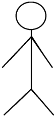
Historical Overview: the Complexity Theoretic Perspective on **PPAD** and Related Classes in **TFNP**

Mark Chen, Tianqi Yang

2024-04-11

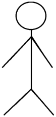
Why This Presentation



Before This Talk

TFNP Expert

Why This Presentation

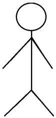


Before This Talk
I took a class on **TFNP** and crypto

$\Longrightarrow$

TFNP Expert

Why This Presentation



Before This Talk
I took a class on **TFNP** and crypto

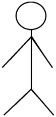
$\Longrightarrow$

TFNP Expert

$\Longleftarrow$

Then you must have learned about Nash equilibrium / hardness of gradient descent!

Why This Presentation



Before This Talk
I took a class on **TFNP** and crypto

TFNP Expert

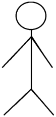
$\Longrightarrow$

What are these?

$\Longleftarrow$

Then you must have learned about Nash equilibrium / hardness of gradient descent!

Why This Presentation



Before This Talk
I took a class on **TFNP** and crypto

$\Longrightarrow$

TFNP Expert

What are these?

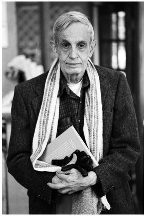
$\Longleftarrow$

Then you must have learned about Nash equilibrium / hardness of gradient descent!

Goal

***TFNP** subclasses, like **PPAD**, are in the title of a lot of the papers we saw. But we have so far focused on stronger results (like showing hardness in **SVL** instead of **PPAD**), which were summarized in Yizhi's talk. Historical context and motivation are important, though.*

Line of Works



John Nash

Game Theory

Foundational Works

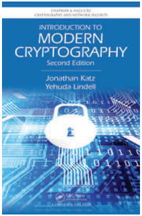
$\Longrightarrow$

C. Papadimitriou

Complexity

Theoretical Questions

$\Longrightarrow$



Entire Class

Cryptography

Specific Connections

Nash Equilibrium

Example, Concepts, and Existence

Example

	Coorporate	Defect
Coorporate	(2,2)	(3,0)
Defect	(0,3)	(1,1)

;

	Go	Stop
Go	(-3,-3)	(0,1)
Stop	(1,0)	(-1,-1)

Example

	Coorporate	Defect
Coorporate	(2,2)	(3,0)
Defect	(0,3)	(1,1)

;

	Go	Stop
Go	(-3,-3)	(0,1)
Stop	(1,0)	(-1,-1)

Some key concepts & assumptions:

- Equilibrium := a set of stable strategies where no individual player has the incentive to change their strategy.
- Equilibrium strategies of other players are known to everyone.

Nash Equilibrium

Example, Concepts, and Existence

Example

	Coorporate	Defect
Coorporate	(2,2)	(3,0)
Defect	(0,3)	(1,1)

;

	Go	Stop
Go	(-3,-3)	(0,1)
Stop	(1,0)	(-1,-1)

Some key concepts & assumptions:

- Equilibrium := a set of stable strategies where no individual player has the incentive to change their strategy.
- Equilibrium strategies of other players are known to everyone.

Theorem (Nash'51)

For every game, a **mixed Nash equilibrium**[a] *always exists (pure equilibrium, on the other hand, does not always exist; for example, rock-paper-scissors).*

[a] Mixed equilibrium is one where at least one player plays a randomized strategy.

Nash Equilibrium

As an algorithmic question

Theorem (Nash'51)

For every game, a mixed Nash equilibrium always exists.

Nash Equilibrium

As an algorithmic question

Theorem (Nash'51)

For every game, a mixed Nash equilibrium always exists.

Now that we have known the **existence** of Nash equilibrium since 70+ years ago, the key question left is:

Question

How **hard** *is it to compute the Nash equilibrium and how* **efficient** *can we make the computation?*

Not surprisingly, this gives us a **TFNP** problem (also efficiently verifiable, which is slightly harder to see for mixed strategy).

Nash Equilibrium

As an algorithmic question

Theorem (Nash'51)

For every game, a mixed Nash equilibrium always exists.

Now that we have known the **existence** of Nash equilibrium since 70+ years ago, the key question left is:

Question

*How **hard** is it to compute the Nash equilibrium and how **efficient** can we make the computation?*

Not surprisingly, this gives us a **TFNP** problem (also efficiently verifiable, which is slightly harder to see for mixed strategy).

Significance

*If we can indeed show that Nash equilibrium is **intractable**, Nash as a concept would be less useful as a ways to predict behaviors of players in the real world (since you can't do so efficiently; e.g., market prediction, etc.).*

Nash Equilibrium → Complexity Theoretic Question

Where (the heck) is it, then?

Remark

Again, by existence theorem, efficient verfiability and the search nature of the computation task (informally defined), **NASH** $\in$ **TFNP***.*

Nash Equilibrium → Complexity Theoretic Question

Where (the heck) is it, then?

Remark

Again, by existence theorem, efficient verfiability and the search nature of the computation task (informally defined), **NASH** $\in$ **TFNP**.

Theorem (Theorem 2.1 [MP91])

Recall NO problem in **TFNP** *is* **NP***-complete, unless* **NP** $=$ **coNP**

So, without the latter condition, it is unlikely to show **NASH** is **NP**-complete (though searching for **NASH** equilibrium with natural additional properties (*e.g.*, maximized sum of utility) could be **NP**-complete).

Nash Equilibrium → Complexity Theoretic Question

Where (the heck) is it, then?

Remark

Again, by existence theorem, efficient verfiability and the search nature of the computation task (informally defined), **NASH** $\in$ **TFNP***.*

Theorem (Theorem 2.1 [MP91])

Recall NO problem in **TFNP** *is* **NP***-complete, unless* **NP** $=$ **coNP**

So, without the latter condition, it is unlikely to show **NASH** is **NP**-complete (though searching for **NASH** equilibrium with natural additional properties (*e.g.*, maximized sum of utility) could be **NP**-complete). Can we show **NASH** is **TFNP**-complete?

Nash Equilibrium → Complexity Theoretic Question

Where (the heck) is it, then?

Remark

Again, by existence theorem, efficient verfiability and the search nature of the computation task (informally defined), **NASH** $\in$ **TFNP**.

Theorem (Theorem 2.1 [MP91])

Recall NO problem in **TFNP** *is* **NP***-complete, unless* **NP** $=$ **coNP**

So, without the latter condition, it is unlikely to show **NASH** is **NP**-complete (though searching for **NASH** equilibrium with natural additional properties (*e.g.*, maximized sum of utility) could be **NP**-complete). Can we show **NASH** is **TFNP**-complete?

Remark

TFNP *is* **unlikely** *to contain any complete problems*[a].

[a] Semantic vs. syntactic. *e.g.*, **TFNP** & **NP** $\cap$ **coNP** are both semantic.

Complexity Theoretic Question

Where (the heck) is it, then?

Iron Chef's

"just basic chemistry"

Nash Equilibrium

Where (the heck) is it, then?

Papa's
ground-laying idea

Nash Equilibrium

Where (the heck) is it, then?

Papa's
ground-laying idea

What if we, instead, find a subclass of **TFNP** problems (which is itself a subclass of **FNP**) based on the type of arguments used, where **NASH** is a complete problem[a]?

[a] Though they could have artificially defined a class of languages that reduce to **NASH** instead, defining based on types of totality arguments turned out to work very well as we have seen.

How It Started - Complexity Theoretic Question

Where (the heck) is **NASH**?

Theorem

[DGP09] As it turns out, **NASH** *is* **PPAD***-complete.*

*Note that, as a total problem, completeness must be shown through a two-way reduction, as all instances of total problems are guaranteed to be YES instances. Therefore, what we need to show is:

- **NASH** can be reduced to **EOTL**.
- **EOTL** can be reduced to **NASH**.

How It Started - Complexity Theoretic Question

Where (the heck) is **NASH**?

Theorem

[DGP09] As it turns out, **NASH** *is* **PPAD***-complete.*

*Note that, as a total problem, completeness must be shown through a two-way reduction, as all instances of total problems are guaranteed to be YES instances. Therefore, what we need to show is:

- **NASH** can be reduced to **EOTL**.
- **EOTL** can be reduced to **NASH**.

Corollary

We have spent weeks constructing **SVL** *and* **rSVL** *hard instances from cryptographic assumptions, which easily translate to hardness in* **PPAD***. So, by completeness, all of such previous hardness results imply hardness in finding* **NASH** *equilibrium.*

Preliminaries of [DGP09]

We first define **NASH** and **Approximate-NASH** formally.

Definition ((Mixed) **NASH**)

Here are the set-ups:

- There are k players, and $p \in [k]$ denotes one of the players.
- Let S_p be a finite set of **strategies** that p can take, then $S = \prod_{p \in [k]} \mathbf{S}_p$ (Cartesian product). S is called **strategy profiles**.
- Let S_{-p} be the set of pure strategies of players other than p. Then, the **payoff** to p when p takes $s \in S_p$ and the other players take $s' \in S_{-p}$ is denoted by $u^p_{ss'} \geq 0$.

Now, let x^p_s denote the probability of p taking $s \in S_p$, finding **NASH** is the restraint problem:

$$x^p_s \geq 0 \text{ and } \sum_{s \in S_p} x^p_j = 1.$$

Frame Title

Definition ((Mixed) NASH, Continued)

Then, a k-mixed strategies is a **NASH** equilibrium if

$$\sum_{s \in S} u_s^p x_s \text{ is maximized } \forall p; x_s = \prod_{p \in [k]} x_{s_p}^p.$$

Frame Title

Definition ((Mixed) **NASH**, Continued)

Then, a k-mixed strategies is a **NASH** equilibrium if

$$\sum_{s \in S} u_s^p x_s \text{ is maximized } \forall p; x_s = \prod_{p \in [k]} x_{s_p}^p.$$

Or, equivalently,

$$\sum_{s' \in S_{-p}} u_{ss'}^p x_{s'} > \sum_{s' \in S_{-p}} u_{s^\circ s'}^p x_{s'} \implies x_{s^\circ}^p = 0.$$

Preliminaries of [DGP09]

Definition (Approximate-NASH)

A set of mixed strategies x is an ϵ-**NASH** equilibrium if (with everything else the same):

$$\sum_{s' \in S_{-p}} u^p_{ss'} x_{s'} > \sum_{s' \in S_{-p}} u^p_{s^\circ s'} x_{s'} + \epsilon \implies x^p_{s^\circ} = 0.$$

Let's give some more intuition about this:

Remark

NASH *can be taken to mean requiring "no incentive to deviate," while* **Approximate-NASH** *is to require "low incentive to deviate." Say, if $\epsilon > 0$ is small, and then ϵ-***NASH** *equilibrium is a profile of mixed strategies where any player can improve its expected payoff by at most ϵ by switching to another strategy.*

Main Theorem

Theorem (Theorem 3.1 [DGP09])

NASH *is* **PPAD***-complete.*

Main Theorem

Theorem (Theorem 3.1 [DGP09])

NASH *is* **PPAD**-*complete.*

Recall the following idea, which is useful for reductions in both directions:

Theorem (Brouwer's Fixed Points Theorem)

Any continuous map from a compact and convex subset of the Euclidean space into itself always has a fixed point (one cannot map a circle continuously [rotate, flip, shrink and stretch] on itself without keeping some point fixed). A natural search problem is to find this point.

Remark

Notice that for the problem to be tractable still, we need the point to be rational, and that is how we use the ϵ-approximate introduced for **NASH**.

Main Theorem

Theorem (Theorem 3.1 [DGP09])

NASH *is* **PPAD***-complete.*

Recall the following idea, which is useful for reductions in both directions:

Theorem (Brouwer's Fixed Points Theorem)

Any continuous map from a compact and convex subset of the Euclidean space into itself always has a fixed point (one cannot map a circle continuously [rotate, flip, shrink and stretch] on itself without keeping some point fixed). A natural search problem is to find this point.

Remark

Notice that for the problem to be tractable still, we need the point to be rational, and that is how we use the ϵ-approximate introduced for **NASH***.*

Definition (**BROUWER**(Π_F, K, ϵ))

Let Π_F be an efficient algorithm for the evaluation of $F : [0,1]^m \to [0,1]^m$. Let K be a constant so that F satisfies Lipschitz continuity:

$$\forall x_1, x_2 \in [0,1]^m : d(F(x_1), F(x_2)) \leq K \cdot d(x_1, x_2).$$

Let ϵ be the desired accuracy. Then, the search problem wants to output x such that $d(F(x), x) \leq \epsilon$.

Main Theorem Proof Overview

NASH $\longrightarrow$ **Approximate-NASH** $\longrightarrow$ **BROUWER** (**PPAD**-complete)

Main Theorem Direction 1 (Pre): **BROUWER** $\in$ **PPAD**

Theorem

BROUWER $\in$ **PPAD** *[Pap94]*.

- Triangulate the domain of F (fill up the domain with a mesh of tiny triangles and each triangle is a vertex of the graph).
- Color the vertices according to the direction in which F displaces them.
- Edges are defined with respect to the colors of the vertices.

Main Theorem Direction 1 (Pre): **BROUWER** $\in$ **PPAD**

Theorem

BROUWER $\in$ **PPAD** *[Pap94]*.

- Triangulate the domain of F (fill up the domain with a mesh of tiny triangles and each triangle is a vertex of the graph).
- Color the vertices according to the direction in which F displaces them.
- Edges are defined with respect to the colors of the vertices.

The idea is that the mesh and its coloring is set up in such a way that if the vertices of a triangle get all possible colors, then F shifts these vertices in conflicting directions, which means we are close to an approximate fixed point.

Main Theorem Direction 1 (Pre): **BROUWER** $\in$ **PPAD**

Theorem

BROUWER $\in$ **PPAD** *[Pap94]*.

- Triangulate the domain of F (fill up the domain with a mesh of tiny triangles and each triangle is a vertex of the graph).
- Color the vertices according to the direction in which F displaces them.
- Edges are defined with respect to the colors of the vertices.

The idea is that the mesh and its coloring is set up in such a way that if the vertices of a triangle get all possible colors, then F shifts these vertices in conflicting directions, which means we are close to an approximate fixed point.

By a combinatorial argument called the **Sperner's lemma**, at least one triangle would satisfy this.

Main Theorem Direction 1: **NASH** $\in$ **PPAD**

Proof.

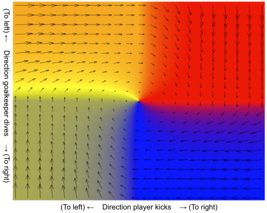
Suffices to show that **Approximate-NASH** $\leq$ **BROUWER**, which was first shown by Nash in 1950. Suppose the players in a game have chosen some mixed strategies. Unless the strategies are already at a Nash equilibrium, at least one of the players will be unsatisfied and will want to change to some other strategies.

Main Theorem Direction 1: **NASH** $\in$ **PPAD**

Proof.

Suffices to show that **Approximate-NASH** $\leq$ **BROUWER**, which was first shown by Nash in 1950. Suppose the players in a game have chosen some mixed strategies. Unless the strategies are already at a Nash equilibrium, at least one of the players will be unsatisfied and will want to change to some other strategies.

We see it as a "preference function" from the set of players' strategies to itself.



Magnitudes & directions are determined by $F_N(x) - x$, where $F_N(x)$ is Nash's function as a preference function for penalty shot game.

Clearly, **Approximate-NASH** equilibrium would be a ϵ-fixed point, and Brouwer's fixed point theorem guarantees its existence, **Approximate-NASH** $\equiv$ finding an approximate fixed point $\implies$ **NASH** $\in$ **PPAD**. ∎

Main Theorem Direction 2: **NASH** is **PPAD**-complete

Proof.

Similarly, we first show that **BROUWER** is **PPAD**-complete and then reduce **BROUWER** to **NASH**.

- (**BROUWER** is **PPAD**-complete): Need to show how to encode a **EOTL** graph as a continuous, easy-to-compute function F. This is non-trivial to show, but is given entirely in [DGP09].

Main Theorem Direction 2: **NASH** is **PPAD**-complete

Proof.

Similarly, we first show that **BROUWER** is **PPAD**-complete and then reduce **BROUWER** to **NASH**.

- (**BROUWER** is **PPAD**-complete): Need to show how to encode a **EOTL** graph as a continuous, easy-to-compute function F. This is non-trivial to show, but is given entirely in [DGP09].
- (Reduce **BROUWER** to **NASH**): $F \in$ **BROUWER** can be efficiently computed using arithmetic circuits built up using a small basis of operators.

Main Theorem Direction 2: **NASH** is **PPAD**-complete

Proof.

Similarly, we first show that **BROUWER** is **PPAD**-complete and then reduce **BROUWER** to **NASH**.

- (**BROUWER** is **PPAD**-complete): Need to show how to encode a **EOTL** graph as a continuous, easy-to-compute function F. This is non-trivial to show, but is given entirely in [DGP09].
- (Reduce **BROUWER** to **NASH**): $F \in$ **BROUWER** can be efficiently computed using arithmetic circuits built up using a small basis of operators. We can write such circuits as a as data flow graph, with one of these small set of operators at each node. Then,
  - We let players be on every node on this data flow graph.
  - Thus, we simulate each arithmetic gate in the circuit by a game.
  - We compose the games to get the overall game.

  The specific ways to do so is non-trivial and are given entirely in [DGP09].

Main Theorem Direction 2: **NASH** is **PPAD**-complete

Proof.

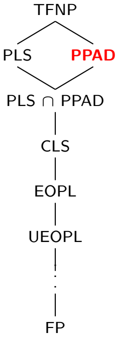
Similarly, we first show that **BROUWER** is **PPAD**-complete and then reduce **BROUWER** to **NASH**.

- (**BROUWER** is **PPAD**-complete): Need to show how to encode a **EOTL** graph as a continuous, easy-to-compute function F. This is non-trivial to show, but is given entirely in [DGP09].
- (Reduce **BROUWER** to **NASH**): $F \in$ **BROUWER** can be efficiently computed using arithmetic circuits built up using a small basis of operators. We can write such circuits as a as data flow graph, with one of these small set of operators at each node. Then,
  - We let players be on every node on this data flow graph.
  - Thus, we simulate each arithmetic gate in the circuit by a game.
  - We compose the games to get the overall game.

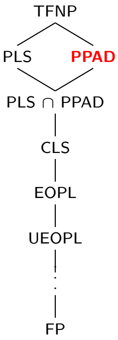
  The specific ways to do so is non-trivial and are given entirely in [DGP09].

Note: Due to methods used, this only proves $k \geq 3$.

Summary of **PPAD**-complete problems we know now



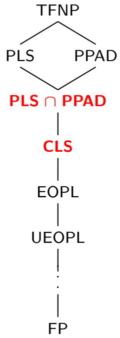
Summary of **PPAD**-complete problems we know now



- **EOTL**.
- **NASH**.
- **BROUWER**.

Next, **CLS** $\in$ **PLS** $\cap$ **PPAD**

Now, we go deeper:



An Old Complete Problems in **PLS** $\cap$ **PPAD**

There was a reason to ask about optimizing for *continuous* functions, since the complete problems known back in the day [DP11] have, so to speak, an awkward flavor. Here's the general formula:

Example (**PPAD-OR-PLS**)

Given an instance $X \in$ **PPAD** and an instance $Y \in$ **PLS**.

- Either solve X.
- Or solve Y.

An Old Complete Problems in **PLS** $\cap$ **PPAD**

There was a reason to ask about optimizing for *continuous* functions, since the complete problems known back in the day [DP11] have, so to speak, an awkward flavor. Here's the general formula:

Example (PPAD-OR-PLS)

Given an instance $X \in$ **PPAD** and an instance $Y \in$ **PLS**.

- Either solve X.
- Or solve Y.

Example (EITHER-FIXEDPOINT)

Let $\epsilon, \delta > 0$. Given three functions f, g and p. Here is the goal:

- Either find an approximate fixed point of f (or violation of f's δ-continuity).
- Or an approximate fixed point of g w.r.t. p (or violation of p's δ-continuity).

The reason why this appears awkward is because it is about finding fixed points of two unrelated functions.

CLS as a New Attempt; CLS $\subseteq$ PLS $\cap$ PPAD

So the question of interest at the time was: can we let f, g coincide in a single function, to make it more natural?

CLS as a New Attempt; CLS $\subseteq$ PLS $\cap$ PPAD

So the question of interest at the time was: can we let f, g coincide in a single function, to make it more natural? Here's an example:

Definition (CONTINUOUS-LOCALOPT)

Let $p : [2^n] \to \mathbb{R}$ and $f : [2^n] \to [2^n]$. Goal is to find $v \in [2^n]$ such that

- $p(f(v)) \geq p(v)$, or
- find a violation of continuity for f and/or p.

You can think of f as the successor function on a DAG line and p (potential) as the value of a node. Then, this condition is a search for the first node with minimal potential, or a sink.

CLS as a New Attempt; $\mathbf{CLS} \subseteq \mathbf{PLS} \cap \mathbf{PPAD}$

So the question of interest at the time was: can we let f, g coincide in a single function, to make it more natural? Here's an example:

Definition (CONTINUOUS-LOCALOPT)

Let $p : [2^n] \to \mathbb{R}$ and $f : [2^n] \to [2^n]$. Goal is to find $v \in [2^n]$ such that

- $p(f(v)) \geq p(v)$, or
- find a violation of continuity for f and/or p.

You can think of f as the successor function on a DAG line and p (potential) as the value of a node. Then, this condition is a search for the first node with minimal potential, or a sink. So, this problem, which defines **CLS**, has a $\mathbf{PLS} \cap \mathbf{PPAD}$ flavor with an additional condition about continuity (not hard to show $\mathbf{CLS} \subseteq \mathbf{PLS} \cap \mathbf{PPAD}$).

Conjecture ([DP11], which is disproved by [FGHS22])

$\mathbf{CLS} \subsetneq \mathbf{PLS} \cap \mathbf{PPAD}$ *(because* **PLS** *and* **PPAD** *in general requires no continuity).*

CLS: **PPAD** $\cap$ **PLS** but with f, p Related to Each Other

In this definition f and p can actually be related!

CLS: **PPAD** $\cap$ **PLS** but with f, p Related to Each Other

In this definition f and p can actually be related! One very natural such attempt is by using a Lipschitz continuous function f and its derivative $p = \lim_{\triangle x \to 0} \frac{f(x+\triangle x) - f(x)}{\triangle x}$. Here is how:

- If f is Lipschitz, then a fixed point must exist $|f(x) - x| = 0$, so it is in **PPAD**.
- Since f is Lipschitz continuous, its derivative must have the following property too: for some $\triangle x > 0$, $|f(x + \triangle x) - f(x)| < \epsilon$, which captures the definition for **PLS**.

CLS = PLS ∩ PPAD

Recall conjecture:

Conjecture ([DP11])

It was conjectured that **CLS** $\subsetneq$ **PLS** ∩ **PPAD**.

$\mathbf{CLS} = \mathbf{PLS} \cap \mathbf{PPAD}$

Recall conjecture:

Conjecture ([DP11])

It was conjectured that $\mathbf{CLS} \subsetneq \mathbf{PLS} \cap \mathbf{PPAD}$.

Next, we present the result that disproves this conjecture, i.e.

Theorem

$\mathbf{CLS} = \mathbf{PLS} \cap \mathbf{PPAD}$.

CLS = PLS $\cap$ PPAD: Preliminaries

In [FGHS22], it was shown that **GRADIENT-DESCENT**, which is a **CLS**-complete problem, is actually also (**PPAD** $\cap$ **PLS**)-complete.

CLS = **PLS** ∩ **PPAD**: Preliminaries

In [FGHS22], it was shown that **GRADIENT-DESCENT**, which is a **CLS**-complete problem, is actually also (**PPAD** ∩ **PLS**)-complete.

Definition (Gradient Descent)

Given a circuit for f and ∂f. Search for an extremum (minimum in particular) of a continuously differentiable function f over some domain D by starting at x_0 and iteratively as:

$$x_{k+1} \leftarrow x_k - \eta \nabla f(x_k).$$

Definition (Karush-Kuhn-Tucker (KKT) Optimality Condition)

Roughly, KKT optimality conditions assert that

- the gradient of f is 0 at x or
- on the boundary of D.

CLS = PLS ∩ PPAD: Preliminaries

In [FGHS22], it was shown that **GRADIENT-DESCENT**, which is a **CLS**-complete problem, is actually also (**PPAD** ∩ **PLS**)-complete.

Definition (Gradient Descent)

Given a circuit for f and ∂f. Search for an extremum (minimum in particular) of a continuously differentiable function f over some domain D by starting at x_0 and iteratively as:

$$x_{k+1} \leftarrow x_k - \eta \nabla f(x_k).$$

Definition (Karush-Kuhn-Tucker (KKT) Optimality Condition)

Roughly, KKT optimality conditions assert that

- the gradient of f is 0 at x or
- on the boundary of D.

Remark

Why does this resemble **PLS** *and* **PPAD** *problems?*

CLS $=$ **PLS** $\cap$ **PPAD**: Complete Search Problem

Definition (Set-Up)

Let $\epsilon, \eta > 0$, domain be D, $f \in C^1_L(D, \mathbb{R})$ (note ∇f is the gradient of f). The goal is to compute a point where gradient descent for f on D terminates.

CLS = PLS ∩ PPAD: Complete Search Problem

Definition (Set-Up)

Let $\epsilon, \eta > 0$, domain be D, $f \in C^1_L(D, \mathbb{R})$ (note ∇f is the gradient of f). The goal is to compute a point where gradient descent for f on D terminates.

Termination is determined using optimality conditions. In particular, if $x \in D$ and $x' = \prod_D (x - \eta \nabla f(x))$, then gradient descent should terminate if one of the following is found:

Definition (GD-LOCAL-SEARCH)

$f(x') \geq f(x) - \epsilon$. ϵ-approximate of the local minimum (**PLS**).

Definition (GD-FIXEDPOINT)

$|x - x'| - \epsilon$. ϵ-approximate of x-fixed point (**PPAD**).

Main Results of the Paper

Recall that GRADIENT-DESCENT as a search problem is defined by searching for either GD-LOCAL-SEARCH or GD-FIXEDPOINT. It turns out the following are true:

Theorem

Given a $f \in C^1_L(D, \mathbb{R})$ and its derivative as circuits, optimizing through GRADIENT-DESCENT is complete for **PLS** $\cap$ **PPAD**.

Theorem

The same problem is also complete for **CLS**.

Main Results of the Paper

Recall that GRADIENT-DESCENT as a search problem is defined by searching for either GD-LOCAL-SEARCH or GD-FIXEDPOINT. It turns out the following are true:

Theorem

Given a $f \in C^1_L(D, \mathbb{R})$ and its derivative as circuits, optimizing through GRADIENT-DESCENT is complete for **PLS** $\cap$ **PPAD**.

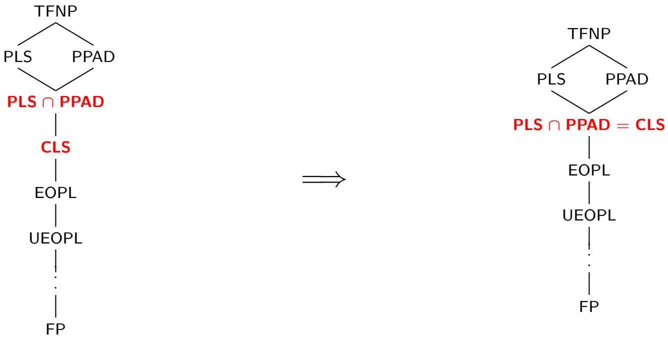
Theorem

The same problem is also complete for **CLS**.

Corollary

CLS $=$ **PLS** $\cap$ **PPAD**. *So, with the weeks of results that we saw for constructing hard* **SVL** *and* **rSVL** *instances from various crypto assumptions, then all imply hardness in* **PLS** $\cap$ **PPAD**, *and so* **CLS** *and the GRADIENT-DESCENT problem.*

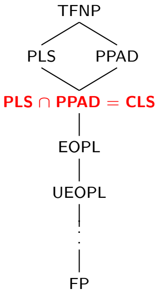
Thus, First Collapse



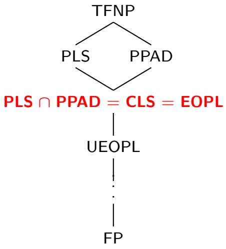
Now, We Have a Second Collapse [GHJ+22]

Theorem (Theorem 1 [GHJ+22])

EOPL $=$ **PLS** $\cap$ **PPAD**.



$\Longrightarrow$



End-of-Potential-Line (**EOPL**)

Recall what an **EOPL** is:

Definition (**EOPL**(S, P, x_0, p))

G is a DAG that is succinctly defined ($|V| = 2^n$) with in/out-degree of at most 1 [can be though of as a disjoint union of directed lines].

End-of-Potential-Line (**EOPL**)

Recall what an **EOPL** is:

Definition (**EOPL**(S, P, x_0, p))

G is a DAG that is succinctly defined ($|V| = 2^n$) with in/out-degree of at most 1 [can be though of as a disjoint union of directed lines].

Starting from x_0 (or, really, any node), we can compute its successor using S circuit, predecessor using P circuit. At any x, we can also compute its potential, $p(x)$, which is guaranteed to increase along the directed line.

End-of-Potential-Line (**EOPL**)

Recall what an **EOPL** is:

Definition (**EOPL**(S, P, x_0, p))

G is a DAG that is succinctly defined ($|V| = 2^n$) with in/out-degree of at most 1 [can be though of as a disjoint union of directed lines].

Starting from x_0 (or, really, any node), we can compute its successor using S circuit, predecessor using P circuit. At any x, we can also compute its potential, $p(x)$, which is guaranteed to increase along the directed line.

Goal is to find any source or sink that is not x_0.

End-of-Potential-Line (**EOPL**)

Recall what an **EOPL** is:

Definition (**EOPL**(S, P, x_0, p))

G is a DAG that is succinctly defined ($|V| = 2^n$) with in/out-degree of at most 1 [can be though of as a disjoint union of directed lines].

Starting from x_0 (or, really, any node), we can compute its successor using S circuit, predecessor using P circuit. At any x, we can also compute its potential, $p(x)$, which is guaranteed to increase along the directed line.

Goal is to find any source or sink that is not x_0.

Remark

EOPL $\subseteq$ **PPAD** $\cap$ **PLS** *should be intuitive, as it is equipped with* S, P *and* p *[***EOTL** *(***PPAD***-complete) is equipped with* S, P*, and SINK-OF-DAG (***PLS***-complete) is equipped with* p, S *and we can arbitrarily define* C *to never be violated].*

Theorem (**EOPL** = **CLS** = **PLS** ∩ **PPAD** [GHJ+22])

Proof.

Theorem 1 [GHJ+22]

$$\textbf{EOPL} \longrightarrow \textbf{CLS} = \textbf{PLS} \cap \textbf{PPAD}$$

∎

Theorem (**EOPL** = **CLS** = **PLS** ∩ **PPAD** [GHJ+22])

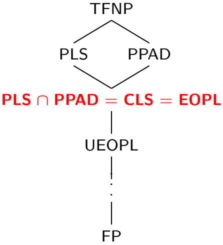
Proof.

Theorem 1 [GHJ+22]

EOPL ⟶ **CLS** = **PLS** ∩ **PPAD**

∎

So, the current state:



[DGP09] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou.
The complexity of computing a nash equilibrium.
Communications of the ACM, 52(2):89–97, 2009.

[DP11] Constantinos Daskalakis and Christos Papadimitriou.
Continuous local search.
In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 790–804. SIAM, 2011.

[FGHS22] John Fearnley, Paul Goldberg, Alexandros Hollender, and Rahul Savani.
The complexity of gradient descent: Cls= ppad pls.
Journal of the ACM, 70(1):1–74, 2022.

[GHJ+22] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao.
Further collapses in tfnp.

arXiv preprint arXiv:2202.07761, 2022.

[MP91] Nimrod Megiddo and Christos H. Papadimitriou.
On total functions, existence theorems and computational complexity.
Theoretical Computer Science, 81(2):317–324, 1991.

[Pap94] Christos H Papadimitriou.
On the complexity of the parity argument and other inefficient proofs of existence.
Journal of Computer and system Sciences, 48(3):498–532, 1994.