On the Existence of Nash Equilibrium in Games with Resource-Bounded Players

Joseph Y. Halpern[1], Rafael Pass[1], and Daniel Reichman[3]

[1] Cornell University, USA {halpern,pass}@cs.cornell.edu
[2] Princeton University, USA daniel.reichman@gmail.com

Abstract. We consider *computational games*, sequences of games $\mathcal{G} = (G_1, G_2, \ldots)$ where, for all n, G_n has the same set of players. Computational games arise in electronic money systems such as Bitcoin, in cryptographic protocols, and in the study of generative adversarial networks in machine learning. Assuming that one-way functions exist, we prove that there is 2-player zero-sum computational game $\mathcal{G}$ such that, for all n, the size of the action space in G_n is polynomial in n and the utility function in G_n is computable in time polynomial in n, and yet there is no ϵ-Nash equilibrium if players are restricted to using strategies computable by polynomial-time Turing machines, where we use a notion of Nash equilibrium that is tailored to computational games. We also show that an ϵ-Nash equilibrium may not exist if players are constrained to perform at most T computational steps in each of the games in the sequence. On the other hand, we show that if players can use arbitrary Turing machines to compute their strategies, then every computational game has an ϵ-Nash equilibrium. These results may shed light on competitive settings where the availability of more running time or faster algorithms can lead to a "computational arms race", precluding the existence of equilibrium. They also point to inherent limitations of concepts such as "best response" and Nash equilibrium in games with resource-bounded players.

Keywords: Nash Equilibrium · Bounded Rationality · Turing Machines.

1 Introduction

One of the most widely used solution concepts in game theory is Nash equilibrium (NE). In a Nash equilibrium, no player can improve his utility by deviating unilaterally from his strategy. A key property of NE is that it exists in every normal-form game, making it a potential candidate for an equilibrium rational players may end up in. However, the proof of existence of NE is silent with respect to the computational resources players may or may not have. But if a Nash equilibrium is hard to compute, it is hard to imagine how computationally bounded players could play it [3]. The importance of taking computational concerns into account in game theory has been recognized since at least the work

[3] The celebrated PPAD-completeness results [3, 5] indicate that finding a NE in a fixed game is intractable. Our setting is very different from the setting that is considered

of Simon [26]. Our goal here is to examine how considering computationally bounded players influences notions such as best response and NE.

We will be mainly interested in players that are polynomially bounded, continuing a long line of work in game theory on resource-bounded players (e.g., [16, 19, 21, 23]). To make sense of polynomial-time players, we need to have a set of inputs that grow as a function of n. But game theorists typically study individual games, which have a fixed size. To deal with this, we consider not single games, but *computational games* [12], which have the form $(G_1, G_2, \ldots)$, where for all n, G_n is a finite game. We assume that each player chooses a Turing machine (TM) that, given n, computes a strategy for the player in G_n. If a player is polynomial-time bounded, then the player's action in the nth game can be computed in time polynomial in n.

Computational games arise in a number of settings of interest. One example is "crypto-currencies" such as Bitcoin. An essential ingredient of Bitcoin [17] is miners who solve challenging cryptographic problems, whose solution is later used in verifying transactions in the system. Bitcoin keeps the average time at which puzzles are solved a constant, despite technological advances, by making the cryptographic problem needed to be solved harder and harder over time, forcing miners to examine a larger number of possible solutions. This can be modeled by viewing Bitcoin as a sequence of games, where in the nth game the miner is required to solve a cryptographic puzzle P_n such that the number of candidate solutions that need to be examined in order to solve P_n is a function of n.

Cryptographic protocols such as *commitment schemes* [2] provide another example of computational games. A commitment scheme consists of two parties; a sender and a receiver. In the first step of this protocol, the sender chooses a bit b and sends an encryption of b to the receiver, committing the sender to b without revealing b to the receiver. Next, the receiver chooses a bit. Finally the sender reveals the bit to the receiver. This protocol can be viewed as a game where the receiver wins if the bit he chooses matches the bit revealed; the sender wins if they do not match. Clearly, if the receiver can break the scheme and deduce the sender's bit, the receiver wins; if the sender can cheat ("reveal" a bit that does not necessarily match what he committed to), the sender wins. The encryption at the first step involves a security parameter k, where larger security parameters provide more security (i.e., more running time is required to break the scheme). This can be modeled as a sequence of games, where in the kth game the sender encrypts the bit using a security parameter k [12]. Many cryptographic protocols, including secret sharing and multiparty computation, can be viewed as computational games in this way.

Yet one more example of computational games arises in the study of GANs *generative adversarial networks* in machine learning; as argued by Oliehook et al. [22], GANs can be viewed as computational games that end up converging to a "local resource-bounded NE".

in these PPAD-hardness results. For more details see the discussion of related work in the end of this section.

The computational games that we consider are actually sequences of *Bayesian* games, where the action of a player may depend on his *type*, which encodes some private information that the player may have. In a computational game, the action spaces and types spaces all have to be finite, and the utility functions and probability distribution over types have to be computable. We focus here on a subclass of computational games that we call *polynomial games*; these are sequences of games where the action space and type space in the nth game have size polynomial in n, and the utility function and probability distribution over types in the nth game can be computed in time polynomial in n. These restrictions all apply to the games that we are interested in, such as Bitcoin.[4]

An analogue of NE can be defined in computational games $(G_1, G_2, \ldots)$ [12]. We assume that every game G_j is a k-player game and that for $1 \leq i \leq k$, player i uses a TM M_i that computes his actions in G_j given j. Roughly speaking, a machine profile $(M_1, \ldots, M_k)$ consisting of TMs is a NE if, for every player, replacing his TM by a different TM gives him at most a negligible improvement to his utility. We can get a notion of *polynomial-time* NE by replacing "TM" with "polynomial-time TM" everywhere in the definition. (There are certain subtleties in this definition; see Definition 5 and the discussion thereafter for more detail.)

In contrast to fixed games, where NE always exists, we show that in computational games, NE may not exist. Specifically, we show (Theorem 1) that, assuming the existence of one-way functions, there are polynomial 2-player zero-sum games for which no polynomial-time Nash equilibrium exists. This is done by simulating the "largest integer game" in this setting, the game where players simultaneously output an integer, and the player who chooses the largest integer wins. Clearly this game has no Nash equilibrium [15]. We can effectively simulate this game by presenting players with multiple one-way function puzzles, requiring players to invert as many puzzles as possible. We can ensure that a player with sufficiently more (but only polynomially more) running time can invert more puzzles. Thus, we get an "arms race" with no equilibrium. This example points to an inherent difficulty in analyzing games with polynomially-bounded players. Namely, in such games, there is often no best response; players can use longer and longer running times to improve their payoffs. Interestingly, a similar phenomenon has been observed in Bitcoin, where miners use increasingly more sophisticated computational devices for the mining operation (see [4] and the reference therein).

We then demonstrate (Theorem 2) that Nash equilibrium may fail to exist even if players are constrained to run for at most T steps for a fixed integer T, without asymptotics kicking in. The idea is to let players first play a game (matching pennies) that requires randomization to achieve equilibrium, and then

[4] The games used to model protocols such as Bitcoin are actually *extensive-form* games, which are played over time. Our impossibility results show that there are computational Bayesian games where there is no NE when we restrict to polynomial-time players. Since Bayesian games are a special case of extensive-form games, our non-existence results carry over to extensive-form games.

effectively give the player with greater remaining running time an additional bonus. Assuming that the generation of a random bit requires computational effort, this game cannot have a Nash equilibrium. Our impossibility results hold even if we replace "Nash equilibrium" by "ϵ-Nash equilibrium". By way of contrast, we show (Theorem 3) that if players are *not* computationally bounded (i.e., can use arbitrary Turing machines), then there is *always* an ϵ-NE in a computational game. The key idea behind Theorem 3 is that an algorithm similar to that of Lipton and Markakais [14] for finding an ϵ-NE in a fixed game can be used by the players to find ϵ-NE in computational games.

It is worthwhile at this point to examine our result in the context of the literature on bounded rationality in game theory. Two high-level approaches to incorporating complexity-theoretic considerations into game theory have been considered:

- Rubinstein [24] did not limit the complexity, but charged for it.
- Neyman [20] limited the players (e.g., to being finite automata).

Halpern and Pass [10] extended Rubinstein's approach to TMs: players choose a TM, and then they are charged for the running time/space used/amount of randomization used by the TM on a given input. The approach of charging for complexity of Turing machines was also considered by Fortnow and Santhanam [9], who discount the payoffs of players by the amount of time they use to compute their response. The effect of charging players for the strategies they use on the convergence of learning dynamics to Nash Equilibrium was considered by Ben-Sasson, Tauman-Kalai, and Kalai [1].

In this work we follow the approach of Neyman [20]: we limit players to using polynomial-time TMs, but don't charge for computation. Thus, unlike Halpern and Pass [10] and Fortnow and Santhamam [9], we limit computation, rather than charging for it. Just as we do, Halpern and Pass [10] prove both the existence and non-existence of NE, depending on assumptions. However, the *reasons* for these results are very much framework-dependent. For example, Halpern and Pass [10] show that NE may not exist if we charge players for randomness and it does exist in their framework if we do not charge for randomness. By way of contrast, our main result concerning the non-existence of NE (Theorem 1) holds even if we do not charge for the time taken to generate a random bit. Fortnow and Samantham's result on the existence of ϵ-NE in their version of computational games [9] depends heavily on their assumption that utilities are discounted; we have no analogue of this assumption, and thus must use quite different techniques in our proof of the existence of ϵ-NE.

Despite all the work on resource-bounded players, to the best of our knowledge, very little work has been done on games where players are limited to using polynomial-time Turing Machines. One exception is the work of Megiddo and Wigderson [16], who consider playing repeated prisoner dilemma (for finitely many rounds) with TMs. Their main interest is whether, in finitely repeated prisoners dilemma, there exist "almost cooperative" equilibria (where "defect" is played $o(n)$ times). They restrict attention to deterministic TM. With this re-

striction it is not difficult to give examples of games (with polynomially-bounded players) for which an ϵ-NE does not exist.

Polynomial games bear some similarities to *succinct games*. In succinct games, there exists a circuit C that calculates the utility $C(x_1, x_2, \ldots, x_k)$ of the players once they choose the actions $x_1, x_2, \ldots, x_k \in \{0,1\}^m$. It is known that, given a 2-player zero-sum succinct game, it is EXP-hard to find a NE [7, 8] (see also [25]). Our results regarding the non-existence of NE in polynomial games are incomparable to these results. We are concerned with *polynomial-time computable* strategies. Considering polynomially-bounded players (as opposed to unbounded players) may drastically change the set of Nash equilibria in succinct games. Indeed, a NE for a computational game $(G_1, G_2, \ldots)$ with polynomially-bounded players may fail to be a Nash equilibrium for G_n for all $n \geq 1$: for an example, see the end of Section 3. Moreover, for any fixed game G, a computational NE for the computational game $(G, G, G, \ldots)$ can always be found in polynomial time. Thus, the PPAD-hardness results of finding a Nash equilibrium in a fixed game [3, 5] cannot be applied in our setting either.

2 Preliminaries

We begin by defining Bayesian games.

Definition 1. *A* k-player normal-form Bayesian game *is described by a tuple* (J, B, T, P, v)*, where*

- J *is a set of* k *players (we identify* J *with* $[k] = \{1, \ldots, k\}$*);*
- $B = \prod_{i=1}^{k} B_i$*, where* B_i *is a finite set for all* $i \in [k]$ *consisting of the available actions of player* i*;*
- $T = \prod_{i=1}^{k} T_i$*, where* T_i *is a finite set called the* type space *of player* i*;*
- P *is a probability distribution over* T*;*
- $v = (v_1, \ldots, v_k)$*, where for all* i*,* v_i *is a function from* $B \times T$ *to the real numbers.*

In our settings, it will often be the case that all types are perfectly correlated: all players have the same type and all players know the type of every other player. Observe that normal-form games can be viewed as a special case of Bayesian games (where the type space is a singleton). Finally, since we are concerned here mainly with Bayesian games, when we write "game" we mean "Bayesian game", unless explicitly stated otherwise.

A *pure strategy* s_i for player i is a map $s_i : T_i \to B_i$; a strategy s_i maps the type $t_i \in T_i$ of player i to an action $s_i(t_i) \in B_i$. We denote by $\Delta(B_i)$ the set of all probability distribution over B_i; let $\Delta = \Pi_{i=1}^{k} \Delta(B_i)$. A mixed-strategy s_i for player i is a function mapping type $t_i \in T_i$ to an element of $\Delta(B_i)$. We denote by $s_i(t_i, b_i)$ the probability assigned by a mixed strategy $s_i(t_i)$ to $b_i \in B_i$. The expected utility of player i with the mixed strategy profile $s = (s_1, \ldots, s_k)$

(where $t = (t_1, \ldots, t_k) \in T$, $b = (b_1, \ldots, b_k) \in B$, and $(s_1(t_1), \ldots, s_k(t_k)) \in \Delta$) is given by

$$V_i(s) = \sum_{t \in T} P(t) \sum_{b \in B} \left(\prod_{j=1}^{k} s_j(t_j, b_j) \right) v_i(t, b). \quad (1)$$

Note that there are two sources of uncertainty in the utility of a player choosing a mixed action: the probability distribution over other players actions and the distribution P over the type space.

Definition 2. *Let $G = (J, B, T, P, v)$ be a k-player Bayesian game and suppose that $\epsilon \geq 0$. A mixed-strategy profile $s = (s_1, \ldots, s_k)$ is an* ϵ-Nash equilibrium *(ϵ-NE for short) if, for all players i and all mixed strategies s_i', we have that*

$$V_i(s) \geq V_i(s_i', s_{-i}) - \epsilon.$$

(As usual, if $s = (s_1, \ldots, s_k)$ then $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_k)$ is the tuple excluding s_i.) When $\epsilon = 0$, we have a Nash equilibrium.

To reason about resource-bounded players in games, we consider a sequence $(G_1, G_2, \ldots)$ of games where, for all n, $G_n = (J, B^n, T^n, P^n, v^n)$ is a k-player game (k is fixed and does not depend on n). We adapt the definition of [12], which in turn is based on earlier definitions by Dodis, Halevy and Rabin [6] and is applied to Bayesian games. For an integer s, recall that $\{0,1\}^{\leq s}$ is the set of all bit strings of length at most s.

Definition 3. *A computational game $\mathcal{G} = (G_1, G_2, \ldots)$ is a sequence of normal-form Bayesian games, where $G_n = ([k], B^n, T^n, P^n, v^n)$, such that*

- *The set of players in G_n, $[k]$, is the same for all n.*[5]
- *For all n and all i, $B_i^n \subseteq \{0,1\}^{\leq m}$ for some finite m (that may depend on n).*
- *For all n and all i, $T_i^n \subseteq \{0,1\}^{\leq r}$ for some finite r (that may depend on n).*
- *For all $i \in [k]$ and n, there is a TM M such that, given $b \in B^n$, $t \in T^n$, and 1^n, computes $v_i^n(b, t)$.*
- *For all $i \in [k]$ and n, there is a TM M' such that given $t \in T^n$ and 1^n, computes $P^n(t)$.*[6]

$\mathcal{G}$ is bounded *if there exist constants $0 < c < C$ such that for all $n, b \in B^n$, and $t \in T^n$ we have that $v_i^n(b, t) \neq 0 \Rightarrow |v_i^n(b, t)| \in [c, C]$.*

When dealing with games with polynomial-time players, we require slightly stronger properties summarized in the definition below. Following the definition of polynomial games for extensive- form games [12], we define polynomial games for a sequence of Bayesian games.

[5] It is also possible to allow k to depend on n, but we focus on the case where k is a constant for concreteness.

[6] We restrict our attention to utilities and probabilities that are rational numbers.

Definition 4. *A computational game* $\mathcal{G} = (G_1, G_2, \ldots)$ *is a* polynomial game *if the following conditions hold:*

- *There exist a polynomial* p *such that, for all* n *and all* i, $B_i^n = \{0,1\}^{\leq p(n)}$.
- *There exist a polynomial* q *such that, for all* n *and all* i, $T_i^n = \{0,1\}^{\leq q(n)}$.
- *For all* $i \in [k]$ *and* n, *there is a TM* M *such that, given* $b = (b_1, \ldots, b_k) \in B^n$, $t \in T^n$, *and* 1^n, *computes* $v_i^n(b,t)$ *and runs in time polynomial in* n.
- *For all* $i \in [k]$ *and* n, *there is a TM* M' *such that given* $t \in T^n$ *and* 1^n, *computes* $P^n(t)$ *in time polynomial in* n.

Throughout, we take the *size* of the action set (or type set) to be the maximal number of bits needed to encode an action (or type). Observe that while we require that size in polynomial games to be polynomial in n for every n, the *cardinality* of the action or type set can be exponential.

A *strategy* for player j in a computational game $\mathcal{G}$ is a TM M_j that, given 1^n and the type $t_j \in T_j^n$, outputs a distribution $M_j(1^n, t_j)$ over actions in B_j^n in the game G_n (so that, given some additional random bits, it outputs an action in B_j^n).[7] $M_j(1^n)$ is the strategy defined by taking $M_j(1^n)(t_j) = M_j(1^n, t_j)$. Observe that there are two sources of randomness in $M_j(1^n)$: the distribution of the type t_j and the randomness of M_j once t_j has been determined. We stress that randomized strategies in our setting are obtained by using probabilistic TMs rather than by mixing over TMs. That is, the randomization is part of the computation, not external to it. The utility of player i in G_n given a machine profile $(M_1 \ldots M_k)$ is $V_i^n(M_1(1^n), \ldots, M_k(1^n))$ (as defined in (1)).

To analyze computational games $\mathcal{G} = (G_1, G_2, \ldots)$, we would like to be able to apply classical game-theoretic notions, such as best response and Nash equilibrium, to sequences of games. However, there are certain difficulties in generalizing these notions to computational games. A first obstacle is that sequences of infinite games may allow resource-bounded players to improve over any strategy by doing additional polynomial-time computations. For example, consider a player who gets a payoff of 1 by breaking an encrypted massage $E(s)$ with $s \in \{0,1\}^n$ and a payoff of 0 if he does not break it, where the player's running time is polynomial in n. Assuming that there is no polynomial-time algorithm (in n) for finding s given $E(s)$, there is no best response in this game, as a player can always make polynomially many additional "guesses" on top of his current action, increasing his expected utility. As pointed out by Dodis, Halevi, and Rabin [6], this observation applies to many problems of interest, such as those arising from cryptographic protocols.

One way around this problem, suggested by Dodis, Haley and Rabin [6] and Halpern, Pass, and Seeman [12], is to ignore *negligible* additive changes in the

[7] One question is how to deal with players who use Turing machines that fail to halt or return an action that does not belong to the action space. We deal with this issue by assigning to each player i a special action a_0^i that we take to be the action played if i's TM does not halt or if i's output is not an action in the action space. Any profile that includes a_0^i gives utility $-\infty$ to all players, thus discouraging players from using TMs that fail to halt or return inappropriate actions.

utility of players, where a sequence $\delta(n)$ is negligible if for every polynomial $p, p(n) = o(\delta(n)^{-1})$. That is, deviations that result in a negligible increase in utility are not considered to be improvements. Ignoring negligible terms suffices to ensure the existence of equilibrium in a number of games of interest for which there would not be an equilibrium otherwise [6].

If we ignore negligible change, then given a machine profile $\overline{M}$, changing the behavior of a TM M in finitely many games will not be a deviation breaking an alleged equilibrium, as altering a sequence $\delta(n)$ on finitely many n's does not change the fact that $\delta(n)$ is negligible. On the other hand, a deviation that improves a given player utility on infinitely many n's by a constant $\delta > 0$ implies that the machine profile is not a NE. Finally, it is worth noting that if the utilities of players are exponentially small (say, on the order of $1/2^n$ in the game G_n), a negligible additive term can have a noticeable effect on the utility of players; on the other hand, if utilities are exponentially large, even a (non-negligible) constant change in utilities would be viewed as negligible. In order to avoid such scaling issues, we deal exclusively with bounded games when considering solution concepts for computational games.[8]

Definition 5. *Let $\mathcal{M}$ be a set of TMs and let $\epsilon \geq 0$ be a constant independent of n. A profile $\overline{M} = (M_1, \ldots, M_k)$ of TMs is an ϵ-$\mathcal{M}$-NE for a bounded computational game $\mathcal{G}$ with respect to $\mathcal{M}$, if (a) for all i, $M_i \in \mathcal{M}$, and (b) there exists a negligible sequence $\delta(n)$ such that, for all $M_i' \in \mathcal{M}$ and all $n > 0$ and all $i \in [k]$ we have that*

$$V_i(M_i(1^n), M_{-i}(1^n)) \geq V_i(M_i'(1^n), M_{-i}(1^n)) - \epsilon - \delta(n). \tag{2}$$

When $\epsilon = 0$, we say that $\overline{M}$ is a $\mathcal{M}$-Nash equilibrium. If $\mathcal{M}$ is the set of all probabilistic polynomial-time TMs, we say $\overline{M}$ is a polynomial *ϵ-NE.*

We can consider polynomial-time players, best response, and equilibrium even if the action space of every player is of super-polynomial size. However, in this case, there are trivial examples showing that a NE may not exist. For example, one can take G_n to be the 2-player zero-sum game where each player outputs an integer of length at most 2^{2^n} (written in binary) and the player outputting the larger integer receives payoff 1, with both players getting 0 in case of equality. Clearly this sequence of games does not have a polynomial equilibrium.

In contrast to previous work [9], we require the utilities of players to be computable. Without this requirement, it is not difficult to give examples of polynomial games that do not have a NE. Indeed, let $x_1, x_2 \ldots$ be an enumeration of $\{0,1\}^*$ and let L be an arbitrary non-recursive language. Furthermore, suppose that for all i, j, $i < j$ implies that $|x_i| \leq |x_j|$ (ensuring that for every n the type x_n can be represented by at most $poly(n)$ bits). Consider the sequence $G = (G_1, G_2, \ldots)$ of two-player games such that the type of each player in G_n is x_n and a player gets a payoff of 1 if it correctly determines whether x_n belongs to L and 0 otherwise. Clearly, G does not have a polynomial-time NE (and the utility function in G is not computable).

[8] Our results also hold in a more general setting where the absolute value of a (nonzero) utility is at most polynomial and at least inversely polynomial in n.

3 Polynomial Games With No Polynomial Equilibrium

As we now show, there is a polynomial game for which there is no polynomial NE, assuming one-way functions exist. We find it convenient to use the definition of one-way function given in [13].

Definition 6. *Given $s : \mathbb{N} \to \mathbb{N}$, $t : \mathbb{N} \to \mathbb{N}$, a* one-way function with security parameter s against a t-bounded inverter *is a family of functions $f_k : \{0,1\}^k \to \{0,1\}^m$, $k = 1, 2, 3, \ldots$, satisfying the following properties:*

- *$m = k^b$ for some positive constant b;*
- *there is a TM M such that, given x with $|x| = k$ computes $f_k(x)$ in time polynomial in k;*
- *for all but finitely many k's and all probabilistic TM M', running in time at most $t(k)$ for a given input $f_k(x)$,*

$$\Pr[f_k(M'(f_k(x))) = f_k(x)] < \frac{1}{s(k)},$$

where the probability Pr *is taken over x sampled uniformly from $\{0,1\}^k$ and the randomness of M'.*

We assume that exponential one-way functions exist. Specifically, we assume that there exists a one-way function that is $2^{k/10}$-secure against a $2^{k/30}$-bounded inverter. The existence of a one-way function with these parameters follows from an assumption made by Wee [27] regarding the existence of exponential non-uniform one-way functions. Given $f_k(x)$, we say an algorithm *inverts* $f_k(x)$ if it finds some z such that $f_k(x) = f_k(z)$.

We can now demonstrate the non-existence of polynomial-time computable equilibrium in a polynomial game.

Theorem 1. *If there exists a one-way function that is $2^{k/10}$-secure against a $2^{k/30}$-inverter, then, for all $\epsilon > 0$, there exists a 2-player zero-sum polynomial game $\mathcal{G}$ that has no polynomial ϵ-NE.*

Proof. Let $\mathcal{G} = (G_1, G_2, \ldots)$ be the following polynomial game, which we call the *one-way function game*. For all n, we define G_n as follows. There are two players, 1 and 2. Fix a one-way function $\{f_k\}_{k \geq 1}$ that is $2^{k/10}$-secure against a $2^{k/30}$-bounded inverter. The type space is the same for each player, and consists of tuples of $l = \lceil \log n \rceil$ bitstrings of the form $(f_{\lceil \log n \rceil}(x_1), \ldots, f_{\lceil \log n \rceil^2}(x_l))$. The distribution on types is generated by choosing $x_i \in \{0,1\}^{i\lceil \log n \rceil}$ uniformly at random, and choosing the x_i's independently. Given his type t_n, player j outputs $y_1^j, \ldots, y_l^j$. A *hit* for player j is an index i such that $f_{i\lceil \log n \rceil}(y_i^j) = f_{i\lceil \log n \rceil}(x_i)$. Let a_j denote how many hits player j gets. The payoff of player j is 1 if $a_j - a_{3-j} > 0$. If $a_j - a_{3-j} = 0$, both players receive a payoff of 0. Observe that the utility function of each player is polynomial-time computable in n. Clearly the length of every action of G_n is polynomial in n and so is the length of the type t_n. Hence the one-way function game is a polynomial game. In the full paper [11], we prove that there cannot be a polynomial-time ϵ-NE for $\mathcal{G}$. ∎

Similar ideas can be applied to show there is a 2-player *extensive-form* polynomial game that has no polynomial ϵ-NE, where we no longer need to use a type space. (See [12] for the definition of extensive-form polynomial game and polynomial ϵ-NE in extensive-form polynomial games; we hope that our discussion suffices to give the reader an intuitive sense.) In the game G_n, instead of the tuple $(f_{\lceil \log n \rceil}(x_1^j), \ldots, f_{l\lceil \log n \rceil}(x_l^j))$ being player j's type, player j chooses $x_1^j, \ldots, x_l^j$ at random and sends this tuple to player $3-j$. Again, player j attempts to invert as many of $f_{\lceil \log n \rceil}(x_1^{3-j}), \ldots, f_{l\lceil \log n \rceil}(x_l^{3-j})$ as it can; their payoffs are just as in the Bayesian game above. A proof similar to that of Theorem 1 shows that this game does not have a polynomial NE.

The one-way function game also shows the effect of restricting strategies to be polynomial-time computable. Clearly, without this restriction, the game has a trivial NE: all players correctly invert every element of their tuple. On the other hand, consider a modification of the game where in G_n, a player's type consists of a single element $f_n(x_n)$, with x_n a bitstring of length n chosen uniformly at random. If both players simultaneously invert or fail to invert $f_n(x_n)$, then both get zero. Otherwise, the player who correctly inverts gets 1 and the other player gets -1. Again, it is easy to see that if we take $\mathcal{M}$ to be the family of all TMs, the only Nash equilibrium is to find y_n, z_n such that $f_n(y_n) = f_n(z_n) = f(x_n)$. But if $\mathcal{M}$ consists of only polynomial-time TMs, then it is a polynomial-time NE for both players to simply output a random string, as neither player can invert f with non-negligible probability, and we ignore negligible additive increase to the utilities of players.

4 Equilibrium With Respect to Concrete Time Bounds

The previous example may lead one to speculate that lack of Nash equilibrium in computational games hinges on asymptotic issues, namely, our ability to consider larger and larger action and type spaces. This raises the question of what happens if we restrict our attention to games where players are constrained to execute at most T computational steps, where $T > 0$ is a fixed integer. It turns out that if the use of randomness is counted as a computational action, then there may not be Nash equilibria, as the following example shows. We assume from now on that $T > 2$.

In our computational game, the family of admissible TMs, which we denote by $\mathcal{M}_\mathcal{T}$, is the set of all probabilistic TMs whose running time is upper-bounded by T. The operation of printing a character takes one computational step, and so does the movement of the cursor to a different location on the tape. The generation of a random bit (or alternatively querying a bit in a designated tape that contains random bits) requires at least one computational step (we allow arbitrary bias of a bit, as it does not affect the proof).

Consider the following 2-player zero-sum normal-form computational game $\mathcal{F}$ between Alice (A) and Bob (B). For every n, F_n is the same game F. The action space of each player is $\{0,1\}^T$. By our choice of $\mathcal{M}_\mathcal{T}$, both players are constrained to perform at most T computational steps. The game proceeds as follows. A

and B use TMs $M_A, M_B \in \mathcal{M}_\mathcal{T}$ respectively, to compute their strategies. M_A outputs a single bit a_1. M_B outputs $b_1 \in \{0,1\}$. Based on a_1 and b_1, a game of *matching pennies* is played. Namely, if $a_1 = b_1$, A gets 1, otherwise B gets 1. In the second phase of the game, the TM of each player prints as many characters as possible without violating the constraint of performing at most T steps. If the final number of characters is the same for both players, then both get a payoff of 0 for the second phase. Otherwise the player with a larger number of printed characters gets an additional bonus of 1, and the player with fewer printed characters incurs a loss of 1.

Theorem 2. *The computational game $\mathcal{F}$ does not have an ϵ-$\mathcal{M}_\mathcal{T}$-NE, for all $\epsilon < 1$.*

Proof. Assume, by way of contradiction, that (M_A, M_B) is a Nash equilibrium for $\mathcal{F}$. Since TMs in $\mathcal{M}_\mathcal{T}$ are constrained to query at most T bits, it follows that the strategy computed by M_A (or M_B) given 1^n, will be the same for all $n > T$. As the outcomes of the games F_m, $m \leq T$, do not effect, by our definition of NE in computational games, whether (M_A, M_B) is an equilibrium, we can assume w.l.o.g that both M_A and M_B compute the same strategy (whether mixed or pure) in all games $F_n, n \geq 1$.

Suppose that one of the players uses randomization. Assume this is player A. Namely, M_A generates a random bit before outputting a_1. Then A can guarantee a payoff for the first phase (the matching pennies game) that is no smaller than his current payoff by choosing a TM M'_A that outputs a *deterministic* best response a_1 against the strategy of B in the matching penny game. Observe that we can assume that a_1 is "hardwired" to M'_A. In particular outputting a_1 can be done in a single computational step. Then A can print strictly more 1's in the second phase of the game by configuring M'_A to print $T-1$ 1's (which can be done in $T-1$ steps). If B prints $T-1$ in the second phase of the game, we have that A can increase its payoff in F_n for all n by switching to M'_A. If, on the other hand, M_B prints less than $T-1$ characters in the second step, an analogous argument shows that B can strictly increase its payoff in F_n for all n, by using a TM that runs in at most T steps. In any event, we get a contradiction to the assumption that (M_A, M_B) is a NE.

Suppose now that A does not use randomization. In this case, it follows immediately by the definition of matching-pennies that either A or B can strictly improve their payoff in the first phase of F_n for all n, by outputting the (deterministic) best response to their opponent and printing $T-1$ characters afterwards. As before, we can assume this response is hardwired to the appropriate TM, such that outputting it consumes one computational step, allowing players to print $T-1$ characters in the second phase of the game.

Finally, it is not difficult to verify that the argument above establishes that $\mathcal{F}$ does not have an ϵ-NE for ϵ-NE for all $\epsilon \in (0,1)$. This concludes the proof. ∎

One might wonder whether the non-existence of NE in computational games follows from the fact that we are dealing with an infinite sequence of games with infinitely many possible TMs (e.g., $|\mathcal{M}| = \infty$). Nash Theorem regarding the

existence of NE requires that the action space of every player is finite; without this requirement a NE may fail to exist. Hence it is natural to ask whether limiting $|\mathcal{M}|$ to be finite (for example, taking $\mathcal{M}$ to be the family of all TMs over a fixed alphabet with at most S states for some bound S) may force the existence of NE in computational games. Theorem 2 illustrates that this is not the case: $\mathcal{F}$ will not have a NE even if we take $\mathcal{M}$ to consist only of TMs whose number of states is upper bounded by a large enough positive number S (S should allow for using the TM that is hardwired to output the appropriate best response in the matching pennies game and print $T-1$ characters in the second phase). The reason why NE does not exist despite the finiteness of $\mathcal{M}$, is that in contrast to ordinary games, where a mixed actions of best responses is a best response, in our setting this is not necessarily true: mixing over actions may consume computational resources, forcing players to choose actions that are suboptimal when using randomized strategies.

5 The Existence of ϵ-NE in Computational Games

Our previous results show that if we restrict players to be computationally bounded, then there are polynomial games with no ϵ-NE. Here we demonstrate that the restriction to computationally bounded players is critical. If we allow players to choose arbitrary TMs (or TMs that are guaranteed to halt on every input), we show that for all $\epsilon > 0$, there is an ϵ-NE in every computational game (and thus, *a fortiori*, in every polynomial game). The reason that we need ϵ-NE rather than NE (although ϵ can be arbitrarily small) is that there are 3-player games in which, in every NE, some actions are chosen with irrational probabilities (even if all utilities are rational and nature's moves are made with rational probabilities) [18]. By considering ϵ-NE, we can avoid representational issues involving irrational numbers.

Let $\epsilon > 0$ be a fixed constant. Suppose that $\mathcal{G} = (G_1, G_2, \ldots)$ is a computational game. Let $\mathcal{M}$ be any set of TMs that includes all TMs that are guaranteed to halt on every input (thus, $\mathcal{M}$ could consist of all TMs). At a high level, the argument for the existence of ϵ-NE is a straightforward application of ideas of Lipton and Markakis [14]. As they observe, given a game G, we can represent the conditions required for a strategy to be a NE using a single algebraic equation (in several variables), where a NE must be a root of the equation. We can compute a strategy profile that is arbitrarily close to a root of this algebraic equation; it can be shown that a strategy vector that is sufficiently close to a root is an ϵ-NE. We can now obtain an ϵ-NE for the computational game $\mathcal{G} = (G_1, G_2, \ldots)$ as follows: Given ϵ, the nth game G_n, and type $t \in T_i$, player i computes a profile $(s_1^n, s_2^n \ldots s_k^n)$ of distributions over actions that is an ϵ-NE of G_n (conditional on t) and plays according to $s_i^n(t)$. (If there are several ϵ-NEs, one is chosen in a consistent way, so that all players are playing a component of the same profile.) Using these ideas we can prove the following result, whose proof can be found in the full paper [11].

Theorem 3. *If $\mathcal{G} = (G_1, G_2 \ldots)$ is a computational game, $\epsilon > 0$, and $\mathcal{M}$ includes all TMs that halt on all inputs, then G has an ϵ-$\mathcal{M}$-NE.*

6 Conclusion

We have considered computational games, where TMs compute strategies of players. We showed that a NE for polynomial-time players may not exist. This suggests that classic notions in game theory, such as best response, must be treated carefully when considering computational games with resource-bounded players.

As we showed, for unbounded players, an ϵ-NE always exists in a computational game. Even for bounded players, there may exist circumstances under which an ϵ-NE exists. For example, it may be that there exists an equilibrium if we bound the number of states in TMs used by players. Studying properties of games or TMs used by players that ensure the existence of (ϵ)-NE in computational games is an interesting direction for future research. It might also prove worthwhile to study the effect of limiting resources other than time, such as space or the amount of randomness used by players. Finally, our paper also leaves open the question as to whether there exists a NE in our model when we restrict players to TM whose running time is at most n^r for a fixed integer r.

Acknowledgments: Halpern was supported in part by NSF grants IIS-178108 and IIS-1703846, a grant from the Open Philanthropy Foundation, ARO grant W911NF-17-1-0592, and MURI grant W911NF-19-1-0217. Pass was supported in part by NSF grant IIS-1703846.

References

1. Ben-Sasson, E., Tauman-Kalai, A., Kalai, E.: An approach to bounded rationality. In: Proc. of the 19th Neural Information Processing Systems Conference. pp. 145–152 (2007)
2. Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. Journal of Computer and System Sciences **37**, 156–189 (1988)
3. Chen, X., Deng, X., Teng, S.H.: Settling the complexity of two-player Nash equilibrium. Journal of the ACM **53**(3) (2009)
4. Courtois, N.T., Bahack, L.: On subversive miner strategies and block with holding attack in bitcoin digital currency (2014), arXiv preprint: http://arxiv.org/abs/1402.1718
5. Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a Nash equilibrium. In: Proc. 38th ACM Symposium on Theory of Computing. pp. 71–78 (2006)
6. Dodis, Y., Halevi, S., Rabin, T.: A cryptographic solution to a game theoretic problem. In: CRYPTO 2000: 20th International Cryptology Conference. pp. 112–130. Springer-Verlag (2000)
7. Feigenbaum, J., Koller, D., Shor, P.W.: A game-theoretic classification of interactive complexity classes. In: Proc. of the Structure in Complexity Theory Conference. pp. 227–237 (1995)

8. Fortnow, L., Impagliazzo, R., Kabanets, V., Umans, C.: On the complexity of succinct zero-sum games. Computational Complexity **17**(3), 353–376 (2008)
9. Fortnow, L., Santhanam, R.: Bounding rationality by discounting time. In: Proc. Innovations in Computer Science Conference. pp. 143–155 (2010)
10. Halpern, J.Y., Pass, R.: Algorithmic rationality: Game theory with costly computation. Journal of Economic Theory **156**, 246–268 (2015). https://doi.org/10.1016/j.jet.2014.04.007
11. Halpern, J.Y., Pass, R., Reichman, D.: On the nonexistence of equilibrium in computational games (2019)
12. Halpern, J.Y., Pass, R., Seeman, L.: Computational extensive-form games. In: Proc. 17th ACM Conference on Electronic Commerce (EC '16). pp. 681–698 (2016)
13. Holenstein, T.: Pseudorandom generators from one-way functions: A simple construction for any hardness. In: Proc. of the Theory of Cryptography Conference. pp. 443–461 (2006)
14. Lipton, R.J., Markakis, E.: Nash equilibria via polynomial equations. In: Proc. LATIN 2004: Theoretical Informatics. pp. 413–422 (2004)
15. Maschler, M., Solan, E., Zamir, S.: Game Theory. Cambridge University Press, Cambridge, U.K. (2013)
16. Megiddo, N., Wigderson, A.: On play by means of computing machines. In: Theoretical Aspects of Reasoning about Knowledge: Proc. 1986 Conference, pp. 259–274 (1986)
17. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), http://www.bitcoin.org/bitcoin.pdf
18. Nash, J.: Non-cooperative games. Annals of Mathematics **54**, 286–295 (1951)
19. Neyman, A.: Bounded complexity justifies cooperation in finitely repeated prisoner's dilemma. Economic Letters **19**, 227–229 (1985)
20. Neyman, A.: The positive value of information. Games and Economic Behavior **3**, 350–355 (1991)
21. Neyman, A.: Finitely repeated games with finite automata. Mathematics Operation Research **23**, 513–552 (1998)
22. Oliehook, F., Savani, R., Gallego, J., van der Poel, E., Gross, R.: Beyond local Nash equilibria for adversarial networks (2018), available at http://arxiv.org/abs/1806.07268.
23. Papadimitriou, C.H., Yannakakis, M.: On complexity as bounded rationality. In: Proc. 26th ACM Symposium on Theory of Computing. pp. 726–733 (1994)
24. Rubinstein, A.: Finite automata play the repeated prisoner's dilemma. Journal of Economic Theory **39**, 83–96 (1986)
25. Schoenebeck, G., Vadhan, S.: The computational complexity of nash equilibria in concisely represented games. Theory of Computing **4**, 270–279 (2006)
26. Simon, H.A.: A behavioral model of rational choice. Quarterly Journal of Economics **49**, 99–118 (1955). https://doi.org/10.2307/1884852
27. Wee, H.: On obfuscating point functions. In: Proc. of the 37th ACM Annual Symposium on Theory of Computing. pp. 523–532 (2005)