Bounded Rationality of Restricted Turing Machines[*]

Lijie Chen, Pingzhong Tang, Ruosong Wang
Institute for Interdisciplinary Information Sciences,
Tsinghua University, Beijing, China.

Abstract

Bounded rationality aims to understand the effects of how limited rationality affects decision-making. The traditional models in game theory and multiagent system research, such as finite automata or unrestricted Turing machine, fall short of capturing how intelligent agents make decision in realistic applications. To address this problem, we model bounded rational agents as restricted Turing machines: restrictions on running time and on storage space. We study our model under the context of two-person repeated games. In the case where the running time of Turing machines is restricted, we show that computing the best response of a given strategy is much harder than the strategy itself. In the case where the storage space of the Turing machines is restricted, we show the best response of a space restricted strategy can not be implemented by machines within the same size (up to a constant factor). Finally, we study how these restrictions affect the set of Nash equilibria in infinitely repeated games. We show restricting the agent's computational resources will give rise to new Nash equilibria.

Introduction

Bounded rationality has been a topic of extensive interest in artificial intelligence and multi-agent system research (Larson and Sandholm 2004; 2005; Cavallo and Parkes 2008; Wright and Leyton-Brown 2010; 2012; Celis et al. 2012; Chen and Tang 2015; Tang and Zhang 2016; Tang et al. 2017). It refers to the limitations (time, space, information, etc) agents encounter that prevent them from making a fully rational decision in realistic settings. This phenomenon has been widely studied in the realm of repeated games (Osborne and Rubinstein 1994). An important feature of repeated games, often modeled as extensive-form games, is their gigantic strategy space. A strategy of a player needs to specify his action choice for any possible history (on or off the actual play path) where it is his turn to move. This leads to the difficulty that the description of a general strategy costs exponential bits in storage and is highly unrealistic. To mitigate this difficulty, a stylized approach models such strategies as *finite automata* (Rubinstein 1986; Osborne and Rubinstein 1994), where "equivalent" histories are grouped into state. Under this compact formulation, the set of equilibria has been characterized (Rubinstein 1986), the computation of the best response against an automata strategy has been investigated (Gilboa 1988; Ben-porath 1990) and the computation of the Stackelberg equilibrium has been investigated (Zuo and Tang 2015).

However, restricting strategies to finite automata loses generality. For example, in infinitely repeated prisoner's dilemma, to model the following strategy:

```
Play D iff the other played D more than
C in the past,
```

one must resort to machines such as pushdown automata.

Indeed, in reality, one can do much better than finite automata: we write computer programs! This inspires researchers to consider the possibility of modeling the bounded rational agents as general Turing machines. Megiddo and Wigderson (1986) model a strategy as a general Turing machine and show that, in finitely repeated games, computing best response against such a machine is trivial by another Turing machine. Knoblauch (1994) shows that in infinitely repeated games and limit-of-means utility, there exists a Turing machine strategy such that no Turing machine can implement its best response. Nachbar and Zame (1996) derive the same results, for discounted utility.

However, the general Turing machine model is also unrealistic in that it assumes an agent can perform computation in arbitrarily long time and use arbitrarily large storage space. Taking this into consideration, existing work has investigated Turing machines with size-restrictions (aka. restrictions on Kolmogorov complexity). Megiddo and Wigderson (1986) show that, under a certain hypothesis, cooperation can be approximately sustained in repeated prisoner's dilemma if we restrict the size of a Turing machine. Lacote (2005) later shows that cooperation can be sustained in finitely repeated games if and only if the Kolmogorv complexity of one player's strategy is substantially smaller than the number of rounds.

In this paper, we explore this direction further, by studying a realistic model of bounded rationality, where agents are confined to use time-restricted or space-restricted Turing machines to implement their strategies. We first use com-

[*]This work was supported by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the Natural Science Foundation of China Grant 61033001, 61361136003, 61303077, 61561146398, a Tsinghua Initiative Scientific Research Grant and a China Youth 1000-talent program.

putational complexity models to rigorously define time and space restrictions. We then study the important game theoretical question of how to compute and implement the best response of such a restricted Turing machine.

For time-restricted case. We show that computing best response against a strategy whose running time is bounded by a polynomial in the number of rounds is NP-*complete*, more generally, computing best response against a strategy with oracle access to a $\sum_{\mathsf{i}}^{\mathsf{P}}$-*complete* language whose running time is bounded by a polynomial in the number of rounds is $\sum_{\mathsf{i+1}}^{\mathsf{P}}$-*complete*. Readers may refer to the full version of this paper or a standard computational complexity textbook (e.g., (Arora and Barak 2009)) for the definition of complexity class $\sum_i^{\mathsf{P}}$.

The above results suggest that finding the best response of a strategy is harder than the strategy itself. It also suggests even if your opponent runs some polynomial time algorithm to decide its decision, you might not be able to efficiently find the best response against him under the conjecture that $\mathsf{P} \neq \mathsf{NP}$.

We study the space-restricted case under two natural models and show that, computing its best response is PSPACE-*complete*. We also show that under one of these models, implementing a strategy's best response requires a super linear expansion in strategy size under a certain reasonable complexity conjecture.

Those results suggest that, in contrast to time-restricted case, finding the best response in polynomial space is possible, but implementing them with linear expansion in size is impossible.

The second question we study is that, if both players have bounded rationality and this is common knowledge, how does it affect on the play of the game? More specifically, how does it change the set of the Nash Equilibria? We show that, in infinitely repeated games, interesting new equilibria will emerge. The intuition behind this result is as follows: For certain strategies, when assuming unbounded computational power of the opponent, these strategies will yield low utilities; however, knowing (by the common knowledge assumption) that the opponent is restricted, these strategies guarantee high utilities and can emerge as new Nash equilibrium! The proof of this part is quite nontrivial and is of independent interest.

Preliminaries

Repeated Games

In this paper, we focus on two-person repeated games.

Definition 1. *$G = \langle S_1, S_2, u_1, u_2\rangle$ is a two-person game in normal form. S_i is the finite set of actions for player i and $u_i : S_1 \times S_2 \to \mathbb{R}$ is the utility function for player i.*

Definition 2. *A super game G^n consists of n consecutive repetitions of a stage game G. At the t-th repetition, each player's strategy is to choose an action based on the history plays in rounds $1 \dots t-1$. That is, a player's strategy in the super game is a function that maps the set of all possible histories to the set of actions.*

1,1	0,5
5,0	3,3

Table 1: Payoff matrix of Prisoner's Dilemma

Definition 3. *In a super game G^n, denote s_i as the strategy of player i. The utility for player i is $U_i(s_1, s_2) = \sum_{t=1}^{n} u_i(a_{t,1}, a_{t,2})$, where $a_{t,i}$ is the action of the player i at round t.*

For ease of exposition, we consider the simplest nontrivial case where the stage game is the well-known Prisoner's Dilemma whose payoff matrix is given in Table 1. Sometimes we may call Prisoner's Dilemma PD game for short. In the remainder of this paper, we use G to denote the Prisoner's Dilemma. We call the two actions of a player *cooperate* and *defect*. Map cooperate to 1 and defect to 0, a strategy is then equivalent to a function: $\{0,1\}^* \to \{0,1\}$.

For brevity, we denote *Turing machine, deterministic Turing machine and nondeterministic Turing machine* by TM, DTM and NTM, respectively. We assume basic knowledge in computational complexity. Readers can refer to the full version of this paper for a list of self-contained conceptions and definitions related to computational complexity, e.g., the definition of time constructible function, the time/space restricted complexity class DTIME, NTIME, DSPACE, NSPACE, PSPACE and $\mathsf{LOGSPACE}$, the polynomial hierarchy (PH) with related complete problem $\sum_i^{\mathsf{P}}\mathsf{SAT}$ and the definition of oracle machine $\mathsf{P}^{\mathcal{O}}$.

We first formally define the language a specific strategy.

Definition 4. *For a strategy s, define the language of s to be the set of histories based on which s plays cooperate. Here the history of a repeated game before the t-th round is the string $a_{1,1}, a_{1,2}, a_{2,1}, a_{2,2}, \dots, a_{t-1,1}, a_{t-1,2}$ in which $a_{i,j}$ is the action that the player j takes in round i. We say a TM M implements a strategy s if M decides the language of s.*

We define a strategy's complexity by its language's complexity class.

Definition 5. *Let $\mathcal{C}$ be a complexity class. A strategy s is a $\mathcal{C}$-strategy if the language of s belongs to $\mathcal{C}$.*

Following the definition above, it is natural to further define time-restricted strategies like P-strategy and space-restricted strategies like PSPACE-strategy, which will be studied in the following sections. Also, we say a strategy s is a TM-strategy if the language of s is a computable language.

Time-restricted Strategies

In this section, we study how much time resource is needed to find or implement a best response of a time-restricted strategy.

As we do not care about the amount of space resources used, implementing a best response of a particular strategy in a super game is simple as one can simply store the optimal action sequence and play according to that. Thus, we focus on studying the computational complexity of finding the best response in a super game G^n against a time-restricted strategy.

Our plan of computing the best response of a polynomial-time strategy (i.e., P-strategy) consists of two steps. First, we compute the cumulative utility of the best response. Second, we construct the best response based on the utility computed during the first step.

The decision version of the first step is whether there exists a strategy that can gain at least utility k from the strategy represented by a polynomial-time TM M? However, this question is in fact not well defined. M may run in super-polynomial time, or even never halts. Meanwhile, by Rice theorem (Rice 1953), deciding whether a TM M runs in polynomial time or always halts on all inputs is uncomputable. Thus, to make this problem computable, we have to put some restriction onto it.

To address the issue mentioned, we resort to complexity theory and restrict the TM's running time explicitly by a time constructible function f.

Definition 6. *Let f be a time constructible function and M be a* TM*, define M_f as a* TM *such that it runs M on input string x of length n for $f(n)$ steps. During the $f(n)$ steps, if M halts, then M_f return the output of M; otherwise M_f rejects.*

It is easy to see that M_f represents a strategy since it always halt. In addition, if f is a polynomial, M_f is indeed a P-strategy.

Let f be a time constructible function, define the decision problem BR_f as follows.

Definition 7. $\mathsf{BR}_f = \{\langle M, 1^n, k\rangle\}$ *such that there exists a strategy that can gain at least utility k against the strategy M_f in the game G^n.*

Here we write n in unary form as if we write n in binary form instead, the best response sequence will have exponential length with respect to the input size.

We first show how to use an oracle to the decision problem to find the best response.

Lemma 1. *For a given strategy M whose running time is bounded by f, finding the best response sequence is in* $\mathsf{P}^{\mathsf{BR}_f}$
.

With Lemma 1, it suffices to study the complexity of the decision problem BR_f. We have the following theorem.

Theorem 1. *There is a polynomial f such that* BR_f *is* NP*-complete. For every polynomial f,* BR_f *is in* NP.

Our plan is to reduce SAT to BR_f. Given a CNF formula φ, the intuition here is to construct an agent that cooperates only if the opponent finds a satisfying assignment for φ. It treats the opponent's actions in the first n rounds as the assignments to the n variables of the SAT instance and checks its validity. If the opponent gives a satisfying assignment for φ, then it cooperates in the following rounds. Otherwise, it defects and thus induces a low cumulative utility for the best response. The detailed proof of Lemma 1 and Theorem 1 can be founded in the full version of this paper.

A direct corollary of Theorem 1 is that for any polynomial g that is always greater than f, BR_g is NP-*complete* as well.

As for a P-strategy, computing the utility of its best response is in NP, then by Lemma 1, we know that we can find the strategy itself in P^{NP}. Meanwhile, Theorem 1 suggests that, in order to compute the best response for a general P-strategy, one must be within the class of P^{NP}.

The natural next question to ask what is the complexity of finding the best response against a P^{NP}-strategy. Informally, we can prove that calculating the best response against a P^{NP}-strategy is $\sum_2^{\mathsf{P}}$-*complete*. Interested reader may refer to the full version of this paper for the definition of $\sum_2^{\mathsf{P}}$-*complete* and the details of this proof.

Space-restricted Strategies

In this section, we study the space restricted strategies. The first natural idea is to study the space complexity to calculate the best response against a PSPACE-strategy.

Lemma 2. *The best response of a* PSPACE*-strategy can be found in* PSPACE.

However, PSPACE-strategy is unrealistic in practice due to the huge amount of space owned by the player. Thus, we switch to study LOGSPACE-strategies that players are limited to use logarithm amount of extra space to calculate the best response.

For LOGSPACE-strategies, notice that the history has large length, which makes it possible for a strategy to "cheat" by gaining extra space via outputting extra information into the history. The idea is to transform a polynomial time TM to a polynomial size circuit in LOGSPACE. At every step, we evaluate one gate in this circuit and output as an action of the LOGSPACE-strategy. Now the game has polynomial number of rounds. By such a method, a LOGSPACE-strategy can do something similar to a P-strategy.

The above result for LOGSPACE-strategies is not interesting. The reason is that, since the space is limited, it is dubious that the strategies afford to store the whole history. Thus, We need to find better models to study space-restricted strategies, in which the whole history is not revealed to the strategy directly.

We propose the following alternative models. The main idea is to model space-restricted strategy as a function that takes the last action of the opponent and the current information bits as input, and outputs the new information bits and the action of this round.

An N-bits-strategy is a function $trans : \{0,1\}^{N+1} \to \{0,1\}^{N+1}$. The function $trans$ takes the information bits and the action of the opponent in last round as input, outputs the information bits and the action for the current round.

The game is played as follows. See Figure 1 for an illustration. Let o_i be the action of the opponent in the i-th round and a_i be the action to be played in the i-th round. We assume $o_0 = 0$. Let b_i be the information bits outputted after the i-th round. We assume $b_0 = 0^N$. In the i-th round, we calculate use the $trans$ function to calculate the new information bits and the action, i.e., $(a_i, b_i) = trans(o_{i-1}, b_{i-1})$.

We remark that this strategy in fact captures the typical way we design a gaming AI: record some information and update the information by an algorithm after the opponent moves. In general, most of the AI does not sweep through

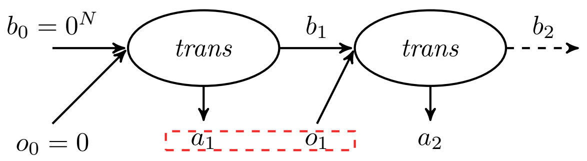

Figure 1: An illustration of the function *trans*. The dashed rectangle indicates the action profile of the first round.

the history every time when making a move, which is costly and typically unaffordable.

In the time-restricted case, we studied how much time resource is needed to find or implement a best response of a time-restricted strategy. In this case, since we focus on space resources, it is natural to study how much space resource is needed for the same tasks of a space-restricted strategy.

To do this, we should first define how to measure the amount of memory used by a strategy in our model. For an agent to work with a space-restricted strategy, it needs space to evaluate the function *trans* and store the description of the function *trans* itself. So it is natural to define the amount of space used by a space-restricted strategy to be the number of storage bits needed to evaluate the function *trans* plus the number of bits needed to specify the function *trans*.

In the following part of this section, we study the amount of space needed for a general TM to compute the best response for a space-restricted strategy and to implement the best response via a space-restricted strategy. We consider two cases. In the first case, we require *trans* to be efficiently computable. This leads to the *circuit strategy model*. In the second case, we drop the computation requirement and consider the *inplace strategy model*.

Circuit Strategy

In this section we consider the case that we require that function *trans* can be efficiently computed. In this case, since the number of information bits N is fixed, we can then represent *trans* as a polynomial size circuit. We can benefit from a circuit representation as a polynomial size circuit will always halt and can always be efficiently computed, which makes the analysis easier.

Definition 8. *An N-bits-circuit strategy is a boolean circuit C which has $N+1$ input gate and $N+1$ output gate. The size of a circuit strategy is the number of gates in C plus the binary description size of C.*

We include the number of gates in the size of C as for each gate we need one bit to record its output in order to evaluate the circuit C.

Inplace Strategy

In this section, we drop the computation efficiency requirement and consider the so-called inplace strategy defined as follows.

Definition 9. *An N-bits-inplace strategy is a* TM *M which runs on input of $N+1$ bits, always halts, and uses only $N+1$ bits of space, returns the content of tape as output when it halts. The size of an inplace strategy is N plus the binary description size of M.*

The name "inplace strategy" is due to the fact that the strategy is implemented by an "inplace" TM, which does not use any extra space other than the input tape itself. Note that in this case, it does not matter whether M accepts or rejects.

As dealing with time-restricted strategies, we are still faced with the same problem: how do we know M is an inplace strategy of N bits? We address this problem by a similar manner as we have done for time-restricted strategies.

Let M be a TM, define M_I as a TM such that it runs M on input string x of length N. If M tries to access tape cells outside the N input bits or does not halt after $Q \cdot N2^N$ steps where Q is the number of the states in M, then M_I halts. M_I returns the content on the tape when it halts.

Lemma 3. *If* TM *M is an $(N-1)$-bits-inplace strategy, the output of M_I is the same as the output of M for every input of length N.*

Complexity of Computing Best Response

Similar to what we have done for time-restricted cases, to study the space complexity for computing best response against a circuit strategy or an inplace strategy, we first study the decision version, and then use the decision version as a subroutine to find the best response.

We first introduce two sets of languages BRCT and BRIP, which are decision versions of finding best response against circuit strategy and inplace strategy, respectively.

Definition 10. BRCT $= \{\langle C, n, k\rangle\}$ *such that there exists a strategy can yield at least utility k against circuit strategy C in the game G^n.* BRIP $= \{\langle M, 1^N, n, k\rangle\}$ *such that there exists a strategy can yield at least utility k against an inplace strategy M_I with N information bits in the game G^n.*

Similar to the time-restricted case, once we have oracle access to the decision version, computing the best response can also be done by an algorithm similar to that of Theorem 1 in polynomial space, which implies the following lemma.

Lemma 4. *For a given inplace (circuit) strategy M, we can find the best response at each round in* $\mathsf{PSPACE}^{\mathsf{BRIP}}$ *or* $\mathsf{PSPACE}^{\mathsf{BRCT}}$.

Lemma 4 suggests that, in order to study the complexity of finding best response against a circuit strategy or an inplace strategy, it suffices to study the decision version of the problem, which was given in the following theorems.

Theorem 2. BRIP *is* PSPACE-*complete.*

An NPSPACE algorithm is to enumerate all possible action sequences and simulate the inplace strategy to check whether it can gain utility at least k. As NPSPACE $=$ PSPACE(Savitch 1970), BRIP $\in$ PSPACE. The hardness part can be found in appendix.

Theorem 3. BRCT *is* PSPACE-*complete.*

Similar to Theorem 2, constructing a PSPACE algorithm is easy. The non-trivial part is to show that BRCT is PSPACE-*complete*. In order to prove that, we first introduce some necessary notations.

Let the alphabet of a TM be $\{0, 1\}$ and the unused cells of the tape be filled with #. Define the configuration of a TM as follows.

Definition 11. *A configuration u of a* TM *M is a triple $(q, pos, content) \in Q \times \mathbb{N} \times \{0, 1\}^*$, where Q is the set of states of M, q is the current state, pos is the location of the head pointer and content is the contents of all non-blank cells of the tape.*

For a NTM M, define $\mathrm{next_c}(M, u)$ to be the configuration after running M for one step on configuration u with the nondeterministic choice to be c. If M have already halted on u, define $\mathrm{next_c}(u) = u$.

The intuition of the proof comes as follows. First, as PSPACE = NPSPACE, it is sufficient to prove BRCT is NPSPACE-*complete*.

For a NPSPACE TM M, its configurations can be described by a polynomial number of bits. We construct a circuit strategy C whose information bits $u \in \{0, 1\}^*$ describe a configuration of M. If u is not halted, C defects and treats the opponent's action as the nondeterministic choice c and outputs $\mathrm{next_c}(M, u)$ as information bits. Otherwise, C outputs u as information bits, cooperates if u is in an accepting state and defects if u is in a rejecting state.

To test whether M accepts a input string s, note that for a sufficient long running, if there is such a sequence of nondeterministic choices that lead M to an accepting state (which means M accepts s), then C will always cooperate after that, so we can gain a relatively high utility. Otherwise, C will always defect, which causes a low utility. Detailed proof for this theorem can be found in the full version of this paper.

The above results demonstrate that computing a best response against a space-restricted strategy can be done in polynomial space, in contrary to the time-restricted case, where it is NP-*complete* to compute the best response.

Complexity of Implementing Best Response

Now, we study the space complexity for implementing a best response of a particular space-restricted strategy, which is equivalent to the question what is the smallest possible size among all best responses.

From previous sections, the algorithm for computing the best response uses polynomial space. Then the natural question is whether it can be done in linear space? Notice that as the input string of the algorithm is $\langle M, n, k\rangle$ which has length $|M| + \log n + \log k = |M| + O(\log n)$, thus we when say polynomial/linear space algorithm, we refer to a polynomial/linear function of $|M| + O(\log n)$.

We have the following theorem which demonstrates that it is impossible to have a linear space best response against an inplace strategy under reasonable complexity conjecture.

Theorem 4. *Unless* $\mathsf{DSPACE}(n) = \mathsf{NSPACE}(n)$*, there does not exist a constant T such that any inplace strategy of size S in super game G^n have a best response implemented by an inplace strategy with size smaller than $T \cdot (S + \log n)$.*

The intuition here is to construct an agent that simulates the behavior of a NTM on a specific input by treating the opponent's actions as the nondeterministic choices. The agent will cooperate only if it is in an accepting state. Thus, the best response strategy will output a sequence of nondeterministic choices which makes the NTM end in an accepting state. If the best response can be implemented in linear space, we can then construct a DTM which enumerates all possible inplace strategies with linear size to find the best response. By finding the best response, we can actually get the nondeterministic choices of the NTM, and thus can simulate the running of a NTM on a DTM, still by using linear space, which contradicts our assumption that $\mathsf{DSPACE}(n) \neq \mathsf{NSPACE}(n)$. Detailed proof can be found in the full version of this paper.

The taken-away message of Theorem 4 is that in general, implementing a best response of a particular strategy need much more space than the strategy itself.

Nash Equilibria via Restricted Turing Machine

In this section, we study the case when both player are using restricted Turing machines strategies and this is a common knowledge. We are going to study infinitely repeated game from now on. For simplicity of analysis, we use the standard limit of mean as the utility notion.

Definition 12. *In an infinitely super game G^∞, denote s_i as the strategy of player i. The utility for player i is $U_i(s_1, s_2) = \liminf_{N\to\infty} \frac{1}{N}\sum_{t=1}^{N} u_i(a_{t,1}, a_{t,2})$, where $a_{t,i}$ is the action of the player i at the t-th round.*

Suppose s is a strategy, Let $\mathcal{S}$ be the set of all possible strategies in G^∞, denote $\mathsf{BR}(s) = \sup\{U_2(s, t) \mid t \in \mathcal{S}\}$. We say a strategy t is a best response of s if $U_2(s, t) = \mathsf{BR}(s)$. Note that it is possible that there is no best response for s.

Suppose s is a strategy, and $\mathcal{C}$ is a complexity class, denote $\mathsf{BR}_{\mathcal{C}}(s) = \sup\{U_2(s, t) \mid \text{t is a } \mathcal{C}\text{-strategy}\}$. We say t is a $\mathcal{C}$-best response of s if $U_2(s, t) = \mathsf{BR}_{\mathcal{C}}(s)$. Based on these notations, we are now ready to define $\mathcal{C}$-Nash Equilibrium ($\mathcal{C}$-NE).

Definition 13. *A $\mathcal{C}$-NE of an infinitely super game G^∞ is a pair of strategy (s_1, s_2) such that both s_1 and s_2 are $\mathcal{C}$-strategies, and none of them can gain higher utility by deviating to another $\mathcal{C}$-strategy.*

Our goal now is to investigate how does such restriction affect the set of NE. At first glance, such a restriction will disqualify some old NEs. Surprisingly, such restriction will also produce some new NEs.

Lemma 5. *There exists a* TM*-NE that is not a NE, and a NE which is not a* TM*-NE.*

Lemma 6. *There exists a* P*-NE that is not a* TM*-NE, and a* TM*-NE which is not a* P*-NE.*

Moreover, we have some stronger results summarized in the following two theorems. We say a $f : \mathbb{N} \to \mathbb{N}$ is a reasonable function, if f is a strictly increasing time constructible function such that $f(0) > 0$.

Theorem 5. *Let $f, g : \mathbb{N} \to \mathbb{N}$ be two reasonable functions, such that $f(n) \log f(n) \in o(g(n))$ and $f(n) \in \Omega(n \log n)$. There exists a* $\mathsf{DTIME}(f(n))$*-NE which is not a* $\mathsf{DTIME}(g(n))$*-NE, and a* $\mathsf{DTIME}(g(n))$*-NE which is not a* $\mathsf{DTIME}(f(n))$*-NE.*

Theorem 6. *Let $f, g : \mathbb{N} \to \mathbb{N}$ be two reasonable functions, such that $f(n) \in o(g(n))$ and $f(n) \in \Omega(\log n)$. There exists a* $\mathsf{DSPACE}(f(n))$*-NE which is not a* $\mathsf{DSPACE}(g(n))$*-NE, and a* $\mathsf{DSPACE}(g(n))$*-NE which is not a* $\mathsf{DSPACE}(f(n))$*-NE.*

We sketch the essence of the proofs here. Full proof can be found in the full version of this paper. Let $\mathcal{C}, \mathcal{D}$ be two complexity classes such that $\mathcal{C} \subset \mathcal{D}$. Each of our results has two parts: there exists a $\mathcal{D}$-NE which is not a $\mathcal{C}$-NE, and there exists a $\mathcal{C}$-NE which is not a $\mathcal{D}$-NE.

We first construct a $\mathcal{C}$-strategy s_1 that s_1 has a $\mathcal{D}$-strategy as the best response but no $\mathcal{C}$-strategy as the best response. For this purpose, we construct a hard problem $\mathcal{P}$. And in some specific rounds, s_1 treats the opponent's action as the answer to "what is the value of $\mathcal{P}(x)$?" where x is dependent on the current round number. s_1 will check whether the opponent's answer is right in later rounds. Once s_1 finds an incorrect answer, s_1 defects forever, otherwise s_1 cooperates. Thus, in order to be the best response of s_1, the opponent should be able to solve all questions correctly. Then we can construct $\mathcal{P}$ in a way that no machine of complexity $\mathcal{C}$ can compute it, but some machine in complexity $\mathcal{D}$ can.

To prove the first part, we further construct a $\mathcal{D}$-strategy s_2 that s_1 and s_2 together constitute a NE. Obviously they constitute a $\mathcal{D}$-NE. And as s_1 has no $\mathcal{C}$-strategy as the best response, so they are not $\mathcal{C}$-NE.

To prove the second part, we construct a hybrid strategy t such that it asks the opponent to make a two-decision choice at the first round. If the opponent choose the first choice, t then acts like a strategy t_1 which is easy to make best response and $\mathsf{BR}(t_1) < \mathsf{BR}(s_1)$, and for the second choice, t will then act like strategy s_1.

Consider another strategy v which chooses the first choice and then behaves the same as t_1's best response. t and v forms a $\mathcal{C}$-NE if we construct them carefully. But they does not form $\mathcal{D}$-NE, as v can make profitable deviation by choosing the second choice and acts like s_1's best response.

Future Works

There are some intriguing problems to be explored.

- How do the restrictions on strategies affect the set of NE in finitely repeated game? Particularly, to what extent should we restrict an inplace(circuit) strategy so that cooperation can be sustained?
- For circuit strategy and inplace strategy, what if we impose the so-called simple machine preference (i.e., prefer machines with fewer states)?

References

Arora, S., and Barak, B. 2009. *Computational complexity*. Cambridge University Press, 1 edition.

Ben-porath, E. 1990. The complexity of computing a best response automaton in repeated games with mixed strategies. *Games and Economic Behavior* 2(1):1–12.

Cavallo, R., and Parkes, D. C. 2008. Efficient metadeliberation auctions. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, 50–56.

Celis, L. E.; Karlin, A. R.; Leyton-Brown, K.; Nguyen, C. T.; and Thompson, D. R. M. 2012. Approximately revenue-maximizing auctions for deliberative agents. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*

Chen, L., and Tang, P. 2015. Bounded rationality of restricted turing machines. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 1673–1674. International Foundation for Autonomous Agents and Multiagent Systems.

Gilboa, I. 1988. The complexity of computing best-response automata in repeated games. *Journal of Economic Theory* 45(2):342–352.

Knoblauch, V. 1994. Computable strategies for repeated prisoner' s dilemma. *Games and Economic Behavior* 7(3):381–389.

Lacôte, G. 2005. Boundedly complex turing machines play the repeated prisoner's dilemma: some results. Technical report.

Larson, K., and Sandholm, T. 2004. Experiments on deliberation equilibria in auctions. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), 19-23 August 2004, New York, NY, USA*, 394–401.

Larson, K., and Sandholm, T. 2005. Mechanism design and deliberative agents. In *4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands*, 650–656.

Megiddo, N., and Wigderson, A. 1986. On play by means of computing machines: preliminary version. 259–274.

Nachbar, J. H., and Zame, W. R. 1996. Non-computable strategies and discounted repeated games. *Economic theory* 8(1):103–122.

Osborne, M. J., and Rubinstein, A. 1994. *A Course in Game Theory*. MIT Press.

Rice, H. G. 1953. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society* 358–366.

Rubinstein, A. 1986. Finite automata play the repeated prisoner's dilemma. *Journal of Economic Theory* 83–96.

Savitch, W. J. 1970. Relationships between nondeterministic and deterministic tape complexities. *Journal of computer and system sciences* 4(2):177–192.

Tang, P., and Zhang, H. 2016. Unit-sphere games. *International Journal of Game Theory, to appear*.

Tang, P.; Teng, Y.; Wang, Z.; Xiao, S.; and Xu, Y. 2017.

Computational issues in time-inconsistent planning. In *Proceedings of AAAI*.

Wright, J. R., and Leyton-Brown, K. 2010. Beyond equilibrium: Predicting human behavior in normal-form games. In *AAAI*.

Wright, J. R., and Leyton-Brown, K. 2012. Behavioral game theoretic models: a bayesian framework for parameter analysis. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 921–930. International Foundation for Autonomous Agents and Multiagent Systems.

Zuo, S., and Tang, P. 2015. Optimal machine strategies to commit to in two-person repeated games. In *Proceedings of AAAI*.