Query Complexity of Approximate Nash Equilibria

Yakov Babichenko

AGT workshop, LSE, 17.10.2013

Query complexity

Consider a normal form game with n players and m actions for each player.

Query complexity

Consider a normal form game with n players and m actions for each player.
n is large, m is constant.

Query complexity

Consider a normal form game with n players and m actions for each player.
n is large, m is constant.
Who hard it is to compute an approximate Nash equilibrium in the game?

Query complexity

Consider a normal form game with n players and m actions for each player.
n is large, m is constant.
Who hard it is to compute an approximate Nash equilibrium in the game?
Warning: The input size is exponential: nm^n.

Query complexity

Consider a normal form game with n players and m actions for each player.
n is large, m is constant.
Who hard it is to compute an approximate Nash equilibrium in the game?
Warning: The input size is exponential: nm^n.
Who to overcome the warning?

Query complexity

Consider a normal form game with n players and m actions for each player.
n is large, m is constant.
Who hard it is to compute an approximate Nash equilibrium in the game?
Warning: The input size is exponential: nm^n.
Who to overcome the warning?
We assume existence of a black box.
The algorithm asks queries about the game and the black box returns answers.

Query complexity

Consider a normal form game with n players and m actions for each player.
n is large, m is constant.
Who hard it is to compute an approximate Nash equilibrium in the game?
Warning: The input size is exponential: nm^n.
Who to overcome the warning?
We assume existence of a black box.
The algorithm asks queries about the game and the black box returns answers.
Each queries is a pure action profile a. The answer is the payoff profile $u(a) = (u_i(a))_i$.

Consider a normal form game with n players and m actions for each player.
n is large, m is constant.
Who hard it is to compute an approximate Nash equilibrium in the game?
Warning: The input size is exponential: nm^n.
Who to overcome the warning?
We assume existence of a black box.
The algorithm asks queries about the game and the black box returns answers.
Each queries is a pure action profile a. The answer is the payoff profile $u(a) = (u_i(a))_i$.
The idea of query-complexity (QC) is to ask: how many queries should the algorithm ask until it knows an answer to the problem?

Deterministic QC vs. Probabilistic QC

- **Deterministic QC**: We allow only deterministic algorithms. The QC is the number of questions for the worst case input.

Deterministic QC vs. Probabilistic QC

- **Deterministic QC**: We allow only deterministic algorithms. The QC is the number of questions for the worst case input.
- **Probabilistic QC**: We allow probabilistic algorithms. The QC is the expected number of questions for the worst case input.

Deterministic QC vs. Probabilistic QC

- **Deterministic QC**: We allow only deterministic algorithms. The QC is the number of questions for the worst case input.
- **Probabilistic QC**: We allow probabilistic algorithms. The QC is the expected number of questions for the worst case input.

Example- The 1-entry problem

INPUT: A vector $v \in \{0,1\}^{2n}$, with n 0s and n 1s (i.e., $|\{i : v_i = 0\}| = |\{i : v_i = 1\}| = n$).

Deterministic QC vs. Probabilistic QC

- **Deterministic QC**: We allow only deterministic algorithms.
  The QC is the number of questions for the worst case input.
- **Probabilistic QC**: We allow probabilistic algorithms.
  The QC is the expected number of questions for the worst case input.

Example- The 1-entry problem

INPUT: A vector $v \in \{0,1\}^{2n}$, with n 0s and n 1s (i.e., $|\{i : v_i = 0\}| = |\{i : v_i = 1\}| = n$).
OUTPUT: An index i s.t. $v_i = 1$.

Deterministic QC vs. Probabilistic QC

- **Deterministic QC**: We allow only deterministic algorithms. The QC is the number of questions for the worst case input.
- **Probabilistic QC**: We allow probabilistic algorithms. The QC is the expected number of questions for the worst case input.

Example- The 1-entry problem

INPUT: A vector $v \in \{0,1\}^{2n}$, with n 0s and n 1s (i.e., $|\{i : v_i = 0\}| = |\{i : v_i = 1\}| = n$).
OUTPUT: An index i s.t. $v_i = 1$.
QUERIES: Each query is an index $i \in [2n]$, and the answer is v_i.

Deterministic QC vs. Probabilistic QC

- **Deterministic QC**: We allow only deterministic algorithms. The QC is the number of questions for the worst case input.
- **Probabilistic QC**: We allow probabilistic algorithms. The QC is the expected number of questions for the worst case input.

Example- The 1-entry problem

INPUT: A vector $v \in \{0,1\}^{2n}$, with n 0s and n 1s (i.e., $|\{i : v_i = 0\}| = |\{i : v_i = 1\}| = n$).
OUTPUT: An index i s.t. $v_i = 1$.
QUERIES: Each query is an index $i \in [2n]$, and the answer is v_i.
Deterministic QC $= n$.

Deterministic QC vs. Probabilistic QC

- **Deterministic QC**: We allow only deterministic algorithms. The QC is the number of questions for the worst case input.
- **Probabilistic QC**: We allow probabilistic algorithms. The QC is the expected number of questions for the worst case input.

Example- The 1-entry problem

INPUT: A vector $v \in \{0,1\}^{2n}$, with n 0s and n 1s (i.e., $|\{i : v_i = 0\}| = |\{i : v_i = 1\}| = n$).
OUTPUT: An index i s.t. $v_i = 1$.
QUERIES: Each query is an index $i \in [2n]$, and the answer is v_i.
Deterministic QC $= n$.
Probabilistic QC ≤ 2.

Related literature, correlated equilibrium

Query complexity of correlated equilibrium:

	Deterministic QC	Probabilistic QC
Exact CE	$\exp(n)$	$\exp(n)$ [HN 2013]
Approximate CE	$\exp(n)$ [HN 2013]	$poly(n)$ Regret minimizing algorithms (e.g. [HM 2000])

Related literature, correlated equilibrium

Query complexity of correlated equilibrium:

	Deterministic QC	Probabilistic QC
Exact CE	$\exp(n)$	$\exp(n)$ [HN 2013]
Approximate CE	$\exp(n)$ [HN 2013]	$poly(n)$ Regret minimizing algorithms (e.g. [HM 2000])

Query complexity of Nash equilibrium:

	Deterministic QC	Probabilistic QC
Exact NE	$\exp(n)$	$\exp(n)$
Approximate NE	$\exp(n)$	?

Related literature, correlated equilibrium

Query complexity of correlated equilibrium:

	Deterministic QC	Probabilistic QC
Exact CE	$\exp(n)$	$\exp(n)$ [HN 2013]
Approximate CE	$\exp(n)$ [HN 2013]	$poly(n)$ Regret minimizing algorithms (e.g. [HM 2000])

Query complexity of Nash equilibrium:

	Deterministic QC	Probabilistic QC
Exact NE	$\exp(n)$	$\exp(n)$
Approximate NE	$\exp(n)$	$\exp(n)$

Notations

- Set of probability distributions over B: $\Delta(B)$.

Notations

- Set of probability distributions over B: $\Delta(B)$.
- Support of a distribution: $supp(x) = \{b \in B : x(b) > 0\}$.

Notations

- Set of probability distributions over B: $\Delta(B)$.
- Support of a distribution: $supp(x) = \{b \in B : x(b) > 0\}$.
- Players: $[n] = \{1, 2, ..., n\}$.

Notations

- Set of probability distributions over B: $\Delta(B)$.
- Support of a distribution: $supp(x) = \{b \in B : x(b) > 0\}$.
- Players: $[n] = \{1, 2, ..., n\}$.
- Actions set of player i: A_i, $|A_i| = m$.

Notations

- Set of probability distributions over B: $\Delta(B)$.
- Support of a distribution: $supp(x) = \{b \in B : x(b) > 0\}$.
- Players: $[n] = \{1, 2, ..., n\}$.
- Actions set of player i: A_i, $|A_i| = m$.
- Action profiles set: $A = \times_{i \in [n]} A_i$.

Notations

- Set of probability distributions over B: $\Delta(B)$.
- Support of a distribution: $supp(x) = \{b \in B : x(b) > 0\}$.
- Players: $[n] = \{1, 2, ..., n\}$.
- Actions set of player i: A_i, $|A_i| = m$.
- Action profiles set: $A = \times_{i \in [n]} A_i$.
- Payoff function of player i: $u_i : A \to [0, 1]$.

Notations

- Set of probability distributions over B: $\Delta(B)$.
- Support of a distribution: $supp(x) = \{b \in B : x(b) > 0\}$.
- Players: $[n] = \{1, 2, ..., n\}$.
- Actions set of player i: A_i, $|A_i| = m$.
- Action profiles set: $A = \times_{i \in [n]} A_i$.
- Payoff function of player i: $u_i : A \to [0, 1]$.
- Payoff function profile: $u = (u_i)_{i \in [n]}$.

Notations

- Set of probability distributions over B: $\Delta(B)$.
- Support of a distribution: $supp(x) = \{b \in B : x(b) > 0\}$.
- Players: $[n] = \{1, 2, ..., n\}$.
- Actions set of player i: A_i, $|A_i| = m$.
- Action profiles set: $A = \times_{i \in [n]} A_i$.
- Payoff function of player i: $u_i : A \to [0, 1]$.
- Payoff function profile: $u = (u_i)_{i \in [n]}$.
- Best-reply value against x_{-i}: $br_i(x_{-i}) := \max_{a_i \in A_i} u_i(a_i, x_{-i})$.

Notations

- Set of probability distributions over B: $\Delta(B)$.
- Support of a distribution: $supp(x) = \{b \in B : x(b) > 0\}$.
- Players: $[n] = \{1, 2, ..., n\}$.
- Actions set of player i: A_i, $|A_i| = m$.
- Action profiles set: $A = \times_{i \in [n]} A_i$.
- Payoff function of player i: $u_i : A \to [0, 1]$.
- Payoff function profile: $u = (u_i)_{i \in [n]}$.
- Best-reply value against x_{-i}: $br_i(x_{-i}) := \max_{a_i \in A_i} u_i(a_i, x_{-i})$.
- $x = (x_i)_i$ is an ε-well supported Nash equilibrium if $u_i(a_i, x_{-i}) \geq br_i(x_{-i}) - \varepsilon$ for every $a_i \in supp(x_i)$.

The Main Theorem

The well supported Nash problem, **WSN**(n, m, ε):

The Main Theorem

The well supported Nash problem, **WSN**(n, m, ε):
INPUT: n-players, m-actions game.

The Main Theorem

The well supported Nash problem, **WSN**(n, m, ε):
INPUT: n-players, m-actions game.
OUTPUT: An ε-well supported Nash equilibrium.

The Main Theorem

The well supported Nash problem, **WSN**(n, m, ε):
INPUT: n-players, m-actions game.
OUTPUT: An ε-well supported Nash equilibrium.
QUERIES: Each queries is a pure action profile a. The answer is the payoff profile $u(a) = (u_i(a))_i$.

The Main Theorem

The well supported Nash problem, **WSN**(n, m, ε):
INPUT: n-players, m-actions game.
OUTPUT: An ε-well supported Nash equilibrium.
QUERIES: Each queries is a pure action profile a. The answer is the payoff profile $u(a) = (u_i(a))_i$.

Theorem

$$QC(\mathbf{WSN}(2n, 10^4, 10^{-8})) \geq \frac{2^{\frac{n}{3}}}{2n^4} \geq 2^{cn}.$$

The Main Theorem

The well supported Nash problem, **WSN**(n, m, ε):
INPUT: n-players, m-actions game.
OUTPUT: An ε-well supported Nash equilibrium.
QUERIES: Each queries is a pure action profile a. The answer is the payoff profile $u(a) = (u_i(a))_i$.

Theorem

$$QC(\mathbf{WSN}(2n, 10^4, 10^{-8})) \geq \frac{2^{\frac{n}{3}}}{2n^4} \geq 2^{cn}.$$

For every probabilistic algorithm that computes an (10^{-8})*-well supported Nash equilibrium in* $(2n)$*-players* (10^4)*-actions games, there exists a game where the expected number of queries will be at least* 2^{cn}.

Very Short Outline of the Proof

1. We reduce the $(2n)$-players ε-well supported Nash equilibrium problem to the problem of computing an approximate fixed point of an n-dimensional Lipschitz function.

Very Short Outline of the Proof

1. We reduce the $(2n)$-players ε-well supported Nash equilibrium problem to the problem of computing an approximate fixed point of an n-dimensional Lipschitz function.
   The reduction holds for constant values of ε!

Very Short Outline of the Proof

1. We reduce the $(2n)$-players ε-well supported Nash equilibrium problem to the problem of computing an approximate fixed point of an n-dimensional Lipschitz function.
   The reduction holds for constant values of ε!
2. We reduce the n-dimensional fixed point problem to the problem of finding end of a simple path on the n-dimensional hyper cube.

Very Short Outline of the Proof

1. We reduce the $(2n)$-players ε-well supported Nash equilibrium problem to the problem of computing an approximate fixed point of an n-dimensional Lipschitz function.
   The reduction holds for constant values of ε!
2. We reduce the n-dimensional fixed point problem to the problem of finding end of a simple path on the n-dimensional hyper cube.
   Hirsch, Papadimitriou, and Vavasis [1989] proved that the deterministic query complexity of the n-dimensional fixed point problem is $\exp(n)$. We prove that it is true even for probabilistic query complexity.
3. We prove that the query complexity of end-of-a-simple-path is $\exp(n)$.

Very Short Outline of the Proof

1. We reduce the $(2n)$-players ε-well supported Nash equilibrium problem to the problem of computing an approximate fixed point of an n-dimensional Lipschitz function.
   The reduction holds for constant values of ε!
2. We reduce the n-dimensional fixed point problem to the problem of finding end of a simple path on the n-dimensional hyper cube.
   Hirsch, Papadimitriou, and Vavasis [1989] proved that the deterministic query complexity of the n-dimensional fixed point problem is $\exp(n)$. We prove that it is true even for probabilistic query complexity.
3. We prove that the query complexity of end-of-a-simple-path is $\exp(n)$.
   Hart and Nisan [2013] proved that the query complexity of end-of-path is $\exp(n)$. We show that even if it is known that the path is simple the query complexity remain $\exp(n)$.

Distribution Queries

Question

What if the algorithm is allowed to ask distribution queries?

QUERIES: Each query is a distribution over action profiles $x \in \Delta(A)$. The answer is $u(x)$.

Distribution Queries

Question

What if the algorithm is allowed to ask distribution queries?

QUERIES: Each query is a distribution over action profiles $x \in \Delta(A)$. The answer is $u(x)$.

Answer

The query complexity remains exponential!

Distribution Queries

Question

What if the algorithm is allowed to ask distribution queries?

QUERIES: Each query is a distribution over action profiles $x \in \Delta(A)$. The answer is $u(x)$.

Answer

The query complexity remains exponential!

Idea: The payoff profile $u(x)$ can be very well approximated (with an error of e^{-cn}) using a sample of $poly(n)$ pure action profiles.

Distribution Queries

Question

What if the algorithm is allowed to ask distribution queries?

QUERIES: Each query is a distribution over action profiles $x \in \Delta(A)$. The answer is $u(x)$.

Answer

The query complexity remains exponential!

Idea: The payoff profile $u(x)$ can be very well approximated (with an error of e^{-cn}) using a sample of $poly(n)$ pure action profiles.

Approximate Nash equilibrium

Question

What about approximate Nash equilibrium (not well supported)?

Approximate Nash equilibrium

Question

What about approximate Nash equilibrium (not well supported)?

Daskalakis, Goldberg, and Papadimitriou [2005] introduced a computationally-efficient, and query-efficient ($poly(n)$) method for constructing an ε-well-supported Nash equilibrium from an $\frac{\varepsilon^2}{n}$-Nash equilibrium.

Approximate Nash equilibrium

Question

What about approximate Nash equilibrium (not well supported)?

Daskalakis, Goldberg, and Papadimitriou [2005] introduced a computationally-efficient, and query-efficient ($poly(n)$) method for constructing an ε-well-supported Nash equilibrium from an $\frac{\varepsilon^2}{n}$-Nash equilibrium.

Corollary

The query complexity of $\frac{c}{n}$-Nash equilibrium ($c = 10^{-16}$) in n-players games with constant number of actions ($m = 10^4$) is $\exp(n)$.

Consequences

Computational Complexity

This result provides evidence that existence of sub-exponential (in n) algorithm for approximate Nash equilibrium is very unlikely. If such an algorithm exists then it must depend on more complex data of the game than payoffs under distributions.

Consequences

Computational Complexity

This result provides evidence that existence of sub-exponential (in n) algorithm for approximate Nash equilibrium is very unlikely. If such an algorithm exists then it must depend on more complex data of the game than payoffs under distributions.

Query Complexity of Approximate Fixed Point

We generalize the $\exp(n)$ lower bound of [HPV 1989], from the case of deterministic algorithms to the case of probabilistic algorithms.

Consequences

Computational Complexity

This result provides evidence that existence of sub-exponential (in n) algorithm for approximate Nash equilibrium is very unlikely. If such an algorithm exists then it must depend on more complex data of the game than payoffs under distributions.

Query Complexity of Approximate Fixed Point

We generalize the $\exp(n)$ lower bound of [HPV 1989], from the case of deterministic algorithms to the case of probabilistic algorithms.

Open question from [HPV 1989]

What if the algorithm is allowed to ask queries which are distributions over the domain (rather than just points in the domain)?

Consequences

Computational Complexity

This result provides evidence that existence of sub-exponential (in n) algorithm for approximate Nash equilibrium is very unlikely. If such an algorithm exists then it must depend on more complex data of the game than payoffs under distributions.

Query Complexity of Approximate Fixed Point

We generalize the $\exp(n)$ lower bound of [HPV 1989], from the case of deterministic algorithms to the case of probabilistic algorithms.

Open question from [HPV 1989]

What if the algorithm is allowed to ask queries which are distributions over the domain (rather than just points in the domain)?

Answer

The query complexity remains $\exp(n)$.

Rate of Convergence of Adaptive Dynamics
Very useful tool for proving lower bounds on the rate of convergence of dynamics to equilibrium, is to study the complexity of equilibrium.

Consequences

Rate of Convergence of Adaptive Dynamics

Very useful tool for proving lower bounds on the rate of convergence of dynamics to equilibrium, is to study the complexity of equilibrium.

$\boxed{\text{Uncoupled Dynamics}} \leftrightarrow \boxed{\text{Communication Complexity}}$

Conitzer and Sandholm [2004]

Rate of Convergence of Adaptive Dynamics
Very useful tool for proving lower bounds on the rate of convergence of dynamics to equilibrium, is to study the complexity of equilibrium.

$\boxed{\text{Uncoupled Dynamics}} \leftrightarrow \boxed{\text{Communication Complexity}}$
Conitzer and Sandholm [2004]

Hart and Mansour [2010] used this idea to prove $\exp(n)$ lower bound on the rate of convergence of uncoupled dynamics to exact Nash equilibrium.
The question regarding the rate of convergence to approximate Nash equilibrium remain open.

$\boxed{k\text{-Queries Dynamics}} \leftrightarrow \boxed{\text{Query Complexity}}$

k-Queries Dynamics

$\boxed{k\text{-Queries Dynamics}} \leftrightarrow \boxed{\text{Query Complexity}}$

A dynamic is called k-queries dynamic if at each time t, k additional queries of payoffs are sufficient are sufficient in order to determine the mixed strategy of every player i at time $t + 1$.

k-Queries Dynamics

$\boxed{k\text{-Queries Dynamics}} \leftrightarrow \boxed{\text{Query Complexity}}$

A dynamic is called k-queries dynamic if at each time t, k additional queries of payoffs are sufficient are sufficient in order to determine the mixed strategy of every player i at time $t+1$.
Most of the studied dynamics are m-queries dynamics, where m is the number of actions of each player. Usually the mixed strategy of player i at time t depends on the set of payoffs $\{u_i(a_i, a_{-i}(t')) : a_i \in A_i, t' \in [t]\}$.

k-Queries Dynamics

$\boxed{k\text{-Queries Dynamics}} \leftrightarrow \boxed{\text{Query Complexity}}$

A dynamic is called k-queries dynamic if at each time t, k additional queries of payoffs are sufficient are sufficient in order to determine the mixed strategy of every player i at time $t+1$.
Most of the studied dynamics are m-queries dynamics, where m is the number of actions of each player. Usually the mixed strategy of player i at time t depends on the set of payoffs $\{u_i(a_i, a_{-i}(t')) : a_i \in A_i, t' \in [t]\}$.
Examples:

- Regret minimizing dynamics (regret matching, smooth fictitious play...).

k-Queries Dynamics

$\boxed{k\text{-Queries Dynamics}} \leftrightarrow \boxed{\text{Query Complexity}}$

A dynamic is called k-queries dynamic if at each time t, k additional queries of payoffs are sufficient are sufficient in order to determine the mixed strategy of every player i at time $t+1$.
Most of the studied dynamics are m-queries dynamics, where m is the number of actions of each player. Usually the mixed strategy of player i at time t depends on the set of payoffs $\{u_i(a_i, a_{-i}(t')) : a_i \in A_i, t' \in [t]\}$.
Examples:

- Regret minimizing dynamics (regret matching, smooth fictitious play...).
- Better reply dynamics (best-reply, log-it response...).

k-Queries Dynamics

$\boxed{k\text{-Queries Dynamics}} \leftrightarrow \boxed{\text{Query Complexity}}$

A dynamic is called k-queries dynamic if at each time t, k additional queries of payoffs are sufficient are sufficient in order to determine the mixed strategy of every player i at time $t+1$.
Most of the studied dynamics are m-queries dynamics, where m is the number of actions of each player. Usually the mixed strategy of player i at time t depends on the set of payoffs
$\{u_i(a_i, a_{-i}(t')) : a_i \in A_i, t' \in [t]\}$.
Examples:

- Regret minimizing dynamics (regret matching, smooth fictitious play...).
- Better reply dynamics (best-reply, log-it response...).
- Evolutionary dynamics (replicator dynamics, Smith dynamics...).

A lower bound to the rate of convergence to approximate well-supported Nash equilibrium, for quite general class of dynamics:

Corollary

For every k-queries dynamic where $k = poly(n)$ there exists an n-players m-actions game ($m = 10^4$) where it will take $\exp(n)$ steps in expectation to converge to an ε-well supported Nash equilibrium ($\varepsilon = 10^{-8}$).

Idea of the Proof of the Main Theorem

The reduction from Nash equilibrium to fixed point.

Proof of Brower's fixed-point Theorem using Nash's Theorem [Shmaya's blog 2012]

The reduction from Nash equilibrium to fixed point.

Proof of Brower's fixed-point Theorem using Nash's Theorem [Shmaya's blog 2012]

Given a mapping $f : [0,1]^n \to [0,1]^n$ we define 2-players game as follows:

$A_1 = A_2 = [0,1]^n$.

Idea of the Proof of the Main Theorem

The reduction from Nash equilibrium to fixed point.

Proof of Brower's fixed-point Theorem using Nash's Theorem [Shmaya's blog 2012]

Given a mapping $f : [0,1]^n \rightarrow [0,1]^n$ we define 2-players game as follows:

$A_1 = A_2 = [0,1]^n$.

$u_1(a_1, a_2) = -||a_1 - a_2||_2^2$.

Idea of the Proof of the Main Theorem

The reduction from Nash equilibrium to fixed point.

Proof of Brower's fixed-point Theorem using Nash's Theorem [Shmaya's blog 2012]

Given a mapping $f : [0,1]^n \to [0,1]^n$ we define 2-players game as follows:

$A_1 = A_2 = [0,1]^n$.
$u_1(a_1, a_2) = -||a_1 - a_2||_2^2$.
$u_2(a_1, a_2) = -||a_2 - f(a_1)||_2^2$.

The reduction from Nash equilibrium to fixed point.

Proof of Brower's fixed-point Theorem using Nash's Theorem [Shmaya's blog 2012]

Given a mapping $f : [0,1]^n \to [0,1]^n$ we define 2-players game as follows:

$A_1 = A_2 = [0,1]^n$.

$u_1(a_1, a_2) = -||a_1 - a_2||_2^2$.

$u_2(a_1, a_2) = -||a_2 - f(a_1)||_2^2$.

For every mixed strategy $x_2 \in \Delta(A_2)$ the unique best-reply of player 1 is $\mathbb{E}_{a \sim x_2}[a]$.

Idea of the Proof of the Main Theorem

The reduction from Nash equilibrium to fixed point.

Proof of Brower's fixed-point Theorem using Nash's Theorem [Shmaya's blog 2012]

Given a mapping $f : [0,1]^n \to [0,1]^n$ we define 2-players game as follows:

$A_1 = A_2 = [0,1]^n$.

$u_1(a_1, a_2) = -||a_1 - a_2||_2^2$.

$u_2(a_1, a_2) = -||a_2 - f(a_1)||_2^2$.

For every mixed strategy $x_2 \in \Delta(A_2)$ the unique best-reply of player 1 is $\mathbb{E}_{a \sim x_2}[a]$.

For every mixed strategy $x_1 \in \Delta(A_1)$ the unique best-reply of player 2 is $\mathbb{E}_{a \sim x_1}[f(a)]$.

Idea of the Proof of the Main Theorem

The reduction from Nash equilibrium to fixed point.

Proof of Brower's fixed-point Theorem using Nash's Theorem [Shmaya's blog 2012]

Given a mapping $f : [0,1]^n \to [0,1]^n$ we define 2-players game as follows:
$A_1 = A_2 = [0,1]^n$.
$u_1(a_1, a_2) = -||a_1 - a_2||_2^2$.
$u_2(a_1, a_2) = -||a_2 - f(a_1)||_2^2$.
For every mixed strategy $x_2 \in \Delta(A_2)$ the unique best-reply of player 1 is $\mathbb{E}_{a \sim x_2}[a]$.
For every mixed strategy $x_1 \in \Delta(A_1)$ the unique best-reply of player 2 is $\mathbb{E}_{a \sim x_1}[f(a)]$.
Therefore every Nash equilibrium of the game is pure.

Idea of the Proof of the Main Theorem

The reduction from Nash equilibrium to fixed point.

Proof of Brower's fixed-point Theorem using Nash's Theorem [Shmaya's blog 2012]

Given a mapping $f : [0,1]^n \to [0,1]^n$ we define 2-players game as follows:
$A_1 = A_2 = [0,1]^n$.
$u_1(a_1, a_2) = -||a_1 - a_2||_2^2$.
$u_2(a_1, a_2) = -||a_2 - f(a_1)||_2^2$.
For every mixed strategy $x_2 \in \Delta(A_2)$ the unique best-reply of player 1 is $\mathbb{E}_{a \sim x_2}[a]$.
For every mixed strategy $x_1 \in \Delta(A_1)$ the unique best-reply of player 2 is $\mathbb{E}_{a \sim x_1}[f(a)]$.
Therefore every Nash equilibrium of the game is pure.
If (a_1, a_2) is a pure Nash equilibrium then $a_1 = a_2$ and $a_2 = f(a_1)$, so $a_1 = f(a_1)$. ■

From Nash equilibrium to fixed point

A discrete version of the above game

Let $f : [0,1]^n \to [0,1]^n$ be a λ-Lipschitz mapping. We are interested in computing an ε-fixed point of f.

From Nash equilibrium to fixed point

A discrete version of the above game
Let $f : [0,1]^n \to [0,1]^n$ be a λ-Lipschitz mapping. We are interested in computing an ε-fixed point of f.

We define $(2n)$-player game where

- player $i \in [1, n]$ chooses the ith coordinate of a_1 from a finite grid $\{\frac{c}{k} : c \in [k]\}$,

From Nash equilibrium to fixed point

A discrete version of the above game
Let $f : [0,1]^n \to [0,1]^n$ be a λ-Lipschitz mapping. We are interested in computing an ε-fixed point of f.

We define $(2n)$-player game where

- player $i \in [1, n]$ chooses the ith coordinate of a_1 from a finite grid $\{\frac{c}{k} : c \in [k]\}$,
- player $n+i \in [n+1, 2n]$ chooses the ith coordinate of a_2 from a finite grid $\{\frac{c}{k} : c \in [k]\}$.

From Nash equilibrium to fixed point

A discrete version of the above game

Let $f : [0,1]^n \to [0,1]^n$ be a λ-Lipschitz mapping. We are interested in computing an ε-fixed point of f.

We define $(2n)$-player game where

- player $i \in [1, n]$ chooses the ith coordinate of a_1 from a finite grid $\{\frac{c}{k} : c \in [k]\}$,
- player $n + i \in [n + 1, 2n]$ chooses the ith coordinate of a_2 from a finite grid $\{\frac{c}{k} : c \in [k]\}$.

$k = \frac{\lambda+3}{\varepsilon}$ does not depend on n.

A discrete version of the above game

Let $f : [0,1]^n \to [0,1]^n$ be a λ-Lipschitz mapping. We are interested in computing an ε-fixed point of f.

We define $(2n)$-player game where

- player $i \in [1, n]$ chooses the ith coordinate of a_1 from a finite grid $\{\frac{c}{k} : c \in [k]\}$,
- player $n + i \in [n + 1, 2n]$ chooses the ith coordinate of a_2 from a finite grid $\{\frac{c}{k} : c \in [k]\}$.

$k = \frac{\lambda+3}{\varepsilon}$ does not depend on n.

Let $\varepsilon' = \frac{3\varepsilon^2}{4(\lambda+3)^2}$, ($\varepsilon'$ does not depend on n).

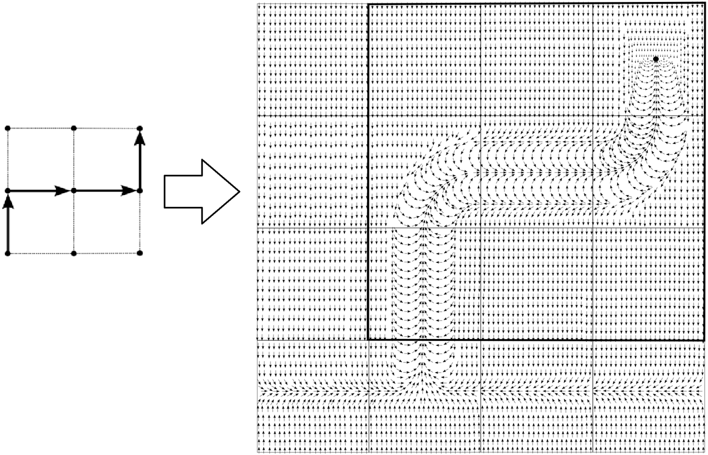
In every ε'-well-supported Nash equilibrium each player i plays at most 2 adjacent points on his grid with positive probability. All the actions in the support of the equilibrium are approximate fixed points of f.

From fixed point to end of path

Hirsch, Papadimitriou, and Vavasis introduced the following n-dimensional reduction from the problem of ε-fixed point of λ-Lipschitz function to the end of simple path on a grid.
The reduction holds for constant ε and λ.

From fixed point to end of path

Hirsch, Papadimitriou, and Vavasis introduced the following *n*-dimensional reduction from the problem of ε-fixed point of λ-Lipschitz function to the end of simple path on a grid. The reduction holds for constant ε and λ.



Open Questions

n-player games with constant number of actions m.

Open Questions

n-player games with constant number of actions m.

1. What is the query complexity of ε-Nash equilibrium (not well-supported), for constant ε?

Open Questions

n-player games with constant number of actions m.

1. What is the query complexity of ε-Nash equilibrium (not well-supported), for constant ε?
2. What is the communication complexity of ε-Nash equilibrium (well-supported or not)?

Open Questions

n-player games with constant number of actions m.

1. What is the query complexity of ε-Nash equilibrium (not well-supported), for constant ε?
2. What is the communication complexity of ε-Nash equilibrium (well-supported or not)?
3. What is the computation complexity of approximate Nash equilibrium?

Open Questions

n-player games with constant number of actions m.

1. What is the query complexity of ε-Nash equilibrium (not well-supported), for constant ε?
2. What is the communication complexity of ε-Nash equilibrium (well-supported or not)?
3. What is the computation complexity of approximate Nash equilibrium?

   $N = nm^n$ is the input size.

   - Evidence that probably sub-exponential (in n) algorithm does not exist.

Open Questions

n-player games with constant number of actions m.

1. What is the query complexity of ε-Nash equilibrium (not well-supported), for constant ε?
2. What is the communication complexity of ε-Nash equilibrium (well-supported or not)?
3. What is the computation complexity of approximate Nash equilibrium?

   $N = nm^n$ is the input size.

   - Evidence that probably sub-exponential (in n) algorithm does not exist.
   - $N^{\log N}$ algorithm exists [Lipton, Markakis, Mehta 2003].

Open Questions

n-player games with constant number of actions m.

1. What is the query complexity of ε-Nash equilibrium (not well-supported), for constant ε?
2. What is the communication complexity of ε-Nash equilibrium (well-supported or not)?
3. What is the computation complexity of approximate Nash equilibrium?

   $N = nm^n$ is the input size.

   - Evidence that probably sub-exponential (in n) algorithm does not exist.
   - $N^{\log N}$ algorithm exists [Lipton, Markakis, Mehta 2003].
   - $N^{\log \log N}$ algorithm exists [Daskalakis, Papadimitriou 2008], [Hemon, Rougemont, Santha 2008].

Open Questions

n-player games with constant number of actions m.

1. What is the query complexity of ε-Nash equilibrium (not well-supported), for constant ε?
2. What is the communication complexity of ε-Nash equilibrium (well-supported or not)?
3. What is the computation complexity of approximate Nash equilibrium?

   $N = nm^n$ is the input size.

   - Evidence that probably sub-exponential (in n) algorithm does not exist.
   - $N^{\log N}$ algorithm exists [Lipton, Markakis, Mehta 2003].
   - $N^{\log \log N}$ algorithm exists [Daskalakis, Papadimitriou 2008], [Hemon, Rougemont, Santha 2008].
   - $N^{\log \log \log N}$ algorithm exists [Babichenko, Peretz 2013].

Open Questions

n-player games with constant number of actions m.

1. What is the query complexity of ε-Nash equilibrium (not well-supported), for constant ε?
2. What is the communication complexity of ε-Nash equilibrium (well-supported or not)?
3. What is the computation complexity of approximate Nash equilibrium?

   $N = nm^n$ is the input size.

   - Evidence that probably sub-exponential (in n) algorithm does not exist.
   - $N^{\log N}$ algorithm exists [Lipton, Markakis, Mehta 2003].
   - $N^{\log \log N}$ algorithm exists [Daskalakis, Papadimitriou 2008], [Hemon, Rougemont, Santha 2008].
   - $N^{\log \log \log N}$ algorithm exists [Babichenko, Peretz 2013].

   Does there exist a $poly(N)$ algorithm?

Thank you!