find a row i such that $R_{i\mathrm{j}^*} > 1/3$ and $C_{i\mathrm{j}^*} > 1/3$.
   - Return (i, j^*).

We argue that this algorithm is correct. For that, we must prove that the row i used in Step 4 actually exists.

Lemma 3 *If Algorithm 1 reaches Step 4, then there exists a row i such that $R_{i\mathrm{j}^*} > 1/3$ and $C_{i\mathrm{j}^*} > 1/3$.*

Proof Let i be a row sampled from $\mathbf{x}^*$. We will show that there is a positive probability that row i satisfies the desired properties.

We begin by showing that the probability that $\Pr(R_{i\mathrm{j}^*} \leq \frac{1}{3}) < 0.5$. Let the random variable $T = 1 - R_{i\mathrm{j}^*}$, where i is chosen according to the probability distribution $\mathbf{x}^*$. Note that $E[R_{i\mathrm{j}^*}] = (\mathbf{x}^* R)_j$. Since $v_r > \frac{2}{3}$, we claim that $E[T] < \frac{1}{3}$. This follows from the fact that x^* is a minmax strategy for the row player, and thus it guarantees a payoff more than $2/3$ against any pure strategy of the column player.

Thus, applying Markov's inequality we obtain:

$$\Pr(T \geq \frac{2}{3}) \leq \frac{E[T]}{2/3} < 0.5.$$

Since $\Pr(R_{i\mathrm{j}^*} \leq \frac{1}{3}) = \Pr(T \geq \frac{2}{3})$ we can therefore conclude that $\Pr(R_{i\mathrm{j}^*} \leq \frac{1}{3}) < 0.5$. The exact same technique can be used to prove that $\Pr(C_{i\mathrm{j}^*} \leq \frac{1}{3}) < 0.5$, by using the fact that $C_{\mathrm{j}^*}^T \cdot \mathbf{x}^* > \frac{2}{3}$.

We can now apply the union bound to argue that:

$$\Pr(R_{i\mathrm{j}^*} \leq \frac{1}{3} \text{ or } C_{i\mathrm{j}^*} \leq \frac{1}{3}) < 1.$$

Hence, there is positive probability that row i satisfies $R_{i\mathrm{j}^*} > \frac{1}{3}$ and $C_{i\mathrm{j}^*} > \frac{1}{3}$, so such a row must exist. □

We now argue that the algorithm always produces a $\frac{2}{3}$-WSNE. There are three possible strategy profiles that can be returned by the algorithm, which we consider individually.

The algorithm returns in Step 2: Since $v_c \leq v_r$ by assumption, and since $v_r \leq \frac{2}{3}$, we have that $(R \cdot \mathbf{y}^*)_i \leq \frac{2}{3}$ for every row i, and $((\hat{\mathbf{x}})^T \cdot C)_j \leq \frac{2}{3}$ for every column j. So, both players can have pure strategy regret at most $\frac{2}{3}$ in $(\hat{\mathbf{x}}, \mathbf{y}^*)$, and thus this profile is a $\frac{2}{3}$-WSNE.

The algorithm returns in Step 3: Much like in the previous case, when the column player plays $\mathbf{y}^*$, the row player can have pure strategy regret at most $\frac{2}{3}$; observe that in this case the row player actually suffers zero regret since $\mathbf{x}^*$ is a best response against $\mathbf{y}^*$. The requirement that $C_j^T \mathbf{x}^* \leq \frac{2}{3}$ also ensures that the column player has pure strategy regret at most $\frac{2}{3}$. Thus, we have that $(\mathbf{x}^*, \mathbf{y}^*)$ is a $\frac{2}{3}$-WSNE.

The algorithm returns in Step 4: Both players have payoff at least $\frac{1}{3}$ under $(i, \mathfrak{j}^*)$ for the sole strategy in their respective supports. Hence, the maximum pure strategy regret that can be suffered by a player is $1 - \frac{1}{3} = \frac{2}{3}$.

Observe that the zero-sum games solved in Step 1 can be solved via linear programming, and so the algorithm runs in polynomial time. Therefore, we have shown the following.

Theorem 4 *Algorithm 1 always produces a $\frac{2}{3}$-WSNE in polynomial time.*

3.1 Win-lose games

The base algorithm can be adapted to provide a communication efficient method for finding a $(0.5 + \epsilon)$-WSNE in win-lose games. In brief, the algorithm can be modified to find a 0.5-WSNE in a win-lose game by making Steps 2 and 3 check against the threshold of 0.5. It can then be shown that if these steps fail, then there exists a pure Nash equilibrium in column $\mathfrak{j}^*$. This can then be implemented as a communication efficient protocol using the algorithm from Lemma 1.

Formally, we will study the following simple modification of Algorithm 1.

Algorithm 2

1. Solve the zero-sum games $(R, -R)$ and $(-C, C)$.
   - Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a NE of $(R, -R)$, and let $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ be a NE of $(C, -C)$.
   - Let v_r be the value secured by $\mathbf{x}^*$ in $(R, -R)$, and let v_c be the value secured by $\hat{\mathbf{y}}$ in $(-C, C)$. Without loss of generality assume that $v_c \leq v_r$.
2. If $v_r \leq 0.5$, then return $(\hat{\mathbf{x}}, \mathbf{y}^*)$.
3. If for all $j \in [n]$ it holds that $C_j^T \cdot \mathbf{x}^* \leq 0.5$, then return $(\mathbf{x}^*, \mathbf{y}^*)$.
4. Otherwise:
   - Let $\mathfrak{j}^*$ be a pure best response to $\mathbf{x}^*$.
   - Find a row i such that $R_{i\mathfrak{j}^*} = 1$ and $C_{ij} = 1$.
   - Return $(i, \mathfrak{j}^*)$.

We will show that this algorithm always finds a 0.5-WSNE in a win-lose game. Firstly, we show that the pure Nash equilibrium found in Step 4 always exists. The following lemma is similar to Lemma 3, but exploits the fact that the game is win-lose to obtain a stronger conclusion.

Lemma 5 *If Algorithm 2 is applied to a win-lose game, and it reaches Step 4, then there exists a row* $i \in \text{supp}(\mathbf{x}^*)$ *such that* $R_{ij^*} = 1$ *and* $C_{ij^*} = 1$.

Proof Let i be a row sampled from $\mathbf{x}^*$. We will show that there is a positive probability that row i satisfies the desired properties.

We begin by showing that the probability that $\Pr(R_{ij^*} = 0) < 0.5$. Let the random variable $T = 1 - R_{ij^*}$. Since $v_r > \frac{1}{2}$, we have that $E[T] < 0.5$. Thus, applying Markov's inequality we obtain:

$$\Pr(T \geq 1) \leq \frac{E[T]}{1} < 0.5.$$

Since $\Pr(R_{ij^*} = 0) = \Pr(T \geq 1)$ we can therefore conclude that $\Pr(R_{ij^*} = 0) < 0.5$. The exact same technique can be used to prove that $\Pr(C_{ij^*} = 0) < 0.5$, by using the fact that $C_{j^*}^T \cdot \mathbf{x}^* > 0.5$.

We can now apply the union bound to argue that:

$$\Pr(R_{ij^*} = 0 \text{ or } C_{ij^*} = 0) < 1.$$

Hence, there is positive probability that row i satisfies $R_{ij^*} > 0$ and $C_{ij^*} > 0$, so such a row must exist. The final step is to observe that, since the game is win-lose, we have that $R_{ij^*} > 0$ implies $R_{ij^*} = 1$, and that $C_{ij^*} > 0$ implies $C_{ij^*} = 1$. □

We now prove that the algorithm always finds a 0.5-WSNE. The reasoning is very similar to the analysis of the base algorithm. The strategy profiles returned by Steps 2 and 3 are 0.5-WSNEs by the same reasoning that was given for the base algorithm. Step 4 always returns a pure Nash equilibrium, which is a 0-WSNE.

Communication complexity. We now show that Algorithm 2 can be implemented in a communication efficient way.

The zero-sum games in Step 1 can be solved by the two players independently without any communication. Then, the players exchange v_r and v_s using $O(\log n)$ rounds of communication. If both v_r and v_s are smaller than 0.5, then the players use the algorithm from Lemma 1 to communicate $\hat{\mathbf{x}}_s$ and $\mathbf{y}_s^*$ between themselves, using the parameter $\frac{\epsilon}{2}$ in place of ϵ. Since the payoffs under the sampled strategies are within $\frac{\epsilon}{2}$ of the originals, we have that all pure strategies have payoff less than or equal to $0.5 + \epsilon$ under $(\hat{\mathbf{x}}_s, \mathbf{y}_s^*)$, so this strategy profile is a $(0.5 + \epsilon)$-WSNE.

We will assume from now on that $v_r > v_c$. If the algorithm reaches Step 3, then the row player uses the algorithm of Lemma 1 to communicate $\mathbf{x}_s^*$ to the column player. The column player then computes a best response j_s^* against

$\mathbf{x}_s^*$, and checks whether the payoff of j_s^* against $\mathbf{x}_s^*$ is less than or equal to $0.5+\epsilon$. If so, then the players output $(\mathbf{x}_s^*, \mathsf{j}_s^*)$, which is a $(0.5+\epsilon)$-WSNE

Otherwise, we claim that there is a pure strategy $i \in \mathrm{supp}(\mathbf{x}_s^*)$ such that (i, j_s^*) is a pure Nash equilibrium. This can be shown by observing that the expected payoff of $\mathbf{x}_s^*$ against j_s^* is at least $0.5-\epsilon$, while the expected payoff of j_s^* against $\mathbf{x}_s^*$ is at least $0.5+\epsilon$. Repeating the proof of Lemma 5 using these inequalities then shows that the pure Nash equilibrium does indeed exist. Since $\mathrm{supp}(\mathbf{x}_s^*)$ has logarithmic size, the row player can simply transmit to the column player all payoffs $R_{i\mathsf{j}_s^*}$ for which $i \in \mathrm{supp}(\mathbf{x}_s^*)$, and the column player can then send back a row corresponding to a pure Nash equilibrium.

In conclusion, we have shown the following theorem.

Theorem 6 *For every win-lose game and $\epsilon > 0$, there is a randomized expected-polynomial-time algorithm that finds a $(0.5+\epsilon)$-WSNE with $O\left(\frac{\log^2 n}{\epsilon^2}\right)$ communication.*

4 An algorithm for finding a 0.6528-WSNE

In this section, we show how Algorithm 1 can be modified to produce a 0.6528-WSNE.

Outline. Our algorithm replaces Step 4 of Algorithm 1 with a more involved procedure. This procedure uses two techniques, that both find an ϵ-WSNE with $\epsilon < \frac{2}{3}$. Firstly, we attempt to turn $(\mathbf{x}^*, \mathsf{j}^*)$ into a WSNE by *shifting probabilities*. Observe that, since j^* is a best response, the column player has a pure strategy regret of 0 in $(\mathbf{x}^*, \mathsf{j}^*)$. On the other hand, we have no guarantees about the row player since $\mathbf{x}^*$ might place a small amount of probability strategies with payoff strictly less than $\frac{1}{3}$. However, since $\mathbf{x}^*$ achieves a high *expected* payoff (due to Step 2,) it cannot place too much probability on these low payoff strategies. Thus, the idea is to shift the probability that $\mathbf{x}^*$ assigns to entries of j^* with payoff less than or equal to $\frac{1}{3}$ to entries with payoff strictly greater than $\frac{1}{3}$, and thus ensure that the row player's pure strategy regret is below $\frac{2}{3}$. Of course, this procedure will increase the pure strategy regret of the column player, but if it is also below $\frac{2}{3}$ once all probability has been shifted, then we have found an ϵ-WSNE with $\epsilon < \frac{2}{3}$.

If shifting probabilities fails to find an ϵ-WSNE with $\epsilon < \frac{2}{3}$, then we show that the game contains a *matching pennies* sub-game. More precisely, we show that there exists a column j', and rows b and s such that the 2×2 sub-game induced by j^*, j', b, and s has the following form:

I \ II	j^*	j'
b	0 ≈ 1	≈ 1 0
s	≈ 1 0	0 ≈ 1

Thus, if both players play uniformly over their respective pair of strategies, then j^*, j', b, and s with have payoff ≈ 0.5, and so this yields an ϵ-WSNE with $\epsilon < \frac{2}{3}$.

The algorithm. We now formalize this approach, and show that it always finds an ϵ-WSNE with $\epsilon < \frac{2}{3}$. In order to quantify the precise ϵ that we obtain, we parametrise the algorithm by a variable z, which we constrain to be in the range $0 \leq z < \frac{1}{24}$. With the exception of the matching pennies step, all other steps of the algorithm will return a $(\frac{2}{3} - z)$-WSNE, while the matching pennies step will return a $(\frac{1}{2} + f(z))$-WSNE for some increasing function f. Optimizing the trade off between $\frac{2}{3} - z$ and $\frac{1}{2} + f(z)$ then allows us to determine the quality of WSNE found by our algorithm.

The algorithm is displayed as Algorithm 3. Steps 1, 2, and 3 are versions of the corresponding steps from Algorithm 1, which have been adapted to produce a $(\frac{2}{3} - z)$-WSNE. Step 4 implements the probability shifting procedure, while Step 5 finds a matching pennies sub-game.

Observe that the probabilities used in $\mathbf{x_{mp}}$ and $\mathbf{y_{mp}}$ are only well defined when $z \leq \frac{1}{24}$, because we have that $\frac{1-15z}{2-39z} > 1$ whenever $z > \frac{1}{24}$, which explains our required upper bound on z.

Algorithm 3

1. Solve the zero-sum games $(R, -R)$ and $(-C, C)$.
   - Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a NE of $(R, -R)$, and let $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ be a NE of $(C, -C)$.
   - Let v_r be the value secured by $\mathbf{x}^*$ in $(R, -R)$, and let v_c be the value secured by $\hat{\mathbf{y}}$ in $(-C, C)$. Without loss of generality assume that $v_c \leq v_r$.
2. If $v_r \leq 2/3 - z$, then return $(\hat{\mathbf{x}}, \mathbf{y}^*)$.
3. If for all $j \in [n]$ it holds that $C_j^T \mathbf{x}^* \leq 2/3 - z$, then return $(\mathbf{x}^*, \mathbf{y}^*)$.
4. Otherwise:
   - Let j^* be a pure best response against $\mathbf{x}^*$. Define:
   $$S := \{i \in \mathrm{supp}(\mathbf{x}^*) : R_{i\mathrm{j}^*} < 1/3 + z\}$$
   $$B := \mathrm{supp}(\mathbf{x}^*) \setminus S$$
   - Define the strategy $\mathbf{x_B}$ as follows. For each $i \in [n]$ we have:
   $$(\mathbf{x_B})_i = \begin{cases} \frac{1}{\Pr(B)} \cdot \mathbf{x}^*_i & \text{if } i \in B \\ 0 & \text{otherwise.} \end{cases}$$
   - If $(\mathbf{x_B}^T \cdot C)_{\mathrm{j}^*} \geq \frac{1}{3} + z$, then return $(\mathbf{x_B}, \mathrm{j}^*)$.
5. Otherwise:
   - Let j' be a pure best response against $\mathbf{x_B}$.
   - If there exists an $i \in \mathrm{supp}(\mathbf{x}^*)$ such that (i, j^*) or (i, j') is a pure $(\frac{2}{3} - z)$-WSNE, then return it.
   - Find a row $b \in B$ such that $R_{b\mathrm{j}^*} > 1 - \frac{18z}{1+3z}$ and $C_{b\mathrm{j}'} > 1 - \frac{18z}{1+3z}$.
   - Find a row $s \in S$ such that $C_{s\mathrm{j}^*} > 1 - \frac{27z}{1+3z}$ and $R_{s\mathrm{j}'} > 1 - \frac{27z}{1+3z}$.
   - Define the row player strategy $\mathbf{x_{mp}}$ and the column player strategy $\mathbf{y_{mp}}$ as follows. For each $i \in [n]$ we have:
   $$\mathbf{x_{mp}}_i = \begin{cases} \frac{1-24z}{2-39z} & \text{if } i = b, \\ \frac{1-15z}{2-39z} & \text{if } i = s, \\ 0 & \text{otherwise.} \end{cases} \qquad \mathbf{y_{mp}}_i = \begin{cases} \frac{1-24z}{2-39z} & \text{if } i = \mathrm{j}^*, \\ \frac{1-15z}{2-39z} & \text{if } i = \mathrm{j}', \\ 0 & \text{otherwise.} \end{cases}$$
   - Return $(\mathbf{x_{mp}}, \mathbf{y_{mp}})$.

The correctness of Step 5. This step of the algorithm relies on the existence of the rows b and s, which is shown in the following lemma.

Lemma 7 *Suppose that the following conditions hold:*

1. $\mathbf{x}^*$ *has payoff at least* $\frac{2}{3} - z$ *against* j^*.
2. j^* *has payoff at least* $\frac{2}{3} - z$ *against* $\mathbf{x}^*$.
3. $\mathbf{x}^*$ *has payoff at least* $\frac{2}{3} - z$ *against* j'.
4. *Neither* j^* *nor* j' *contains a pure* $(\frac{2}{3} - z)$*-WSNE* (i, j) *with* $i \in \mathrm{supp}(\mathbf{x}^*)$.

Then, both of the following are true:

- *There exists a row* $b \in B$ *such that* $R_{b\mathrm{j}^*} > 1 - \frac{18z}{1+3z}$ *and* $C_{b\mathrm{j}'} > 1 - \frac{18z}{1+3z}$.
- *There exists a row* $s \in S$ *such that* $C_{s\mathrm{j}^*} > 1 - \frac{27z}{1+3z}$ *and* $R_{s\mathrm{j}'} > 1 - \frac{27z}{1+3z}$.

We defer the proof of this lemma to Section 4.1. Observe that the preconditions are indeed true if the Algorithm reaches Step 5. The first and third conditions hold because, due to Step 2, we know that $\mathbf{x}^*$ is a min-max strategy that secures payoff at least $v_r > \frac{2}{3} - z$. The second condition holds because Step 3 ensures that the column player's best response payoff is at least $\frac{2}{3} - z$. The fourth condition holds because Step 5 explicitly checks for these pure strategy profiles.

Quality of approximation. We now analyse the quality of WSNE our algorithm produces. Steps 2, 3, 4, 5 each return a strategy profile. Observe that Steps 2 and 3 are the same as the respective steps in the base algorithm, but with the threshold changed from $\frac{2}{3}$ to $\frac{2}{3} - z$. Hence, we can use the same reasoning as we gave for the base algorithm to argue that these steps return $(\frac{2}{3} - z)$-WSNE. We now consider the other two steps.

The algorithm returns in Step 4: By definition all rows $r \in B$ satisfy $R_{ij^*} \geq \frac{1}{3} + z$, so since $\text{supp}(\mathbf{x_B}) \subseteq B$, the pure strategy regret of the row player can be at most $1 - (\frac{1}{3} + z) = \frac{2}{3} - z$. For the same reason, since $(\mathbf{x_B}^T \cdot C)_{\mathsf{j}^*} \geq \frac{1}{3} + z$ holds, the pure strategy regret of the column player can also be at $\frac{2}{3} - z$. Thus, the profile $(\mathbf{x_B}, \mathsf{j}^*)$ is a $(\frac{2}{3} - z)$-WSNE.

The algorithm returns in Step 5: Since $R_{b\mathsf{j}^*} > 1 - \frac{18z}{1+3z}$, the payoff of b when the column player plays $\mathbf{y_{mp}}$ is at least:

$$\frac{1 - 24z}{2 - 39z} \cdot \left(1 - \frac{18z}{1 + 3z}\right) = \frac{1 - 39z + 360z^2}{2 - 33z - 117z^2}$$

Similarly, since $R_{s\mathsf{j}'} > 1 - \frac{27z}{1+3z}$, the payoff of s when the column player plays $\mathbf{y_{mp}}$ is at least:

$$\frac{1 - 15z}{2 - 39z} \cdot \left(1 - \frac{27z}{1 + 3z}\right) = \frac{1 - 39z + 360z^2}{2 - 33z - 117z^2}$$

In the same way, one can show that the payoffs of j^* and j' are also $\frac{1-39z+360z^2}{2-33z-117z^2}$ when the row player plays $\mathbf{x_{mp}}$. Thus, we have that $(\mathbf{x_{mp}}, \mathbf{y_{mp}})$ is a $(1 - \frac{1-39z+360z^2}{2-33z-117z^2})$-WSNE.

To find the optimal value for z, we need to find the largest value of z for which the following inequality holds.

$$1 - \frac{1 - 39z + 360z^2}{2 - 33z - 117z^2} \leq \frac{2}{3} - z.$$

Setting the inequality to an equality and rearranging gives us a cubic polynomial equation: $117\,z^3 + 432\,z^2 - 30\,z + \frac{1}{3} = 0$. Since the discriminant of this polynomial is positive, this polynomial has three real roots, which can be found via the trigonometric method. Only one of these roots lies in the range

$0 \le z < \frac{1}{24}$, which is the following:

$$z = \frac{1}{117}\sqrt{3}\left(\sqrt{2434}\sqrt{3}\cos\left(\frac{1}{3}\arctan\left(\frac{39}{240073}\sqrt{9749}\sqrt{3}\right)\right)\right.$$
$$\left. - 3\sqrt{2434}\sin\left(\frac{1}{3}\arctan\left(\frac{39}{240073}\sqrt{9749}\sqrt{3}\right)\right) - 48\sqrt{3}\right).$$

Thus, we get $z \approx 0.013906376$, and we have found an algorithm that always produces a 0.6528-WSNE. So we have the following theorem.

Theorem 8 *There is a polynomial time algorithm that, given a bimatrix game, finds a* 0.6528*-WSNE.*

4.1 Proof of Lemma 7

In this section we assume that Steps 1 through 4 of our algorithm did not return a $(\frac{2}{3} - z)$-WSNE, and that neither j^* nor j' contained a pure $(\frac{2}{3} - z)$-WSNE. We show that, under these assumptions, the rows b and s required by Step 5 do indeed exist.

Probability bounds. We begin by proving bounds on the amount of probability that $\mathbf{x}^*$ can place on S and B. The following lemma uses the fact that $\mathbf{x}^*$ secures an expected payoff of at least $\frac{2}{3} - z$ to give an upper bound on the amount of probability that $\mathbf{x}^*$ can place on S. To simplify notation, we use $\Pr(B)$ to denote the probability assigned by $\mathbf{x}^*$ to the rows in B, and we use $\Pr(S)$ to denote the probability assigned by $\mathbf{x}^*$ to the rows in S.

Lemma 9 $\Pr(S) \le \frac{1+3z}{2-3z}$.

Proof We will prove our claim using Markov's inequality. Consider the random variable $T = 1 - R_{ij^*}$ where i is sampled from $\mathbf{x}^*$. Since by our assumption the expected payoff of the row player is greater than $2/3 - z$ we get that $E(T) \le 1/3 + z$. If we apply Markov's inequality we get

$$Pr(T \ge \frac{2}{3} - z) \le \frac{E(T)}{\frac{2}{3} - z} \le \frac{1 + 3z}{2 - 3z}$$

which is the claimed result. □

Next we show an upper bound on $\Pr(B)$. Here we use the fact that j^* does not contain a $(\frac{2}{3} - z)$-WSNE to argue that all column player payoffs in B are smaller than $\frac{1}{3} + z$. Since we know that the payoff of j^* against $\mathbf{x}^*$ is at least $\frac{2}{3} - z$, this can be used to prove a upper bound on the amount of probability that $\mathbf{x}^*$ assigns to B.

Lemma 10 $\Pr(B) \le \frac{1+3z}{2-3z}$.

Proof Since there is no $i \in \text{supp}(\mathbf{x}^*)$ such that (i, j^*) is a pure $(\frac{2}{3} - z)$-WSNE, and since each row $i \in B$ satisfies $R_{i\mathrm{j}^*} \geq \frac{1}{3} + z$, we must have that $C_{i\mathrm{j}^*} < \frac{1}{3} + z$ for every $i \in B$. By assumption we know that $C_{\mathrm{j}^*}^T \mathbf{x}^* > 2/3 - z$. So, we have the following inequality:

$$\frac{2}{3} - z < \Pr(B) \cdot (\frac{1}{3} + z) + \big(1 - \Pr(B)\big) \cdot 1.$$

Solving this inequality for $\Pr(B)$ gives the desired result. □

Payoff inequalities for j^*. We now show properties about the average payoff obtained from the rows in B and S. Recall that $\mathbf{x_B}$ was defined in Step 4 of our algorithm, and that it denotes the normalization of the probability mass assigned by $\mathbf{x}^*$ to rows in B. The following lemma shows that the expected payoff to the row player in the strategy profile $(\mathbf{x_B}, \mathrm{j}^*)$ is close to 1.

Lemma 11 *We have* $(\mathbf{x_B}^T \cdot R)_{\mathrm{j}^*} > \frac{1-6z}{1+3z}$.

Proof By definition we have that:

$$(\mathbf{x_B}^T \cdot R)_{\mathrm{j}^*} = \frac{1}{\Pr(B)} \cdot \sum_{i \in B} \mathbf{x}_i^* \cdot R_{i\mathrm{j}^*}. \tag{1}$$

We begin by deriving a lower bound for $\sum_{i \in B} \mathbf{x}_i^* \cdot R_{i\mathrm{j}^*}$. Using the fact that $\mathbf{x}^*$ secures an expected payoff of at least $2/3 - z$ against j^* and then applying the bound from Lemma 9 gives:

$$\begin{aligned} \frac{2}{3} - z &< \sum_{i \in B} \mathbf{x}_i^* \cdot R_{i\mathrm{j}^*} + (\frac{1}{3} + z) \cdot \Pr(S) \\ &\leq \sum_{i \in B} \mathbf{x}_i^* \cdot R_{i\mathrm{j}^*} + (\frac{1}{3} + z) \cdot \frac{1+3z}{2-3z}. \end{aligned}$$

Hence we can conclude that:

$$\begin{aligned} \sum_{i \in B} \mathbf{x}_i^* \cdot R_{i\mathrm{j}^*} &> \frac{2}{3} - z - \frac{1}{3} \cdot \frac{(1+3z)^2}{2-3z} \\ &= \frac{1-6z}{2-3z}. \end{aligned}$$

Substituting this into Equation (1), along with the upper bound on $\Pr(B)$ from Lemma 10, allows us to conclude that:

$$\begin{aligned} (\mathbf{x_B}^T \cdot R)_{\mathrm{j}^*} &\geq \frac{2-3z}{1+3z} \cdot \sum_{i \in B} \mathbf{x}_i^* \cdot R_{i\mathrm{j}^*} \\ &> \frac{2-3z}{1+3z} \cdot \frac{1-6z}{2-3z} \\ &= \frac{1-6z}{1+3z}. \end{aligned}$$

□

Next we would like to show a similar bound on the expected payoff to the column player of the rows in S. To do this, we define $\mathbf{x_S}$ to be the normalisation of the probability mass that $\mathbf{x}^*$ assigns to the rows in S. More formally, for each $i \in [n]$, we define:

$$(\mathbf{x_S})_i = \begin{cases} \frac{1}{\Pr(S)} \cdot \mathbf{x}^*_i & \text{if } i \in S \\ 0 & \text{otherwise.} \end{cases}$$

The next lemma shows that the expected payoff to the column player in the profile $(\mathbf{x_S}, \mathrm{j}^*)$ is close to 1.

Lemma 12 *We have* $(\mathbf{x_S}^T \cdot C)_{\mathrm{j}^*} > \frac{1-6z}{1+3z}$.

Proof By definition we have that:

$$(\mathbf{x_S}^T \cdot C)_{\mathrm{j}^*} = \frac{1}{\Pr(S)} \cdot \sum_{i \in S} \mathbf{x}^*_i \cdot C_{i\mathrm{j}^*}. \tag{2}$$

We begin by deriving a lower bound for $\sum_{i \in S} \mathbf{x}^*_i \cdot C_{i\mathrm{j}^*}$. By assumption, we know that $C^T_{\mathrm{j}^*}\mathbf{x}^* > 2/3 - z$. Moreover, since j^* does not contain a $(\frac{2}{3} - z)$-WSNE we have that all rows i in B satisfy $C_{i\mathrm{j}^*} \leq 1/3 + z$. If we combine these facts that with Lemma 10 we obtain:

$$\begin{aligned} \frac{2}{3} - z &< \sum_{i \in S} \mathbf{x}^*_i \cdot C_{i\mathrm{j}^*} + (\frac{1}{3} + z) \cdot \Pr(B) \\ &\leq \sum_{i \in S} \mathbf{x}^*_i \cdot C_{i\mathrm{j}^*} + (\frac{1}{3} + z) \cdot \frac{1+3z}{2-3z}. \end{aligned}$$

Hence we can conclude that:

$$\begin{aligned} \sum_{i \in S} \mathbf{x}^*_i \cdot C_{i\mathrm{j}^*} &> \frac{2}{3} - z - \frac{1}{3} \cdot \frac{(1+3z)^2}{2-3z} \\ &= \frac{1-6z}{2-3z}. \end{aligned}$$

Substituting this into Equation (2), along with the upper bound on $\Pr(S)$ from Lemma 10, allows us to conclude that:

$$\begin{aligned} (\mathbf{x_B}^T \cdot R)_{\mathrm{j}^*} &\geq \frac{2-3z}{1+3z} \cdot \sum_{i \in B} \mathbf{x}^*_i \cdot R_{i\mathrm{j}^*} \\ &> \frac{2-3z}{1+3z} \cdot \frac{1-6z}{2-3z} \\ &= \frac{1-6z}{1+3z}. \end{aligned}$$

□

Payoff inequalities for j'. We now want to prove similar inequalities for the column j'. The next lemma shows that the expected payoff for the column player in the profile $(\mathbf{x_B}, \mathsf{j}')$ is close to 1. This is achieved by first showing a lower bound on the payoff to the column player in the profile $(\mathbf{x_B}, \mathsf{j}^*)$, and then using the fact that j^* is not a $(\frac{2}{3} - z)$-best-response against $\mathbf{x_B}$, and that j' is a best response against $\mathbf{x_B}$.

Lemma 13 *We have* $(\mathbf{x_B}^T \cdot C)_{\mathsf{j}'} > \frac{1-6z}{1+3z}$.

Proof We first establish a lower bound on $(\mathbf{x_B}^T \cdot C)_{\mathsf{j}^*}$. By assumption, we know that $C_{\mathsf{j}^*}^T \mathbf{x}^* > 2/3 - z$. Using this fact, along with the bounds from Lemmas 9 and 10 gives:

$$\begin{aligned}\frac{2}{3} - z &< \Pr(B) \cdot (\mathbf{x_B}^T \cdot C)_{\mathsf{j}^*} + \Pr(S) \cdot 1 \\ &\leq \frac{1+3z}{2-3z} \cdot (\mathbf{x_B}^T \cdot C)_{\mathsf{j}^*} + \frac{1+3z}{2-3z}.\end{aligned}$$

Solving this inequality for $(\mathbf{x_B}^T \cdot C)_{\mathsf{j}^*}$ yields:

$$(\mathbf{x_B}^T \cdot C)_{\mathsf{j}^*} > \frac{1}{3} \cdot \frac{1 - 21z + 9z^2}{1+3z}.$$

Now we can prove the lower bound on $(\mathbf{x_B}^T \cdot C)_{\mathsf{j}'}$. Since j^* is not a $(\frac{2}{3} - z)$-best-response against $\mathbf{x_B}$, and since j' is a best response against $\mathbf{x_B}$ we obtain:

$$\begin{aligned}(\mathbf{x_B}^T \cdot C)_{\mathsf{j}'} &> (\mathbf{x_B}^T \cdot C)_{\mathsf{j}^*} + \frac{2}{3} - z \\ (\mathbf{x_B}^T \cdot C)_{\mathsf{j}'} &> \frac{1}{3} \cdot \frac{1 - 21z + 9z^2}{1+3z} + \frac{2}{3} - z \\ &= \frac{1-6z}{1+3z}.\end{aligned}$$

□

The only remaining inequality that we require is a lower bound on the expected payoff to the row player in the profile $(\mathbf{x_S}, \mathsf{j}')$. However, before we can do this, we must first prove an upper bound on the expected payoff to the row player in $(\mathbf{x_B}, \mathsf{j}')$, which we do in the following lemma. Here we first prove that most of the probability mass of $\mathbf{x_B}$ is placed on rows i in which $C_{i\mathsf{j}'} > \frac{1}{3} + z$, which when combined with the fact that there is no $i \in \text{supp}(\mathbf{x}^*)$ such that (i, j') is a pure $(\frac{2}{3} - z)$-WSNE, is sufficient to provide an upper bound.

Lemma 14 *We have* $(\mathbf{x_B}^T \cdot R)_{\mathsf{j}'} < \frac{1}{3} \cdot \frac{1+33z+9z^2}{1+3z}$.

Proof We begin by proving an upper bound on the amount of probability mass assigned by $\mathbf{x_B}$ to rows i with $C_{i\mathsf{j}'} < \frac{1}{3} + z$. Let $T = 1 - C_{i\mathsf{j}'}$ be a random variable where the row i is sampled according to $\mathbf{x_B}$. Lemma 13 implies that:

$$E[T] < 1 - \frac{1-6z}{1+3z} = \frac{9z}{1+3z}.$$

Observe that $\Pr(T \geq 1 - (\frac{1}{3} + z)) = \Pr(T \geq \frac{2}{3} - z)$ is equal to the amount of probability that $\mathbf{x_B}$ assigns to rows i with $C_{i\mathsf{j}'} < \frac{1}{3} + z$. Applying Markov's inequality then establishes our bound.

$$\Pr(T \geq \frac{2}{3} - z) \leq \frac{\frac{9z}{1+3z}}{\frac{2}{3} - z}.$$

So, if $p = \frac{9z}{(1+3z)(2/3-z)}$ then we know that at least $1 - p$ probability is assigned by $\mathbf{x_B}$ to rows i such that $C_{i\mathsf{j}'} \geq \frac{1}{3} + z$. Since we have assumed that there is no $i \in \mathrm{supp}(\mathbf{x}^*)$ such that (i, j') is a pure $(\frac{2}{3} - z)$-WSNE, we know that any such row i must satisfy $R_{i\mathsf{j}'} < \frac{1}{3} + z$. Hence, we obtain the following bound:

$$\begin{aligned}(\mathbf{x_B}^T \cdot R)_{\mathsf{j}'} &< (1-p) \cdot (\frac{1}{3} + z) + p \\ &= \frac{1}{3} \cdot \frac{1 + 33z + 9z^2}{1 + 3z}.\end{aligned}$$

□

Finally, we show that the expected payoff to the row player in the profile $(\mathbf{x_S}, \mathsf{j}')$ is close to 1. Here we use the fact that $\mathbf{x}^*$ is a min-max strategy along with the bound from Lemma 14 to prove our lower bound.

Lemma 15 *We have* $(\mathbf{x_S}^T \cdot R)_{\mathsf{j}'} > \frac{1-15z}{1+3z}$.

Proof Since $\mathbf{x}^*$ is a min-max strategy that secures a value strictly larger than $\frac{2}{3} - z$, we have:

$$\frac{2}{3} - z < \Pr(B) \cdot (\mathbf{x_B}^T \cdot R)_{\mathsf{j}'} + \Pr(S) \cdot (\mathbf{x_S}^T \cdot R)_{\mathsf{j}'}.$$

Substituting the bounds from Lemmas 9, 10, and 14 then gives:

$$\frac{2}{3} - z < \frac{1+3z}{2-3z} \cdot \frac{1}{3} \cdot \frac{1 + 33z + 9z^2}{1+3z} + \frac{1+3z}{2-3z} \cdot (\mathbf{x_S}^T \cdot R)_{\mathsf{j}'}.$$

Solving for $(\mathbf{x_S}^T \cdot R)_{\mathsf{j}'}$ then yields the desired result. □

Finding rows b and u. So far, we have shown that the expected payoff to the row player in $(\mathbf{x_B}, \mathsf{j}^*)$ is close to 1, and that the expected payoff to the column player in $(\mathbf{x_B}, \mathsf{j}')$ is close to 1. We now show that there exists a row $b \in B$ such that $R_{b\mathsf{j}^*}$ is close to 1, and $C_{b\mathsf{j}'}$ is close to 1, and that there exists a row $s \in S$ in which $C_{s\mathsf{j}^*}$ and $R_{s\mathsf{j}'}$ are both close to 1. The following lemma uses Markov's inequality to show a pair of probability bounds that will be critical in showing the existence of b.

Lemma 16 *We have:*

- $\mathbf{x_B}$ *assigns strictly more than* 0.5 *probability to rows* i *with* $R_{i\mathsf{j}^*} > 1 - \frac{18z}{1+3z}$.
- $\mathbf{x_B}$ *assigns strictly more than* 0.5 *probability to rows* i *with* $C_{i\mathsf{j}'} > 1 - \frac{18z}{1+3z}$.

Proof We begin with the first case. Consider the random variable $T = 1 - R_{ij^*}$ where i is sampled from $\mathbf{x_B}$. By Lemma 11, we have that:

$$E[T] < 1 - \frac{1-6z}{1+3z} = \frac{9z}{1+3z}.$$

We have that $T \geq \frac{18z}{1+3z}$ whenever $R_{ij^*} \leq 1 - \frac{18z}{1+3z}$, so we can apply Markov's inequality to obtain:

$$\Pr(T \geq \frac{18z}{1+3z}) < \frac{\frac{9z}{1+3z}}{\frac{18z}{1+3z}} = 0.5.$$

The proof of the second case is identical to the proof given above, but uses the (identical) bound from Lemma 13. □

The next lemma uses the same techniques to prove a pair of probability bounds that will be used to prove the existence of s.

Lemma 17 *We have:*

- $\mathbf{x_S}$ *assigns strictly more than* $\frac{1}{3}$ *probability to rows* i *with* $C_{ij^*} > 1 - \frac{27z}{1+3z}$.
- $\mathbf{x_S}$ *assigns strictly more than* $\frac{2}{3}$ *probability to rows* i *with* $R_{ij'} > 1 - \frac{27z}{1+3z}$.

Proof We begin with the first claim. Consider the random variable $T = 1 - C_{ij^*}$ where i is sampled from $\mathbf{x_S}$. By Lemma 12, we have that:

$$E[T] < 1 - \frac{1-6z}{1+3z} = \frac{9z}{1+3z}.$$

We have that $T \geq \frac{27z}{1+3z}$ whenever $C_{ij^*} \leq 1 - \frac{27z}{1+3z}$, so we can apply Markov's inequality to obtain:

$$\Pr(T \geq \frac{27z}{1+3z}) < \frac{\frac{9z}{1+3z}}{\frac{27z}{1+3z}} = \frac{1}{3}.$$

We now move on to the second claim. Consider the random variable $T = 1 - R_{ij^*}$ where i is sampled from $\mathbf{x_B}$. By Lemma 15, we have that:

$$E[T] < 1 - \frac{1-15z}{1+3z} = \frac{18z}{1+3z}.$$

We have that $T \geq \frac{27z}{1+3z}$ whenever $R_{ij^*} \leq 1 - \frac{27z}{1+3z}$, so we can apply Markov's inequality to obtain:

$$\Pr(T \geq \frac{27z}{1+3z}) < \frac{\frac{18z}{1+3z}}{\frac{27z}{1+3z}} = \frac{2}{3}.$$

□

Finally, we can formally prove the existence of b and s, which completes the proof of correctness for our algorithm.

Proof (of Lemma 7) We begin by proving the first claim. If we sample a row b randomly from $\mathbf{x_B}$, then Lemma 16 implies that probability that $R_{bj^*} \leq 1 - \frac{18z}{1+3z}$ is strictly less than 0.5 and that the probability that $C_{bj'} \leq 1 - \frac{18z}{1+3z}$ is strictly less than 0.5. Hence, by the union bound, the probability that at least one of these events occurs is strictly less than 1. So, there is a positive probability that neither of the events occurs, which implies that there exists at least one row b that satisfies the desired properties.

The second claim is proved using exactly the same technique, but using the bounds from Lemma 17, again observing that the probability that a randomly sampled row from $\mathbf{x_S}$ satisfies the desired properties with positive probability. □

5 Communication complexity

We claim that Algorithm 3 can be adapted for the limited communication setting. We make the following modification to our algorithm. After computing $\mathbf{x}^*, \mathbf{y}^*, \hat{\mathbf{x}}$, and $\hat{\mathbf{y}}$, we then use Lemma 1 to construct and communicate the sampled strategies $\mathbf{x}^*_s, \mathbf{y}^*_s, \hat{\mathbf{x}}_s$, and $\hat{\mathbf{y}}_s$. These strategies are communicated between the two players using $4 \cdot (\log n)^2$ bits of communication, and the players also exchange $v_r = (\mathbf{x}^*_s)^T \cdot R\mathbf{y}^*_s$ and $v_c = \hat{\mathbf{x}}^T_s C\hat{\mathbf{y}}_s$ using $\log n$ rounds of communication. The algorithm then continues as before, except the sampled strategies are used in place of their non-sampled counterparts. Finally, in Steps 2 and 3, we test against the threshold $\frac{2}{3} - z + \epsilon$ instead of $\frac{2}{3} - z$.

Observe that, when sampled strategies are used, all steps of the algorithm can be carried out in at most $(\log n)^2$ communication. In particular, to implement Step 4, the column player can communicate j^* to the row player, and then the row player can communicate R_{ij^*} for all rows $i \in \text{supp}(\mathbf{x}^*_s)$ using $(\log n)^2$ bits of communication, which allows the column player to determine j'. Once j' has been determined, there are only $2 \cdot \log n$ payoffs in each matrix that are relevant to the algorithm (the payoffs in rows $i \in \text{supp}(\mathbf{x}^*_s)$ in columns j^* and j',) and so the two players can communicate all of these payoffs to each other, and then no further communication is necessary.

Now, we must argue that this modified algorithm is correct. Firstly, we argue that if the modified algorithm reaches Step 5, then the rows b and s exist. To do this, we observe that the required preconditions of Lemma 7 are satisfied by $\mathbf{x}^*_s$, j^*, and j'. Condition 2 holds because the modified Step 3 ensures that the column player's best response payoff is at least $\frac{2}{3} - z + \epsilon > \frac{2}{3} - z$, while Condition 4 is ensured by the explicit check in Step 5. For Conditions 1 and 3, we use the fact that $(\mathbf{x}^*, \mathbf{y}^*)$ is an ϵ-Nash equilibrium of the zero-sum game $(R, -R)$. The following lemma shows that any approximate Nash equilibrium of a zero-sum game behaves like an approximate min-max strategy.

Lemma 18 *If $(\mathbf{x}, \mathbf{y})$ is an ϵ-NE of a zero-sum game $(M, -M)$, then for every strategy $\mathbf{y}'$ we have:*

$$\mathbf{x}^T \cdot M \cdot \mathbf{y}' \geq \mathbf{x}^T \cdot M \cdot \mathbf{y} - \epsilon.$$

Proof Let $v = \mathbf{x}^T \cdot M \cdot \mathbf{y}$ be the payoff to the row player under $(\mathbf{x}, \mathbf{y})$. Suppose, for the sake of contradiction, that there exists a column player strategy $\mathbf{y}'$ such that:

$$\mathbf{x}^T \cdot M \cdot \mathbf{y}' < v - \epsilon.$$

Since the game is zero-sum, this implies that the column player's payoff under $(\mathbf{x}, \mathbf{y}')$ is strictly larger than $-v + \epsilon$, which then directly implies that the best response payoff for the column player against $\mathbf{x}$ is strictly larger than $-v + \epsilon$. However, since the column player's expected payoff under $(\mathbf{x}, \mathbf{y})$ is $-v$, this then implies that $(\mathbf{x}, \mathbf{y})$ is not an ϵ-NE, which provides our contradiction. □

Since Step 2 implies that the row player's payoff in $(\mathbf{x}^*, \mathbf{y}^*)$ is at least $\frac{2}{3} - z + \epsilon$, Lemma 18 implies that $\mathbf{x}^*$ secures a payoff of $\frac{2}{3} - z$ no matter what strategy the column player plays, which then implies that Conditions 1 and 3 of Lemma 7 hold.

Finally, we argue that the algorithm finds a $(0.6528 + \epsilon)$. The modified Steps 2 and 3 now return a $(\frac{2}{3} - z + \epsilon)$-WSNE, whereas the approximation guarantees of the other steps are unchanged. Thus, we our original analysis gives the following theorem.

Theorem 19 *For every $\epsilon > 0$, there is a randomized expected-polynomial-time algorithm that uses $O\left(\frac{\log^2 n}{\epsilon^2}\right)$ communication and finds a $(0.6528+\epsilon)$-WSNE.*

6 Query complexity

We now show that Algorithm 3 can be implemented in a payoff-query efficient manner. Let $\epsilon > 0$ be a positive constant. We now outline the changes needed in the algorithm.

- In Step 1 we use the algorithm of Lemma 2 to find $\frac{\epsilon}{2}$-NEs of $(R, -R)$, and $(C, -C)$. We denote the mixed strategies found as $(\mathbf{x}_a^*, \mathbf{y}_a^*)$ and $(\hat{\mathbf{x}}_a, \hat{\mathbf{y}}_a)$, respectively, and we use these strategies in place of their original counterparts throughout the rest of the algorithm. We also compute $\frac{\epsilon}{2}$-approximate payoff vectors for each of these strategies, and use them whenever we need to know the payoff of a particular strategy under one of these strategies. In particular, we set v_r to be the payoff of $\mathbf{x}_a^*$ according to the approximate payoff vector of $\mathbf{y}_a^*$, and we set v_c to be the payoff of $\hat{\mathbf{y}}_a$ according to the approximate payoff vector for $\hat{\mathbf{x}}_a$.
- In Steps 2 and 3 we test against the threshold of $\frac{2}{3} - z + \epsilon$ rather than $\frac{2}{3} - z$.
- In Step 4 we select j^* to be the column that is maximal in the approximate payoff vector against $\mathbf{x}_a^*$. We then spend n payoff queries to query every row in column j^*, which allow us to proceed with the rest of this step as before.
- In Step 5 we use the algorithm of Lemma 2 to find an approximate payoff vector v for the column player against $\mathbf{x_B}$. We then select j' to be a column

that maximizes v, and then spend n payoff queries to query every row in j^*, which allows us to proceed with the rest of this step as before.

Observe that the query complexity of the algorithm is $O(\frac{n \cdot \log n}{\epsilon^2})$, where the dominating term arises due to the use of the algorithm from Lemma 2 to approximate solutions to the zero-sum games.

We now argue that this modified algorithm produces a $(0.6528+\epsilon)$-WSNE. Firstly, we need to reestablish the existence of the rows b and s used in Step 5. To do this, we observe that the preconditions of Lemma 7 hold for $\mathbf{x}_a^*$. We start with Conditions 1 and 3. Note that the payoff for the row player under $(\mathbf{x}_a^*, \mathbf{y}_a^*)$ is at least $v_r - \frac{\epsilon}{2}$ (since v_r was estimated with approximate payoff vectors,) and Step 2 ensures that $v_r > \frac{2}{3} - z + \epsilon$. Hence, we can apply Lemma 18 to argue that $\mathbf{x}_a^*$ secures payoff at least $\frac{2}{3} - z$ against every strategy of the column player, which proves that Conditions 1 and 3 hold. Condition 2 holds because the check in Step 3, ensures that the approximate payoff of j^* against $\mathbf{x}^*$ is at least $\frac{2}{3} - z + \epsilon$, and therefore the actual payoff of of j^* against $\mathbf{x}^*$ is at least $\frac{2}{3} - z + \frac{\epsilon}{2}$. Finally, Condition 4 holds because pure strategy profiles of this form are explicitly checked for in Step 5.

Steps 2 and 3 in the modified algorithm return a $(\frac{2}{3} - z + \epsilon)$-WSNE, while the other steps provided the same approximation guarantee as the original algorithm. So, we can reuse the analysis for the original algorithm to prove the following theorem.

Theorem 20 *There is a randomized algorithm that, with high probability, finds a* $(0.6528 + \epsilon)$*-WSNE using* $O(\frac{n \cdot \log n}{\epsilon^2})$ *payoff queries.*

7 A communication-efficient algorithm for finding a $(0.382 + \epsilon)$-NE

We study the following algorithm.

Algorithm 4

1. Solve the zero-sum games $(R, -R)$ and $(-C, C)$.
   - Let $(\mathbf{x}^*, \mathbf{y}^*)$ be a NE of $(R, -R)$, and let $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ be a NE of $(C, -C)$.
   - Let v_r be the value secured by $\mathbf{x}^*$ in $(R, -R)$, and let v_c be the value secured by $\hat{\mathbf{y}}$ in $(-C, C)$. Without loss of generality assume that $v_c \leq v_r$.
   - If $v_r \leq \frac{3-\sqrt{5}}{2}$, return $(\hat{\mathbf{x}}, \mathbf{y}^*)$.
2. Otherwise:
   - Let j be a best response for the column player against $\mathbf{x}^*$.
   - Let r be a best response for the row player against j.
   - Define the strategy profile $\mathbf{x}' = \frac{1}{2-v_r} \cdot \mathbf{x}^* + \frac{1-v_r}{2-v_r} \cdot r$.
   - Return $(\mathbf{x}', j)$.

We show that this algorithm always produces a $\frac{3-\sqrt{5}}{2}$-NE. We start by considering the strategy profile returned by Step 1. The maximum payoff that

the row player can achieve against $\mathbf{y}^*$ is v_r, so the row player's regret can be at most v_r. Similarly, the maximum payoff that the column layer can achieve against $\hat{\mathbf{x}}$ is $v_c \leq v_r$, so the column player's regret can be at most v_r. Step 1 only returns a strategy profile in the case where $v_r \leq \frac{3-\sqrt{5}}{2}$, so this step always produces a $\frac{3-\sqrt{5}}{2}$-NE.

To analyse the quality of approximate equilibrium found by Step 2, we use the following Lemma.

Lemma 21 *The strategy profile $(\mathbf{x}', j)$ is a $\frac{1-v_r}{2-v_r}$-NE.*

Proof We start by analysing the regret of the row player. By definition, row r is a best response against column j. So, the regret of the row player can be expressed as:

$$\begin{aligned} R_{rj} - (\mathbf{x}' \cdot R)_j &= R_{rj} - \frac{1}{2-v_r} \cdot ((\mathbf{x}^*)^T \cdot R)_j - \frac{1-v_r}{2-v_r} \cdot R_{rj} \\ &\leq \frac{1}{2-v_r} \cdot R_{rj} - \frac{1}{2-v_r} \cdot v_r \\ &\leq \frac{1}{2-v_r} \cdot 1 - \frac{1}{2-v_r} \cdot v_r \\ &= \frac{1-v_r}{2-v_r}, \end{aligned}$$

where in the first inequality we use the fact that $\mathbf{x}^*$ is a min-max strategy that secures payoff at least v_r, and the second inequality uses the fact that $R_{rj} \leq 1$.

We now analyse the regret of the column player. Let c be a best response for the column player against $\mathbf{x}'$. The regret of the column player can be expressed as:

$$\begin{aligned} &((\mathbf{x}')^T \cdot C)_c - ((\mathbf{x}')^T \cdot C)_j \\ &= \frac{1}{2-v_r} \cdot ((\mathbf{x}^*)^T \cdot C)_c + \frac{1-v_r}{2-v_r} \cdot C_{rc} - \frac{1}{2-v_r} \cdot ((\mathbf{x}^*)^T \cdot C)_{x^* j} - \frac{1-v_r}{2-v_r} \cdot C_{rj} \\ &\leq \frac{1-v_r}{2-v_r} \cdot C_{rc} - \frac{1-v_r}{2-v_r} \cdot C_{rj} \\ &\leq \frac{1-v_r}{2-v_r}. \end{aligned}$$

The first inequality holds since j is a best response against x^* , and therefore $((\mathbf{x}^*)^T \cdot C)_c \leq (\mathbf{x}^*)^T \cdot C)_j$, and the second inequality holds since $C_{rc} \leq 1$ and $C_{rj} \geq 0$. Thus, we have shown that both players have regret at most $\frac{1-v_r}{2-v_r}$ under $(\mathbf{x}', j)$, and therefore $(\mathbf{x}', j)$ is a $\frac{1-v_r}{2-v_r}$-NE. □

Step 2 is only triggered in the case where $v_r > \frac{3-\sqrt{5}}{2}$, and we have that $\frac{1-v_r}{2-v_r} = \frac{3-\sqrt{5}}{2}$ when $v_r = \frac{3-\sqrt{5}}{2}$. Since $\frac{1-v_r}{2-v_r}$ decreases as v_r increases, we therefore have that Step 2 always produces a $\frac{3-\sqrt{5}}{2}$-NE. This completes the proof of correctness for the algorithm.

Communication complexity. We now argue that, for every $\epsilon > 0$ the algorithm can be used to find a $\left(\frac{3-\sqrt{5}}{2}+\epsilon\right)$-NE using $O\left(\frac{\log^2 n}{\epsilon^2}\right)$ rounds of communication.

We begin by considering Step 1. Obviously, the zero-sum games can be solved by the two players independently without any communication. Then, the players exchange v_r and v_c using $O(\log n)$ rounds of communication. If both v_r and v_c are smaller than $\frac{3-\sqrt{5}}{2}$, then the algorithm from Lemma 1 is applied to communicate $\hat{\mathbf{x}}_s$ to the row player, and $\mathbf{y}_s^*$ to the column player. Since the payoffs under the sampled strategies are within ϵ of the originals, we have that $(\hat{\mathbf{x}}_s, \mathbf{y}_s^*)$ is a $\left(\frac{3-\sqrt{5}}{2}+\epsilon\right)$-NE.

If the algorithm reaches Step 2, then the row player uses the algorithm of Lemma 1 to communicate $\mathbf{x}_s^*$ to the column player. The column player then computes a best response j_s against $\mathbf{x}_s^*$, and uses $\log n$ communication rounds to transmit it to the row player. The row player then computes a best response r_s against j_s, then computes: $\mathbf{x}'_s = \frac{1}{2-v_r} \cdot \mathbf{x}_s^* + \frac{1-v_r}{2-v_r} \cdot r$, and the players output $(\mathbf{x}'_s, j_s)$. To see that this produces a $\left(\frac{3-\sqrt{5}}{2}+\epsilon\right)$-NE, observe that $\mathbf{x}_s^*$ secures a payoff of at least $v_r - \epsilon$ for the row player, and repeating the proof of Lemma 21 with this weaker inequality gives that this strategy profile is a $\left(\frac{1-v_r}{2-v_r}+\epsilon\right)$-NE.

Therefore, we have shown the following theorem.

Theorem 22 *For every $\epsilon > 0$, there is a randomized expected-polynomial-time algorithm that uses $O\left(\frac{\log^2 n}{\epsilon^2}\right)$ communication and finds a $\left(\frac{3-\sqrt{5}}{2}+\epsilon\right)$-NE.*

8 Lower bounds

Lower bound against Algorithm 3. We start by showing a tight lower bound against the base algorithm (Algorithm 1). Consider the following game, which we will denote as (R, C).

I \ II	l	r
t	1, 0	0.9, 1
b	0, $\frac{2}{3}$	$\frac{2}{3}$, 0.9

In the game $(R, -R)$, the unique Nash equilibrium is (b, l), which can be found by iterated elimination of dominated strategies. Similarly, in the game $(-C, C)$, the unique Nash equilibrium is (b, r), which can again be found by elimination of dominated strategies. Note, however, that the game itself does not contain any dominated strategies. Hence, we have $v_R = v_C = \frac{2}{3}$, so Step 2

is triggered, and the resulting strategy profile is (b, l). Under this strategy profile, the column player receives payoff 0, while the best response payoff to the column player is $\frac{2}{3}$, so this is a $\frac{2}{3}$-WSNE and no better.

This lower bound can be modified to work against Algorithm 3 changing both $\frac{2}{3}$ payoffs to 0.6528. Then, by the same reasoning given above, Step 2 is triggered, and the algorithm returns a 0.6528-WSNE.

Lower bounds against a better implementation. The lower bound given above exploits the fact that, as specified, our algorithm will return a 0.6528-WSNE as soon as it finds one. In fact, given the game above, the algorithm will return in Step 2, and the vast majority of the algorithm will never run.

A more thoughtful implementation of the algorithm would be to run all of the steps, and then return the best WSNE found during this process. In particular:

- Step 2 should check the quality of WSNE provided by $(\hat{\mathbf{x}}, \mathbf{y}^*)$.
- Step 3 should check the quality of WSNE provided by $(\mathbf{x}^*, \mathbf{y}^*)$ and $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$.
- If $\Pr(B) > 0$, then Step 4 should check the quality of the WSNE provided by $(\mathbf{x_B}, \mathrm{j}^*)$.
- If $\Pr(B) > 0$, then Step 5 should:
  - find the best pure WSNE that can be found in the support of $\mathbf{x_B}$ and the column j^*.
  - determine if there are two rows b and s that satisfy the payoff constraints listed in Step 5, and if so, find the quality of the WSNE provided by the specified strategy profile.

The algorithm should then return the best WSNE found by one of these steps. Note that Steps 4 and 5 cannot be applied to all games, since their precondition is only guaranteed to hold in games where the previous steps failed to find a good WSNE.

A recent paper of Fearnley, Igwe, and Savani successfully applied genetic algorithms to find lower bounds against algorithms that compute approximate Nash equilibria [12]. Using the same approach, along with some hand tweaking of the output, we found the following game.

$$R = \begin{bmatrix} 0.35056 & 0.99 & 1 \\ 0 & 0 & 0 \\ 1 & 0.25 & 0.3 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0.3 \\ 0 & 0 & 0 \\ 0.3 & 0 & 1 \end{bmatrix}$$

In this game:

- The row player can secure payoff 0.64944 in $(R, -R)$, and we have that $\mathbf{x}^* = (0.53979, 0, 0.46021)^T$ and $\mathbf{y}^* = (0.53259, 0.46741, 0)$.
- The column player can secure payoff 0 in $(-C, C)$. Note that the column player can actually play any strategy to secure this payoff, but the row player must play the middle row. We will focus on the case where $\hat{\mathbf{y}}$ plays the middle column, and $\hat{\mathbf{x}}$ plays the middle row.

It can be verified that under these strategy profiles, all steps of the algorithm produce no better than a 0.64944-WSNE. We remark that this lower bound is within 0.0034 of the theoretical upper bound.

9 Conclusion

In this paper, we have developed a new technique for computing approximate Nash equilibria, and approximate well-supported Nash equilibria. This new technique has allowed us to improve upon the best known results in multiple settings. For well-supported Nash equilibria, we have presented a polynomial-time algorithm for finding a 0.6528-WSNE, and we have shown how to implement it in a communication efficient manner, and a query efficient manner, improving upon the best known results in those settings. For approximate Nash equilibria, our techniques obtain a 0.382-NE, and again we showed how this can be carried out in a communication efficient manner, improving the best known results in that setting.

References

1. Y. Babichenko and A. Rubinstein. Communication complexity of approximate Nash equilibria. *CoRR*, abs/1608.06580, 2016. URL `http://arxiv.org/abs/1608.06580`.
2. H. Bosse, J. Byrka, and E. Markakis. New algorithms for approximate Nash equilibria in bimatrix games. *Theoretical Computer Science*, 411(1):164–173, 2010.
3. 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.
4. V. Conitzer and T. Sandholm. Communication complexity as a lower bound for learning in games. In *Proc. of ICML*, pages 185–192, 2004.
5. A. Czumaj, M. Fasoulakis, and M. Jurdzinski. Approximate well-supported Nash equilibria in symmetric bimatrix games. In *Proc. of SAGT*, pages 244–254, 2014.
6. C. Daskalakis, A. Mehta, and C. H. Papadimitriou. Progress in approximate Nash equilibria. In *Proc. of EC*, pages 355–358, 2007.
7. C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.
8. C. Daskalakis, A. Mehta, and C. H. Papadimitriou. A note on approximate Nash equilibria. *Theoretical Computer Science*, 410(17):1581–1588, 2009.
9. J. Fearnley and R. Savani. Finding approximate Nash equilibria of bimatrix games via payoff queries. In *Proc. of EC*, pages 657–674, 2014.
10. J. Fearnley, P. W. Goldberg, R. Savani, and T. B. Sørensen. Approximate well-supported Nash equilibria below two-thirds. In *Proc. of SAGT*, pages 108–119, 2012. To appear in *Algorithmica*.
11. J. Fearnley, M. Gairing, P. W. Goldberg, and R. Savani. Learning equilibria of games via payoff queries. In *Proc. of EC*, pages 397–414, 2013.
12. J. Fearnley, T. P. Igwe, and R. Savani. An empirical study of finding approximate equilibria in bimatrix games. In *Proc. of SEA*, pages 339–351, 2015.
13. P. W. Goldberg and A. Pastink. On the communication complexity of approximate Nash equilibria. *Games and Economic Behavior*, 85:19–31, 2014.
14. P. W. Goldberg and A. Roth. Bounds for the query complexity of approximate equilibria. In *Proc. of EC*, pages 639–656, 2014.
15. S. Hart and Y. Mansour. How long to equilibrium? the communication complexity of uncoupled equilibrium procedures. *Games and Economic Behavior*, 69(1):107–126, 2010.

16. S. C. Kontogiannis and P. G. Spirakis. Well supported approximate equilibria in bimatrix games. *Algorithmica*, 57(4):653–667, 2010.
17. R. J. Lipton, E. Markakis, and A. Mehta. Playing large games using simple strategies. In *Proc. of EC*, pages 36–41, 2003.
18. J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.
19. A. Rubinstein. Settling the complexity of computing approximate two-player Nash equilibria. In *Proc. of FOCS*, pages 258–265, 2016.
20. H. Tsaknakis and P. G. Spirakis. An optimization approach for approximate Nash equilibria. *Internet Mathematics*, 5(4):365–382, 2008.