On the Complexity of Succinct Zero-Sum Games

Lance Fortnow[*]
Department of Computer Science
University of Chicago
Chicago, IL
USA
fortnow@cs.uchicago.edu

Russell Impagliazzo[†]
Department of Computer Science
University of California - San Diego
La Jolla, CA
USA
russell@cs.ucsd.edu

Valentine Kabanets[‡]
School of Computing Science
Simon Fraser University
Vancouver, BC
Canada
kabanets@cs.sfu.ca

Christopher Umans [§]
Department of Computer Science
California Institute of Technology
Pasadena, CA
USA
umans@cs.caltech.edu

Abstract

We study the complexity of solving succinct zero-sum games, i.e., the games whose payoff matrix M is given implicitly by a Boolean circuit C such that $M(i,j) = C(i,j)$. We complement the known EXP*-hardness of computing the* exact *value of a succinct zero-sum game by several results on* approximating *the value. (1) We prove that approximating the value of a succinct zero-sum game to within an* additive *factor is* complete *for the class* promise-S_2^p*, the "promise" version of S_2^p. To the best of our knowledge, it is the first natural problem shown complete for this class. (2) We describe a* $\mathsf{ZPP}^{\mathsf{NP}}$ *algorithm for constructing approximately optimal strategies, and hence for approximating the value, of a given succinct zero-sum game. As a corollary, we obtain, in a uniform fashion, several complexity-theoretic results, e.g., a* $\mathsf{ZPP}^{\mathsf{NP}}$ *algorithm for learning circuits for SAT [7] and a recent result by Cai [9] that $S_2^p \subseteq$* $\mathsf{ZPP}^{\mathsf{NP}}$*. (3) We observe that approximating the value of a succinct zero-sum game to within a* multiplicative *factor is in* PSPACE*, and that it* cannot *be in* promise-S_2^p *unless the polynomial-time hierarchy collapses. Thus, under a reasonable complexity-theoretic assumption, multiplicative-factor approximation of succinct zero-sum games is strictly harder than additive-factor approximation.*

[*]This research was done while the author was at NEC Laboratories America.

[†]Research supported by NSF Award CCR-0098197 and USA-Israel BSF Grant 97-00188

[‡]Most of this research was done while the author was at the University of California, San Diego, supported by an NSERC postdoctoral fellowship.

[§]Research supported in part by NSF grant CCF-0346991.

1. Introduction

1.1. Zero-Sum Games

A two-person zero-sum game is specified by a matrix M. The row player chooses a row i, and, simultaneously, the column player chooses a column j. The row player then pays the amount $M(i,j)$ to the column player. The goal of the row player is to minimize its loss, while the goal of the column player is to maximize its gain.

Given probability distributions (mixed strategies) P and Q over the rows and the columns of M, respectively, the expected payoff is defined as $M(P,Q) = \sum_{i,j} P(i)M(i,j)Q(j)$. The fundamental Minmax Theorem of von Neumann [22] states that even if the two players were to play sequentially, the player who moves last would not have any advantage over the player who moves first, i.e.,

$$\min_P \max_Q M(P,Q) = \max_Q \min_P M(P,Q) = v,$$

where v is called the *value* of the game M. This means that there are strategies P^* and Q^* such that $\max_Q M(P^*,Q) \leqslant v$ and $\min_P M(P,Q^*) \geqslant v$. Such strategies P^* and Q^* are called *optimal* strategies. It is well-known that optimal strategies, and hence the value of the game, can be found in polynomial time by linear programming (see, e.g., [24]); moreover, finding optimal strategies is equivalent to solving linear programs, and hence is P-hard.

Sometimes it may be sufficient to approximate the value v of the given zero-sum game M to within a small additive factor ϵ, and to find approximately optimal strategies $\tilde{P}$ and $\tilde{Q}$ such that $\max_Q M(\tilde{P}, Q) \leqslant v+\epsilon$ and $\min_P M(P, \tilde{Q}) \geqslant v-\epsilon$. Unlike the case of exactly optimal strategies, finding approximately optimal strategies can be done efficiently in parallel [15, 16, 21, 26], as well as sequentially in sublinear time by a randomized algorithm [16].

Zero-sum games also play an important role in computational complexity and computational learning. In complexity theory, Yao [30, 32] shows how to apply zero-sum games to proving lower bounds on the running time of randomized algorithms; Goldmann, Håstad, and Razborov [14] prove a result about the power of circuits with weighted threshold gates; Lipton and Young [20] use Yao's ideas to show that estimating (to within a linear factor) the Boolean circuit complexity of a given NP language is in the second level of the polynomial-time hierarchy Σ_2^p; Impagliazzo [18] gets an alternative proof of Yao's XOR Lemma [31]. In learning theory, Freund and Schapire [12, 13] show how an algorithm for playing a repeated zero-sum game can be used for both on-line prediction and boosting.

1.2. Succinct Zero-Sum Games

A *succinct* two-person zero-sum game is defined by an implicitly given payoff matrix M. That is, one is given a Boolean circuit C such that the value $M(i, j)$ can be obtained by evaluating the circuit C on the input i, j. Note that the circuit C can be much smaller than the matrix M (e.g., polylogarithmic in the size of M).

Computing the exact value of a succinct zero-sum game is EXP-complete, as shown, e.g., in [11, Theorem 4.6]. For the sake of completeness, we give an alternative proof in Appendix A.

The language decision problems for several complexity classes can be efficiently reduced to the task of computing (or approximating) the value of an appropriate succinct zero-sum game. For example, consider a language $L \in \mathsf{MA}$ [2, 4] with polynomial-time computable predicate $R(x, y, z)$ such that if x is in L then there exists a y such that $\mathbf{Pr}_z[R(x, y, z) = 1] > 2/3$ and if x is not in L then for all y, $\mathbf{Pr}_z[R(x, y, z) = 1] < 1/3$, where $|y| = |z| \in \mathsf{poly}(|x|)$. For every x, we define the payoff matrix $M_x(w; y, z) = R(x, y, z \oplus w)$ whose rows are labeled by w's and whose columns are labeled by the pairs (y, z)'s, where $|y| = |z| = |w|$ and $z \oplus w$ denotes the bitwise XOR of the binary strings z and w. It is easy to see that the value of the game M_x is greater than $2/3$ if $x \in L$, and is less than $1/3$ if $x \notin L$.

Recall that the class S_2^p [10, 28] consists of those languages L that have polynomial-time predicates $R(x, y, z)$ such that if x is in L then $\exists y \forall z\ R(x, y, z) = 1$ and if x is not in L then $\exists z \forall y\ R(x, y, z) = 0$. For every x, define the payoff matrix $M_x(z, y) = R(x, y, z)$. Now, if $x \in L$, then there is a column of all 1's, and hence the value of the game M_x is 1; if $x \notin L$, then there is a row of all 0's, and hence the value of the game is 0.

1.3. Our Results

We have three main results about the complexity of computing the value of a given succinct zero-sum game.

(1) We prove that approximating the value of a succinct zero-sum game to within an additive factor is *complete* for the class promise-S_2^p, the "promise" version of S_2^p. To the best of our knowledge, it is the first natural problem shown complete for this class; the existence of a natural complete problem should make the class S_2^p more interesting to study.

(2) We describe a $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm for constructing approximately optimal strategies, and hence for approximating the value, of a given succinct zero-sum game. As a corollary, we obtain, in a uniform fashion, several previously known results: $\mathsf{MA} \subseteq S_2^p$ [28], $S_2^p \subseteq \mathsf{ZPP}^{\mathsf{NP}}$ [9], a $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm for learning polynomial-size Boolean circuits for SAT, assuming such circuits exist [7], and a $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm for deciding if a given Boolean circuit computing some Boolean function f is approximately optimal for f (i.e., if there is no significantly smaller Boolean circuit computing f) [7].

(3) We also observe that approximating the value of a succinct zero-sum game to within a *multiplicative* factor is in PSPACE, and that it is not in promise-S_2^p unless the polynomial-time hierarchy collapses to the second level. Thus, under a reasonable complexity-theoretic assumption, multiplicative-factor approximation of succinct zero-sum games is strictly harder than additive-factor approximation.

Remainder of the paper Section 2 contains necessary definitions and some known results needed later in the paper. In Section 3, we prove that approximating the value of a succinct zero-sum game is complete for the class promise-S_2^p, the "promise" version of S_2^p. Section 4 presents a $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm for approximately solving a given succinct zero-sum game, as well as for finding approximately optimal strategies and give several applications of our results by proving some old and new results in a uniform fashion. In Section 5, we consider the problem of approximating the value of a succinct zero-sum game to within a multiplicative factor. Section 6 contains concluding remarks. In the appendices, we give an alternative proof that computing the exact value of a succinct zero-sum game is EXP-hard, and give an example of another promise-S_2^p-complete problem, a version of Succinct Set Cover.

2. Preliminaries

Let M be a given $0-1$ payoff matrix with value v. For $\epsilon > 0$, we say that a row mixed strategy P and a column mixed strategy Q are *ϵ-optimal* if $\max_Q M(P,Q) \leqslant v + \epsilon$ and $\min_P M(P,Q) \geqslant v - \epsilon$. For $k \in \mathbb{N}$, we say that a mixed strategy is *k-uniform* if it chooses uniformly from a multiset of k pure strategies.

The following result by Newman [23], Althöfer [1], and Lipton and Young [20] shows that every zero-sum game has k-uniform ϵ-optimal strategies for small k.

Theorem 1 ([23, 1, 20]). *Let M be a $0-1$ payoff matrix on n rows and m columns. For any $\epsilon > 0$, let $k \geqslant \max\{\ln n, \ln m\}/(2\epsilon^2)$. Then there are k-uniform ϵ-optimal mixed strategies for both the row and the column player of the game M.*

We use standard notation for complexity classes P, NP, ZPP, BPP, PH, EXP, and P/poly [25]. We use $\mathsf{BPP}^{\mathsf{NP}}$ to denote the class of (not necessarily Boolean) functions that can be computed with high probability by a probabilistic polynomial-time Turing machine given access to an NP-oracle. The error-free class, $\mathsf{ZPP}^{\mathsf{NP}}$, denotes the class of (not necessarily Boolean) functions that can be computed by a probabilistic Turing machine with an NP-oracle such that the Turing machine always halts in polynomial time, and either outputs the correct value of the function, or, with small probability, outputs FAIL.

Let $R(x,y)$ be any polynomial-time relation for $|y| \in \mathsf{poly}(|x|)$, let $R_x = \{y \mid R(x,y) = 1\}$ be the set of witnesses associated with x, and let $L_R = \{x \mid R_x \neq \varnothing\}$ be the NP language defined by R. Bellare, Goldreich, and Petrank [5] show that witnesses for $x \in L_R$ can be generated uniformly at random, using an NP-oracle; the following theorem is an improvement on an earlier result by Jerrum, Valiant, and Vazirani [19].

Theorem 2 ([5]). *For R, R_x, and L_R as above, there is a $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm that takes as input $x \in L_R$, and outputs a uniformly distributed element of the set R_x, or outputs* FAIL*; the probability of outputting* FAIL *is bounded by a constant strictly less than 1.*

3. Promise-S_2^p-Completeness

A *promise problem* Π is a collection of pairs $\Pi = \cup_{n>0}(\Pi_n^+, \Pi_n^-)$, where $\Pi_n^+, \Pi_n^- \subseteq \{0,1\}^n$ are disjoint subsets, for every $n > 0$. The strings in $\Pi^+ = \cup_{n>0}\Pi_n^+$ are called *positive* instances of Π, while the strings in $\Pi^- = \cup_{n>0}\Pi_n^-$ are *negative* instances. If $\Pi^+ \cup \Pi^- = \{0,1\}^*$, then Π defines a language.

The "promise" version of the class S_2^p, denoted as promise-S_2^p, consists of those promise problems Π for which there is a polynomial-time computable predicate $R(x,y,z)$, for $|y| = |z| \in \mathsf{poly}(|x|)$, satisfying the following: for every $x \in \Pi^+ \cup \Pi^-$, $x \in \Pi^+ \Rightarrow \exists y \forall z \;\; R(x,y,z) = 1$ and $x \in \Pi^- \Rightarrow \exists z \forall y \;\; R(x,y,z) = 0$.

Let C be an arbitrary Boolean circuit defining some succinct zero-sum game with the payoff matrix M_C, and let $0 \leqslant u \leqslant 1$ and $k \in \mathbb{N}$ be arbitrary. We define the promise problem

Succinct zero-sum Game Value (SGV)
POSITIVE INSTANCES: $(C, u, 1^k)$ if the value of the game M_C is at least $u + 1/k$.
NEGATIVE INSTANCES: $(C, u, 1^k)$ if the value of the game M_C is at most $u - 1/k$.

The main result of this section is the following.

Theorem 3. *The promise problem SGV is* promise-S_2^p-*complete.*

First, we argue that every problem in promise-S_2^p is polynomial-time reducible to the promise problem SGV, i.e., the problem SGV is *hard* for the class promise-S_2^p.

Lemma 4. *The problem SGV is* promise-S_2^p*-hard.*

Proof. Let Π be an arbitrary promise problem in promise-S_2^p. Let $R(x,y,z)$ be the polynomial-time computable predicate such that $\forall x \in \Pi^+ \exists y \forall z \;\; R(x,y,z) = 1$ and $\forall x \in \Pi^- \exists z \forall y \;\; R(x,y,z) = 0$.

For any x, consider the succinct zero-sum game with the payoff matrix $M_x(z,y) \overset{\text{def}}{=} R(x,y,z)$. Note that for every $x \in \Pi^+$, there is a pure strategy y for the column player that achieves the payoff 1. On the other hand, for every $x \in \Pi^-$, there is a pure strategy z for the row player that achieves the payoff 0. That is, the value of the game M_x is 1 for $x \in \Pi^+$, and is 0 for $x \in \Pi^-$. Defining the SGV problem on $(C, u, 1^k)$ by setting $C(z,y) = R(x,y,z)$, $u = 1/2$, and $k = 3$ completes the proof. $\square$

Next, we show that the problem SGV is in the class promise-S_2^p.

Lemma 5. *SGV$\in$* promise-S_2^p.

Proof. Let (C, u, k) be an instance of SGV such that the value of the game M defined by C is either at least $u + 1/k$, or at most $u - 1/k$. Let the payoff matrix M have n rows and m columns, where $n, m \leqslant 2^{|C|}$.

Using Theorem 1, we can choose the parameter $s \in \mathsf{poly}(|C|, k)$ so that the game M has s-uniform $1/(2k)$-optimal strategies for both the row and the column player. Now we define a new payoff matrix $\hat{M}$ whose rows are labeled by $\hat{n} = n^s$ size-s multisets from $\{1, \ldots, n\}$, and whose columns are labeled by $\hat{m} = m^s$ size-s multisets from $\{1, \ldots, m\}$. For $1 \leqslant i \leqslant \hat{n}$ and $1 \leqslant j \leqslant \hat{m}$, let S_i

and T_j denote the ith and the jth multisets from $\{1, \ldots, n\}$ and $\{1, \ldots, m\}$, respectively. We define

$$\hat{M}(i,j) = \begin{cases} 1 & \text{if } \frac{1}{|S_i||T_j|} \sum_{a \in S_i, b \in T_j} M(a,b) > u \\ 0 & \text{otherwise} \end{cases}$$

Consider the case where the value v of the game M is at least $u + 1/k$. Then there is an s-uniform $1/(2k)$-optimal strategy for the column player. Let T_j be the size-s multiset corresponding to this strategy. By the definition of $1/(2k)$-optimality, we have for every $1 \leqslant a \leqslant n$ that

$$\frac{1}{|T_j|} \sum_{b \in T_j} M(a,b) \geqslant v - 1/(2k) \geqslant u + 1/k - 1/(2k) > u.$$

It follows that $\hat{M}(i,j) = 1$ for every $1 \leqslant i \leqslant \hat{n}$. A symmetrical argument shows that, if the value of the game M is at most $u - 1/k$, then there is a row $1 \leqslant i \leqslant \hat{n}$ such that $\hat{M}(i,j) = 0$ for all $1 \leqslant j \leqslant \hat{m}$. Defining the predicate $R((C, u, 1^k), j, i) = \hat{M}(i,j)$ puts the problem SGV in the class promise-S_2^p. □

Now we can prove Theorem 3.

Proof of Theorem 3. The proof follows from Lemmas 4 and 5. □

4. Approximating the Value of a Succinct Zero-Sum Game

Here we will show how to approximate the value and how to learn sparse approximately optimal strategies for a given succinct zero-sum game in $\mathsf{ZPP}^{\mathsf{NP}}$.

4.1. Learning to play repeated games

Our learning algorithm will be based on the "multiplicative-weights" algorithm of Freund and Schapire [13] for learning how to play repeated zero-sum games; a similar algorithm was proposed earlier by Grigoriadis and Khachiyan [16], motivated by the classical deterministic iterative method for solving zero-sum games due to Brown and Robinson [6, 27].

We first describe the repeated game setting of [13]. Say M is a payoff matrix. In each round t of the repeated game, the row player has a mixed strategy P_t over the rows. With full knowledge of P_t, the column player chooses a (pure) strategy Q_t; in principle, Q_t can be arbitrary, but in the adversarial setting of zero-sum games, the column player is likely to choose Q_t to maximize its expected payoff given the current strategy P_t of the row player. After round t, the row player suffers the loss $M(P_t, Q_t)$. The row player observes its loss $M(i, Q_t)$ for each row i, and chooses the mixed strategy P_{t+1} to use in the next round of the game. The goal of the row player is to minimize its total loss $\sum_{t=1}^{T} M(P_t, Q_t)$, where T is the total number of rounds in the repeated play.

Freund and Schapire propose and analyze the following learning algorithm, called MW for "multiplicative weights". The algorithm MW starts with the uniform distribution on the rows. After each round t, the new mixed strategy of the row player is computed by the following rule:

$$P_{t+1}(i) = P_t(i) \frac{\beta^{M(i,Q_t)}}{Z_t}, \tag{1}$$

where $Z_t = \sum_i P_t(i)\beta^{M(i,Q_t)}$ is a normalizing factor, and $\beta \in [0,1)$ is a parameter of the algorithm. In words, the new mixed strategy of the row player re-weighs the rows by reducing the probability of row i proportionately to the loss suffered by i given the current strategy Q_t of the column player: the higher the loss, the lower the new probability.

Theorem 6 ([13]). *For any matrix M with n rows and entries in $[0,1]$, and for any sequence of mixed strategies $Q_1, \ldots, Q_T$ played by the column player, the sequence of mixed strategies $P_1, \ldots, P_T$ produced by the algorithm MW with $\beta = \left(1 + \sqrt{\frac{2 \ln n}{T}}\right)^{-1}$ satisfies the following:*

$$\sum_{t=1}^{T} M(P_t, Q_t) \leqslant \min_P \sum_{t=1}^{T} M(P, Q_t) + 3\sqrt{T \ln n}.$$

In other words, the algorithm MW plays only slightly worse than the algorithm with full knowledge of all the mixed strategies $Q_1, \ldots, Q_T$ to be played by the column player.

Now, suppose that the column player picks its mixed strategies in the most adversarial fashion, i.e., in each round t,

$$Q_t = \arg\max_Q M(P_t, Q).$$

Then the probability distribution $\bar{P} = \frac{1}{T}\sum_{t=1}^{T} P_t$, the average of the mixed strategies produced by the algorithm MW of Theorem 6, will be an approximately optimal strategy for the game M whenever T is sufficiently large.

Theorem 7 ([13]). *Let M be a payoff matrix with n rows whose entries are in $[0,1]$. Let v be the value of the game M. Let the mixed strategies $P_1, \ldots, P_T$ be chosen by the algorithm MW of Theorem 6, while the column strategies $Q_1, \ldots, Q_T$ are chosen so that $Q_t = \arg\max_Q M(P_t, Q)$, for each $1 \leqslant t \leqslant T$. Then the mixed strategies $\bar{P} = \frac{1}{T}\sum_{t=1}^{T} P_t$ and $\bar{Q} = \frac{1}{T}\sum_{t=1}^{T} Q_t$ are ϵ-optimal for $\epsilon = 3\sqrt{\frac{\ln n}{T}}$, i.e., $\max_Q M(\bar{P}, Q) \leqslant v + \epsilon$ and $\min_P M(P, \bar{Q}) \geqslant v - \epsilon$.*

Hence, we have $v - \epsilon \leqslant M(\bar{P}, \bar{Q}) \leqslant v + \epsilon$.

Proof. The following sequence of inequalities proves the theorem:

$$\begin{aligned}
v &= \min_P \max_Q M(P,Q) && \text{by the Minmax Theorem}\\
&\leqslant \max_Q M(\bar{P},Q)\\
&= \max_Q \frac{1}{T}\sum_{t=1}^{T} M(P_t,Q) && \text{by definition of } \bar{P}\\
&\leqslant \frac{1}{T}\sum_{t=1}^{T} \max_Q M(P_t,Q)\\
&= \frac{1}{T}\sum_{t=1}^{T} M(P_t,Q_t) && \text{by definition of } Q_t\\
&\leqslant \min_P \frac{1}{T}\sum_{t=1}^{T} M(P,Q_t)+\epsilon && \text{by Theorem 6}\\
&= \min_P M(P,\bar{Q})+\epsilon && \text{by definition of } \bar{Q}\\
&\leqslant \max_Q \min_P M(P,Q)+\epsilon\\
&= v+\epsilon && \text{by the Minmax Theorem.}
\end{aligned}$$

□

Thus, we can use the algorithm MW to approximate the value of a given zero-sum game to within an additive factor δ, by setting $T \in O(\ln n/\delta^2)$.

4.2. Computing approximately optimal strategies in ZPP$^{\mathsf{NP}}$

Now we will show how to adapt the algorithm MW of [13] to obtain a ZPP$^{\mathsf{NP}}$ algorithm for computing sparse, approximately optimal strategies of succinct zero-sum games. Let M be a payoff matrix of n rows and m columns implicitly given by some Boolean circuit C so that $C(i,j) = M(i,j)$ for all $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$. Note that $n, m \leqslant 2^{|C|}$. We need to construct an algorithm that runs in time polynomial in $|C|$.

Obviously, we do not have enough time to write down the mixed strategies of the row player as they are computed by the algorithm MW by rule (1). Fortunately, each such strategy P_{t+1} has a succinct description: it only depends on the t pure strategies $Q_1, \dots, Q_t$ used by the column player in the previous t rounds of the game, and each pure strategy is just an index $1 \leqslant j \leqslant m$ of the column of the matrix M. Thus, P_{t+1} is completely defined by the circuit C plus at most $t \log m$ bits of information. Using Theorem 2, we are able to sample according to the distribution P_{t+1}.

Lemma 8. *Let M be a payoff matrix specified by a Boolean circuit C. There is a* ZPP$^{\mathsf{NP}}$ *algorithm that, given the t column indices $j_1, \dots, j_t$ corresponding to pure strategies $Q_1, \dots, Q_t$, outputs a row index i distributed according to the mixed strategy P_{t+1} as defined by rule* (1) *of the algorithm MW.*

Proof. We assume that the parameter β of the algorithm MW from Theorem 6 is a rational number $\beta = b_1/b_2$, for some integers b_1, b_2 (by taking a sufficiently good rational approximation of β, if necessary). For integers $1 \leqslant i \leqslant n$ and $1 \leqslant r \leqslant b_2^t$, define the relation $R(j_1, \dots, j_t; i, r) = 1$ iff $r \leqslant \beta^{\sum_{k=1}^{t} M(i,j_k)} b_2^t$. Viewing the pair (i,r) a witness of the relation R and applying Theorem 2, we get a pair (i_0, r_0) uniformly distributed among the witnesses of R. Observe that, for every $1 \leqslant i \leqslant n$, the probability of sampling a pair whose first element is i is exactly

$$P_{t+1}(i) = \beta^{\sum_{k=1}^{t} M(i,j_k)}/Z,$$

where $Z = \sum_{i=1}^{n} \beta^{\sum_{k=1}^{t} M(i,j_k)}$ is a normalizing factor. Thus, uniformly sampling a witness of R and outputting the first element i_0 of the sampled pair yields us the required ZPP$^{\mathsf{NP}}$ algorithm sampling according to P_{t+1}. □

In order to compute an approximately optimal strategy $\bar{P}$ using Theorem 7, we would need to select in each round t of the game a best possible column strategy $Q_t = \arg\max_Q M(P_t, Q)$ given the current mixed strategy P_t of the row player. It is not clear if this can be accomplished in BPP$^{\mathsf{NP}}$. However, the proof of Theorem 7 can be easily modified to argue that if each Q_t is chosen so that $M(P_t, Q_t) \geqslant \max_Q M(P_t, Q) - \sigma$, for some $\sigma > 0$, then the resulting mixed strategies $\bar{P}$ and $\bar{Q}$ will be $(\epsilon + \sigma)$-optimal (rather than ϵ-optimal). In other words, choosing in each round t an *almost* best possible column strategy Q_t is sufficient for obtaining approximately optimal strategies $\bar{P}$ and $\bar{Q}$.

We now explain how to choose such almost best possible column strategies Q_t in BPP$^{\mathsf{NP}}$. The reader should not be alarmed by the fact that we are considering a BPP$^{\mathsf{NP}}$ algorithm, rather than a ZPP$^{\mathsf{NP}}$ algorithm. This BPP$^{\mathsf{NP}}$ algorithm will only be used as a subroutine in our final, error-free ZPP$^{\mathsf{NP}}$ algorithm.

Fix round t, $1 \leqslant t \leqslant T$. We assume that we have already chosen strategies $Q_1, \dots, Q_{t-1}$, and hence the mixed strategy P_t is completely specified; the base case is for $t = 1$, where P_1 is simply the uniform distribution over the rows of the matrix M.

Lemma 9. *There is a* BPP$^{\mathsf{NP}}$ *algorithm that, given column indices $j_1, \dots, j_{t-1}$ of the matrix M for $t \geqslant 1$ and $\sigma > 0$, outputs a column index j_t such that, with high probability over the random choices of the algorithm, $M(P_t, j_t) \geqslant \max_j M(P_t, j) - \sigma$. The running time of the algorithm is polynomial in $t, 1/\sigma, |C|$, where C is the Boolean circuit that defines the matrix M.*

Proof. Let $\sigma' = \sigma/2$. For integer k to be specified later, form the multiset S by sampling k times independently at random according to the distribution P_t; this can be achieved in $\mathsf{ZPP}^{\mathsf{NP}}$ by Lemma 8. For any fixed column $1 \leqslant j \leqslant m$, the probability that $|\frac{1}{|S|}\sum_{i\in S} M(i,j) - M(P_t,j)| > \sigma'$ is at most $2e^{-2k\sigma'^2}$ by the Hoeffding bounds [17]. Thus, with probability at least $1-2me^{-2k\sigma'^2}$, we have that $|\frac{1}{|S|}\sum_{i\in S} M(i,j) - M(P_t,j)| \leqslant \sigma'$ for every column $1 \leqslant j \leqslant m$. Let us call such a multiset S *good*. Choosing $k \in \mathsf{poly}(\log m, 1/\sigma)$, we can make the probability of constructing a good multiset S sufficiently high.

Assuming that we have constructed a good multiset S, we can now pick $j = \arg\max_j \sum_{i\in S} M(i,j)$ in P^{NP} as follows. First, we compute $w^* = \max_j \sum_{i\in S} M(i,j)$ by going through all possible integers $w = 0,\dots,k$, asking the NP-query: Is there a column $1 \leqslant j \leqslant m$ such that $\sum_{i\in S} M(i,j) > w$? The required w^* will be the last value of w for which our query is answered positively. (To speed up things a little, we could also do the binary search over the interval of integers between 0 and k.) Once we have computed w^*, we can do the binary search over the column indices $1 \leqslant j \leqslant m$, asking the NP-query: Is there a column j in the upper half of the current interval such that $\sum_{i\in S} M(i,j) = w^*$? After at most $\log m$ steps, we will get the required $j^* = \arg\max_j \sum_{i\in S} M(i,j)$. Finally, since S is a good set, we have that $M(P_t,j^*) \geqslant \frac{1}{|S|}\sum_{i\in S} M(i,j^*) - \sigma' \geqslant \max_j M(P_t,j) - 2\sigma' = \max_j M(P_t,j) - \sigma$, as promised. □

Running the $\mathsf{BPP}^{\mathsf{NP}}$ algorithm of Lemma 9 for $T \in O(\ln n/\sigma^2)$ steps, we construct a sequences of pure strategies $Q_1,\dots,Q_T$ such that, with high probability over the random choices of the algorithm, the mixed strategies $\bar{P} = \frac{1}{T}\sum_{t=1}^{T} P_t$ (determined by rule (1)) and $\bar{Q} = \frac{1}{T}\sum_{t=1}^{T} Q_t$ are 2σ-optimal. That is, $\max_Q M(\bar{P},Q) \leqslant v + 2\sigma$ and $\min_P M(P,\bar{Q}) \geqslant v - 2\sigma$, where v is the value of the game M. Hence, we have with high probability that $v - 2\sigma \leqslant M(\bar{P},\bar{Q}) \leqslant v + 2\sigma$.

Since both the mixed strategies $\bar{P}$ and $\bar{Q}$ have small descriptions, they both can be sampled by a $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm. The case of $\bar{Q}$ is trivial since it is a sparse strategy on at most T columns. To sample from $\bar{P}$, we pick $1 \leqslant t \leqslant T$ uniformly at random, sample from P_t using the algorithm of Lemma 8, and output the resulting row index.

Finally, we can prove the main theorem of this section.

Theorem 10. *There is a $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm that, given a $\delta > 0$ and Boolean circuit C defining a payoff matrix M of unknown value v, outputs a number u and multisets S_1 and S_2 of row and column indices, respectively, such that $|S_1| = k_1$ and $|S_2| = k_2$ for $k_1, k_2 \in \mathsf{poly}(|C|, 1/\delta)$,*

$$v - \delta \leqslant u \leqslant v + \delta,$$

and the multisets S_1 and S_2 give rise to sparse approximately optimal strategies, i.e.,

$$\max_j \frac{1}{k_1}\sum_{i\in S_1} M(i,j) \leqslant v + \delta,$$

and

$$\min_i \frac{1}{k_2}\sum_{j\in S_2} M(i,j) \geqslant v - \delta.$$

The running time of the algorithm is polynomial in $|C|$ and $1/\delta$.

Proof. Let us set $\sigma = \delta/12$. As explained in the discussion preceding the theorem, we can construct in $\mathsf{BPP}^{\mathsf{NP}}$ the descriptions of two mixed strategies $\bar{P}$ and $\bar{Q}$ such that $v - 2\sigma \leqslant M(\bar{P},\bar{Q}) \leqslant v + 2\sigma$, where the running time of the $\mathsf{BPP}^{\mathsf{NP}}$ algorithm is $\mathsf{poly}(|C|, 1/\sigma)$. That is, with high probability, both the strategies are approximately optimal to within the additive factor 2σ. Let S_2 be the multiset of column indices given by the sequence of pure strategies $Q_1,\dots,Q_T$ used to define $\bar{Q}$, where $T = k_2 \in \mathsf{poly}(|C|, 1/\sigma)$. To construct S_1, we sample from $\bar{P}$ independently k_1 times. Obviously, both multisets can be constructed in $\mathsf{ZPP}^{\mathsf{NP}}$.

By uniformly sampling from S_1, we can approximate $M(\bar{P},\bar{Q})$ to within an additive factor σ in probabilistic $\mathsf{poly}(|C|, 1/\sigma)$ time with high probability, by the Hoeffding bounds [17]. That is, with high probability, the resulting estimate u will be such that $v - 3\sigma \leqslant u \leqslant v + 3\sigma$, and the sparse mixed strategies given by S_1 and S_2 will be approximately optimal to within the additive factor 3σ.

Finally, we show how to eliminate the error of our probabilistic construction. Given the estimate u and the sparse strategies S_1 and S_2, we can test in P^{NP} whether

$$\max_j \frac{1}{|S_1|}\sum_{i\in S_1} M(i,j) \leqslant u + 6\sigma, \tag{2}$$

and

$$\min_i \frac{1}{|S_2|}\sum_{j\in S_2} M(i,j) \geqslant u - 6\sigma. \tag{3}$$

If both tests (2) and (3) succeed, then we output u, S_1, and S_2; otherwise, we output FAIL.

To analyze correctness, we observe that, with high probability, u, S_1, and S_2 are such that $v - 3\sigma \leqslant u \leqslant v + 3\sigma$,

$$\max_j \frac{1}{|S_1|}\sum_{i\in S_1} M(i,j) \leqslant v + 3\sigma \leqslant u + 6\sigma,$$

and, similarly,

$$\min_i \frac{1}{|S_2|}\sum_{j\in S_2} M(i,j) \geqslant v - 3\sigma \geqslant u - 6\sigma.$$

Hence, with high probability, our tests (2) and (3) will succeed. Whenever they succeed, the output u will approximate the value v of the game M to within the additive factor $6\sigma < \delta$, while the sparse strategies given by S_1 and S_2 will be approximately optimal to within the additive factor $12\sigma = \delta$, as required. $\square$

4.3. Applications

In this section, we show how our Theorems 3 and 10 can be used to derive several old and new results in a very uniform fashion.

Theorem 11 ([28]). $\mathsf{MA} \subseteq S_2^p$

Proof. Let $L \in \mathsf{MA}$ be any language. As we argued in Section 1.2 of the Introduction, for every x there is a succinct zero-sum game M_x defined by a Boolean circuit C_x such that the value of M_x is at least $2/3$ if $x \in L$, and is at most $1/3$ if $x \notin L$.

Let us associate with every x the triple $(C_x, 1/2, 1^8)$ in the format of instances of SGV. By Theorem 3, the resulting problem SGV is in promise-S_2^p, defined by some polynomial-time predicate R. Defining the new predicate $\hat{R}(x, y, z) \overset{\text{def}}{=} R((C_x, 1/2, 1^8), y, z)$ shows that $L \in S_2^p$, as required. $\square$

Theorem 12 ([9]). $S_2^p \subseteq \mathsf{ZPP}^{\mathsf{NP}}$

Proof. Let $L \in S_2^p$ be any language. As we argued in Section 1.2 of the Introduction, the definition of S_2^p implies that for every x there exists a succinct zero-sum game whose value is 1 if $x \in L$, and is 0 if $x \notin L$. Since approximating the value of any succinct zero-sum game to within a $1/4$ is in $\mathsf{ZPP}^{\mathsf{NP}}$ by Theorem 10, the result follows. $\square$

The proofs of the following results utilize the notion of a zero-sum game between algorithms and inputs proposed by Yao [30, 32].

Theorem 13 ([7]). *If SAT is in* $\mathsf{P/poly}$ *then there is a* $\mathsf{ZPP}^{\mathsf{NP}}$ *algorithm for learning polynomial-size circuits for SAT.*

Proof. Let $s'(n) \in \mathsf{poly}(n)$ be the size of a Boolean circuit deciding the satisfiability of any size-n Boolean formula. By the well-known self-reducibility property of SAT, we get the existence of size $s(n) \in \mathsf{poly}(s'(n))$ circuits that, given a Boolean formula ϕ of size n as input, output FALSE if ϕ is unsatisfiable, and TRUE together with a satisfying assignment for ϕ if ϕ is satisfiable.

For any n, consider the succinct zero-sum game given by the payoff matrix M whose rows are labeled by circuits C of size $s(n)$, and whose columns are labeled by the pairs (ϕ, x) where ϕ is a Boolean formula of size n and x is an assignment to the variables of ϕ. We define

$$M(C,(\phi, x)) = \begin{cases} 1 & \text{if } \phi(x) \text{ is True, and } C(\phi) \text{ outputs FALSE} \\ 0 & \text{otherwise.} \end{cases}$$

In words, the matrix M is defined to penalize the row player for using incorrect circuits for SAT.

By our assumption, there is a size-$s(n)$ circuit C that correctly decides SAT. Hence, the row C of the matrix M will consist entirely of 0's, and so the value v of the game M is 0.

Applying Theorem 10 to the succinct zero-sum game M (with $\delta = 1/4$), we obtain a $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm for learning a size-k multiset S of circuits, for $k \in \mathsf{poly}(s(n))$, such that, for every column j of M,

$$\frac{1}{|S|} \sum_{i \in S} M(i, j) \leqslant 1/4.$$

This means that, for every satisfiable Boolean formula ϕ of size n, at least $3/4$ of the circuits in the multiset S will produce a satisfying assignment for ϕ. Therefore, the following polynomial-size Boolean circuit will correctly decide SAT: On input ϕ, output 1 iff at least one of the circuits in S produces a satisfying assignment for ϕ. $\square$

Using similar ideas, we also obtain the following improvements on some results from [20], which are implicit in [7].

Theorem 14 ([7]). *Let C be a Boolean circuit over n-bit inputs, and let s be the size of the smallest possible Boolean circuit equivalent to C. There is a* $\mathsf{ZPP}^{\mathsf{NP}}$ *algorithm that, given a Boolean circuit C, outputs an equivalent circuit of size $O(ns + n \log n)$.*

Proof. For every $1 \leqslant i \leqslant |C|$, consider the succinct zero-sum game with the payoff matrix M_i whose rows are labeled by size-i Boolean circuit A and whose columns are labeled by n-bit strings x. We define $M_i(A, x) = 0$ if $A(x) = C(x)$, and $M_i(A, x) = 1$ if $A(x) \neq C(x)$.

Clearly, the value of the game M_i is 0 for all $i \geqslant s$. Applying the $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm of Theorem 10 to every $i = 1, \ldots, |C|$ in turn, we can find the first $i_0 \leqslant s$ such that the value of the game M_{i_0} is at most 1/4. Similarly to the proof of Theorem 13, we get a small multiset of size-i_0 circuits such that their majority agrees with C on all inputs. It is not difficult to verify that the size of this constructed circuit is at most $O(ni_0 + n \log n)$, as claimed. $\square$

Theorem 14 can also be used to check in $\mathsf{ZPP}^{\mathsf{NP}}$ if a given Boolean circuit is approximately the smallest possible, i.e., if there is no equivalent circuit of significantly smaller size.

5. Multiplicative-Factor Approximation

In the previous sections, we studied the problem of approximating the value of a given succinct zero-sum game to within an *additive* factor. It is natural to consider the problem of *multiplicative*-factor approximation: Given a Boolean circuit C computing the payoff matrix of a zero-sum game of unknown value v, and a parameter ϵ (written in unary), compute $w = (1 \pm \epsilon)v$.

It follows from the work of Luby and Nisan [21] that approximating the value of a given succinct zero-sum game to within a multiplicative factor ϵ (written in unary) is in PSPACE. The result in [21] talks about explicitly given linear programming problems where the input constraint matrix and constraint vector are positive; zero-sum games are a special case of such "positive linear programming" problems. The algorithm in [21] uses a polynomial number of processors and runs in time polynomial in ϵ and polylogarithmic in the size of the input matrix. By scaling it up, one obtains a PSPACE algorithm for implicitly given zero-sum games.

We do not know whether multiplicative-factor approximation of succinct zero-sum games can be done in the polynomial-time hierarchy. This is an interesting open question. However, we can show that, unless the polynomial-time hierarchy collapses to the second level, multiplicative-factor approximation is strictly harder than additive-factor approximation.

Theorem 15. *If the value of every succinct zero-sum game can be approximated to within some multiplicative constant factor $\epsilon < 1$ in Σ_2^p, then $\mathsf{PH} = \Sigma_2^p$.*

The proof of Theorem 15 relies on the following simple lemma.

Lemma 16. *The problem of approximating the value of a succinct zero-sum game to within a multiplicative constant factor $\epsilon < 1$ is Π_2^p-hard.*

Proof. Let $L \in \Pi_2^p$ be an arbitrary language, and let R be a polynomial-time computable ternary relation such that, for all inputs x, $x \in L \Leftrightarrow \forall y \exists z R(x, y, z)$, where $|y|$ and $|z|$ are polynomial in $|x|$. For every input x, consider the following zero-sum game M_x:

$$M_x(y, z) = \begin{cases} 1 & \text{if } R(x, y, z) \text{ is true} \\ 0 & \text{otherwise.} \end{cases}$$

We claim that if $x \in L$, then the value of the game M_x is greater than 0; and if $x \notin L$, then the value of M_x is 0. Indeed, if $x \notin L$, then the row player has a pure strategy y that achieves the payoff 0 for any strategy z of the column player. On the other hand, if $x \in L$, then the uniform distribution over the columns achieves the payoff at least $2^{-|z|} > 0$ for any strategy y of the row player.

It follows that an algorithm for approximating the value of a succinct zero-sum game to within a multiplicative factor $\epsilon < 1$ can be used to decide the language L, which proves the lemma. $\square$

Proof of Theorem 15. By the assumption of the theorem and by Lemma 16, we get that $\Pi_2^p \subseteq \Sigma_2^p$, implying the collapse $\mathsf{PH} = \Sigma_2^p$. $\square$

6. Conclusions

We have shown that the problem of approximating the value of a succinctly given zero-sum game is complete for the "promise" version of the class S_2^p. This appears to be the first natural problem proved to capture the complexity of S_2^p. We would like to point out that this is not the only interesting problem complete for promise-S_2^p. In Appendix B we show that a version of Succinct Set Cover is also promise-S_2^p-complete.

We presented a $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm for learning the approximate value and approximately optimal sparse strategies for the given succinct zero-sum game. Our algorithm allowed us to prove a few results from [7, 9, 20] in a completely uniform fashion, via the connection between zero-sum games and computational complexity discovered by Yao [30, 32]. We conclude by observing that derandomizing our $\mathsf{ZPP}^{\mathsf{NP}}$ algorithm is impossible without proving superpolynomial circuit lower bounds for the class $\mathsf{EXP}^{\mathsf{NP}}$.

Theorem 17. *If there is a P^{NP} algorithm for approximating the value of any given succinct zero-sum game to within an additive factor, then $\mathsf{EXP}^{\mathsf{NP}} \not\subset \mathsf{P/poly}$.*

Proof Sketch. Our proof is by contradiction. If $\mathsf{EXP}^{\mathsf{NP}} \subset \mathsf{P/poly}$, then $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{EXP} = \mathsf{MA} \subseteq S_2^p$ by combining the results of Buhrman and Homer [8], Babai, Fortnow, and Lund [3], and Russell and Sundaram [28]. Since approximating the value of a succinct zero-sum game is complete for promise-S_2^p by Theorem 3, the assumed existence of a P^{NP} algorithm for that problem would imply the collapse $S_2^p = \mathsf{P}^{\mathsf{NP}}$. Hence, we would get $\mathsf{EXP}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}}$, which is impossible by diagonalization. $\square$

Acknowledgements We want to thank Joe Kilian for bringing [11] to our attention, and Christos Papadimitriou for answering our questions on linear programming and zero-sum games. We also thank David Zuckerman for helpful discussions.

References

[1] I. Althöfer. On sparse approximations to randomized strategies and convex combinations. *Linear Algebra and its Applications*, 199, 1994.

[2] L. Babai. Trading group theory for randomness. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, pages 421–429, 1985.

[3] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[4] L. Babai and S. Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[5] M. Bellare, O. Goldreich, and E. Petrank. Uniform generation of NP-witnesses using an NP-oracle. *Information and Computation*, 163:510–526, 2000.

[6] G. Brown. Iterative solution of games by fictitious play. In T. Koopmans, editor, *Activity Analysis of Production and Allocation*, volume 13 of *Cowles Commision Monograph*, pages 129–136. Wiley, New York, 1951.

[7] N. Bshouty, R. Cleve, R. Gavaldà, S. Kannan, and C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52(3):421–433, 1996.

[8] H. Buhrman and S. Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In R. Shyamasundar, editor, *Proceedings of the Twelfth Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 652 of *Lecture Notes in Computer Science*, pages 116–127, Berlin, Germany, 1992. Springer Verlag.

[9] J.-Y. Cai. $S_2^p \subseteq \mathsf{ZPP}^{\mathsf{NP}}$. In *Proceedings of the Forty-Second Annual IEEE Symposium on Foundations of Computer Science*, pages 620–629, 2001.

[10] R. Canetti. On BPP and the polynomial-time hierarchy. *Information Processing Letters*, 57:237–241, 1996.

[11] J. Feigenbaum, D. Koller, and P. Shor. A game-theoretic classification of interactive complexity classes. In *Proceedings of the Tenth Annual IEEE Conference on Computational Complexity*, pages 227–237, 1995.

[12] Y. Freund and R. Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 325–332, 1996.

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

[14] M. Goldmann, J. Håstad, and A. Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

[15] M. Grigoriadis and L. Khachiyan. Approximate solution of matrix games in parallel. In P. Pardalos, editor, *Advances in Optimization and Parallel Computing*, pages 129–136. Elsevier, Amsterdam, 1992.

[16] M. Grigoriadis and L. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2):53–58, 1995.

[17] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Journal*, pages 13–30, 1963.

[18] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of the Thirty-Sixth Annual IEEE Symposium on Foundations of Computer Science*, pages 538–545, 1995.

[19] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structuress from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[20] R. Lipton and N. Young. Simple strategies for large zero-sum games with applications to complexity theory. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, pages 734–740, 1994.

[21] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 448–457, 1993.

[22] J. Neumann. Zur Theorie der Gesellschaftspiel. *Mathematische Annalen*, 100:295–320, 1928.

[23] I. Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39:67–71, 1991.

[24] G. Owen. *Game Theory*. Academic Press, 1982.

[25] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[26] S. Plotkin, D. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301, 1995.

[27] J. Robinson. An iterative method of solving a game. *Annals of Mathematics*, 54:296–301, 1951.

[28] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Computational Complexity*, 7(2):152–162, 1998.

[29] C. Umans. Hardness of approximating Σ_2^p minimization problems. In *Proceedings of the Fortieth Annual IEEE Symposium on Foundations of Computer Science*, pages 465–474, 1999.

[30] A. Yao. Probabilistic complexity: Towards a unified measure of complexity. In *Proceedings of the Eighteenth Annual IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977.

[31] A. Yao. Theory and applications of trapdoor functions. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[32] A. Yao. Lower bounds by probabilistic arguments. In *Proceedings of the Twenty-Fourth Annual IEEE Symposium on Foundations of Computer Science*, pages 420–428, 1983.

A. Computing the Value of a Succinct Zero-Sum Game

In this section, we show that computing the exact value of a given succinct zero-sum game is EXP-hard. To this end, we first show that computing the exact value of an explicit (rather than succinct) zero-sum game is P-hard. The EXP-hardness of the succinct version of the problem will then follow by standard arguments.

Theorem 18. *Given a payoff matrix M of a zero-sum game, it is* P*-hard to compute the value of the game M.*

Proof. The proof is by a reduction from the Monotone Circuit Value Problem. Fix a circuit with m wires (gates). We will construct a payoff matrix as follows.

For every wire w, we have two columns w and w' in the matrix (w' is intended to mean the complement of w). We will create the rows to have the following property:

1. If the circuit is true, there is a probability distribution that can be played by the column player that achieves a guaranteed nonnegative payoff. For each wire w, it will place $1/m$ probability on w if wire w carries a 1 or on w' if wire w carries a 0.
2. If the circuit is false, then for every distribution on the columns, there is a choice of a row that forces a negative payoff for the column player.

Construction of rows (unspecified payoffs are 0):

- For every pair of wires u and v, have a row with u and u' have a payoff of -1 and v and v' have a payoff of 1. This guarantees the column player must put the same probability on each wire.
- For the output wire o, have a row with the payoff of -1 for o'.
- For every input wire i with a value 1 have a row with a payoff of -1 for i'.
- For every input wire i with a value 0 have a row with a payoff of -1 for i.
- If wire w is the OR of wires u and v, have a row with payoffs of -1 for w and 1 for u and 1 for v.
- If wire w is the AND of wires u and v, have a row with payoff of -1 for w and 1 for u and another row with a payoff of -1 for w and 1 for v.

It is not difficult to verify that the constructed zero-sum game has a nonnegative value iff the circuit is true. □

By standard complexity-theoretic arguments, we immediately get from Theorem 18 the following.

Corollary 19 ([11]). *Computing the exact value of a given succinct zero-sum game is* EXP*-hard.*

B. A Version of Succinct Set Cover is Promise-S_2^p-complete

An instance of Succinct Set Cover (SSC) is given by a 0-1 incidence matrix A with rows ("sets") indexed by $\{0,1\}^m$ and columns ("elements") indexed by $\{0,1\}^n$. The matrix is presented as a circuit C where $C(i,j)$ outputs the (i,j) entry of A, which is 1 iff element j is in set i. The goal is to find the smallest set cover, i.e., a set I such that $\forall j \exists i \in I\ C(i,j) = 1$.

We define the promise problem $n/\log n$-SSC whose positive instances are $(C, 1^k)$ such that there is a set cover of size at most k, and whose negative instances are $(C, 1^k)$ such that there is no set cover of size less than $k(n/\log n)$. This is a variant of a problem introduced by Umans [29].

Theorem 20. *$n/\log n$-SSC is* $\mathsf{promise}$-S_2^p*-complete.*

Proof Sketch. First, it is $\mathsf{promise}$-S_2^p-hard. Given any relation $R(x,y,z)$ defining a $\mathsf{promise}$-S_2^p problem and an input x, just take A to be $A(i,j) = R(x,i,j)$. If $R(x,y,z)$ has an all-1 row, then there is a set cover of size 1. If $R(x,y,z)$ has an all-0 column, then there is *no* set cover of any size.

Now we prove that $n/\log n$-SSC is in $\mathsf{promise}$-S_2^p. We are given an instance $(C, 1^k)$. The game matrix M has rows indexed by $\{0,1\}^n$ and columns indexed by k-subsets of $\{0,1\}^m$. Entry $M(i; j_1, j_2, \ldots, j_k)$ is 1 iff there is a $1 \leqslant \ell \leqslant k$ for which $C(j_\ell, i) = 1$.

It is clear that if we start with a positive instance, then there is an all-1 column, and so the value of the game is 1. On the other hand, if we start with a negative instance, then there is no cover of size $k(n/\log n)$, and we claim that there exists a set I of rows with $|I| = \mathsf{poly}(n,m,k)$ such that $\forall J \exists i \in I\ M(i;J) = 0$. Thus, by playing the uniform distribution over the rows in I, the row player achieves the value at most $1 - 1/|I|$.

Now we prove our claim. First, observe that there must exist a i such that $\mathbf{Pr}_J[M(i;J) = 0] > 1/n$. Indeed, suppose otherwise, i.e., that every i is covered by a random column of M with probability greater than $1 - 1/n$. Then the probability that a fixed i is *not* covered by any of $n/\log n$ columns of M, chosen independently and uniformly at random, is less than $1/n^{n/\log n} = 2^{-n}$. Thus, there exists a set of $n/\log n$ columns of M that covers all i's. This means that the original Succinct Set Cover instance has a set cover of size $k(n/\log n)$, contradicting our assumption.

Let us add this i to our set I, and delete all columns J for which $M(i;J) = 0$. Repeating this procedure $\mathsf{poly}(n,k,m)$ times will eliminate all columns of M, yielding the requisite set I. □