The Complexity of Gradient Descent: CLS = PPAD ∩ PLS

ALEXANDROS HOLLENDER

JOINT WORK WITH JOHN FEARNLEY, PAUL GOLDBERG AND RAHUL SAVANI
Some interesting computational problems
Some interesting computational problems

**NASH:**
Find a mixed Nash equilibrium of a game.
Some interesting computational problems

**NASH:**
Find a mixed Nash equilibrium of a game.

**FACTORING:**
Find a prime factor of a number \( n \geq 2 \).
Some interesting computational problems

**NASH:**
Find a mixed Nash equilibrium of a game.

**FACTORIZING:**
Find a prime factor of a number $n \geq 2$.

**BROUWER:**
Find a fixpoint of a continuous function $f : [0,1]^3 \rightarrow [0,1]^3$. 
Some interesting computational problems

**NASH:**
Find a mixed Nash equilibrium of a game.

**FACTORING:**
Find a prime factor of a number $n \geq 2$.

**BROUWER:**
Find a fixpoint of a continuous function $f : [0,1]^3 \to [0,1]^3$.

**CONTRACTION:**
Find the unique fixpoint of a contraction $f : [0,1]^n \to [0,1]^n$. 
Some interesting computational problems

**NASH:**
Find a mixed Nash equilibrium of a game.

**FACTORING:**
Find a prime factor of a number $n \geq 2$.

**BROUWER:**
Find a fixpoint of a continuous function $f : [0,1]^3 \rightarrow [0,1]^3$.

**CONTRACTION:**
Find the unique fixpoint of a contraction $f : [0,1]^n \rightarrow [0,1]^n$.

**PURE-CONGESTION:**
Find a pure Nash equilibrium of a congestion game.
Some interesting computational problems

**NASH:**
Find a mixed Nash equilibrium of a game.

**FACTORING:**
Find a prime factor of a number \( n \geq 2 \).

**BROUWER:**
Find a fixpoint of a continuous function \( f: [0,1]^3 \rightarrow [0,1]^3 \).

**CONTRACTION:**
Find the unique fixpoint of a contraction \( f: [0,1]^n \rightarrow [0,1]^n \).

**PURE-CONGESTION:**
Find a pure Nash equilibrium of a congestion game.

What do these problems have in common?
Some interesting computational problems

NASH:
Find a mixed Nash equilibrium of a game.

FACTORING:
Find a prime factor of a number $n \geq 2$.

BROUWER:
Find a fixpoint of a continuous function $f : [0,1]^3 \rightarrow [0,1]^3$.

CONTRACTION:
Find the unique fixpoint of a contraction $f : [0,1]^n \rightarrow [0,1]^n$.

PURE-CONGESTION:
Find a pure Nash equilibrium of a congestion game.

What do these problems have in common?
They are NP Total Search (TFNP) problems!
- Total: there is always a solution
- NP: it is easy to verify solutions
Some interesting computational problems

**NASH:**
Find a mixed Nash equilibrium of a game.

**FACTORING:**
Find a prime factor of a number \( n \geq 2 \).

**BROUWER:**
Find a fixpoint of a continuous function \( f: [0,1]^3 \rightarrow [0,1]^3 \).

**CONTRACTION:**
Find the unique fixpoint of a contraction \( f: [0,1]^n \rightarrow [0,1]^n \).

**PURE-CONGESTION:**
Find a pure Nash equilibrium of a congestion game.

What do these problems have in common?
They are NP Total Search (TFNP) problems!
- Total: there is always a solution
- NP: it is easy to verify solutions

Can a TFNP problem be NP-hard?
Some interesting computational problems

NASH:
Find a mixed Nash equilibrium of a game.

FACTORING:
Find a prime factor of a number \( n \geq 2 \).

BROUWER:
Find a fixpoint of a continuous function \( f: [0,1]^3 \rightarrow [0,1]^3 \).

CONTRACTION:
Find the unique fixpoint of a contraction \( f: [0,1]^n \rightarrow [0,1]^n \).

PURE-CONGESTION:
Find a pure Nash equilibrium of a congestion game.

What do these problems have in common?
They are NP Total Search (TFNP) problems!
• Total: there is always a solution
• NP: it is easy to verify solutions

Can a TFNP problem be NP-hard?
Not unless co-NP = NP...
The class TFNP [Megiddo-Papadimitriou, 1991]

Total NP search problems:

• “search”: looking for a solution, not just YES or NO
• “NP”: any solution can be checked efficiently
• “total”: there always exists at least one solution
The class TFNP [Megiddo-Papadimitriou, 1991]

Total NP search problems:

- “search”: looking for a solution, not just YES or NO
- “NP”: any solution can be checked efficiently
- “total”: there always exists at least one solution

TFNP lies between P and NP (search versions)
The class TFNP [Megiddo-Papadimitriou, 1991]

Total NP search problems:
- “search”: looking for a solution, not just YES or NO
- “NP”: any solution can be checked efficiently
- “total”: there always exists at least one solution

How do we show that a TFNP-problem is hard:
The class TFNP [Megiddo-Papadimitriou, 1991]

Total NP search problems:
- “search” : looking for a solution, not just YES or NO
- “NP”: any solution can be checked efficiently
- “total”: there always exists at least one solution

How do we show that a TFNP-problem is hard:
- No TFNP-problem can be NP-hard, unless NP = coNP...
The class TFNP [Megiddo-Papadimitriou, 1991]

Total NP search problems:

• “search” : looking for a solution, not just YES or NO
• “NP”: any solution can be checked efficiently
• “total”: there always exists at least one solution

How do we show that a TFNP-problem is hard:

▪ No TFNP-problem can be NP-hard, unless NP = coNP...

\[ 3\text{-SAT} \leq \text{NASH} \implies \text{certificate for unsatisfiable 3-SAT formulas} \]
The class TFNP [Megiddo-Papadimitriou, 1991]

Total NP search problems:

• “search”: looking for a solution, not just YES or NO
• “NP”: any solution can be checked efficiently
• “total”: there always exists at least one solution

How do we show that a TFNP-problem is hard:

▪ No TFNP-problem can be NP-hard, unless NP = coNP...
The class TFNP [Megiddo-Papadimitriou, 1991]

Total NP search problems:
• “search” : looking for a solution, not just YES or NO
• “NP” : any solution can be checked efficiently
• “total” : there always exists at least one solution

How do we show that a TFNP-problem is hard:
- No TFNP-problem can be NP-hard, unless NP = coNP...
- Believed that no TFNP-complete problems exists...
The TFNP landscape
The TFNP landscape

Pigeonhole Principle

TFNP

PPP

P
The TFNP landscape

Pigeonhole Principle

Parity Argument

Borsuk-Ulam

TFNP

PPA

PPP

P
The TFNP landscape

- Pigeonhole Principle
- Parity Argument
- Borsuk-Ulam
- Local Search Argument
- PURE-CONGESTION
- LOCAL-MAX-CUT
The TFNP landscape

- Pigeonhole Principle
- Directed Graph Argument
- NASH BROUWER
- P
- Parity Argument
  - Borsuk-Ulam
- Local Search Argument
  - PURE-CONGESTION
  - LOCAL-MAX-CUT

TFNP

PPA

PPP

PPAD

PLS
The TFNP landscape

- Pigeonhole Principle
- Directed Graph Argument
- NASH
- BROUWER
- Parity Argument
- Borsuk-Ulam
- Local Search Argument
- PURE-CONGESTION
- LOCAL-MAX-CUT

TFNP

PPA

FACTORIZING

PPP

PPAD

PLS

P
TFNP subclasses

What reasons are there to believe that PPAD ≠ P, PLS ≠ P, etc?
TFNP subclasses

What reasons are there to believe that PPAD $\neq P$, PLS $\neq P$, etc?

- many seemingly hard problems lie in PPAD, PLS etc...
What reasons are there to believe that PPAD ≠ P, PLS ≠ P, etc?

- many seemingly hard problems lie in PPAD, PLS etc...
- oracle separations between the classes (in particular PPAD ≠ PLS)
What reasons are there to believe that PPAD ≠ P, PLS ≠ P, etc?

- many seemingly hard problems lie in PPAD, PLS etc...
- oracle separations between the classes (in particular PPAD ≠ PLS)
- hard under cryptographic assumptions
TFNP

PPAD

BROUWER  NASH

PPAD \cap PLS

CONTRACTION
MIXED-CONGESTION

PLS

LOCAL-MAX-CUT
PURE-CONGESTION

P
TFNP

PPAD
BROUWER NASH

PPAD \cap PLS
CONTRACTION
MIXED-CONGESTION
P-LCP

PLS
LOCAL-MAX-CUT
PURE-CONGESTION

P
TFNP

\[
\text{PPAD} \cap \text{PLS} \\
\text{BROUWER} \quad \text{NASH} \\
\text{CONTRACTION} \\
\text{MIXED-CONGESTION} \\
\text{P-LCP} \\
\text{TARSKI} \\
\text{P} \\
\text{LOCAL-MAX-CUT} \\
\text{PURE-CONGESTION}
\]
TFNP

PPAD
- Brouwer
- Nash

PPAD \cap PLS
- Contraction
- Mixed-congestion
- P-LCP
- SSGs
- Tarski

P

PLS
- Local-max-cut
- Pure-congestion
PPAD ∩ PLS seems unnatural...
PPAD \cap PLS seems unnatural...

Problem $A$ : PPAD-complete
Problem $B$ : PLS-complete
PPAD ∩ PLS seems unnatural...

Problem $A$ : PPAD-complete
Problem $B$ : PLS-complete

**EITHER-SOLUTION($A,B$):**
*Input:* instance $I_A$ of $A$, instance $I_B$ of $B$
*Goal:* find a solution of $I_A$, or a solution of $I_B$
PPAD \cap PLS seems unnatural...

Problem $A$ : PPAD-complete
Problem $B$ : PLS-complete

\textbf{EITHER-SOLUTION}(A,B):
\textit{Input:} instance $I_A$ of $A$, instance $I_B$ of $B$
\textit{Goal:} find a solution of $I_A$, or a solution of $I_B$

\implies \textbf{EITHER-SOLUTION}(A,B) is (PPAD \cap PLS)-complete!
PPAD ∩ PLS seems unnatural...

**BROUWER:**

*Input*: a continuous function $f: [0,1]^n \rightarrow [0,1]^n$

*Goal*: find a fixpoint $x$

\[ f(x) = x \]
PPAD $\cap$ PLS seems unnatural...

**BROUWER:**

*Input:* a continuous function $f: [0,1]^n \rightarrow [0,1]^n$, precision $\varepsilon > 0$

*Goal:* find an approximate fixpoint $x$

\[ \| f(x) - x \| \leq \varepsilon \]
PPAD ∩ PLS seems unnatural...

**BROUWER:**
*Input:* a continuous function $f: [0,1]^n \to [0,1]^n$, precision $\epsilon > 0$
*Goal:* find an approximate fixpoint $x$

$$\|f(x) - x\| \leq \epsilon$$

**REAL-LOCAL-OPT:**
*Input:*
- a continuous function $p: [0,1]^n \to [0,1]$
- a (possibly non-continuous) function $g: [0,1]^n \to [0,1]^n$
PPAD ∩ PLS seems unnatural…

**BROUWER:**
*Input:* a continuous function $f: [0,1]^n \rightarrow [0,1]^n$, precision $\varepsilon > 0$
*Goal:* find an approximate fixpoint $x$
\[ \| f(x) - x \| \leq \varepsilon \]

**REAL-LOCAL-OPT:**
*Input:*
- a continuous function $p: [0,1]^n \rightarrow [0,1]$
- a (possibly non-continuous) function $g: [0,1]^n \rightarrow [0,1]^n$
*Goal:* find a local minimum of $p$ with respect to $g$
\[ p(g(x)) \geq p(x) \]
PPAD ∩ PLS seems unnatural...

**BROUWER:**
*Input:* a continuous function $f: [0,1]^n \rightarrow [0,1]^n$, precision $\varepsilon > 0$
*Goal:* find an approximate fixpoint $x$
\[ \| f(x) - x \| \leq \varepsilon \]

**REAL-LOCAL-OPT:**
*Input:*
- a continuous function $p : [0,1]^n \rightarrow [0,1]$ 
- a (possibly non-continuous) function $g : [0,1]^n \rightarrow [0,1]^n$
*Goal:* find a local minimum of $p$ with respect to $g$
\[ p(g(x)) \geq p(x) - \varepsilon \]
PPAD ∩ PLS seems unnatural...

**BROUWER:**

*Input:* a continuous function $f: [0,1]^n \rightarrow [0,1]^n$, precision $\varepsilon > 0$

*Goal:* find an approximate fixpoint $x$

$$\|f(x) - x\| \leq \varepsilon$$

**REAL-LOCAL-OPT:**

*Input:*
- a continuous function $p: [0,1]^n \rightarrow [0,1]$
- a (possibly non-continuous) function $g: [0,1]^n \rightarrow [0,1]^n$

*Goal:* find a local minimum of $p$ with respect to $g$

$$p(g(x)) \geq p(x) - \varepsilon$$

$\Rightarrow$ **EITHER-SOLUTION(BROUWER,LOCAL-OPT) is (PPAD ∩ PLS)-complete.**
Continuous Local Search

But EITHER-SOLUTION(BROUWER, LOCAL-OPT) is not very natural...
Continuous Local Search

But EITHER-SOLUTION(BROUWER,LOCAL-OPT) is not very natural...

**CONTINUOUS-LOCAL-OPT:**

*Input*: continuous functions $g: [0,1]^n \rightarrow [0,1]^n$ and $p: [0,1]^n \rightarrow [0,1]$

*Goal*: find $x$ such that

$$p(g(x)) \geq p(x) - \varepsilon$$
Continuous Local Search

But EITHER-SOLUTION(BROUWER,LOCAL-OPT) is not very natural...

**CONTINUOUS-LOCAL-OPT:**
*Input:* continuous functions $g: [0,1]^n \rightarrow [0,1]^n$ and $p: [0,1]^n \rightarrow [0,1]$
*Goal:* find $x$ such that

$$p(g(x)) \geq p(x) - \epsilon$$

$\rightarrow$ class **Continuous Local Search (CLS)** [Daskalakis-Papadimitriou, 2011]
PPAD ∩ PLS

EITHER-SOLUTION(A, B)
PPAD ∩ PLS

EITHER-SOLUTION(𝐴, 𝑃)

CLS

CONTINUOUS-LOCAL-OPT

CONTRACTION

MIXED-CONGESTION

SSGs

P-LCP

P
PPAD ∩ PLS

EITHER-SOLUTION\((A, B)\)

CLS

CONTINUOUS-LOCAL-OPT
BANACH

CONTRACTION

MIXED-CONGESTION

SSGs P-LCP

[Daskalakis-Tzamos-Zampetakis, 2018]
Motivation behind the classes
Motivation behind the classes

**PPAD**: “all problems that can be solved by a path-following algorithm”
(Lemke-Howson algorithm for NASH)
Motivation behind the classes

**PPAD**: “all problems that can be solved by a path-following algorithm” (Lemke-Howson algorithm for NASH)

**PLS**: “all problems that can be solved by a local search algorithm”
Motivation behind the classes

**PPAD:** “all problems that can be solved by a path-following algorithm” (Lemke-Howson algorithm for NASH)

**PLS:** “all problems that can be solved by a local search algorithm”

**CLS:** “all problems that can be solved by a continuous local search algorithm”
Motivation behind the classes

**PPAD**: “all problems that can be solved by a path-following algorithm” (Lemke-Howson algorithm for NASH)

**PLS**: “all problems that can be solved by a local search algorithm”

**CLS**: “all problems that can be solved by a *continuous* local search algorithm”

**GD**: “all problems that can be solved by gradient descent”
Gradient Descent Problems

**GD**: “all problems that can be solved by gradient descent”
Gradient Descent Problems

**GD:** “all problems that can be solved by gradient descent”

**Input:** $C^1$-function $f: [0,1]^n \rightarrow [0,1]$, step size $\eta > 0$, precision $\varepsilon > 0$
Gradient Descent Problems

**GD:** “all problems that can be solved by gradient descent”

*Input:* $C^1$-function $f: [0,1]^n \rightarrow [0,1]$, step size $\eta > 0$, precision $\varepsilon > 0$

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

**GD:** “all problems that can be solved by gradient descent”

**Input:** $C^1$-function $f: [0,1]^n \rightarrow [0,1]$, step size $\eta > 0$, precision $\varepsilon > 0$

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

**Goal:** find a point where gradient descent terminates
Gradient Descent Problems

**GD:** “all problems that can be solved by gradient descent”

**Input:** $C^1$-function $f: [0,1]^n \rightarrow [0,1]$, step size $\eta > 0$, precision $\varepsilon > 0$

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

**Goal:** find a point where gradient descent terminates

$$[x' := x - \eta \nabla f(x)]$$

**GD-Local-Search:**

**Goal:** find $x$ such that $f(x') \geq f(x) - \varepsilon$ \hspace{1cm} (the next iterate decreases $f$ by at most $\varepsilon$)
Gradient Descent Problems

**GD:** “all problems that can be solved by gradient descent”

**Input:** $C^1$-function $f: [0,1]^n \rightarrow [0,1]$, step size $\eta > 0$, precision $\varepsilon > 0$

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

**Goal:** find a point where gradient descent terminates

$$[x' := x - \eta \nabla f(x)]$$

**GD-Local-Search:**

**Goal:** find $x$ such that $f(x') \geq f(x) - \varepsilon$ \hspace{1cm} (the next iterate decreases $f$ by at most $\varepsilon$)

$\Rightarrow$ in CLS: $p(x) := f(x)$ and $g(x) := x - \eta \nabla f(x)$
Gradient Descent Problems

**GD:** “all problems that can be solved by gradient descent”

**Input:** $C^1$-function $f: [0,1]^n \rightarrow [0,1]$, step size $\eta > 0$, precision $\varepsilon > 0$

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

**Goal:** find a point where gradient descent terminates

$$x' := x - \eta \nabla f(x)$$

**GD-Local-Search:**

**Goal:** find $x$ such that $f(x') \geq f(x) - \varepsilon$ 

*(the next iterate decreases $f$ by at most $\varepsilon$)*
Gradient Descent Problems

**GD:** “all problems that can be solved by gradient descent”

**Input:** $C^1$-function $f : [0,1]^n \rightarrow [0,1]$, step size $\eta > 0$, precision $\varepsilon > 0$

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

**Goal:** find a point where gradient descent terminates

$$[x' := x - \eta \nabla f(x)]$$

**GD-Local-Search:**
**Goal:** find $x$ such that $f(x') \geq f(x) - \varepsilon$ \quad (the next iterate decreases $f$ by at most $\varepsilon$)

**GD-Fixed-Point:**
**Goal:** find $x$ such that $\|x' - x\| \leq \varepsilon$ \quad (the next iterate is $\varepsilon$-close)
Gradient Descent Problems

**GD:** “all problems that can be solved by gradient descent”

*Input:* $C^1$-function $f: [0,1]^n \rightarrow [0,1]$, step size $\eta > 0$, precision $\varepsilon > 0$

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

*Goal:* find a point where gradient descent terminates

$$[x' := x - \eta \nabla f(x)]$$

**GD-Local-Search:**
*Goal:* find $x$ such that $f(x') \geq f(x) - \varepsilon$ (the next iterate decreases $f$ by at most $\varepsilon$)

**GD-Fixed-Point:**
*Goal:* find $x$ such that $\|x' - x\| \leq \varepsilon$ (the next iterate is $\varepsilon$-close)

$\rightarrow$ polynomial-time equivalent!
PPAD \cap PLS

EITHER-SOLUTION(\(A, B\))

CLS

CONTINUOUS-LOCAL-OPT
BANACH

CONTRACTION

MIXED-CONGESTION

SSGs \quad P-LCP

P
PPAD \cap PLS

EITHER-SOLUTION(\(A, B\))

CLS

CONTINUOUS-LOCAL-OPT

BANACH

GD

GD-FIXED-POINT

CONTRACTION

SSGs

P-LCP

MIXED-CONGESTION

P
PPAD ∩ PLS

EITHER-SOLUTION(A, B)

CLS

CONTINUOUS-LOCAL-OPT

BANACH

GD

GD-FIXED-POINT

2D-GD-FIXED-POINT

MIXED-CONGESTION

P

CONTRACTION

SSGs

P-LCP
PPAD $\cap$ PLS

EITHER-SOLUTION($A, B$)

CLS

CONTINUOUS-LOCAL-OPT
BANACH

GD

GD-FIXED-POINT

CONTRACTION

SSGs

P-LCP

2D-GD-FIXED-POINT

MIXED-CONGESTION

P
$\text{PPAD} \cap \text{PLS} = \text{CLS} = \text{GD}$

Either-Solution($A, B$)
Continuous-Local-Opt
Banach
2D-GD-Fixed-Point

Contraction
SSGs
P-LCP

Mixed-Congestion
Consequences
Consequences

• PPAD ∩ PLS is an interesting class!
Consequences

• PPAD ∩ PLS is an interesting class!

• It captures continuous local search, and even gradient descent
Consequences

• PPAD ∩ PLS is an interesting class!

• It captures continuous local search, and even gradient descent

• CLS and GD are robust with respect to:
  ➢ dimension
  ➢ domain
  ➢ arithmetic circuits
  ➢ ...

Proof Sketch
PPAD

Canonical complete problem: END-OF-LINE
PPAD

Canonical complete problem: END-OF-LINE

Input: directed graph of paths and cycles, and a source
PPAD

Canonical complete problem: **END-OF-LINE**

Input: directed graph of paths and cycles, and a source
PPAD

Canonical complete problem: END-OF-LINE

Input: directed graph of paths and cycles, and a source
PPAD

Canonical complete problem: **END-OF-LINE**

Input: directed graph of paths and cycles, and a source

Goal: find a sink, or another source
PPAD

Canonical complete problem: END-OF-LINE

Input: directed graph of paths and cycles, and a source

Goal: find a sink, or another source
PPAD

Canonical complete problem: END-OF-LINE

Input: directed graph of paths and cycles, and a source
Goal: find a sink, or another source

The catch: the graph is given \textit{implicitly}

- Vertex set $\{0,1\}^n$
- Boolean circuits $S$ and $P$
  - successor circuit $S$: $\{0,1\}^n \to \{0,1\}^n$
  - predecessor circuit $P$: $\{0,1\}^n \to \{0,1\}^n$
Reduction: high level
Reduction: high level

Goal: reduction from EITHER-SOLUTION(END-OF-LINE, LOCAL-OPT) to 2D-GD-FIXED-POINT
Reduction: high level

Goal: reduction from EITHER-SOLUTION(END-OF-LINE, LOCAL-OPT) to 2D-GD-FIXED-POINT

→ Construct a continuously differentiable function \( f: [0,1]^2 \to \mathbb{R} \) such that any gradient descent fixed point yields a solution to the EITHER-SOLUTION instance
Reduction: high level

Goal: reduction from EITHER-SOLUTION(END-OF-LINE, LOCAL-OPT) to 2D-GD-FIXED-POINT

→ Construct a continuously differentiable function $f: [0,1]^2 \rightarrow \mathbb{R}$ such that any gradient descent fixed point yields a solution to the EITHER-SOLUTION instance
Reduction: high level

Goal: reduction from EITHER-SOLUTION(END-OF-LINE, LOCAL-OPT) to 2D-GD-FIXED-POINT

→ Construct a continuously differentiable function \( f: [0,1]^2 \rightarrow \mathbb{R} \) such that any gradient descent fixed point yields a solution to the EITHER-SOLUTION instance
Goal: reduction from EITHER-SOLUTION(END-OF-LINE, LOCAL-OPT) to 2D-GD-FIXED-POINT

→ Construct a continuously differentiable function $f: [0,1]^2 \rightarrow \mathbb{R}$ such that any gradient descent fixed point yields a solution to the EITHER-SOLUTION instance
Reduction: high level

Goal: reduction from EITHER-SOLUTION(END-OF-LINE, LOCAL-OPT) to 2D-GD-FIXED-POINT

→ Construct a continuously differentiable function $f: [0,1]^2 \rightarrow \mathbb{R}$ such that any gradient descent fixed point yields a solution to the EITHER-SOLUTION instance
Warm up: Monotone-End-of-Line
Warm up: Monotone-End-of-Line
Warm up: Monotone-End-of-Line

Special case of END-OF-LINE: No backward edges allowed!
Warm up: Monotone-End-of-Line

Special case of END-OF-LINE: No backward edges allowed!
Warm up: Monotone-End-of-Line

Special case of END-OF-LINE: No backward edges allowed!
Locally computable!

[Hubáček-Yoge, 2017] for CLS
Back to standard End-of-Line
Back to standard End-of-Line
Back to standard End-of-Line
green edges: forward
red edges: backwards
Requires solving the PLS instance!
→ to find a gradient descent fixed point, we have to solve the PPAD problem or the PLS problem.
Future Directions

• are there other intersections of classes that are interesting?
Future Directions

• are there other intersections of classes that are interesting?

• candidates for \((\text{PPAD} \cap \text{PLS})\)-completeness:
  - CONTRACTION
  - TARKSI
  - POLYNOMIAL-KKT
  - MIXED-CONGESTION
Future Directions

• are there other intersections of classes that are interesting?

• candidates for \((\text{PPAD} \cap \text{PLS})\)-completeness:
  - CONTRACTION
  - TARSKI
  - POLYNOMIAL-KKT
  - MIXED-CONGESTION

Solved!
Future Directions

• are there other intersections of classes that are interesting?

• candidates for (PPAD ∩ PLS)-completeness:
  ➢ CONTRACTION
  ➢ TARSKI
  ➢ POLYNOMIAL-KKT
  ➢ MIXED-CONGESTION

  \[ \text{Solved!} \]

[Babichenko-Rubinstein, 2020]

\[ 2D-GD-FIXED-POINT \leq \text{MIXED-CONGESTION} \leq \text{POLYNOMIAL-KKT} \]
Thank You!