The Complexity of Gradient Descent: CLS = PPAD ∩ PLS

ALEXANDROS HOLLENDER

JOINT WORK WITH JOHN FEARNLEY, PAUL GOLDBERG AND RAHUL SAVANI

UNIVERSITY OF OXFORD
UNIVERSITY OF LIVERPOOL

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.

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

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

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 $f: [0,1]^n \rightarrow [0,1]^n$.

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 $f: [0,1]^n \to [0,1]^n$.

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

Some interesting computational problems

What do these problems have in common?

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 $f: [0,1]^3 \to [0,1]^3$.

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

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…

$$\text{3-SAT} \leq \text{NASH} \quad \Rightarrow \quad \text{certificate for unsatisfiable 3-SAT formulas}$$

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]

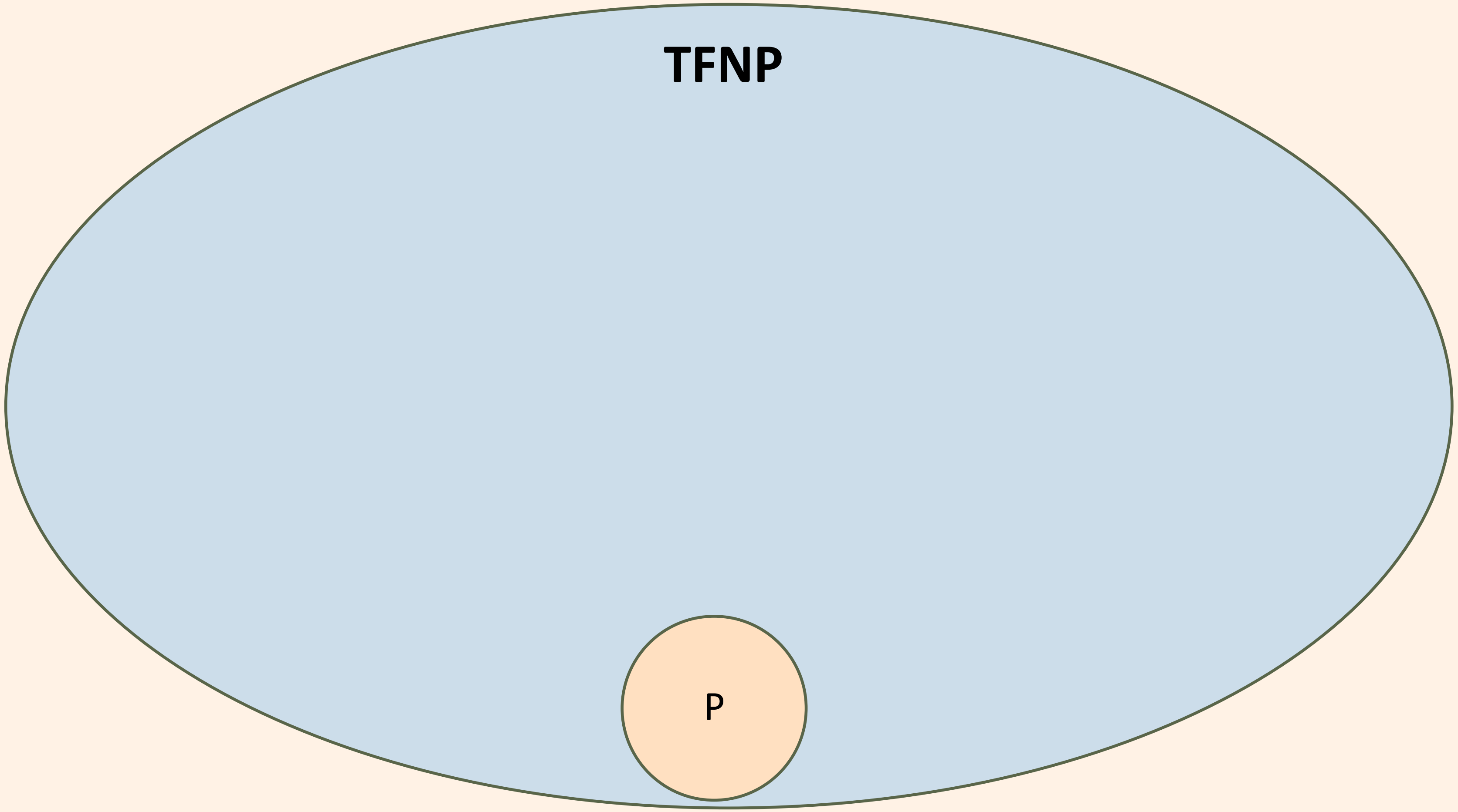
Total NP search problems:

- “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

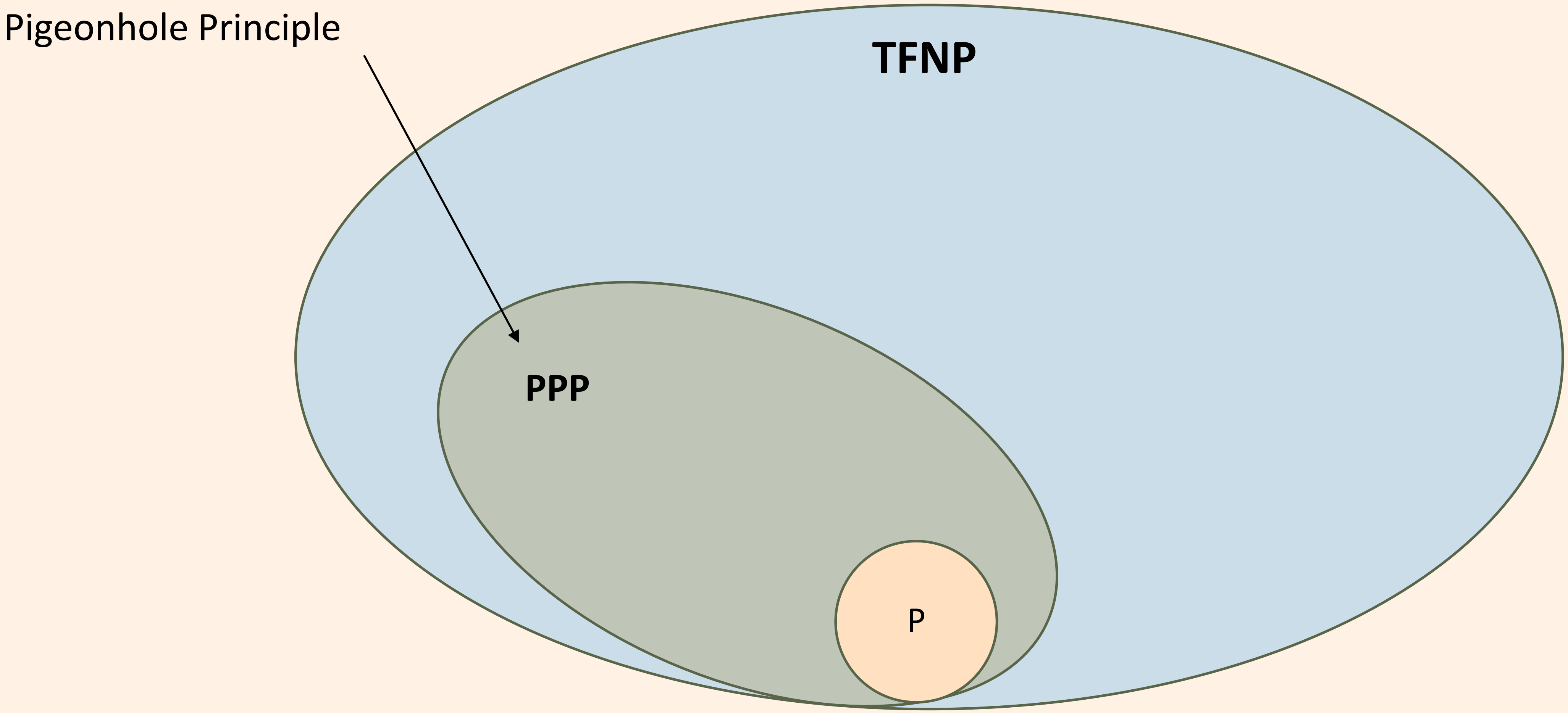
How do we show that a TFNP-problem is hard:

- No TFNP-problem can be NP-hard, unless NP = coNP…
- Believed that no TFNP-complete problems exists…

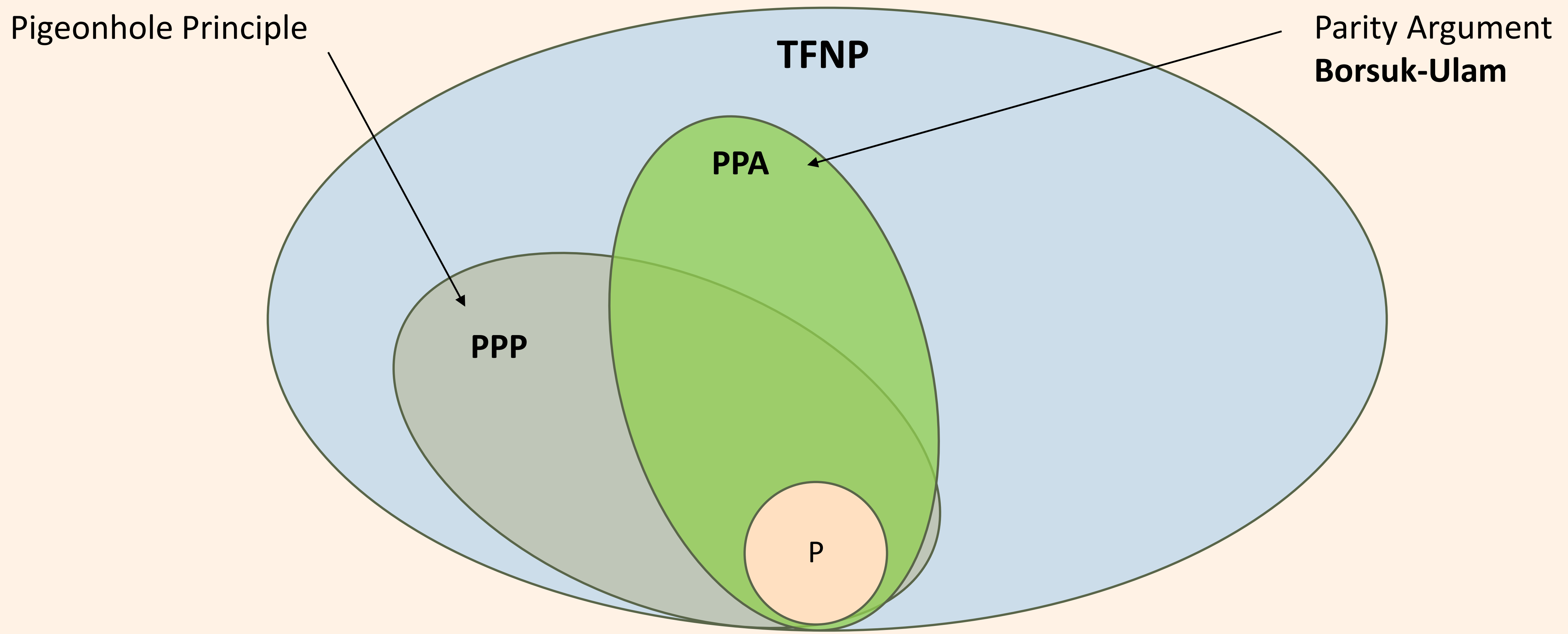
The TFNP landscape



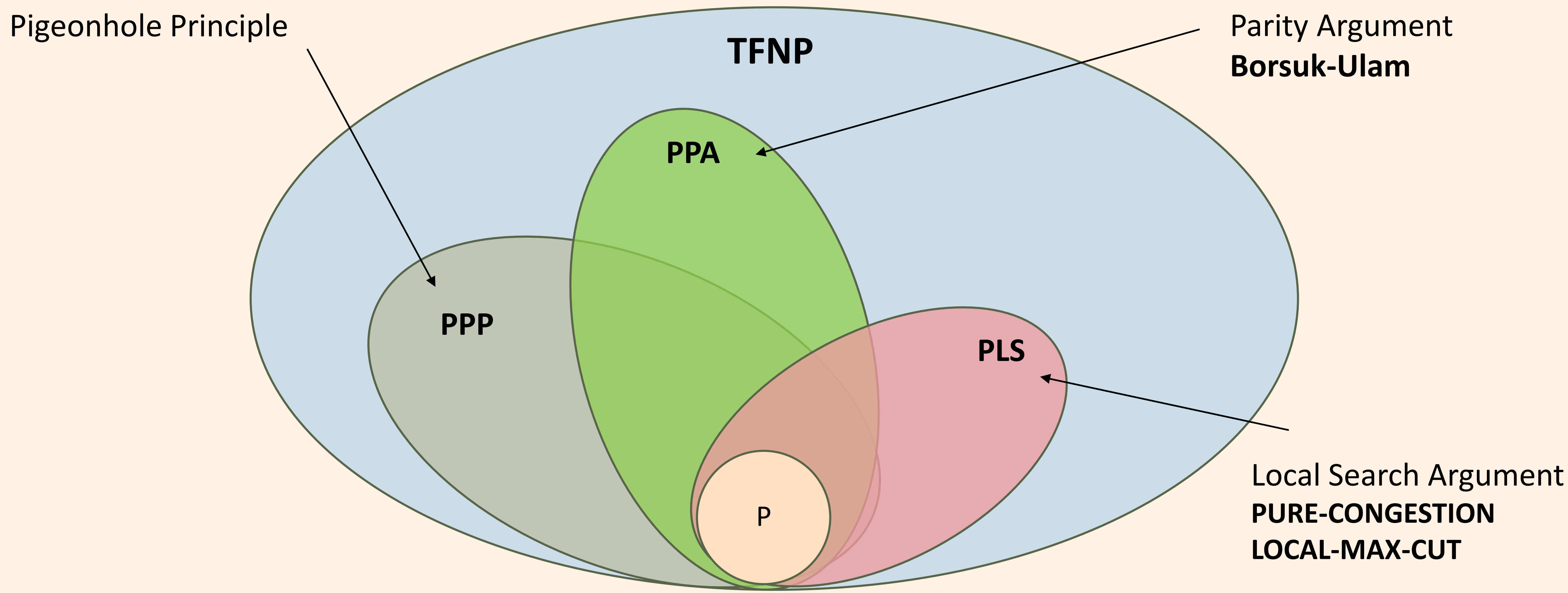
The TFNP landscape



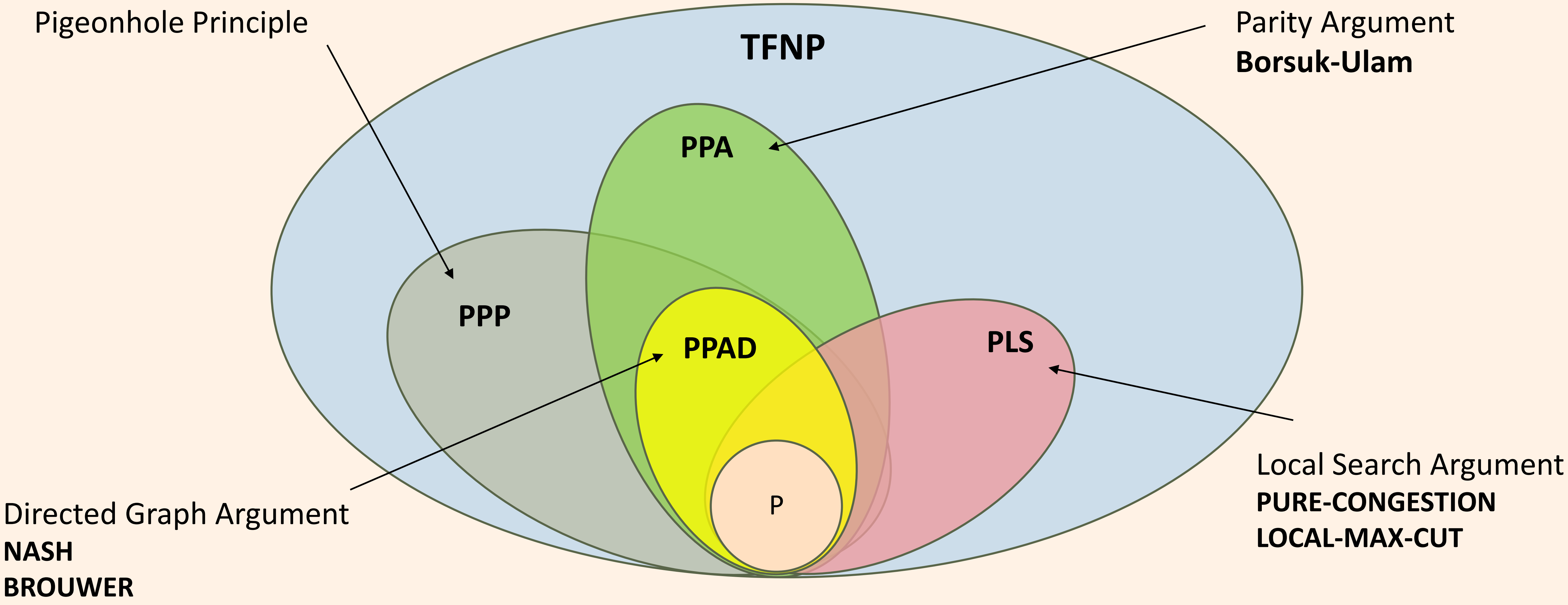
The TFNP landscape



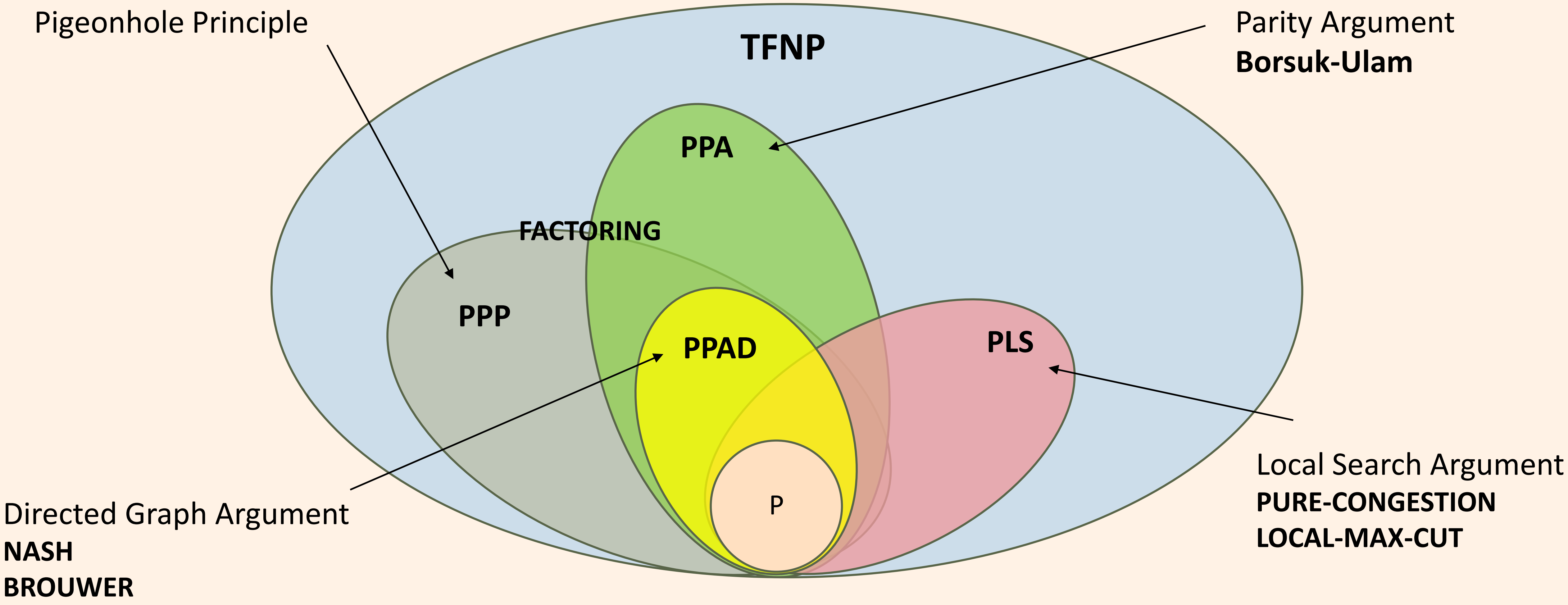
The TFNP landscape



The TFNP landscape



The TFNP landscape



TFNP subclasses

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

TFNP subclasses

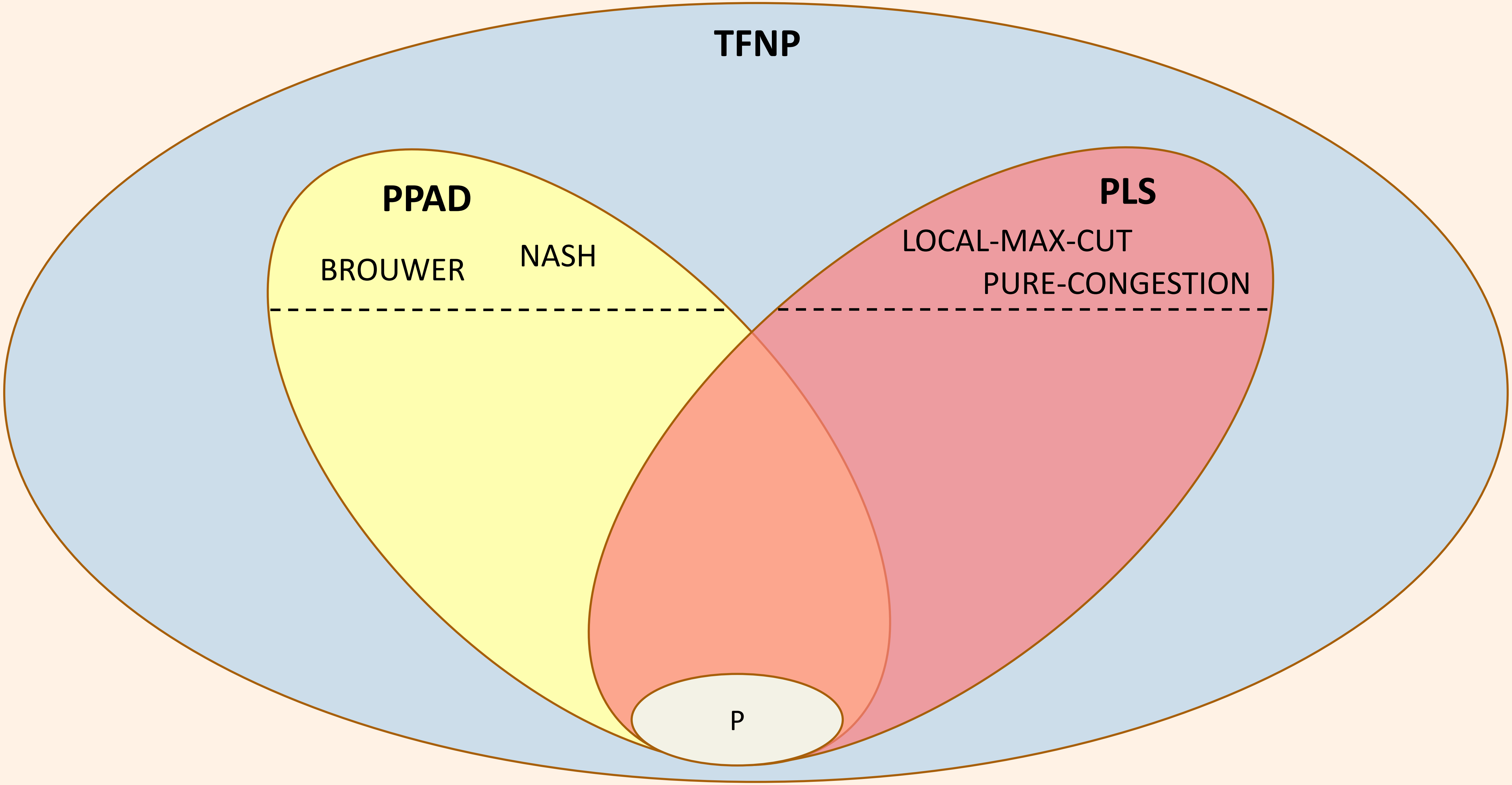
What reasons are there to believe that PPAD ≠ P, PLS ≠ P, etc?

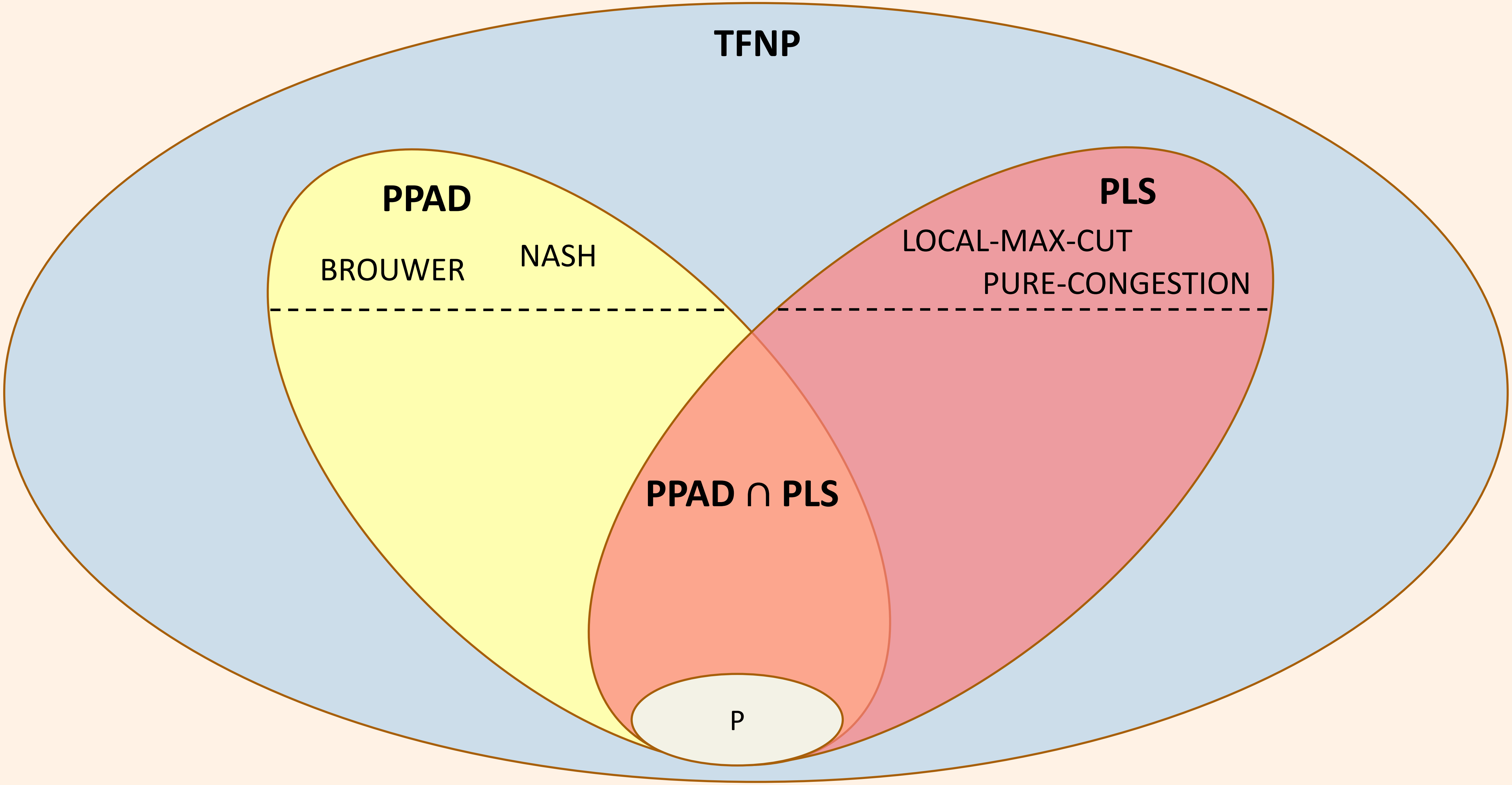
- many seemingly hard problems lie in PPAD, PLS etc…
- oracle separations between the classes (in particular PPAD ≠ PLS)

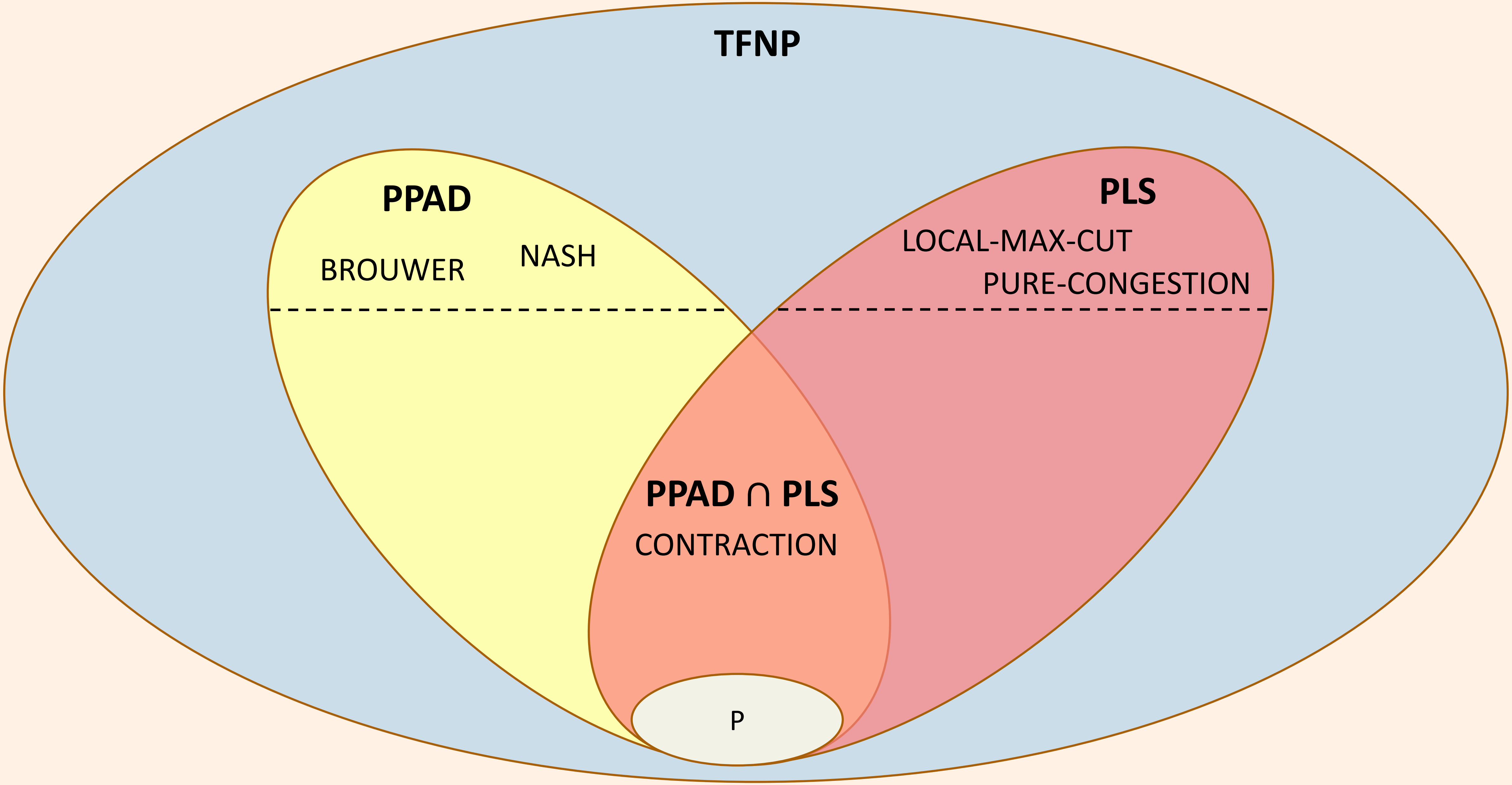
TFNP subclasses

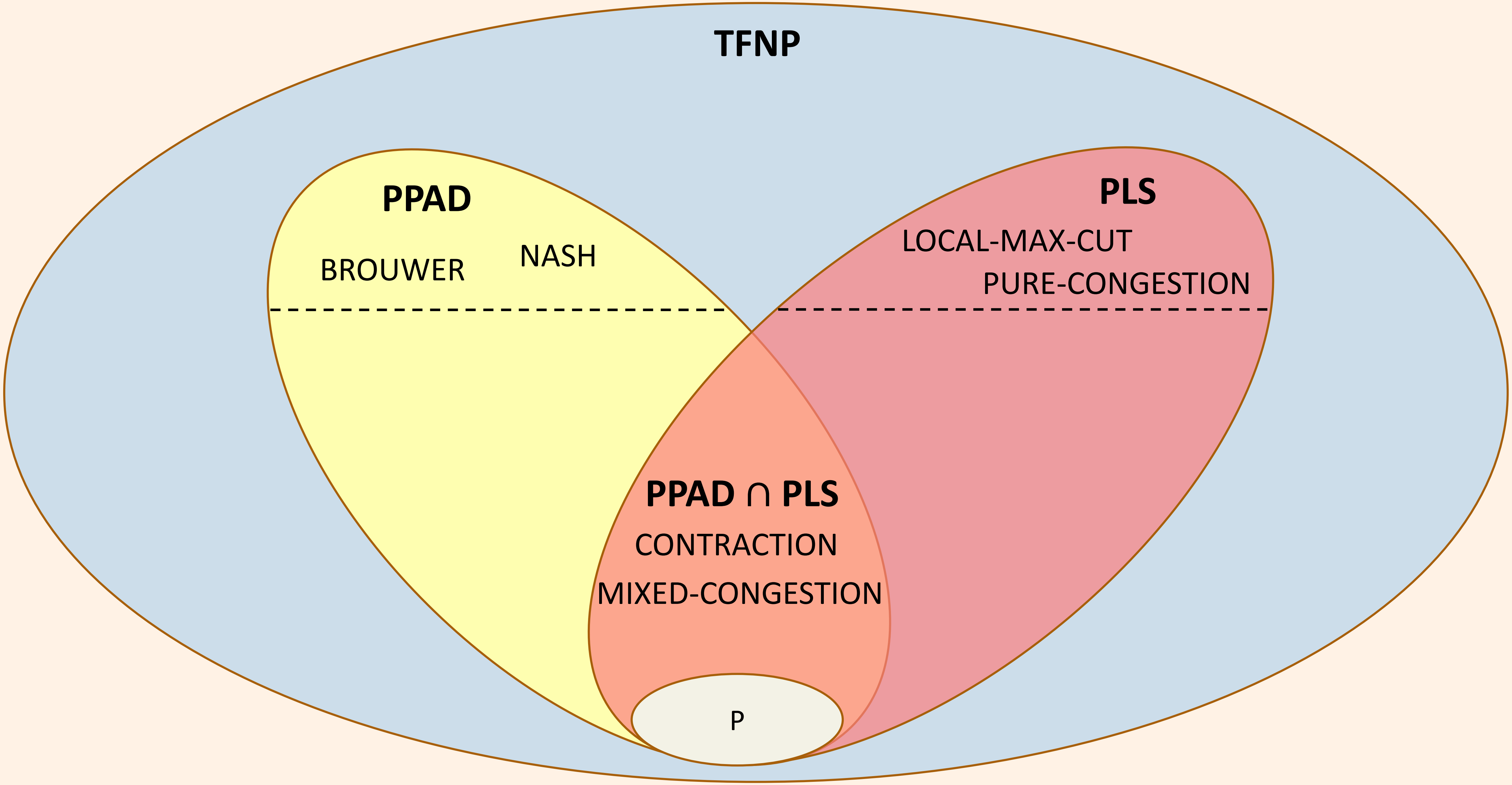
What reasons are there to believe that PPAD ≠ P, PLS ≠ P, etc?

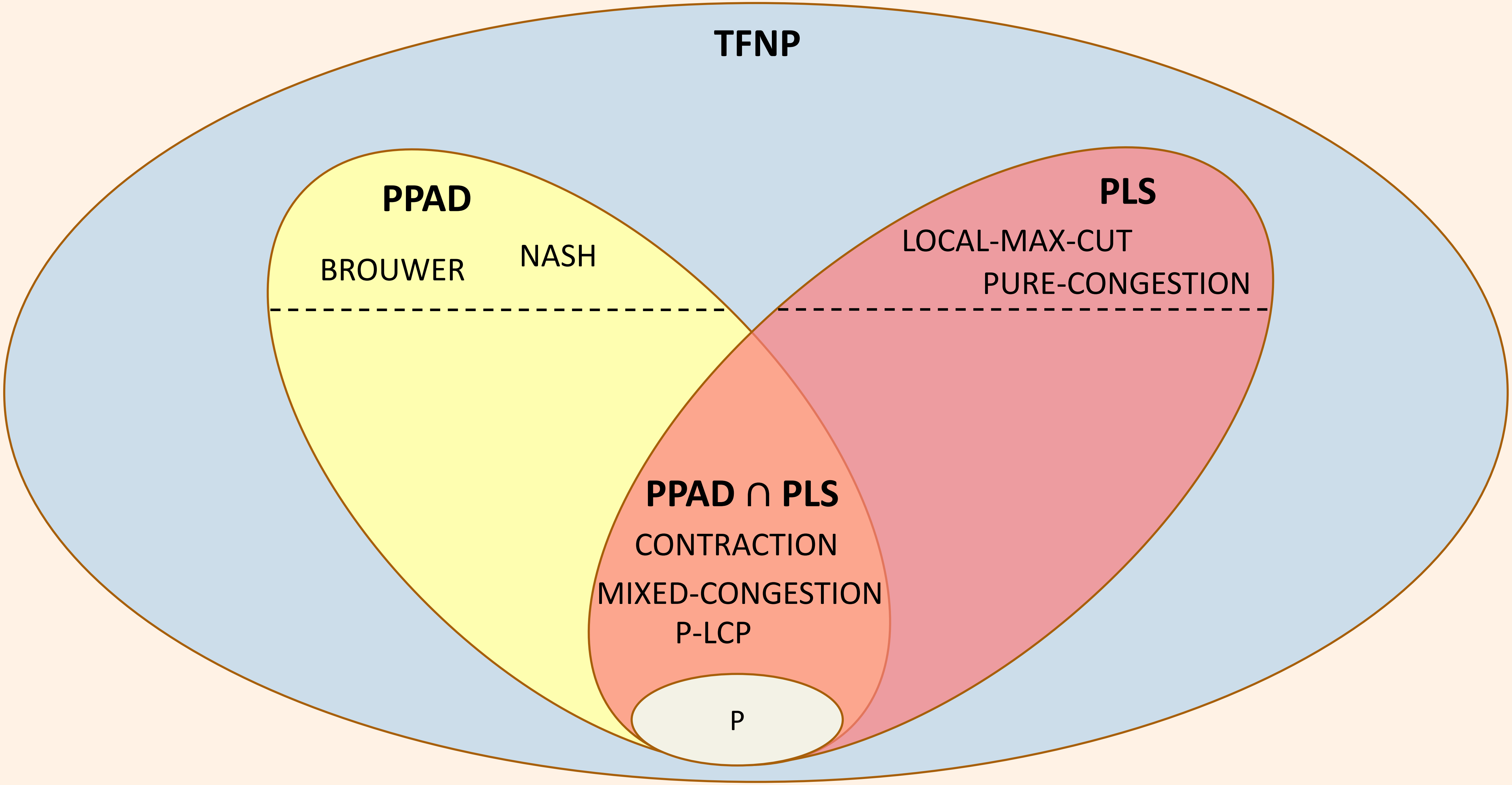
- many seemingly hard problems lie in PPAD, PLS etc…
- oracle separations between the classes (in particular PPAD ≠ PLS)
- hard under cryptographic assumptions

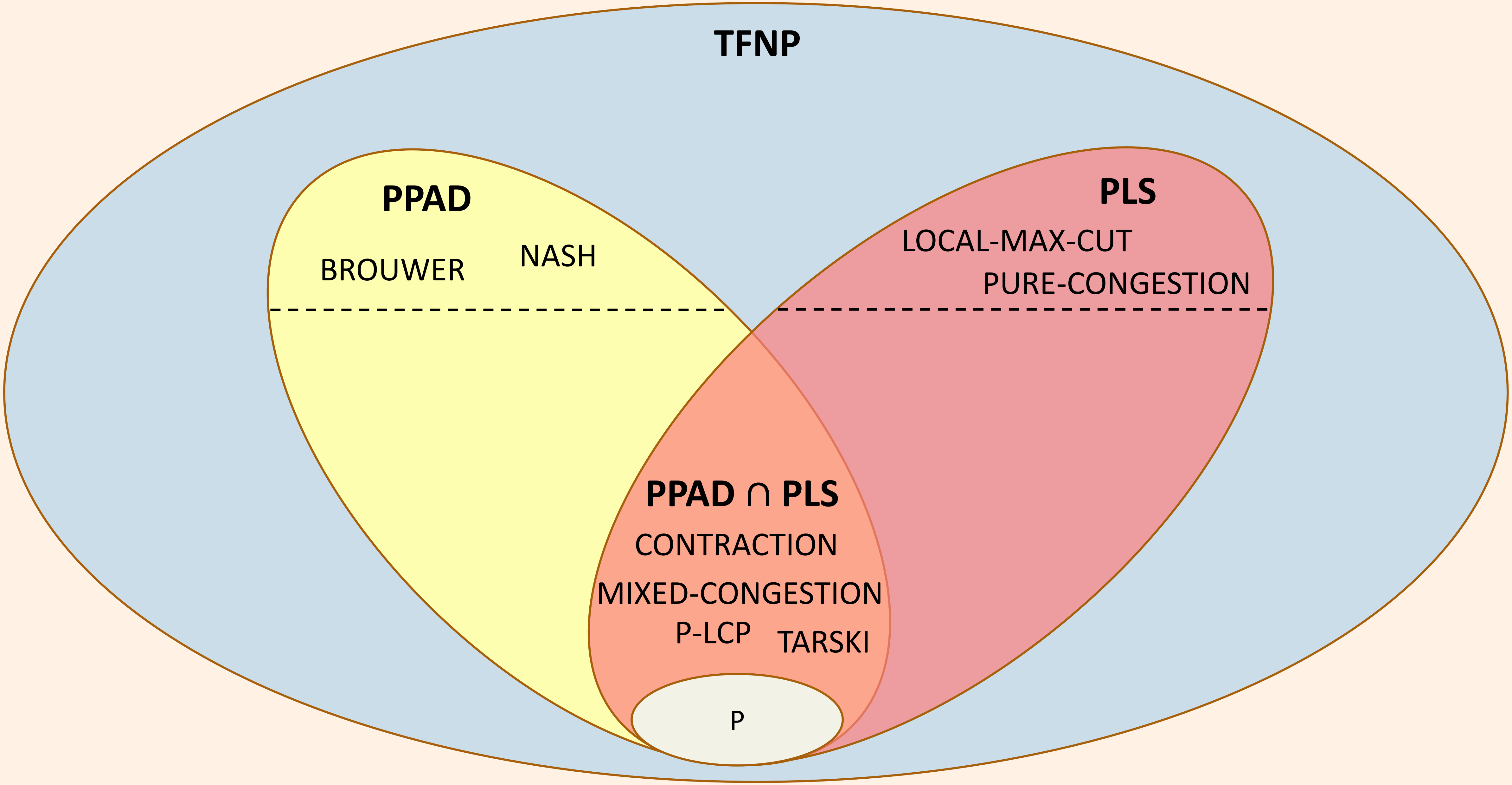

TFNP
PPAD
BROUWER
NASH
PLS
LOCAL-MAX-CUT
PURE-CONGESTION
P

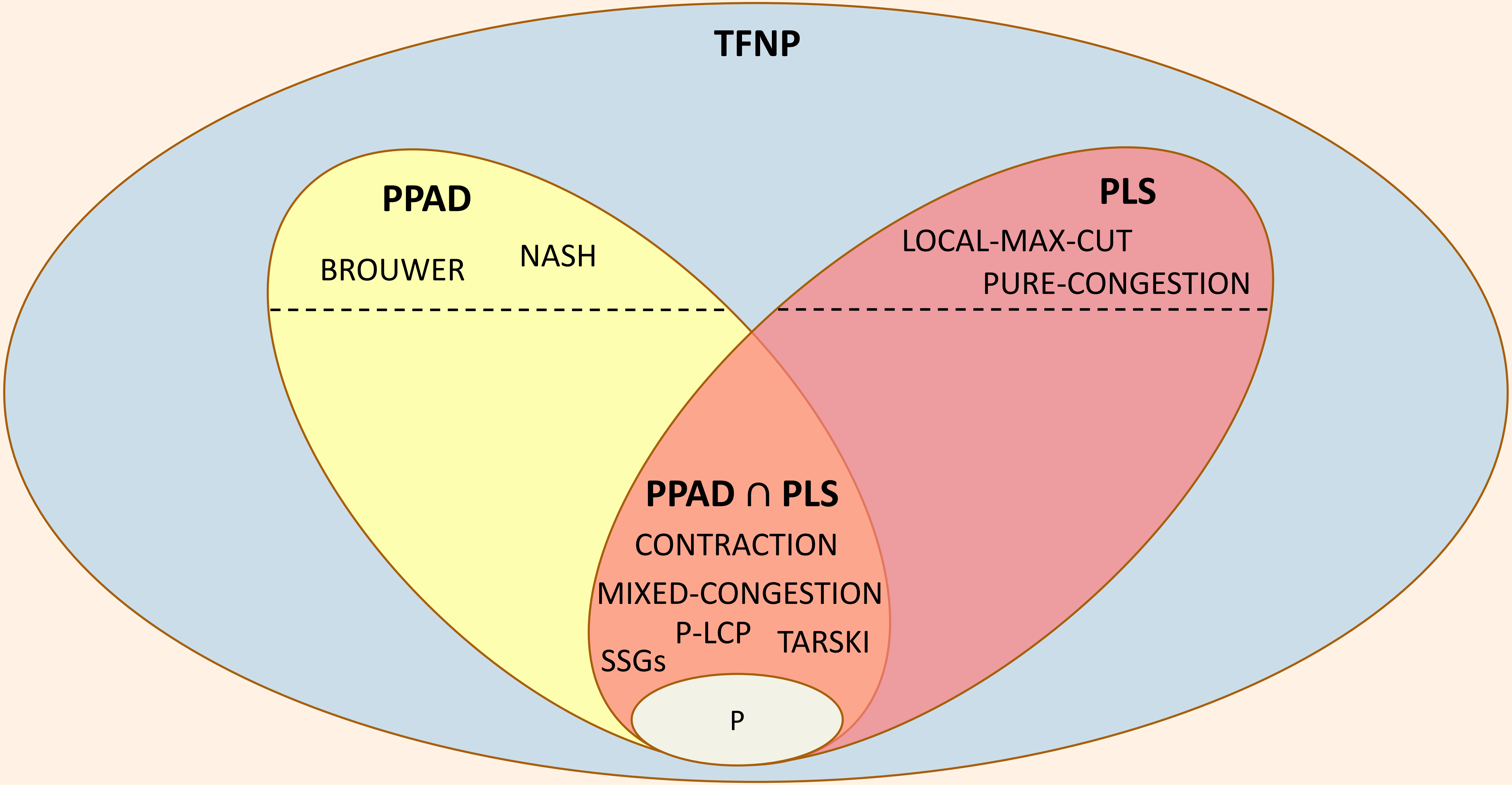

TFNP
PPAD
BROUWER
NASH
PLS
LOCAL-MAX-CUT
PURE-CONGESTION
PPAD ∩ PLS
P


TFNP
PPAD
BROUWER
NASH
PLS
LOCAL-MAX-CUT
PURE-CONGESTION
PPAD ∩ PLS
CONTRACTION
P


TFNP
PPAD
BROUWER
NASH
PLS
LOCAL-MAX-CUT
PURE-CONGESTION
PPAD ∩ PLS
CONTRACTION
MIXED-CONGESTION
P


TFNP
PPAD
BROUWER
NASH
PLS
LOCAL-MAX-CUT
PURE-CONGESTION
PPAD ∩ PLS
CONTRACTION
MIXED-CONGESTION
P-LCP
P


TFNP
PPAD
BROUWER
NASH
PLS
LOCAL-MAX-CUT
PURE-CONGESTION
PPAD ∩ PLS
CONTRACTION
MIXED-CONGESTION
P-LCP
TARSKI
P


TFNP
PPAD
BROUWER
NASH
PLS
LOCAL-MAX-CUT
PURE-CONGESTION
PPAD ∩ PLS
CONTRACTION
MIXED-CONGESTION
P-LCP
TARSKI
SSGs
P

PPAD $\cap$ PLS seems unnatural…

PPAD ∩ 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($\boldsymbol{A}$,$\boldsymbol{B}$):
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 ∩ PLS seems unnatural...

Problem A : PPAD-complete

Problem B : PLS-complete

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

→ **EITHER-SOLUTION($\boldsymbol{A}$,$\boldsymbol{B}$) is (PPAD ∩ PLS)-complete!**

PPAD ∩ PLS seems unnatural…

BROUWER:
Input: a continuous function $f: [0,1]^n \to [0,1]^n$
Goal: find a fixpoint x

$$f(x) = x$$

PPAD ∩ PLS seems unnatural…

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

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

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 $f\colon [0,1]^n \to [0,1]^n$, precision $\varepsilon > 0$
Goal: find an approximate fixpoint x

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

REAL-LOCAL-OPT:
Input:
- a continuous function $p\colon [0,1]^n \to [0,1]$
- a (possibly non-continuous) function $g\colon [0,1]^n \to [0,1]^n$

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

$$p\big(g(x)\big) \geq p(x)$$

PPAD ∩ PLS seems unnatural…

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

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

REAL-LOCAL-OPT:
Input:
- a continuous function $p\colon\ [0,1]^n \to [0,1]$
- a (possibly non-continuous) function $g\colon [0,1]^n \to [0,1]^n$

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

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

PPAD ∩ PLS seems unnatural...

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

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

REAL-LOCAL-OPT:
Input:
- a continuous function p: $[0,1]^n \to [0,1]$
- a (possibly non-continuous) function $g: [0,1]^n \to [0,1]^n$

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

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

→ **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

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

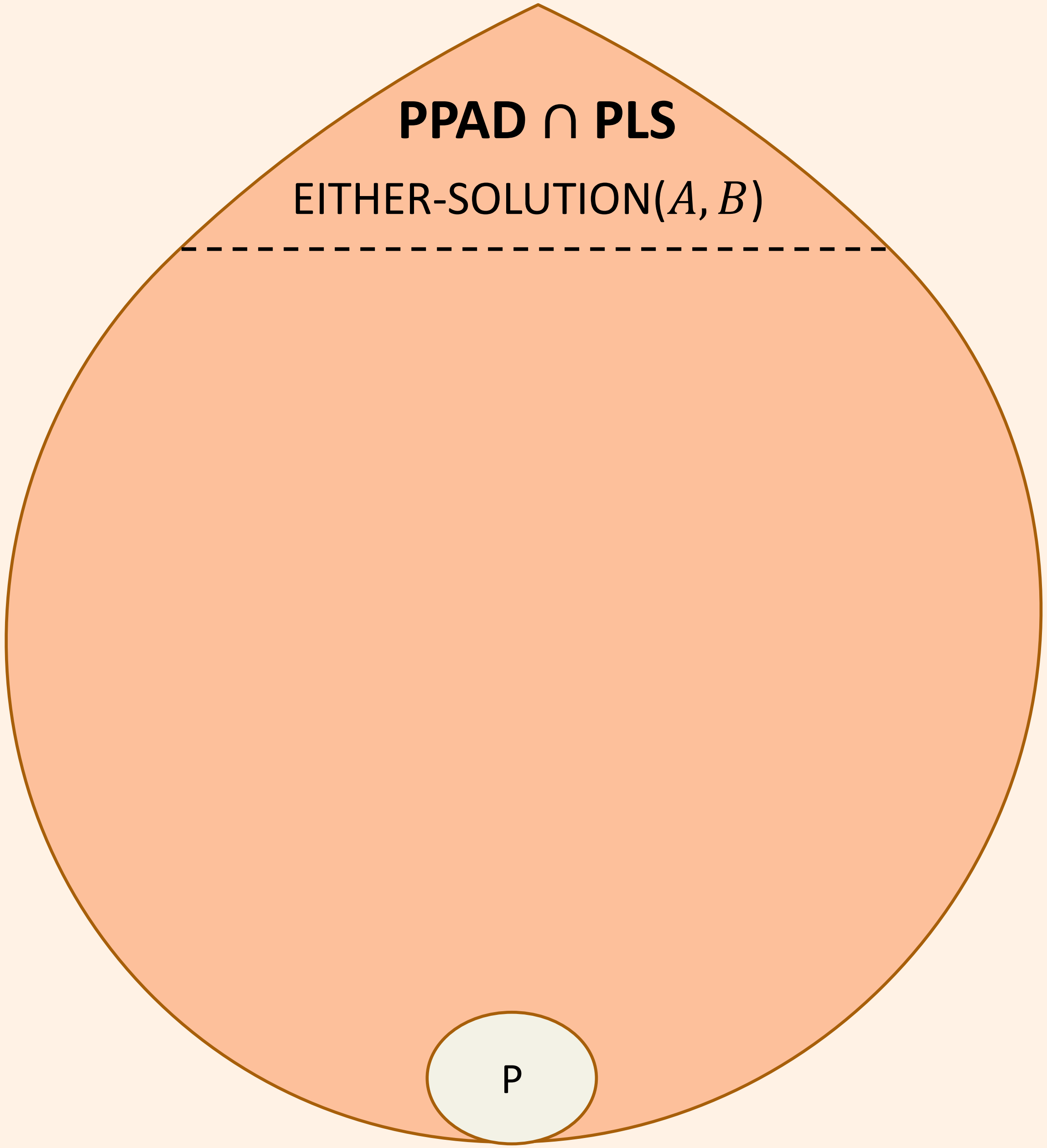
Continuous Local Search

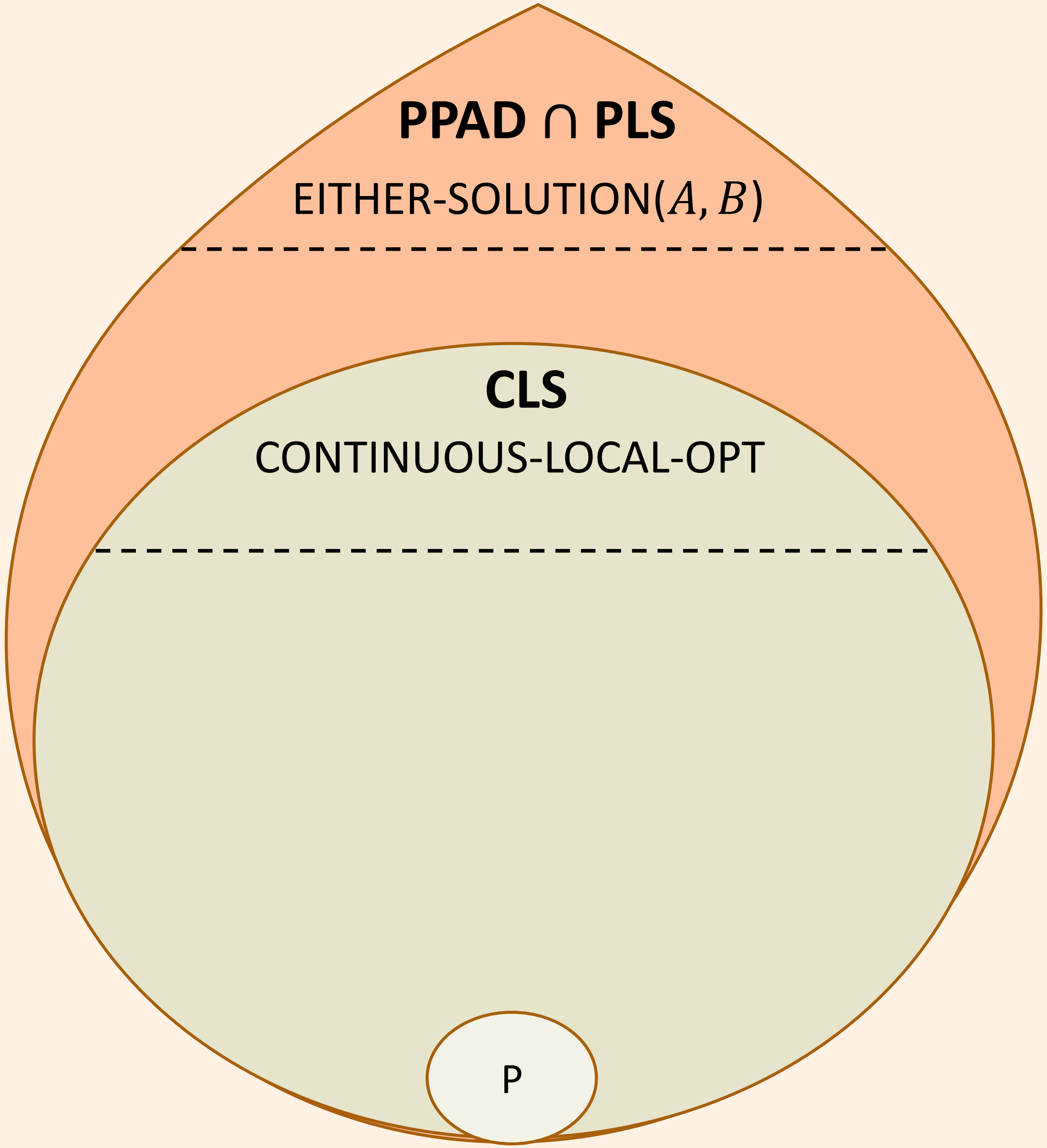
But EITHER-SOLUTION(BROUWER,LOCAL-OPT) is not very natural…

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

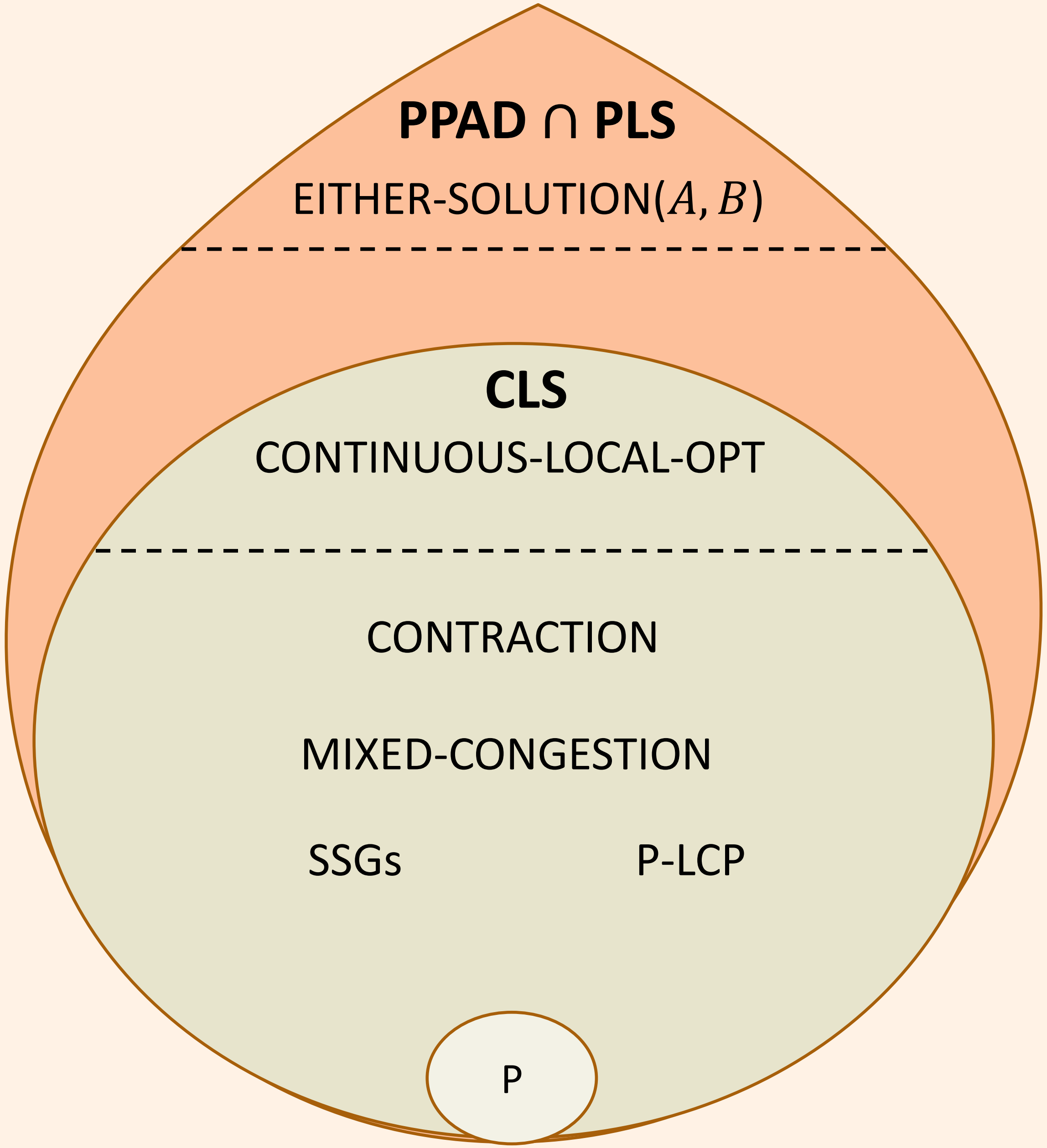
$$p\big(g(x)\big) \geq p(x) - \varepsilon$$

→ class **Continuous Local Search (CLS)** [Daskalakis-Papadimitriou, 2011]

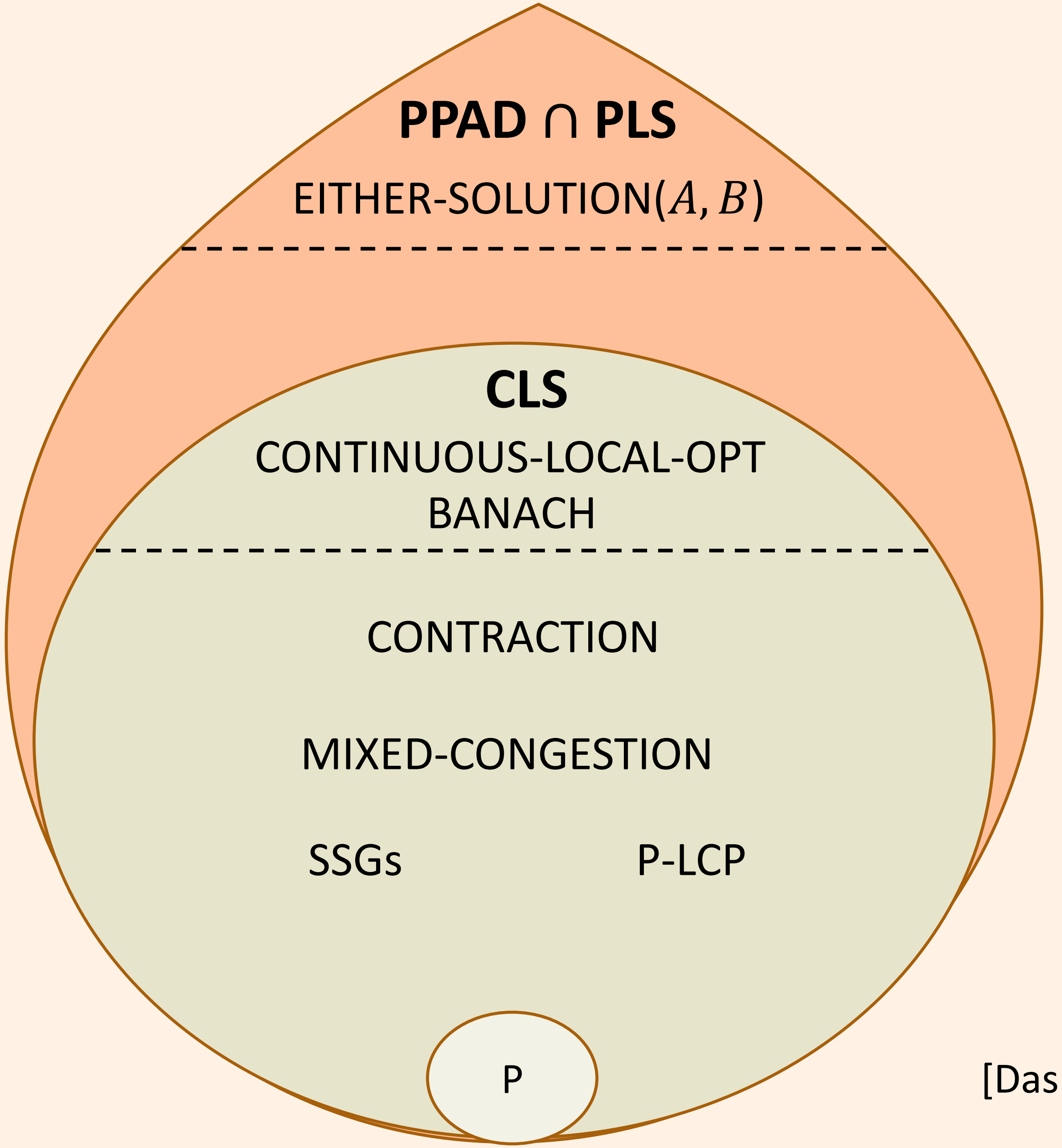

PPAD ∩ PLS
EITHER-SOLUTION(A, B)
P



PPAD ∩ PLS
EITHER-SOLUTION(A, B)
CLS
CONTINUOUS-LOCAL-OPT
P



PPAD ∩ PLS
EITHER-SOLUTION(A, B)
CLS
CONTINUOUS-LOCAL-OPT
CONTRACTION
MIXED-CONGESTION
SSGs
P-LCP
P



[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 $f: [0,1]^n \to [0,1]$, 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 $\varepsilon > 0$

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

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 $\varepsilon > 0$

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

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 $f\colon [0,1]^n \to [0,1]$, step size $\eta > 0$, precision $\varepsilon > 0$

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

Goal: find a point where gradient descent terminates

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

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

Gradient Descent Problems

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

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

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

Goal: find a point where gradient descent terminates

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

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

→ in CLS: $p(x) \coloneqq f(x)$ and $g(x) \coloneqq x - \eta \nabla f(x)$

Gradient Descent Problems

GD: "all problems that can be solved by gradient descent"

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

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

Goal: find a point where gradient descent terminates

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

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

Gradient Descent Problems

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

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

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

Goal: find a point where gradient descent terminates

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

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

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

Gradient Descent Problems

GD: "all problems that can be solved by gradient descent"

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

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

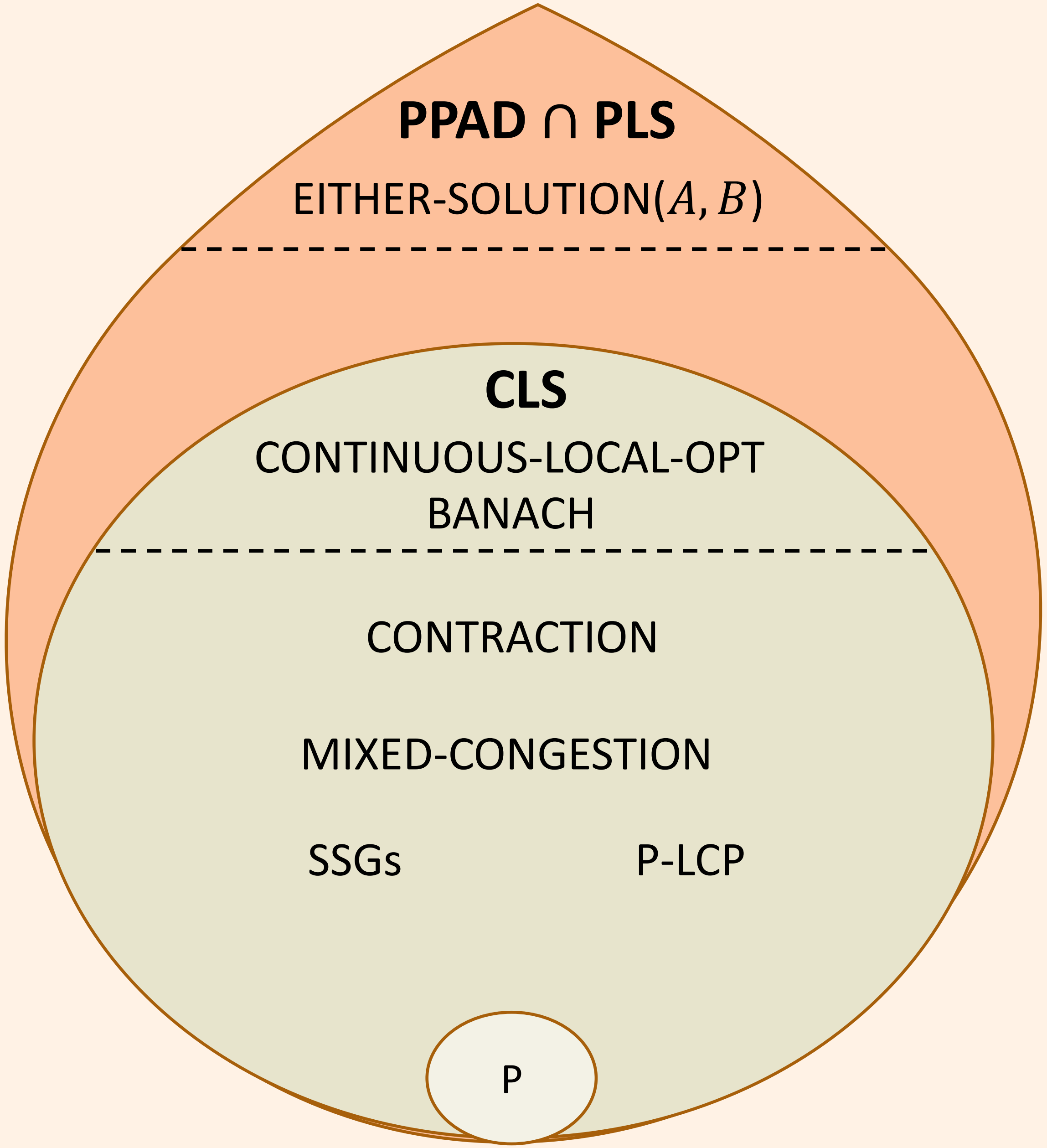
Goal: find a point where gradient descent terminates

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

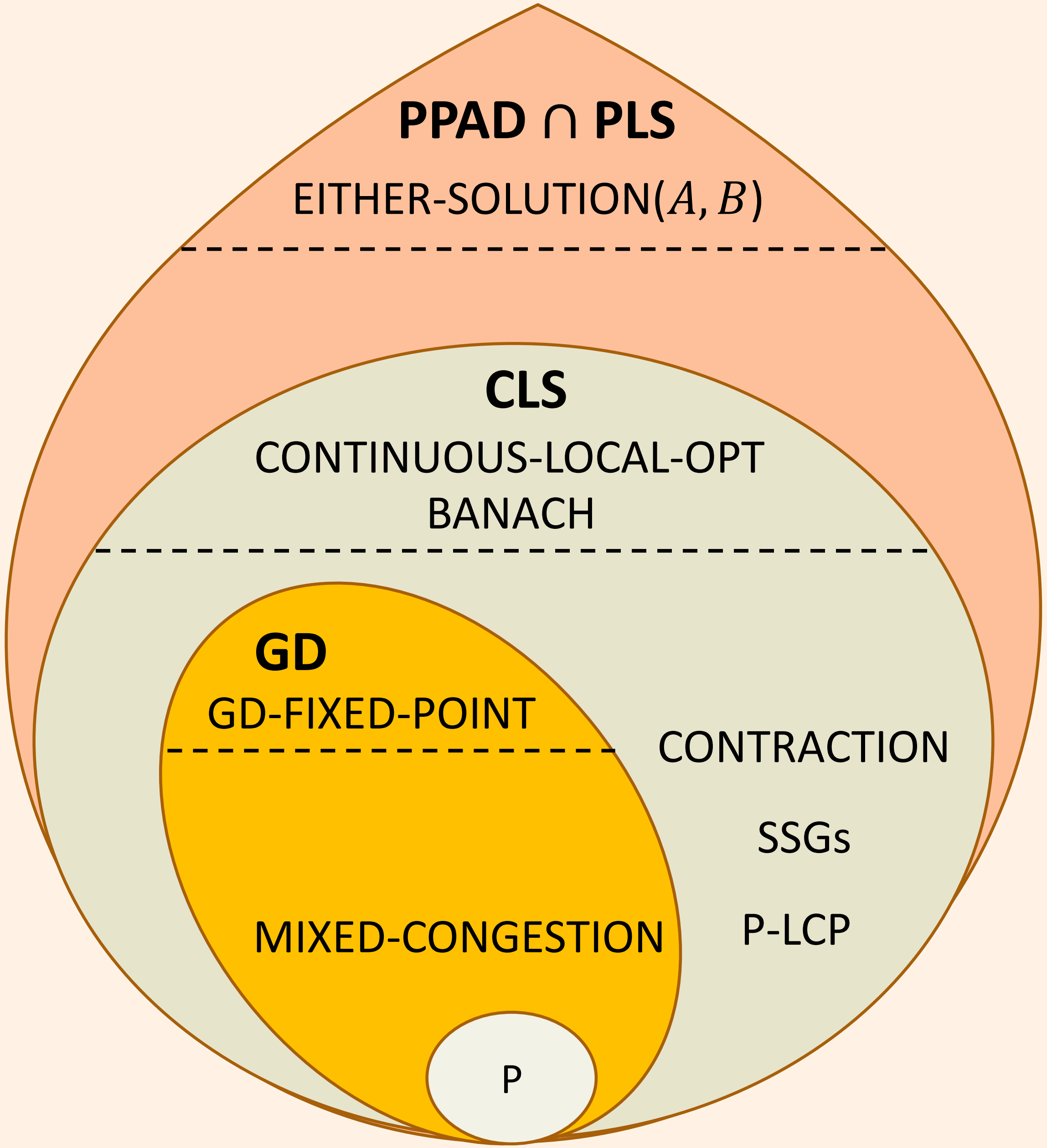
GD-Local-Search:
Goal: find x such that $f(x') \geq f(x) - \varepsilon$ *(the next iterate decreases f by at most ε)*

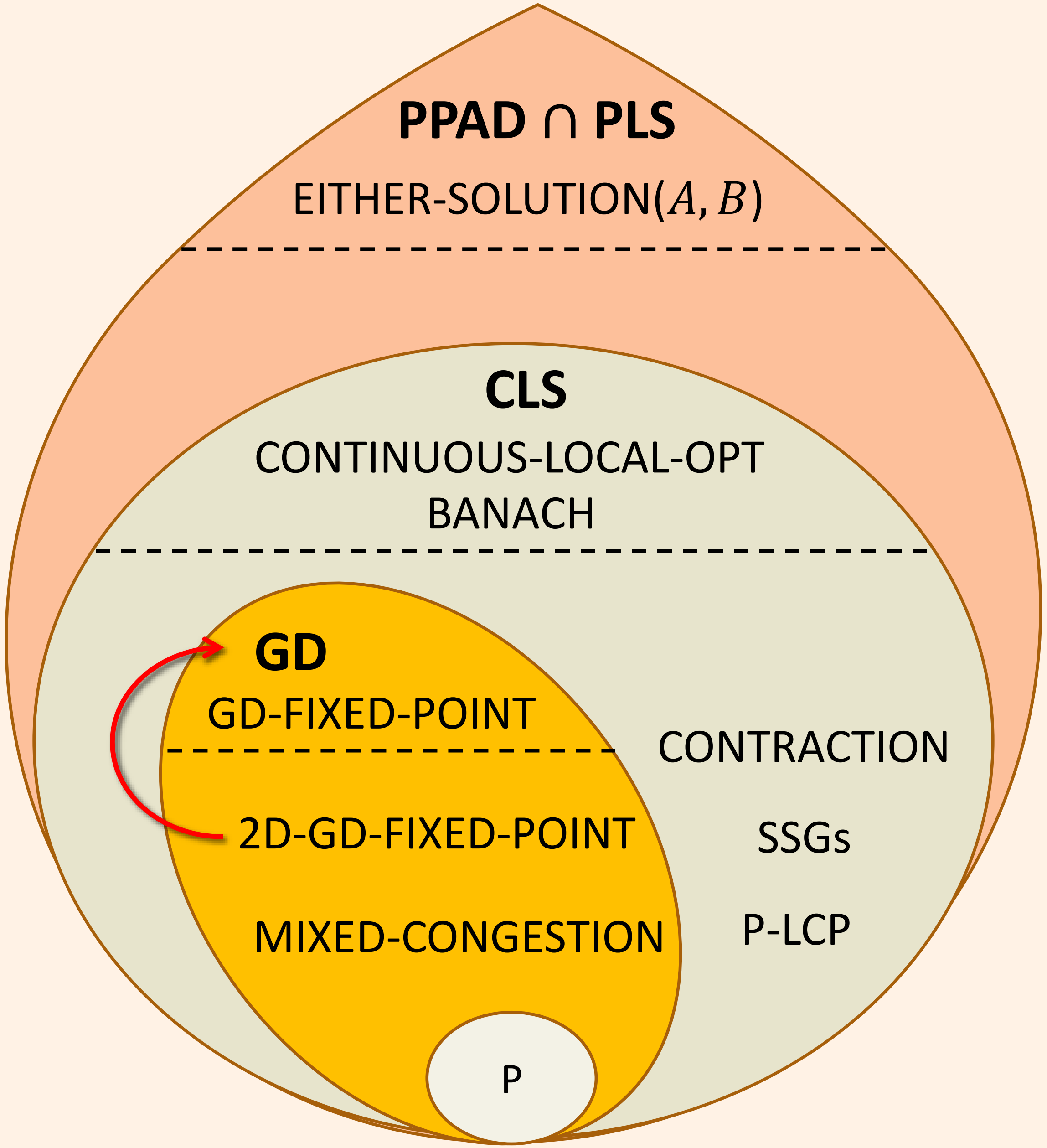
GD-Fixed-Point:
Goal: find x such that $\|x' - x\| \leq \varepsilon$ *(the next iterate is ε-close)*

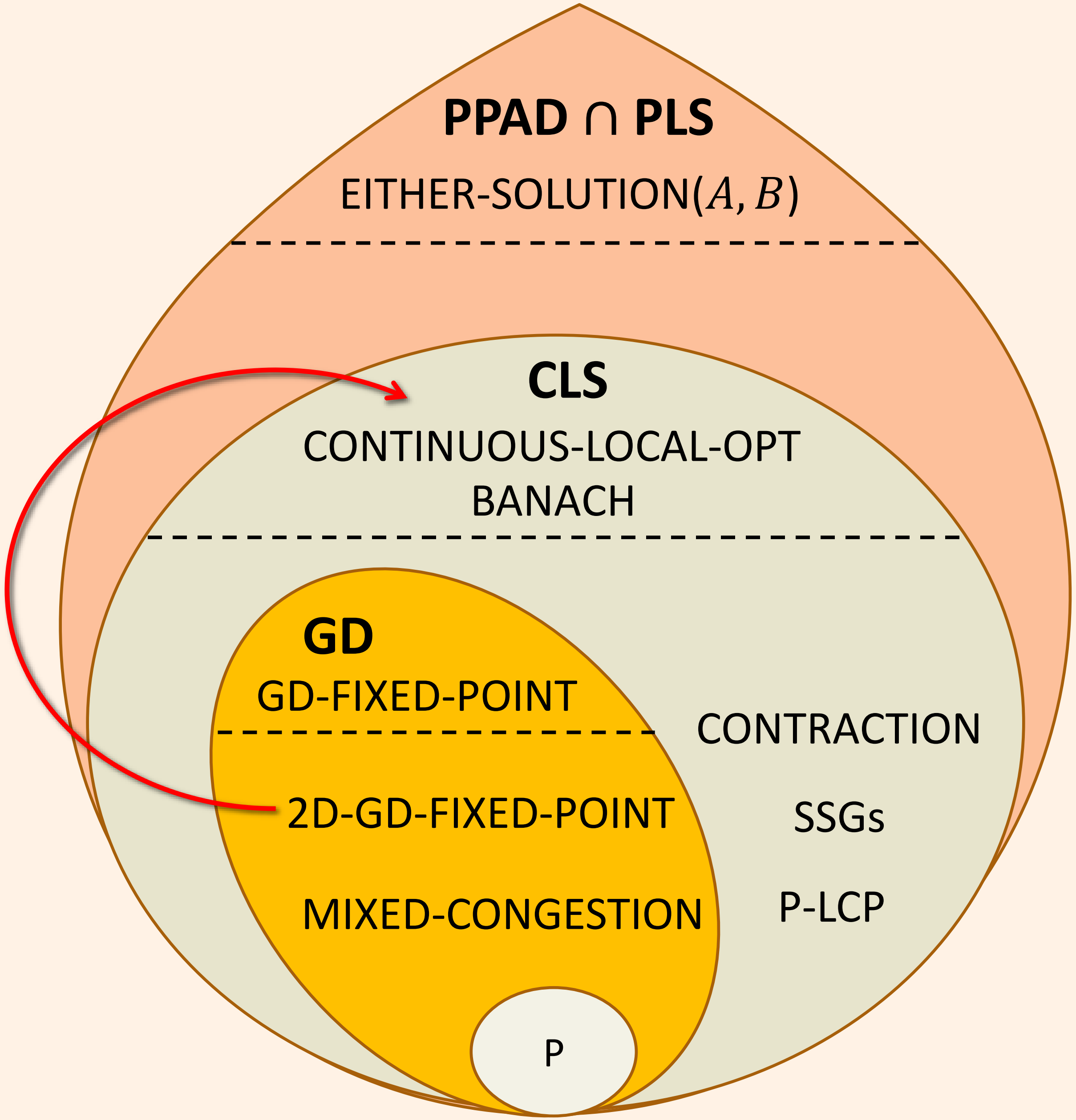
→ polynomial-time equivalent!



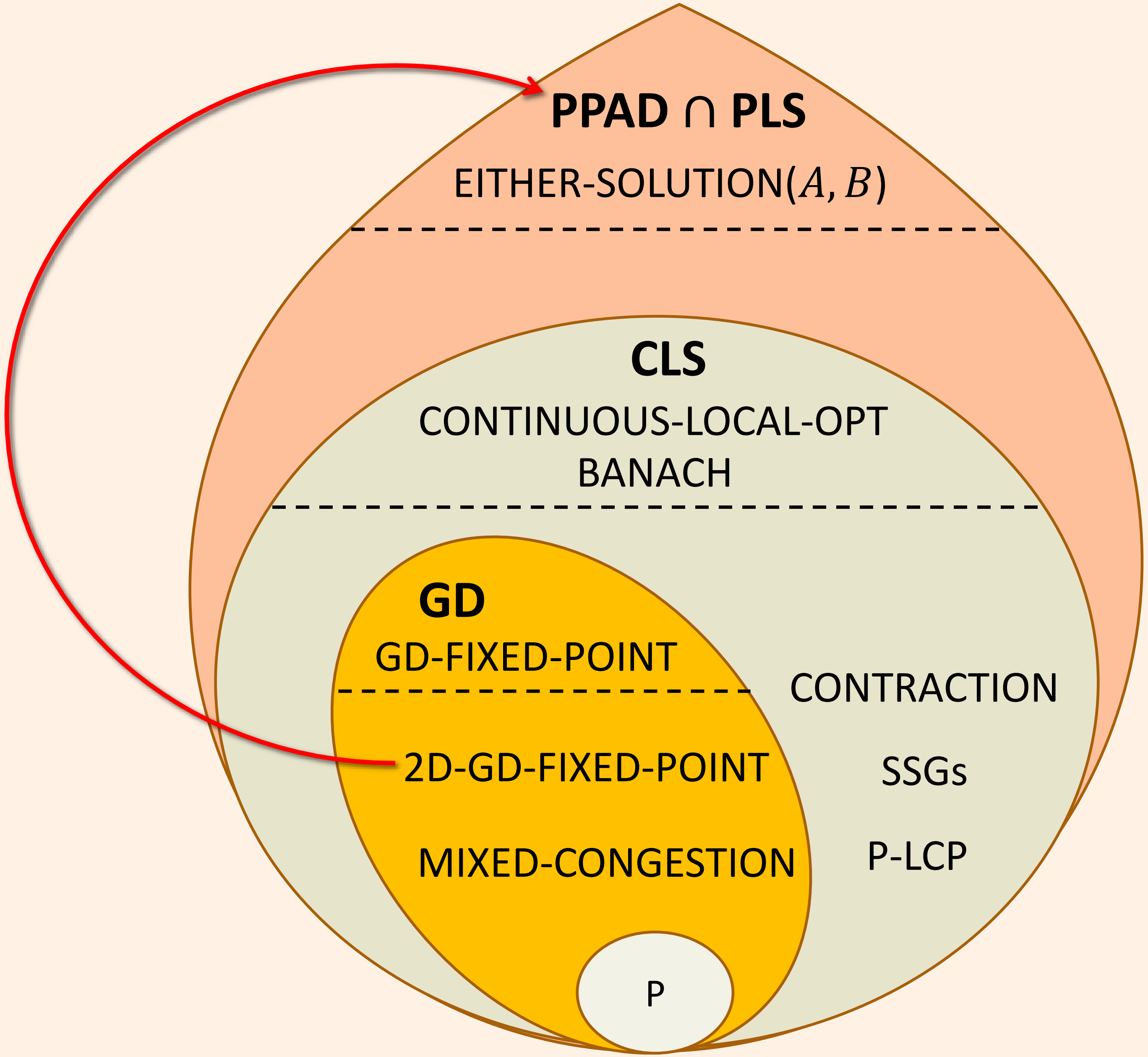
PPAD ∩ PLS
EITHER-SOLUTION(A, B)
CLS
CONTINUOUS-LOCAL-OPT
BANACH
CONTRACTION
MIXED-CONGESTION
SSGs
P-LCP
P

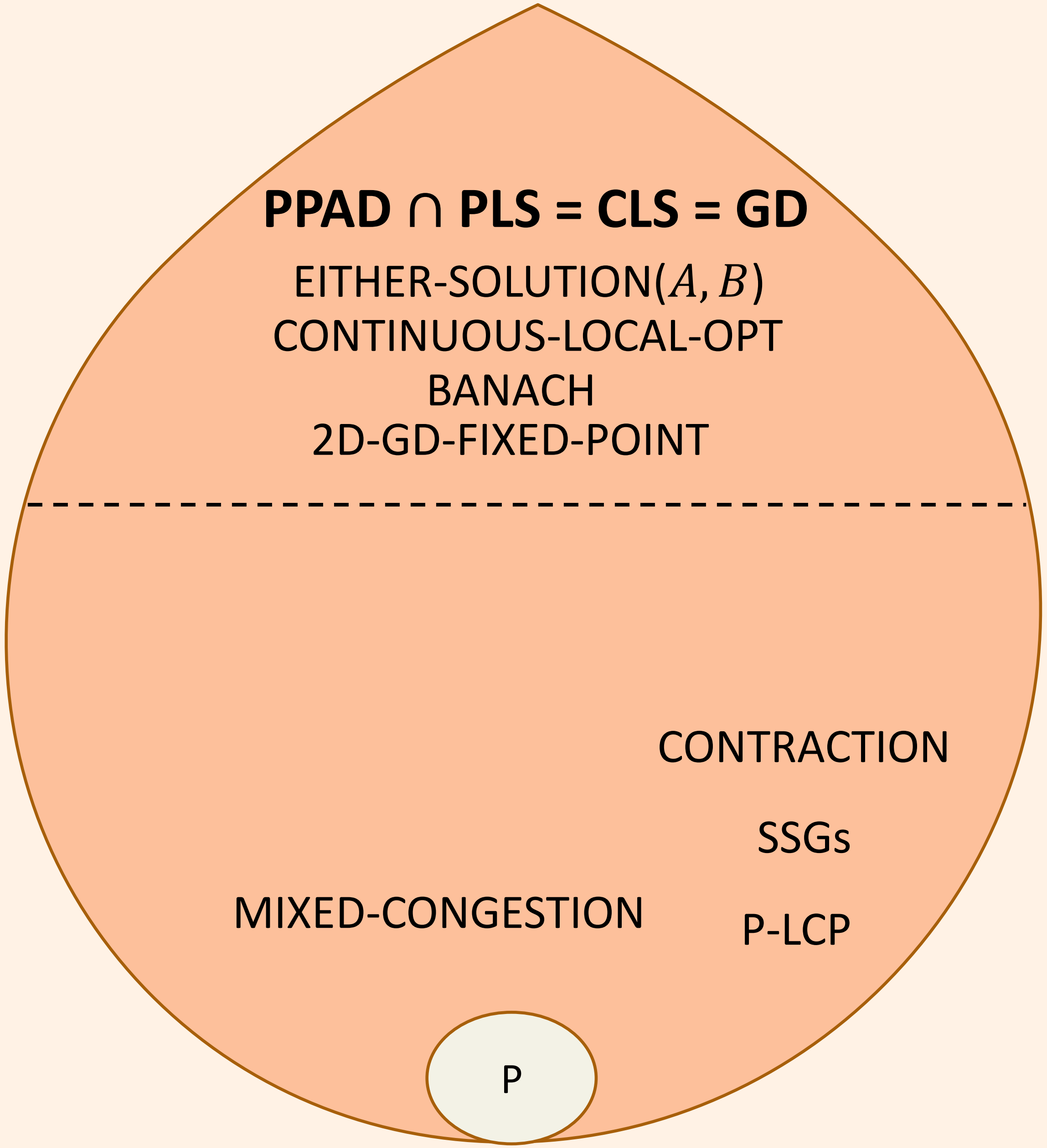

PPAD ∩ PLS
EITHER-SOLUTION(A, B)
CLS
CONTINUOUS-LOCAL-OPT
BANACH
GD
GD-FIXED-POINT
CONTRACTION
SSGs
P-LCP
MIXED-CONGESTION
P


PPAD ∩ PLS
EITHER-SOLUTION(A, B)
CLS
CONTINUOUS-LOCAL-OPT
BANACH
GD
GD-FIXED-POINT
2D-GD-FIXED-POINT
MIXED-CONGESTION
CONTRACTION
SSGs
P-LCP
P



PPAD ∩ PLS
EITHER-SOLUTION(A, B)
CLS
CONTINUOUS-LOCAL-OPT
BANACH
GD
GD-FIXED-POINT
2D-GD-FIXED-POINT
MIXED-CONGESTION
CONTRACTION
SSGs
P-LCP
P


PPAD ∩ PLS
EITHER-SOLUTION(A, B)
CLS
CONTINUOUS-LOCAL-OPT
BANACH
GD
GD-FIXED-POINT
2D-GD-FIXED-POINT
MIXED-CONGESTION
CONTRACTION
SSGs
P-LCP
P


PPAD ∩ PLS = CLS = GD
EITHER-SOLUTION(A, B)
CONTINUOUS-LOCAL-OPT
BANACH
2D-GD-FIXED-POINT
CONTRACTION
SSGs
MIXED-CONGESTION
P-LCP
P

Consequences

Consequences

- PPAD $\cap$ PLS is an interesting class!

Consequences

- PPAD $\cap$ 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

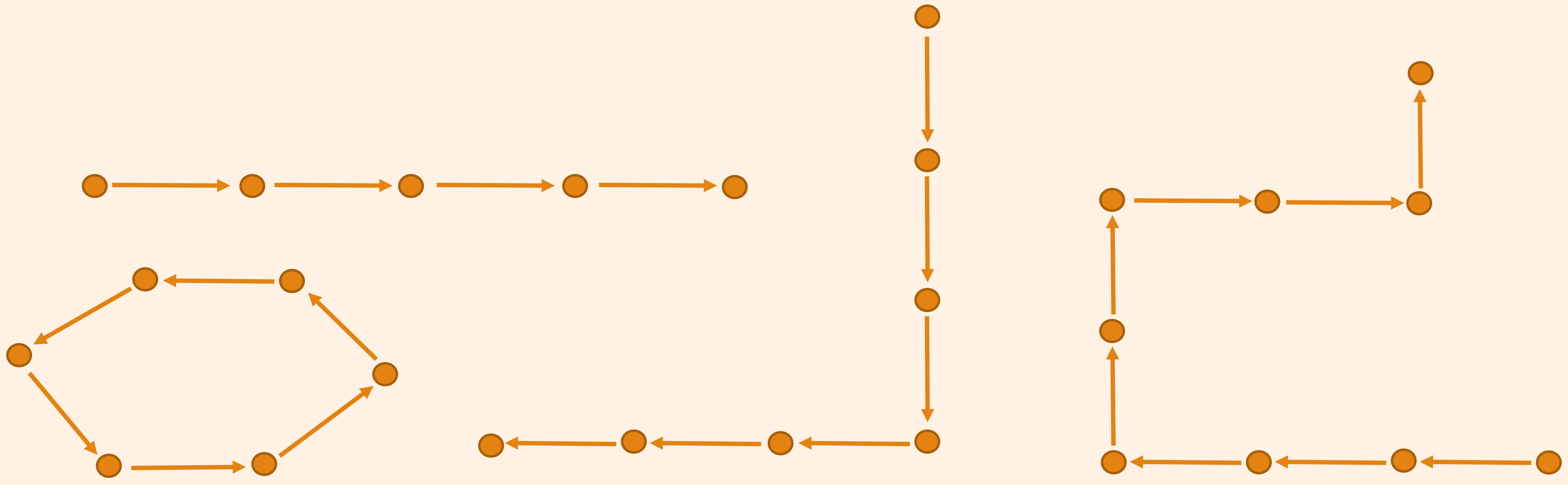
Canonical complete problem: **END-OF-LINE**

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

PPAD

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

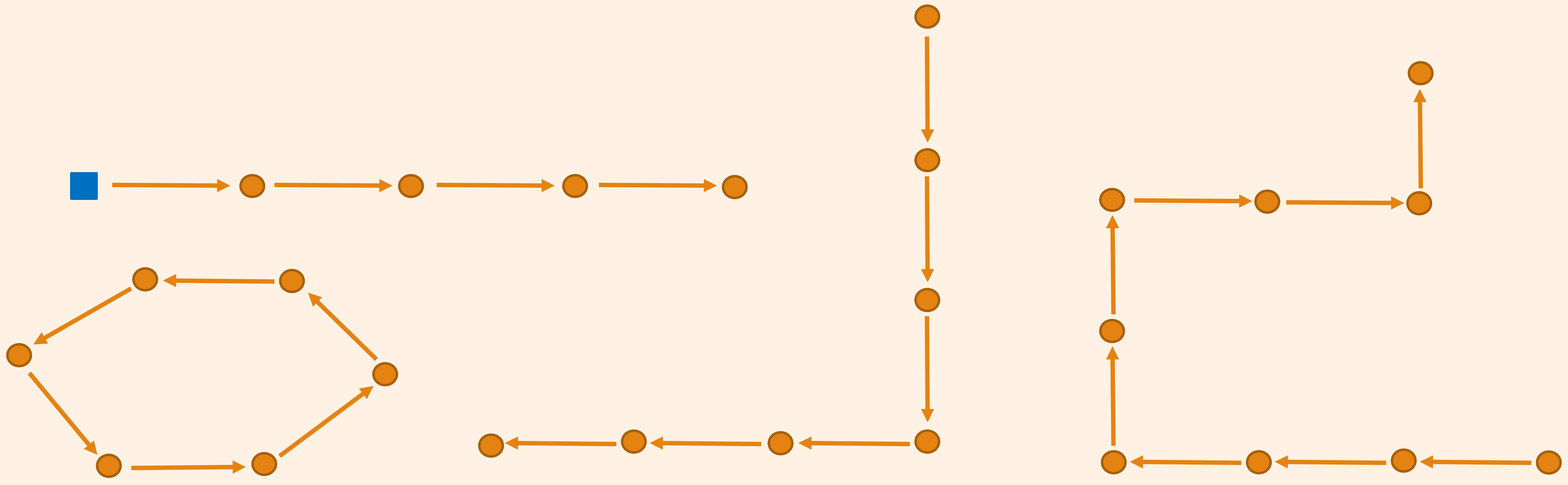
Input: directed graph of paths and cycles, and a source



PPAD

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

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

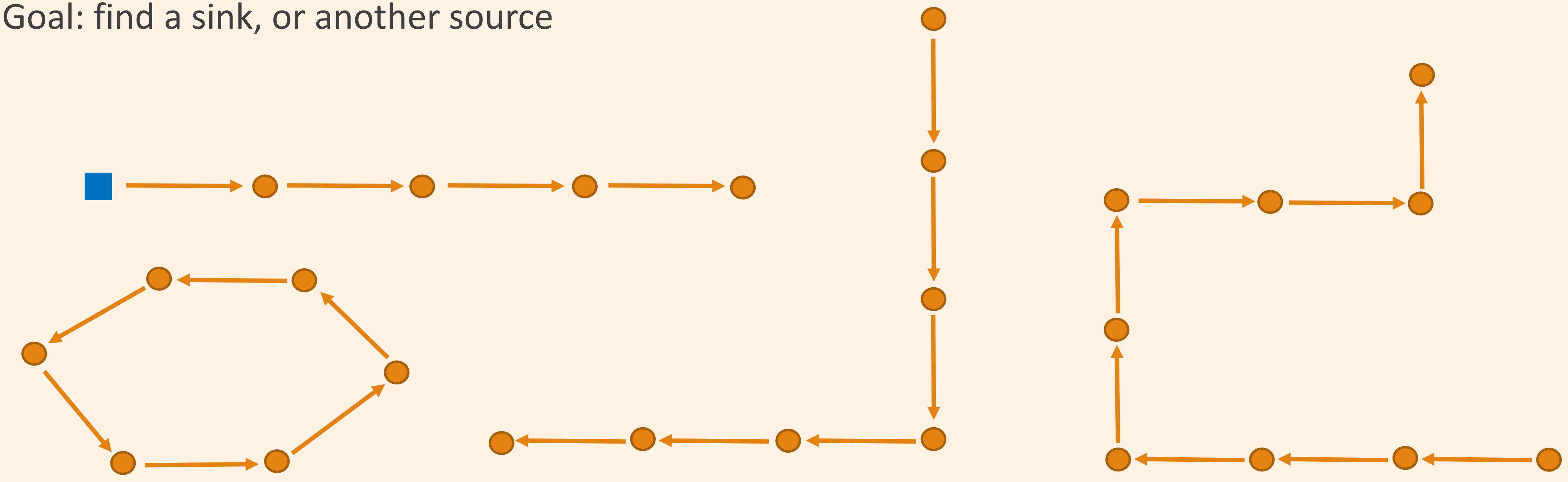


PPAD

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

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

Goal: find a sink, or another source

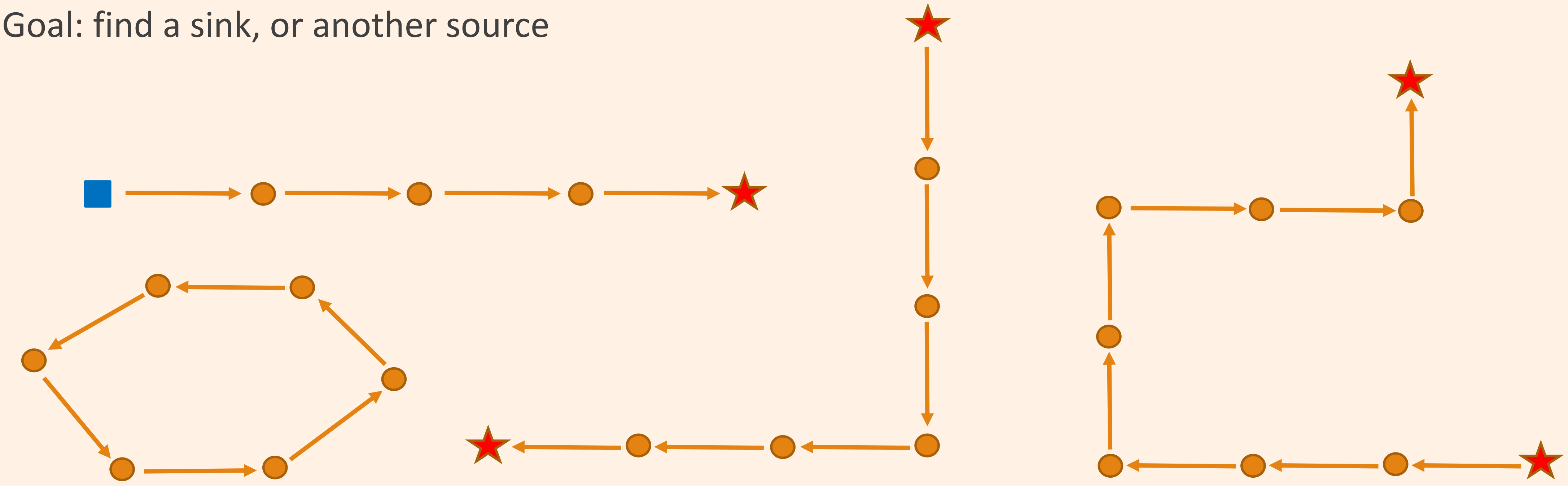


PPAD

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

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

Goal: find a sink, or another source

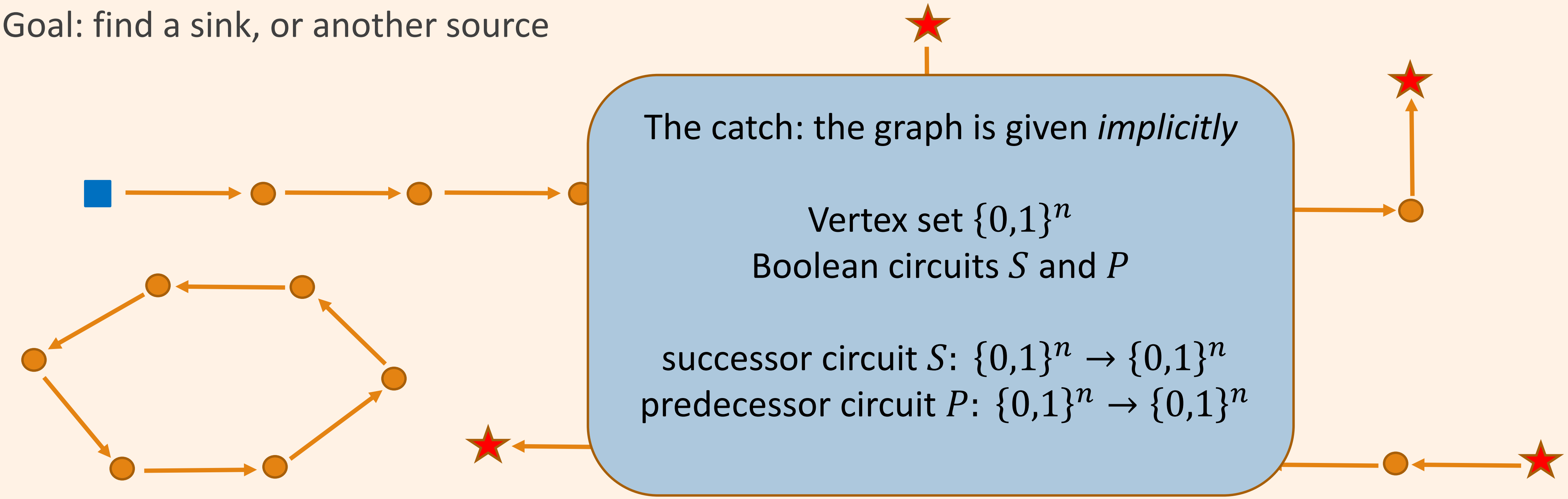


PPAD

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 *implicitly*

Vertex set $\{0,1\}^n$
Boolean circuits S and P

successor circuit S: $\{0,1\}^n \rightarrow \{0,1\}^n$
predecessor circuit P: $\{0,1\}^n \rightarrow \{0,1\}^n$

Reduction: high level

Reduction: high level

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

Reduction: high level

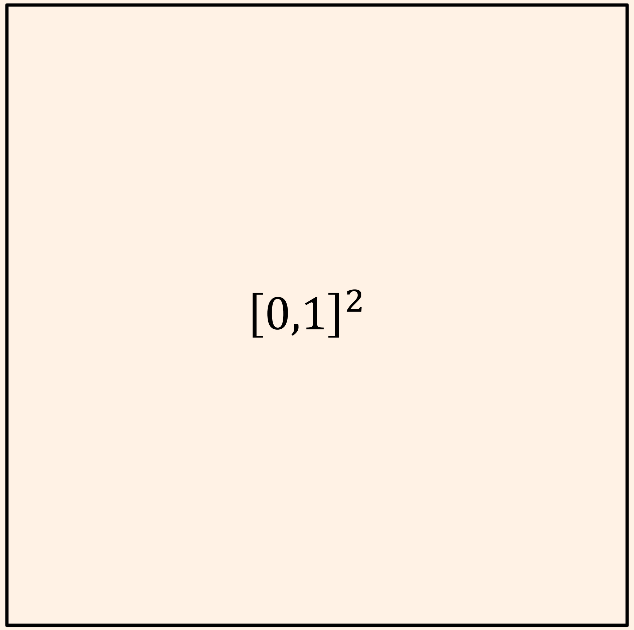
Goal: reduction from EITHER-SOLUTION(END-OF-LINE, LOCAL-OPT) to 2D-GD-FIXED-POINT

→ Construct a continuously differentiable function $f: [0,1]^2 \rightarrow \mathbb{R}$ 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

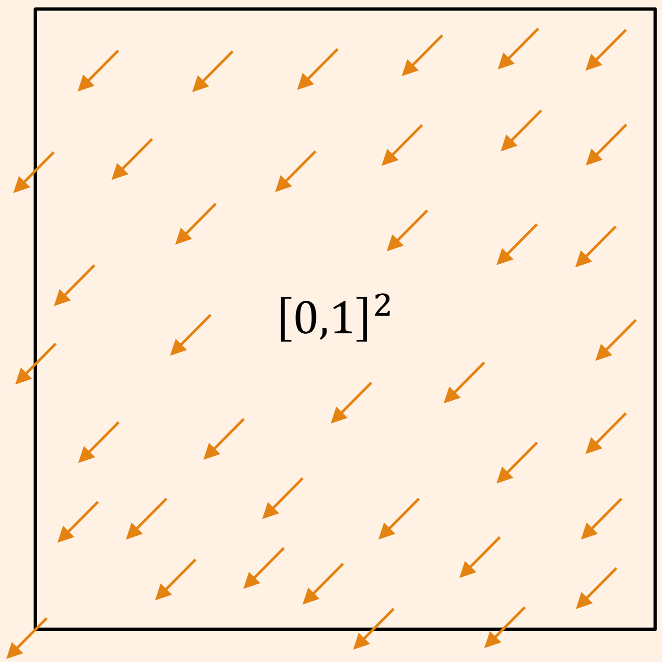
→ Construct a continuously differentiable function $f: [0,1]^2 \to \mathbb{R}$ such that any gradient descent fixed point yields a solution to the EITHER-SOLUTION instance



Reduction: high level

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

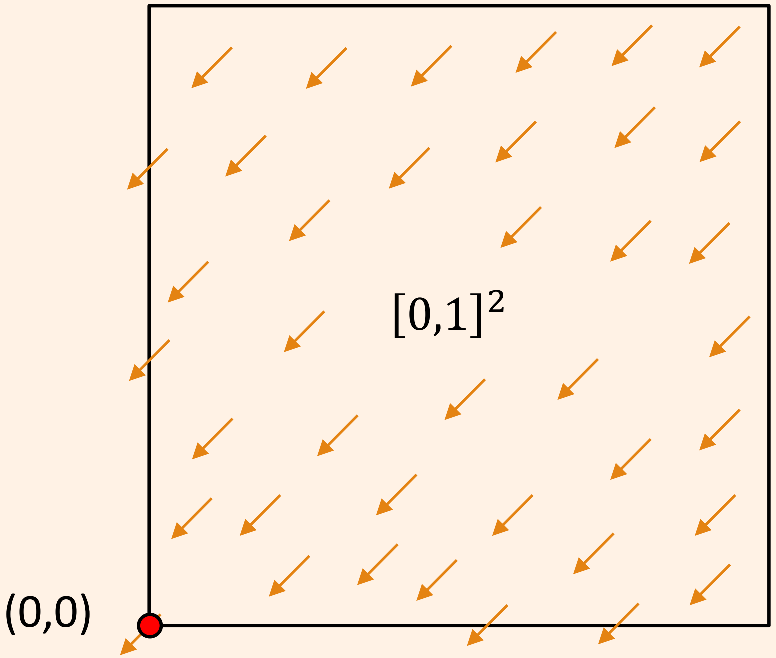
→ Construct a continuously differentiable function $f: [0,1]^2 \rightarrow \mathbb{R}$ such that any gradient descent fixed point yields a solution to the EITHER-SOLUTION instance



Reduction: high level

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

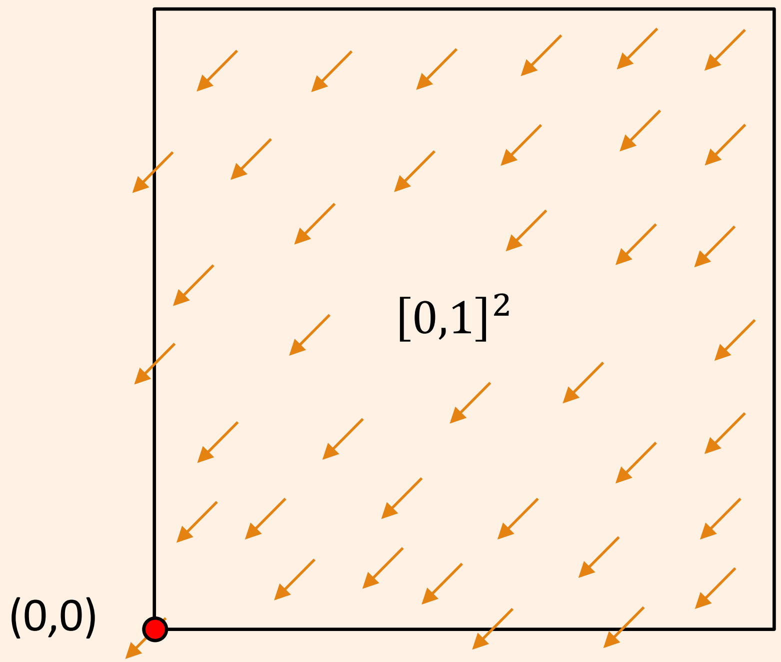
→ Construct a continuously differentiable function $f: [0,1]^2 \to \mathbb{R}$ such that any gradient descent fixed point yields a solution to the EITHER-SOLUTION instance

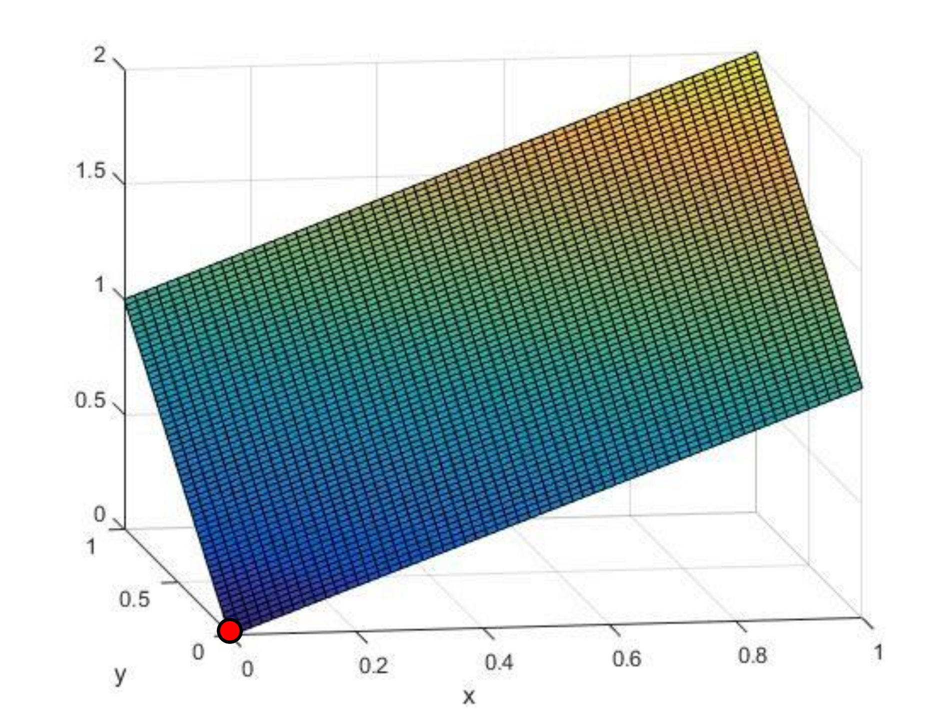


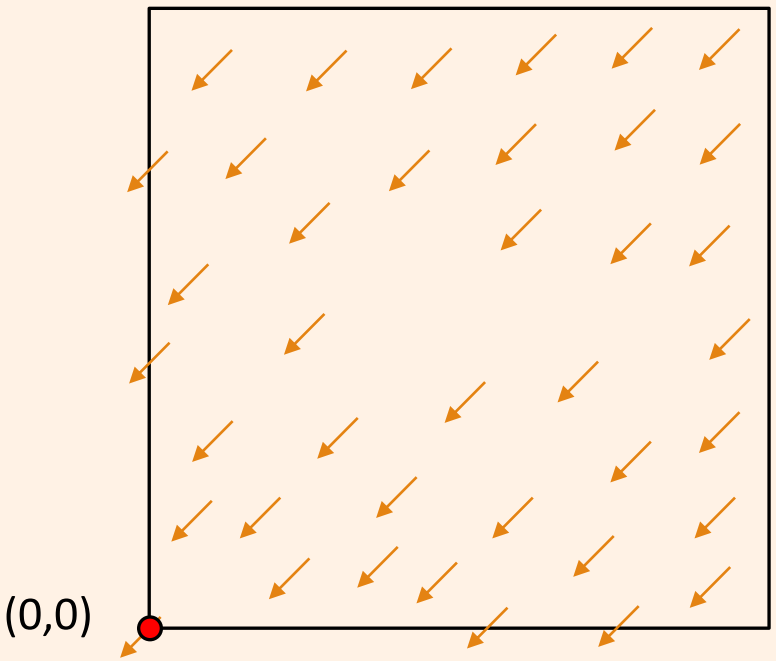
Reduction: high level

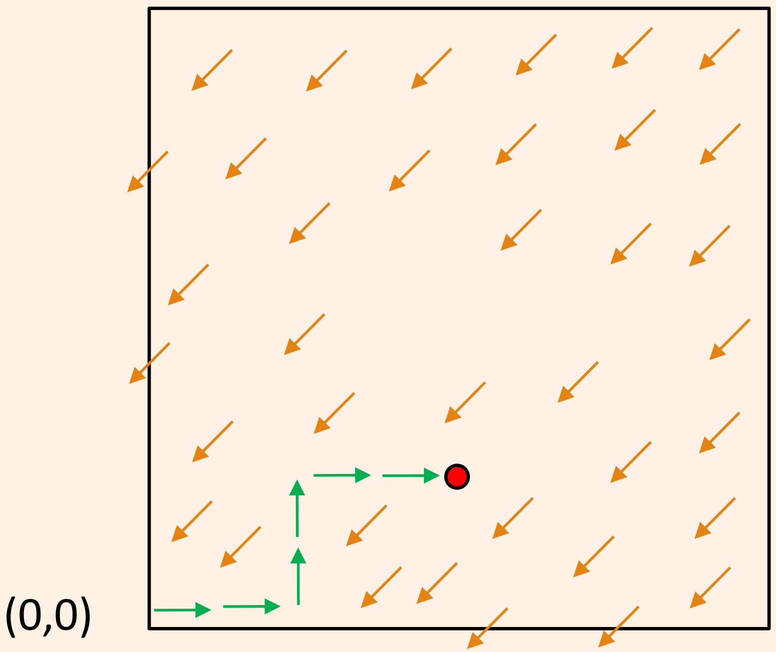
Goal: reduction from EITHER-SOLUTION(END-OF-LINE, LOCAL-OPT) to 2D-GD-FIXED-POINT

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





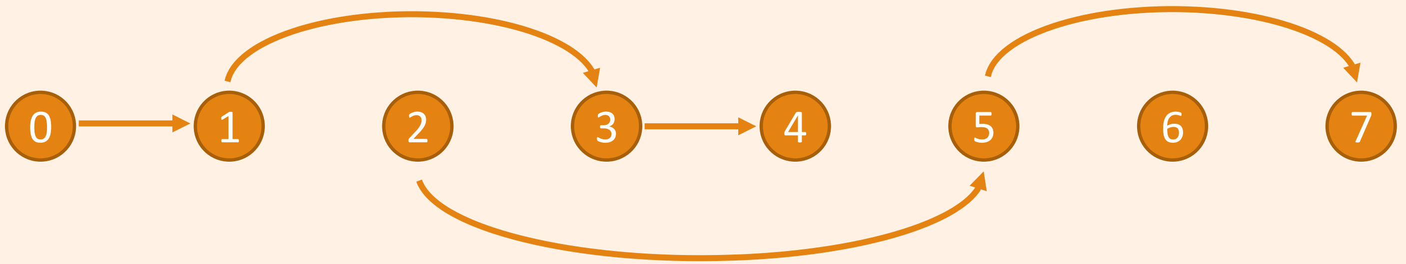

(0,0)


(0,0)

Warm up: Monotone-End-of-Line

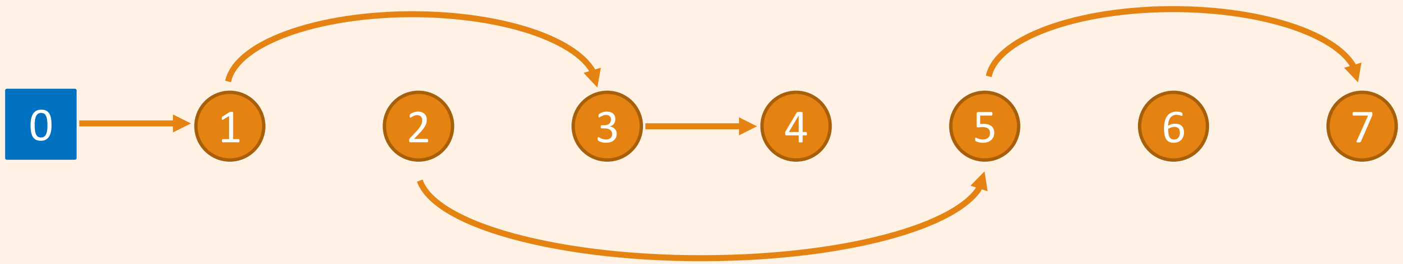
Warm up: Monotone-End-of-Line

Warm up: Monotone-End-of-Line



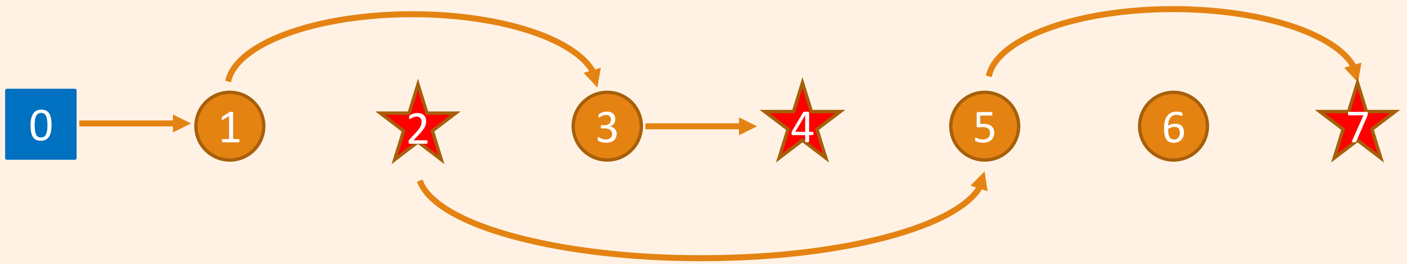
Special case of END-OF-LINE: No backward edges allowed!

Warm up: Monotone-End-of-Line

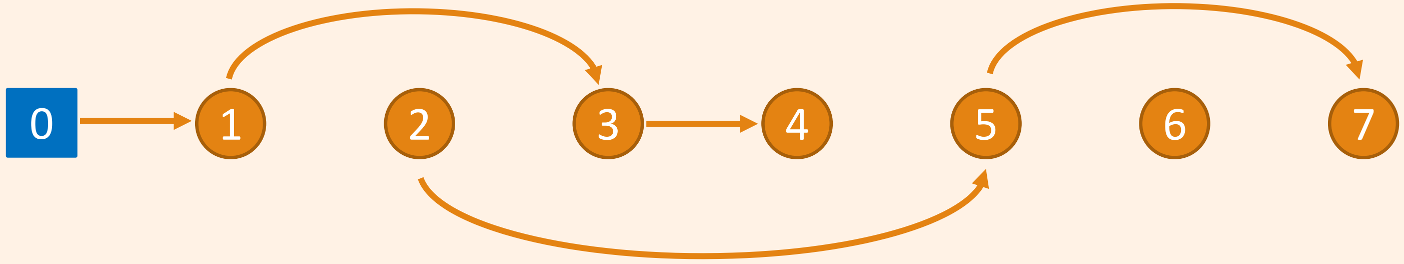


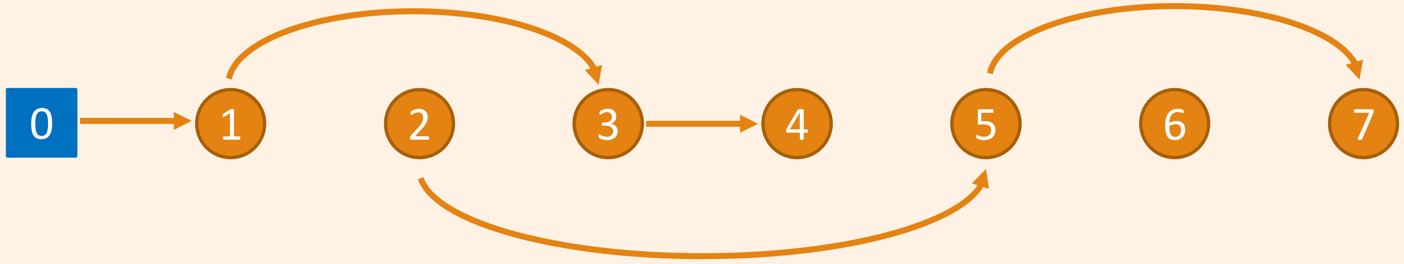
Special case of END-OF-LINE: No backward edges allowed!

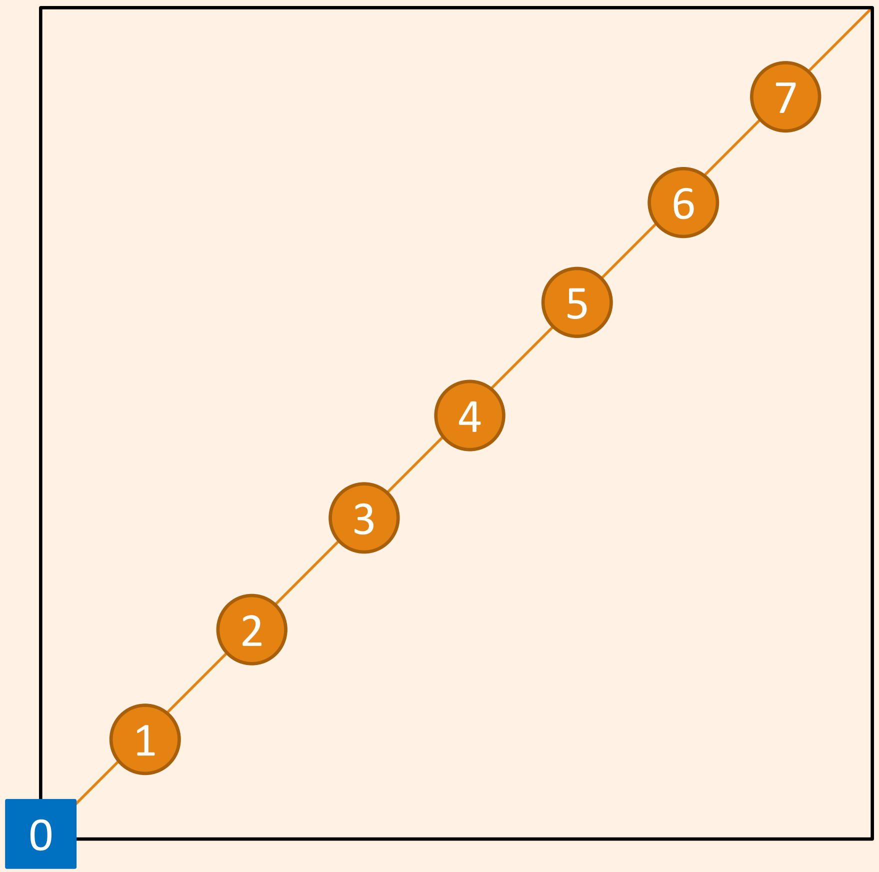
Warm up: Monotone-End-of-Line

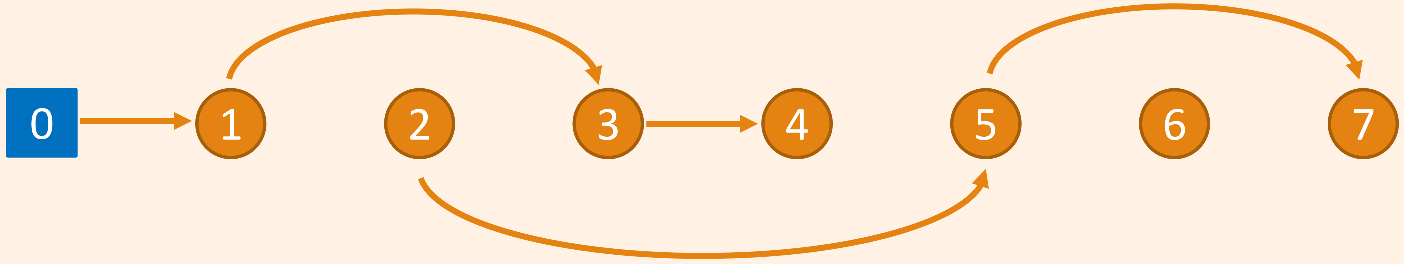


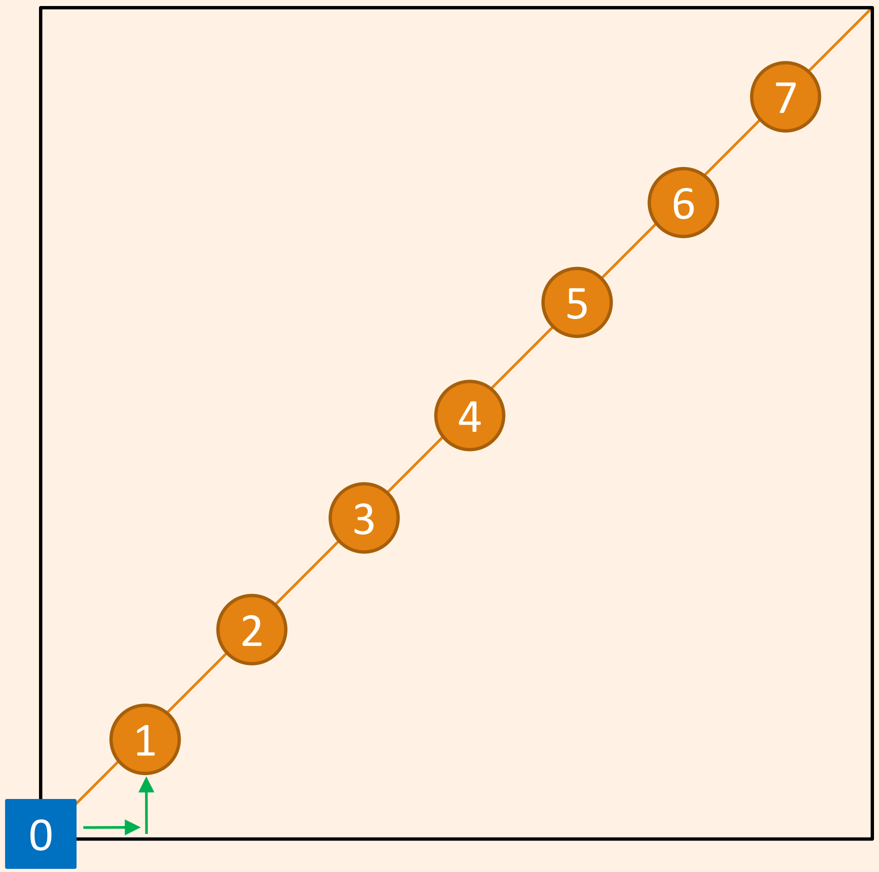
Special case of END-OF-LINE: No backward edges allowed!

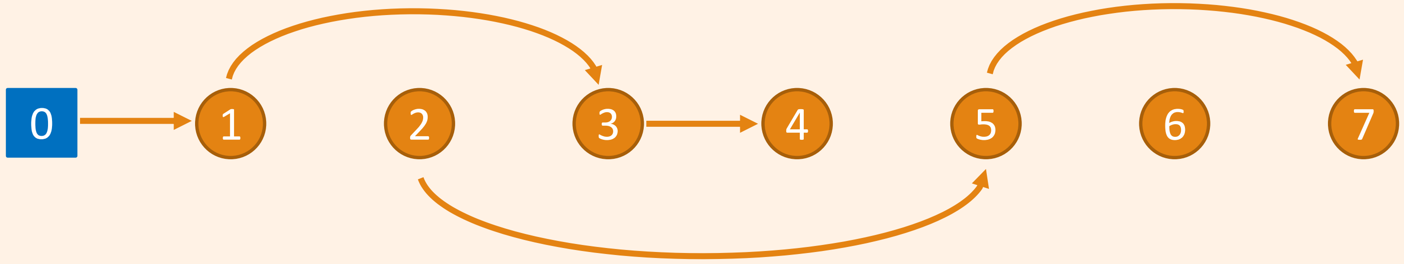

0
1
2
3
4
5
6
7

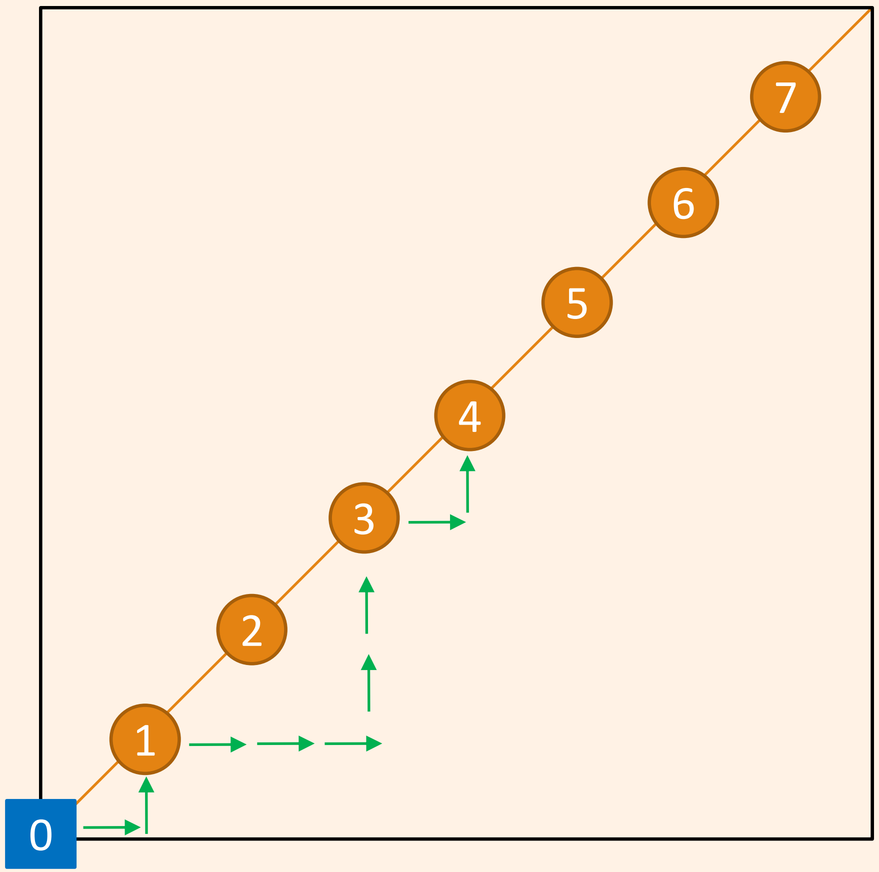

0
1
2
3
4
5
6
7

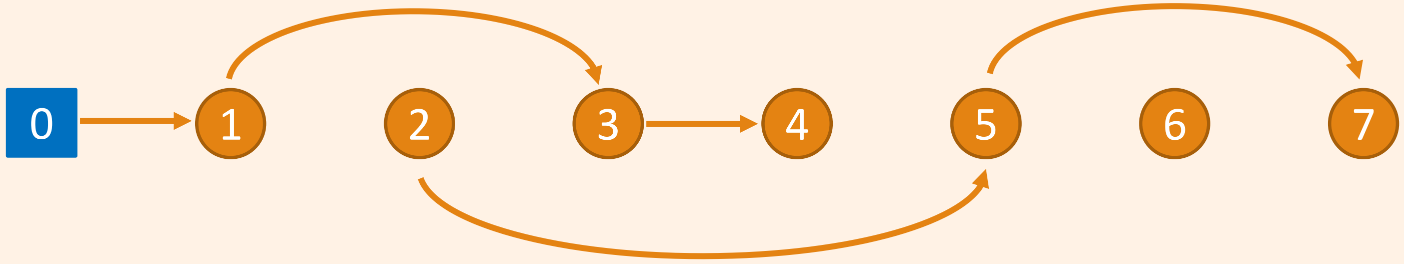

0
1
2
3
4
5
6
7

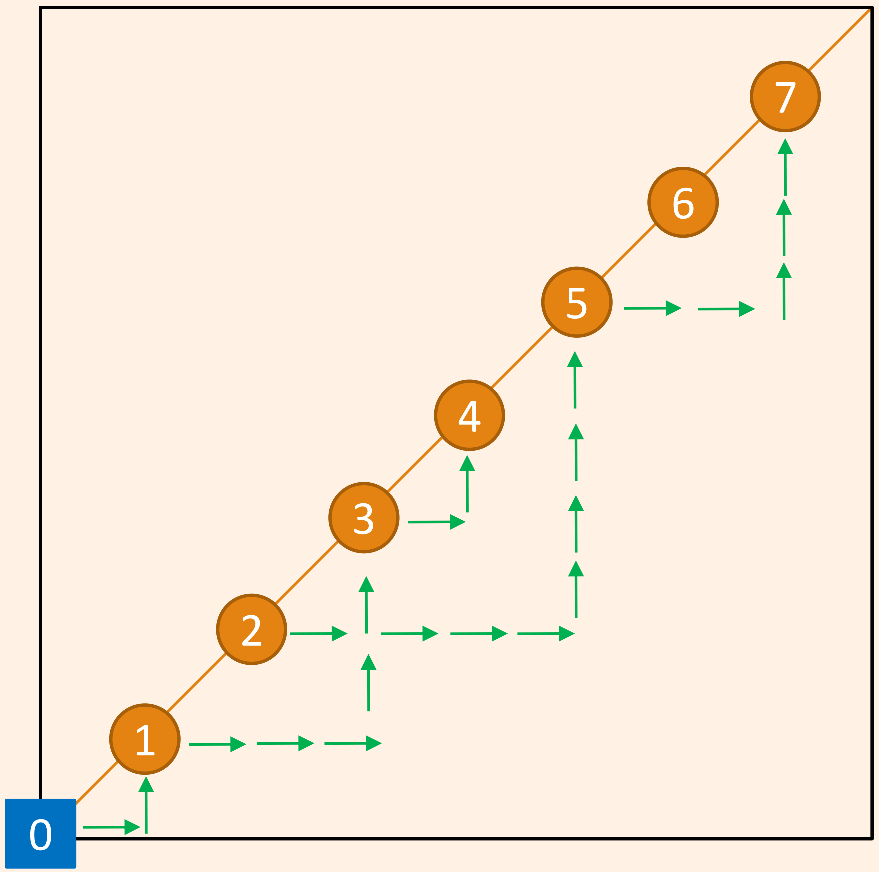

0
1
2
3
4
5
6
7

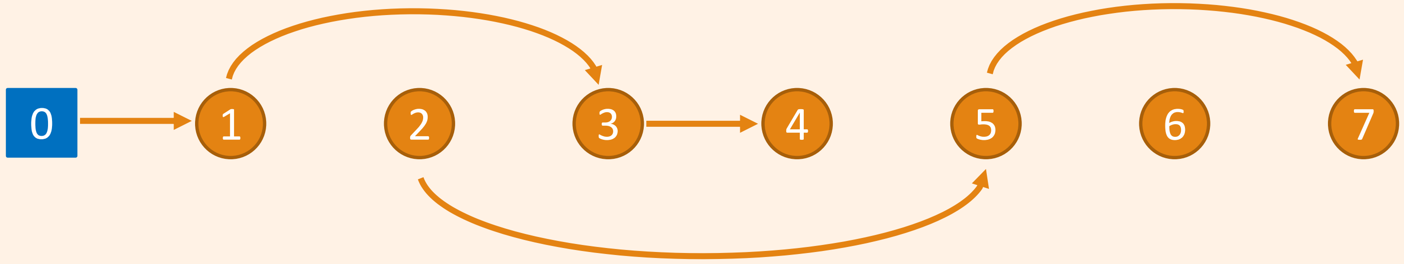

7
6
5
4
3
2
1
0

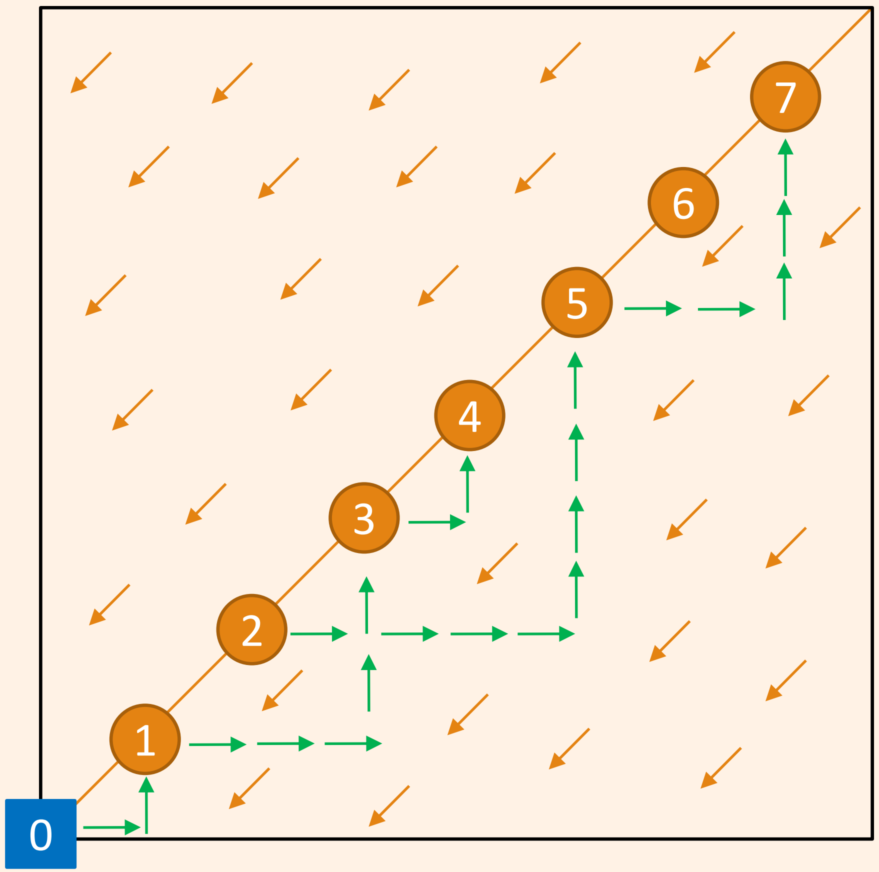

0
1
2
3
4
5
6
7

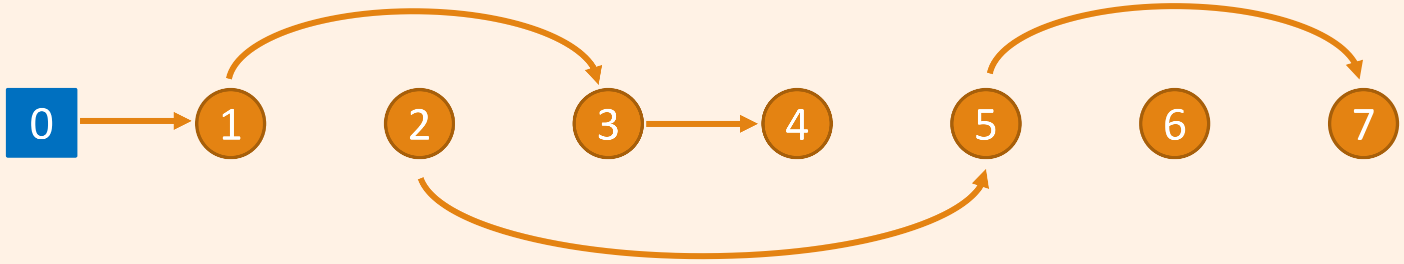

0
1
2
3
4
5
6
7

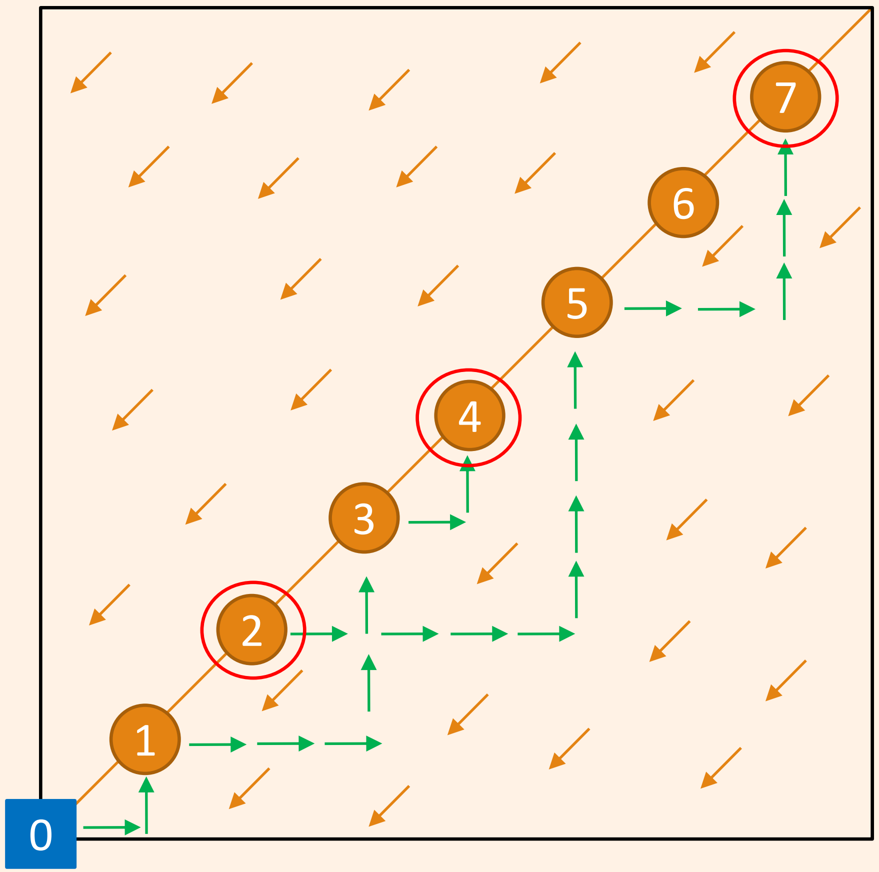

0
1
2
3
4
5
6
7

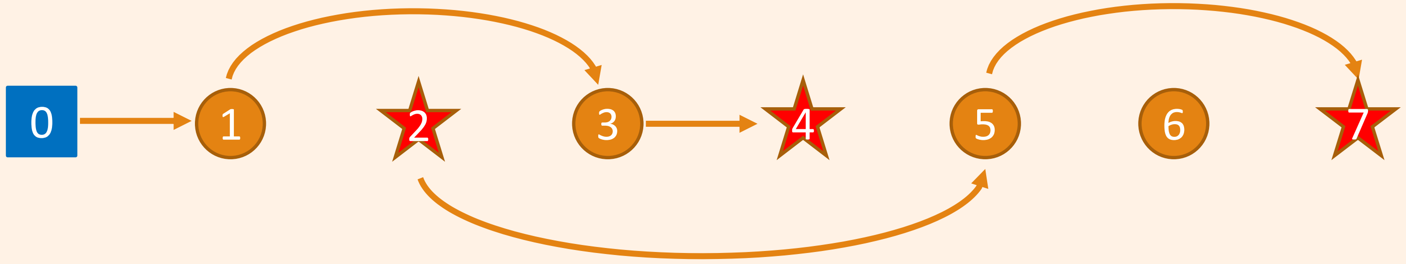

7
6
5
4
3
2
1
0

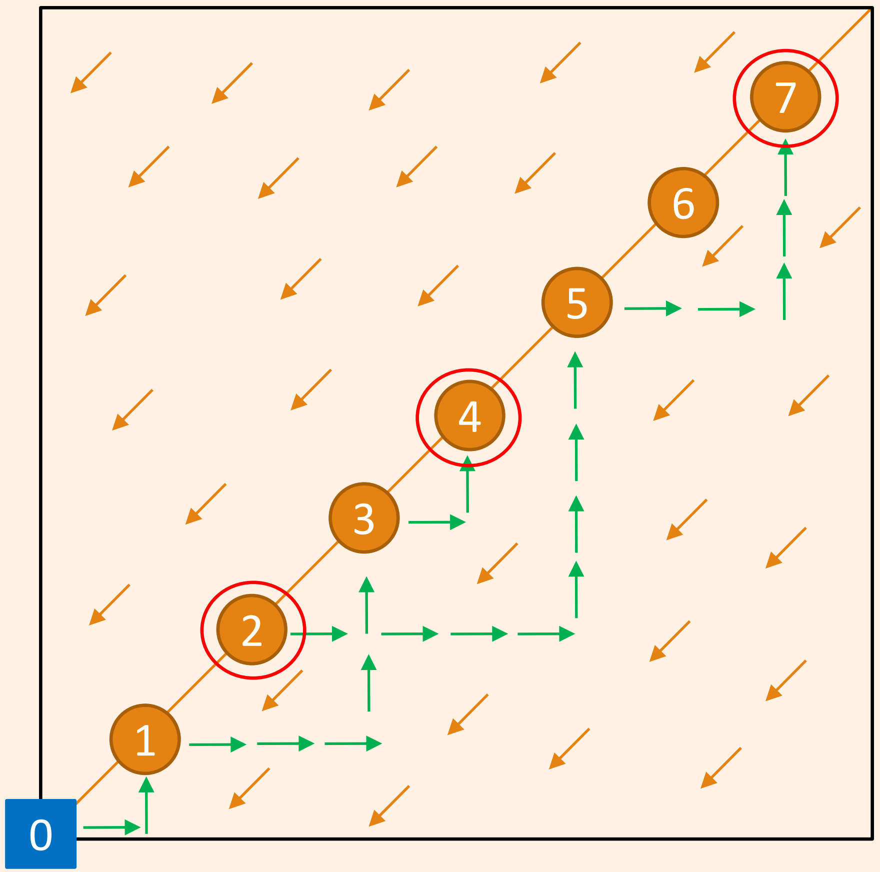

0
1
2
3
4
5
6
7

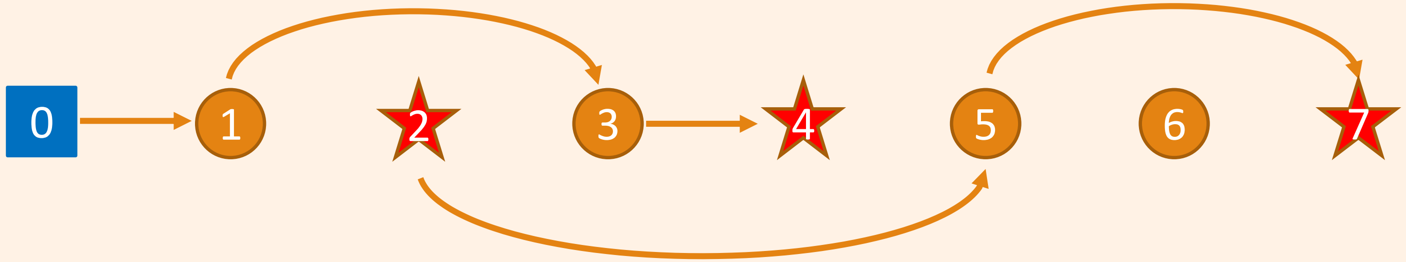

7
6
5
4
3
2
1
0

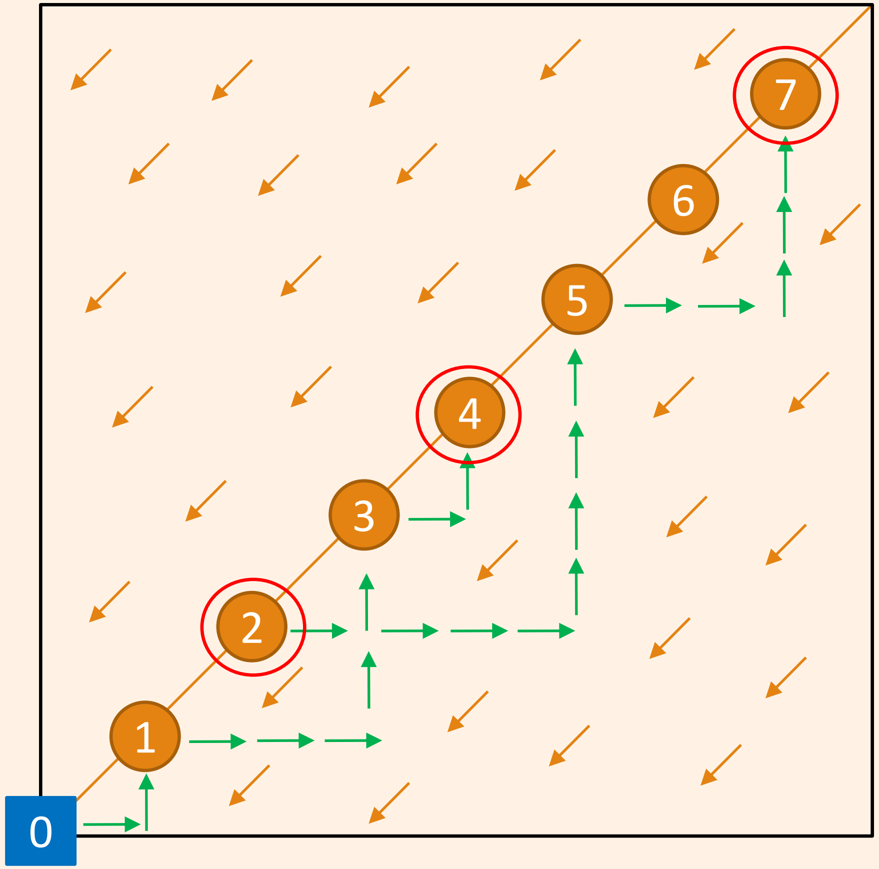

0
1
2
3
4
5
6
7


0
1
2
3
4
5
6
7


0
1
2
3
4
5
6
7


0
1
2
3
4
5
6
7





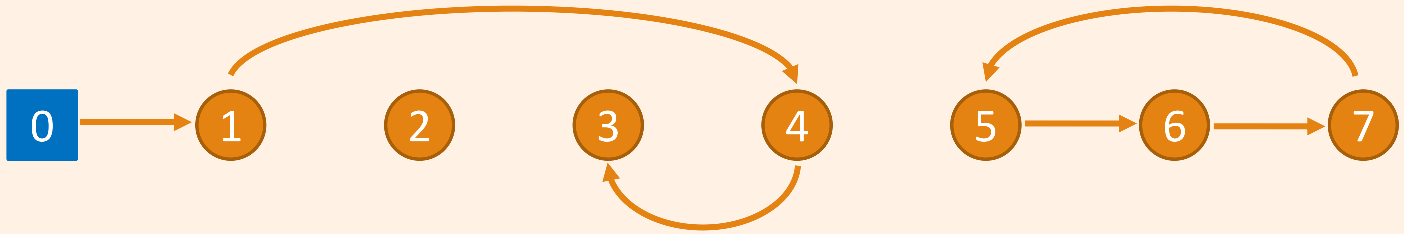
Locally computable!

[Hubáček-Yogev, 2017] for CLS

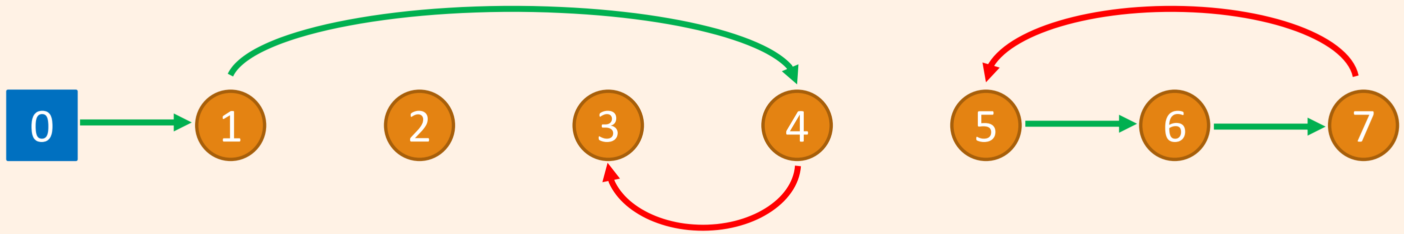
Back to standard End-of-Line

Back to standard End-of-Line

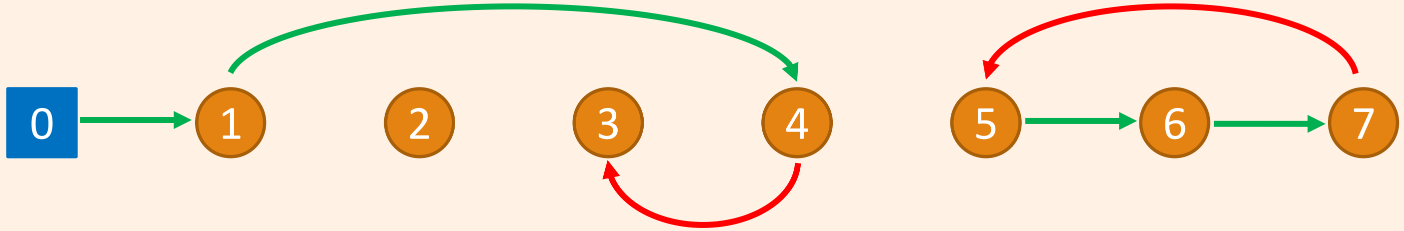
Back to standard End-of-Line

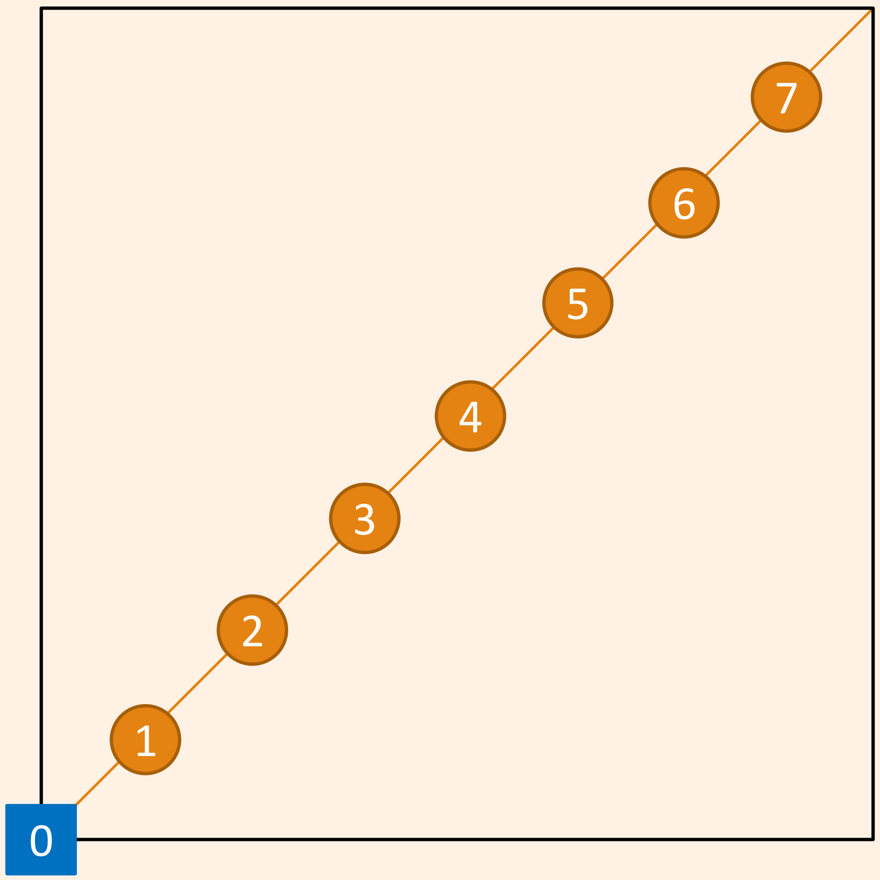


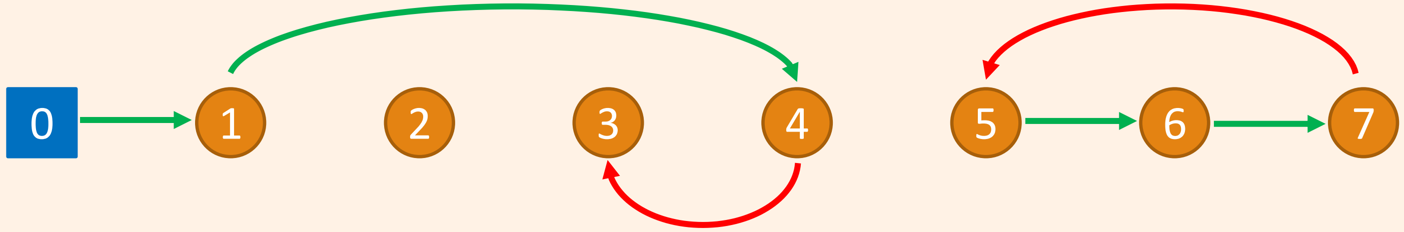
Back to standard End-of-Line

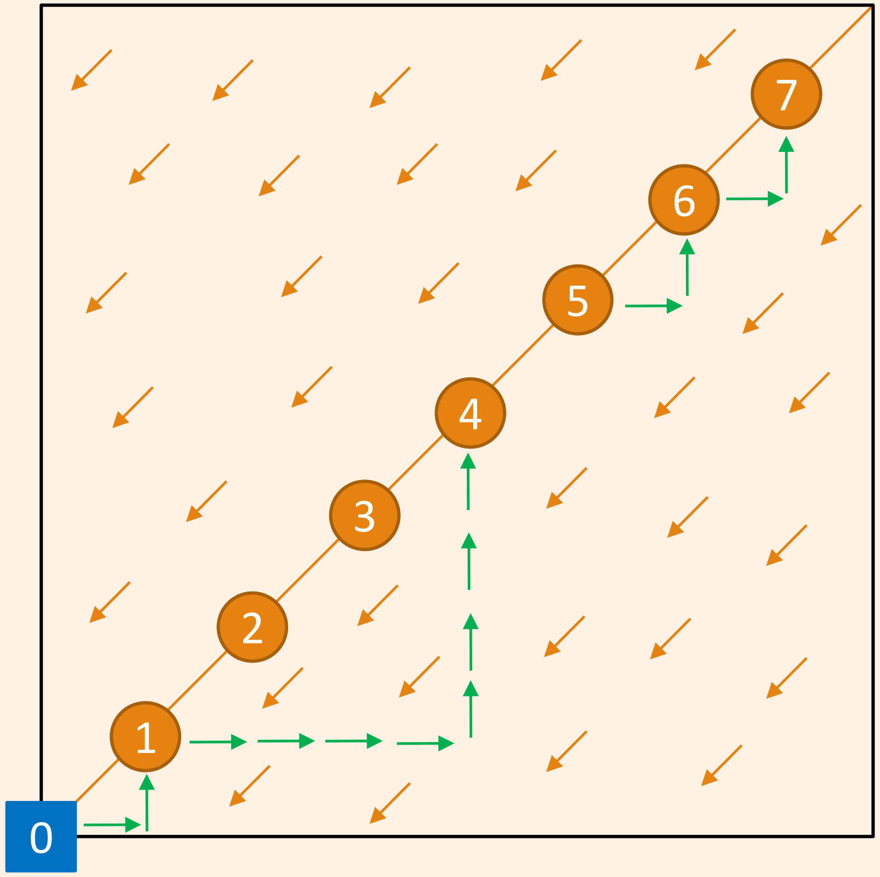


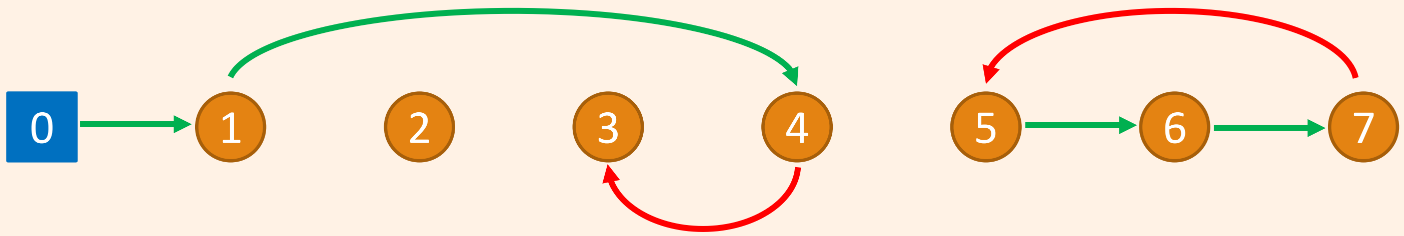
green edges: forward
red edges: backwards

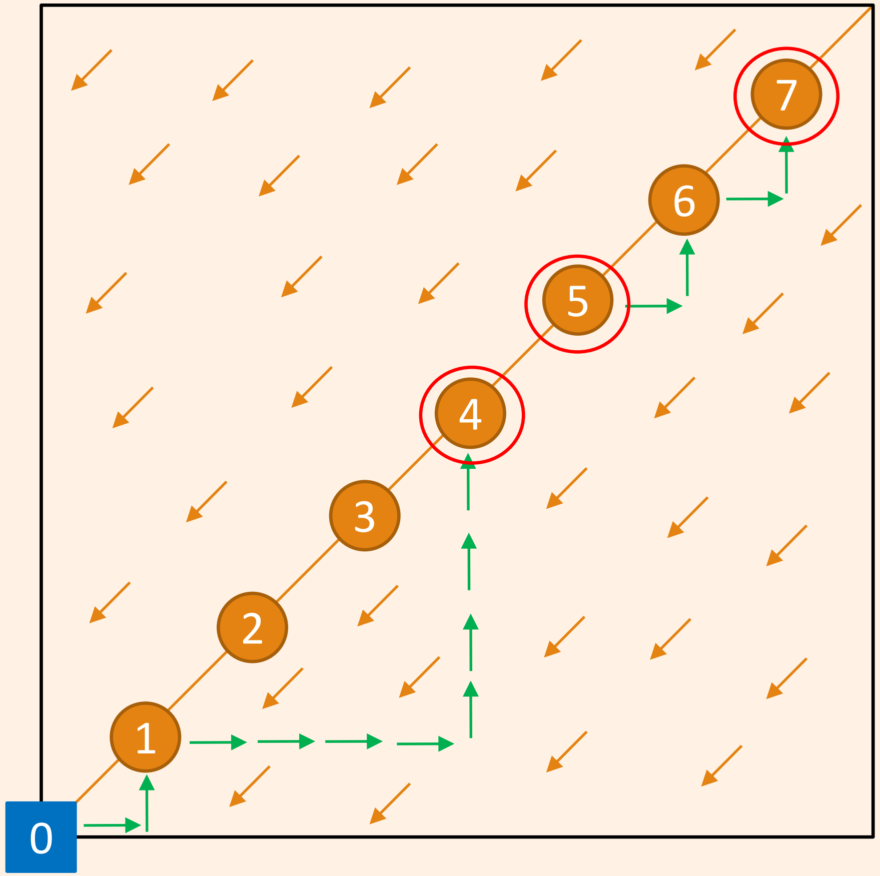

0
1
2
3
4
5
6
7

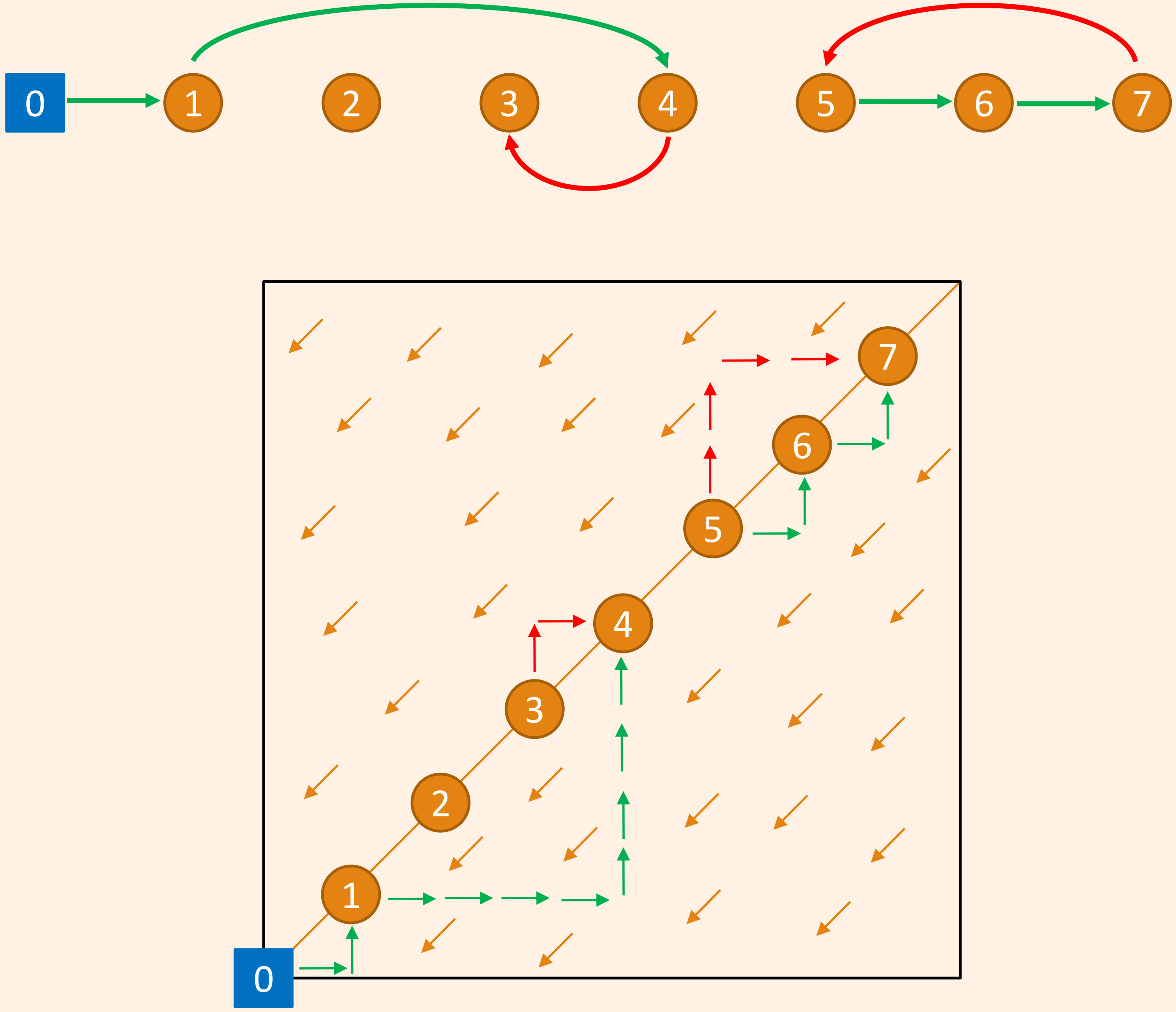

0
1
2
3
4
5
6
7

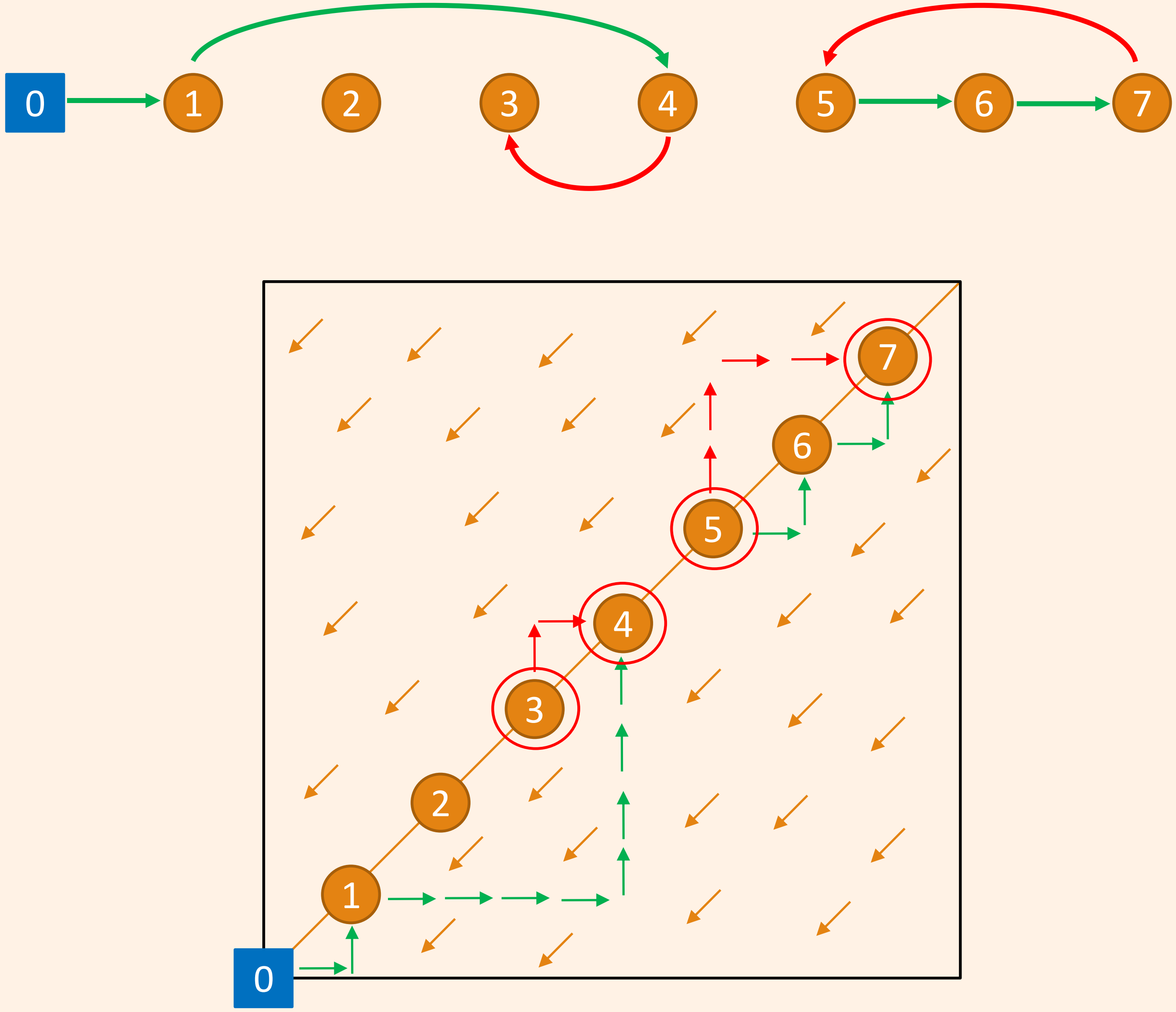

0
1
2
3
4
5
6
7

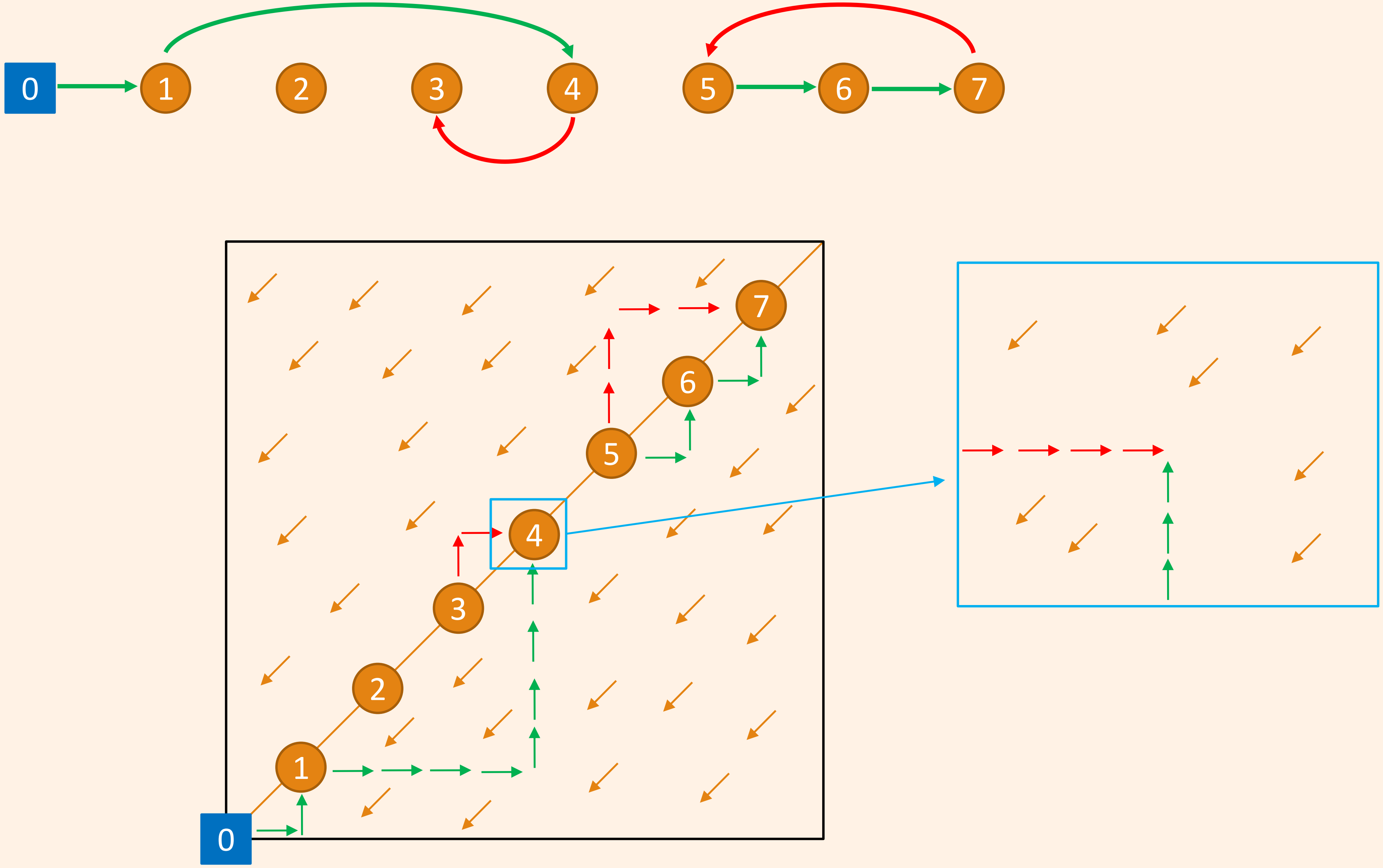

0
1
2
3
4
5
6
7


0
1
2
3
4
5
6
7

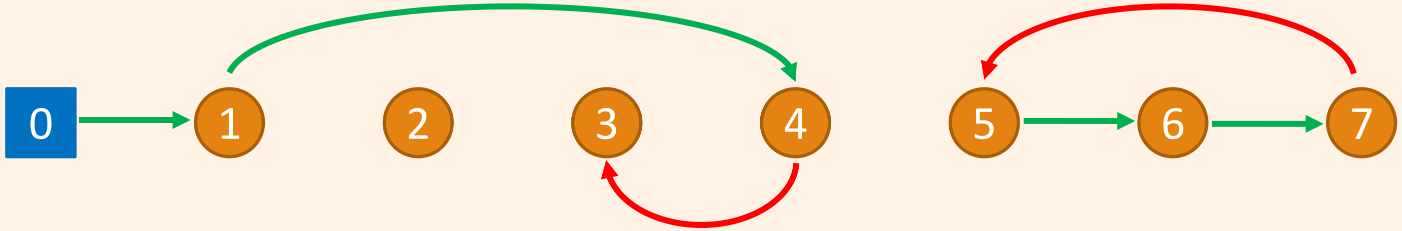

7
6
5
4
3
2
1
0

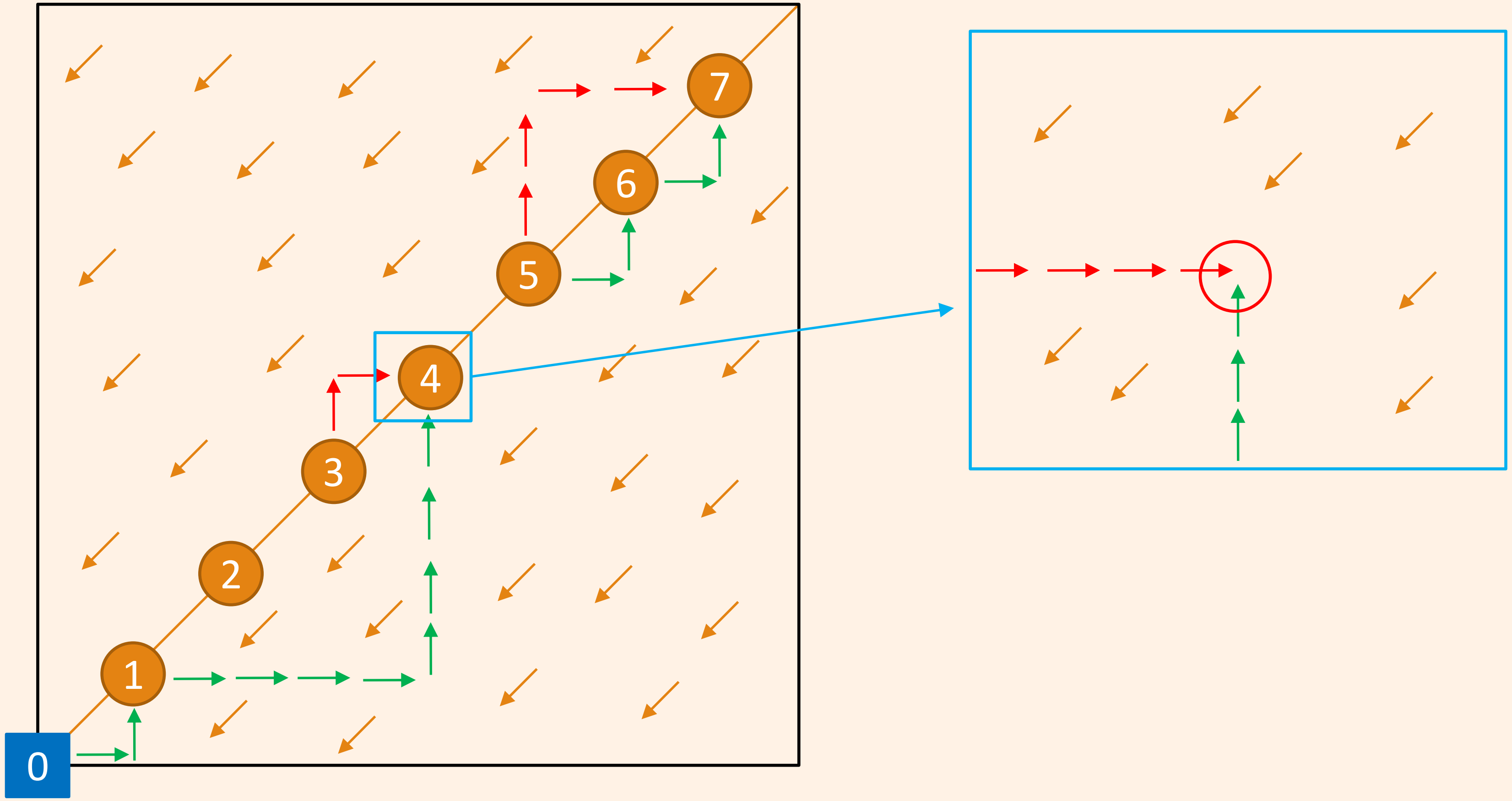

0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7

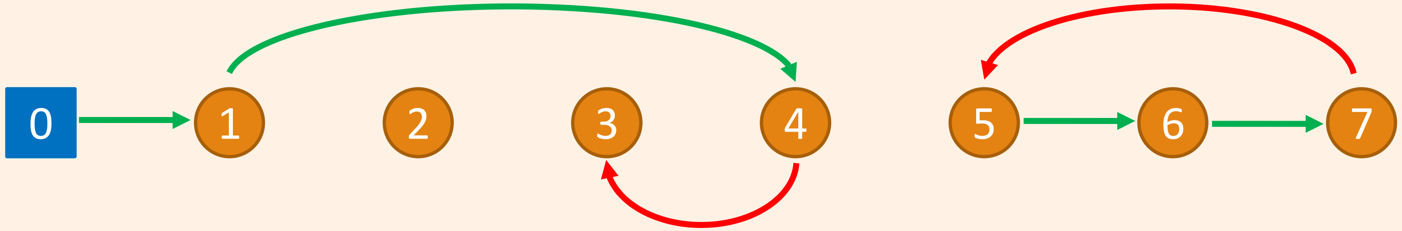

0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7

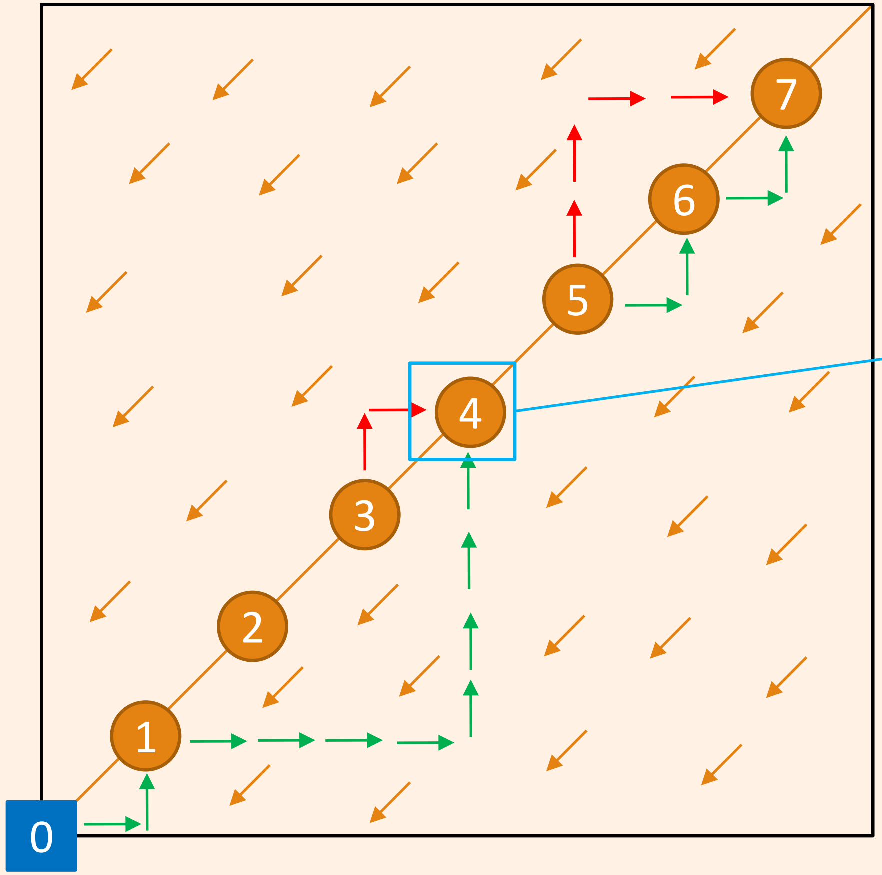


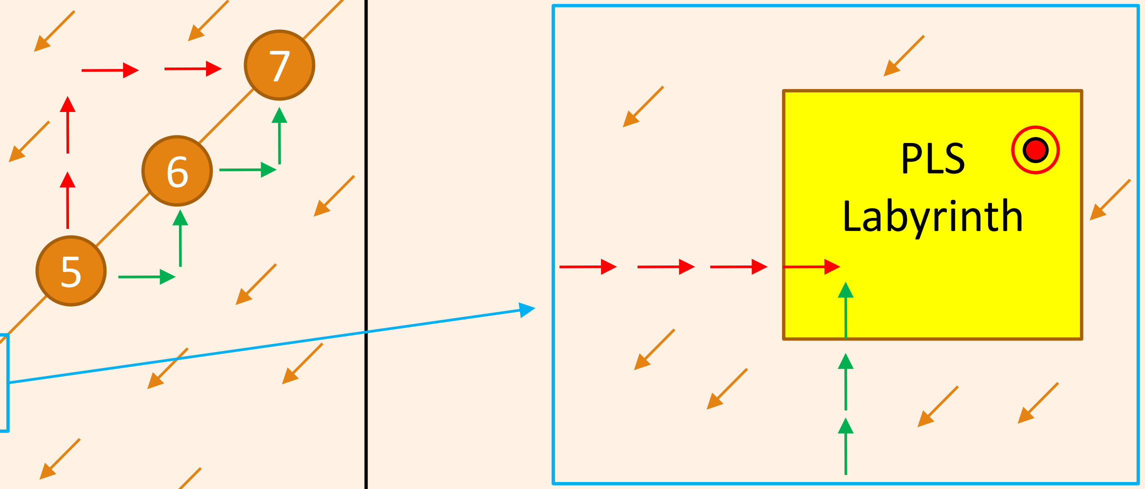
0
1
2
3
4
5
6
7


0
1
2
3
4
5
6
7

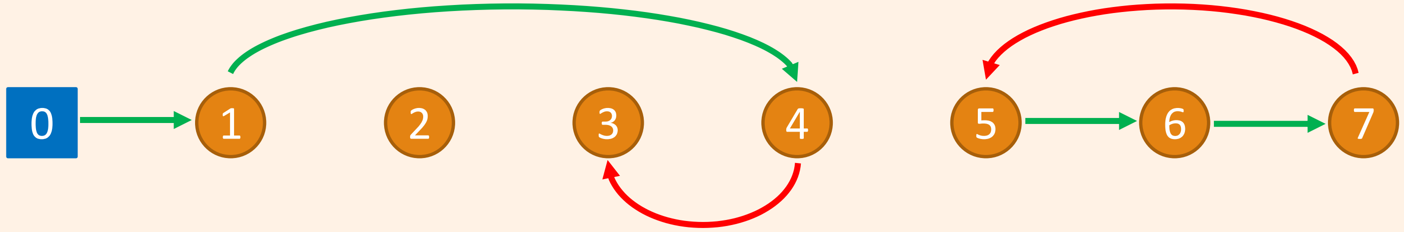

0
1
2
3
4
5
6
7

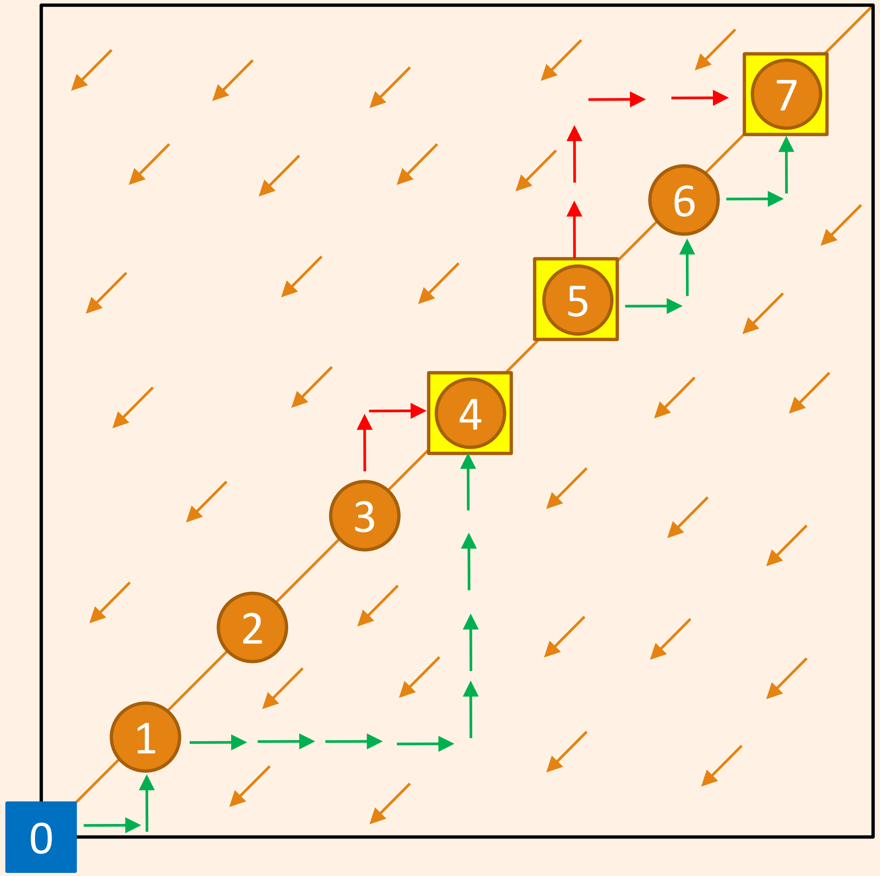


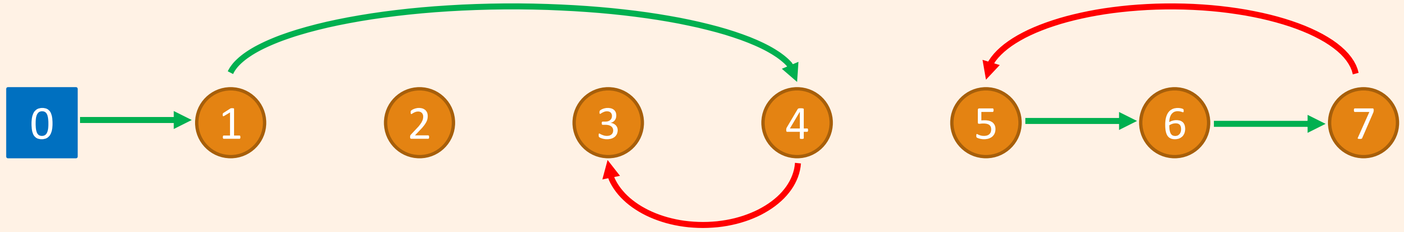


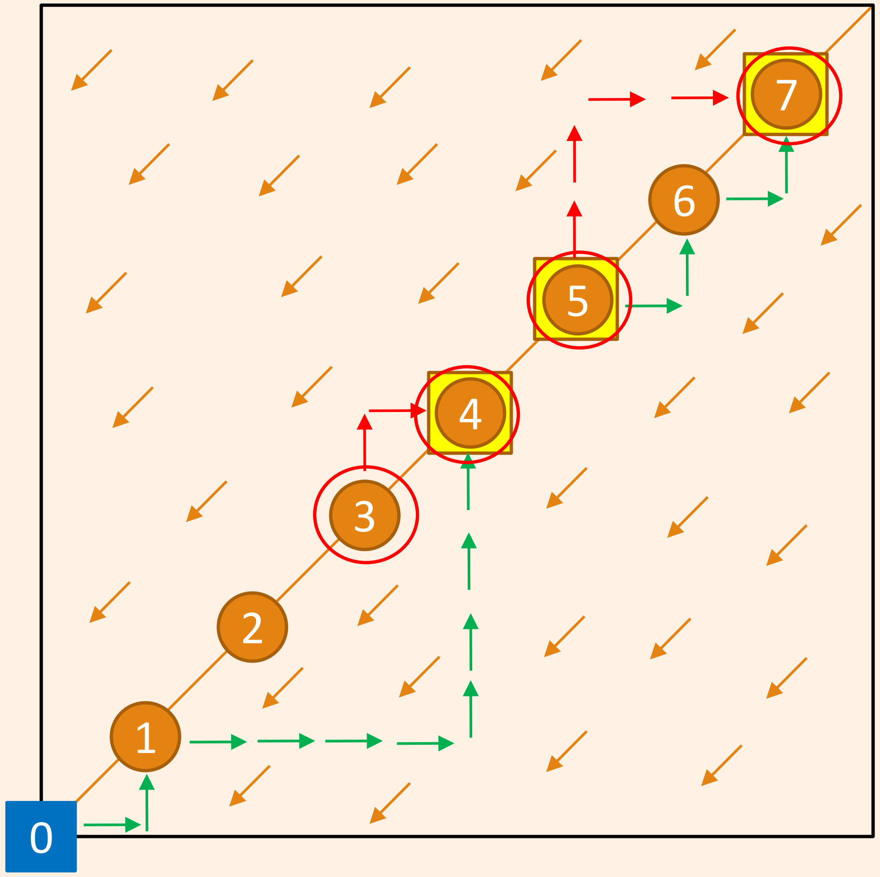


Requires solving the PLS instance!

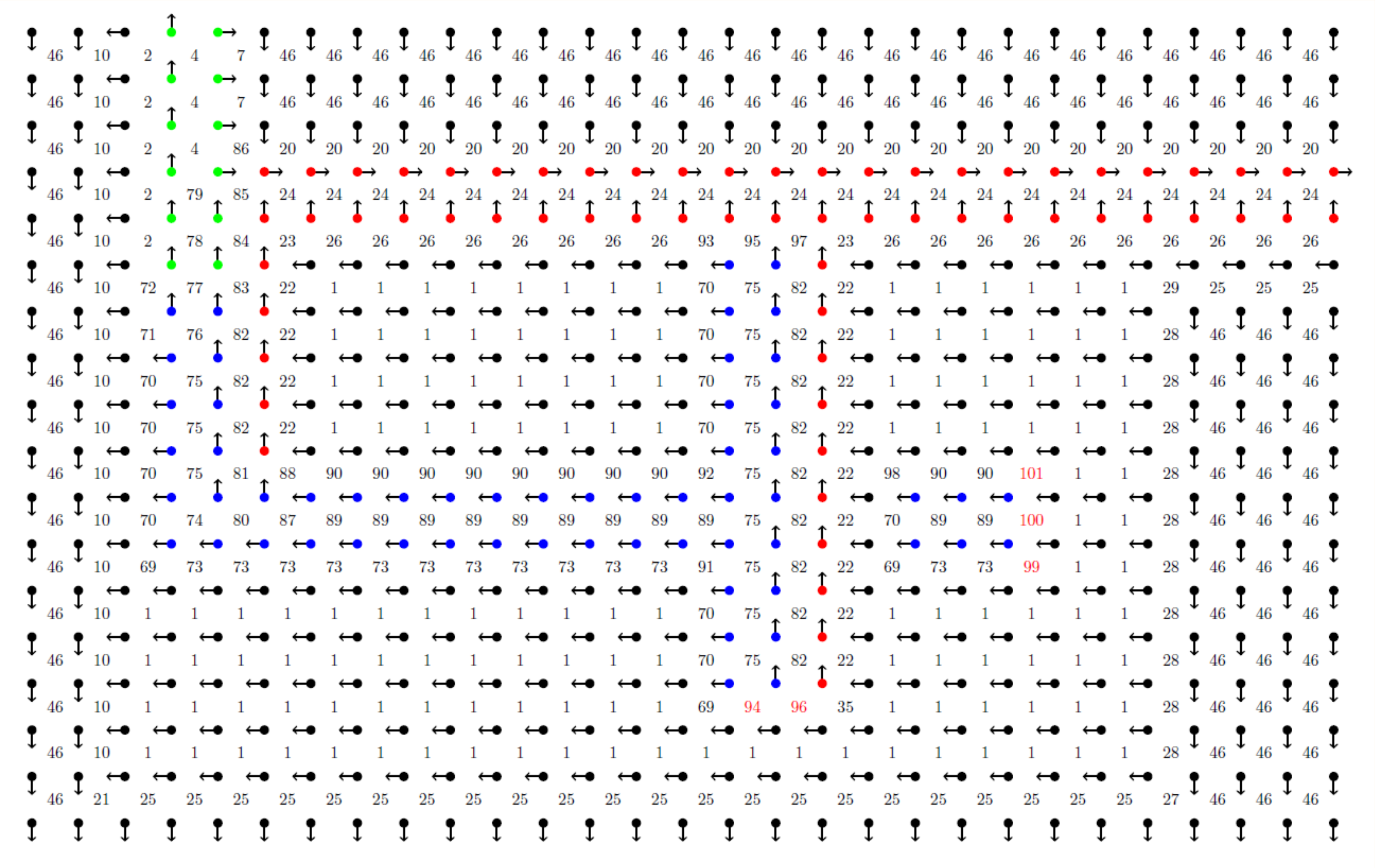

0
1
2
3
4
5
6
7


0
1
2
3
4
5
6
7





→ to find a gradient descent fixed point, we have to solve the PPAD problem or the PLS problem


46 10 2 4 7 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46
46 10 2 4 7 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46 46
46 10 2 4 86 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
46 10 2 79 85 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
46 10 2 78 84 23 26 26 26 26 26 26 26 26 93 95 97 23 26 26 26 26 26 26 26 26 26 26
46 10 72 77 83 22 1 1 1 1 1 1 1 1 70 75 82 22 1 1 1 1 1 1 29 25 25 25
46 10 71 76 82 22 1 1 1 1 1 1 1 1 70 75 82 22 1 1 1 1 1 1 28 46 46 46
46 10 70 75 82 22 1 1 1 1 1 1 1 1 70 75 82 22 1 1 1 1 1 1 28 46 46 46
46 10 70 75 82 22 1 1 1 1 1 1 1 1 70 75 82 22 1 1 1 1 1 1 28 46 46 46
46 10 70 75 81 88 90 90 90 90 90 90 90 90 92 75 82 22 98 90 90 101 1 1 28 46 46 46
46 10 70 74 80 87 89 89 89 89 89 89 89 89 89 75 82 22 70 89 89 100 1 1 28 46 46 46
46 10 69 73 73 73 73 73 73 73 73 73 73 73 91 75 82 22 69 73 73 99 1 1 28 46 46 46
46 10 1 1 1 1 1 1 1 1 1 1 1 1 70 75 82 22 1 1 1 1 1 1 28 46 46 46
46 10 1 1 1 1 1 1 1 1 1 1 1 1 70 75 82 22 1 1 1 1 1 1 28 46 46 46
46 10 1 1 1 1 1 1 1 1 1 1 1 1 69 94 96 35 1 1 1 1 1 1 28 46 46 46
46 10 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 28 46 46 46
46 21 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 25 27 46 46 46

Future Directions

- are there other intersections of classes that are interesting?

Future Directions

- are there other intersections of classes that are interesting?
- candidates for (PPAD ∩ PLS)-completeness:
  - CONTRACTION
  - TARSKI
  - POLYNOMIAL-KKT
  - MIXED-CONGESTION

Future Directions

- are there other intersections of classes that are interesting?
- candidates for (PPAD ∩ 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

Solved! (POLYNOMIAL-KKT, MIXED-CONGESTION)

[Babichenko-Rubinstein, 2020]

2D-GD-FIXED-POINT $\leq$ MIXED-CONGESTION $\leq$ POLYNOMIAL-KKT

Thank You!