Computing a Nash Equilibrium of a Congestion Game: PLS-completeness

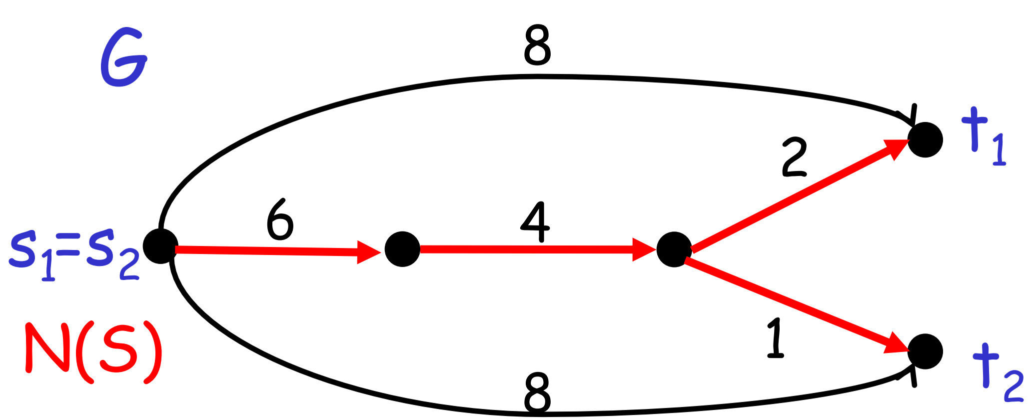
based on Chapters 19 & 20 of
Twenty Lectures on Algorithmic Game Theory,
Tim Roughgarden

Global Connection Game

- $G=(V,E)$: directed graph
- c_e: non-negative cost of the edge $e \in E$
- player i has a source node s_i and a sink node t_i
- Strategy for player i: a path P_i from s_i to t_i
- Given a strategy vector S, the cost of player i

$$\mathrm{cost}_i(S) = \sum_{e \in P_i} c_e / k_e(S)$$

$k_e(S)$: number of players whose path contains e



$cost_1=7$
$cost_2=6$

- Global Connection Game
  - potential game
  - a NE always exists
  - better-response dynamics always converge to a NE
- Facts
  - no one knows how to define a dynamic converging to a NE in poly-time
  - no one knows how to compute a NE in poly-time
- question:
  - can we derive an evidence that the probem is hard?
- (tricky) answer:
  - theory of PLS-completeness

Congestion Game

- E: set of resources
- k players
- player i picks a strategy S_i from an explicit set of strategies $\mathcal{S}_i \subseteq 2^E$
- each resource $e \in E$ has possible costs $c_e(1)$, $c_e(2)$,…, $c_e(k)$
- Given a strategy vector S, the cost of player i is:

$$\mathrm{cost}_i(S) = \sum_{e \in S_i} c_e(k_e(S))$$

$k_e(S)$: number of players whose chosen strategy contains e

properties of CG

- Congestion Game is a potential game

- Rosenthal potential function:

$$\Phi(S) = \sum_{e \in E} \sum_{i=0}^{k_e(S)} c_e(i)$$

⇨ a NE always exists (any local minimum of Φ is a NE)

⇨ better response dynamic converges to a NE

CG-NE problem

Given an instance of Congestion Game, find any NE

can we prove that CG-NE is NP-hard?

....if yes, this would yield to quite surprising consequences.

Addressing a typechecking error

- an NP problem is a decision problem admitting short (polynomial size) witnesses for YES-instances and poly-time verifier
  - inputs accepted by the verifier are called witnesses
- CG-NE is not a decision problem

- class FNP (Functional NP): problem just like NP probems except that, for YES-instances, a witness must be provided
  - also called search problems

- An algorithm for an FNP problem:
  - takes as input an instance
  - outputs a witness for a YES-instance or say "No".

Reduction from one search problem L_1 to onother one L_2

Two polynomial-time algorithms:

- A_1 mapping instances $x \in L_1$ to instances $A_1(x)$ of L_2
- A_2 mapping witnesses of $A_1(x)$ to witnesses of x (and "no" to "no")

Notice: if L_2 is solvable in poly-time then L_1 is solvable in poly-time as well.

Theorem

CG-NE is not FNP-complete unless NP=coNP

proof

Assume we have two poly-time algs

- A_1 that maps every SAT formula ϕ to instances of CG-NE $A_1(\phi)$
- A_2 that maps every NE S of $A_1(\phi)$ to a satisfying assignment $A_2(S)$ of ϕ, if one exists, or to the string "no" otherwise.

Then NP=CoNP.

Let ϕ be unsatisfiable SAT formula, S be a NE of $A_1(\phi)$.

S is a short, efficiently verifiable proof of the unsatisfiability of ϕ

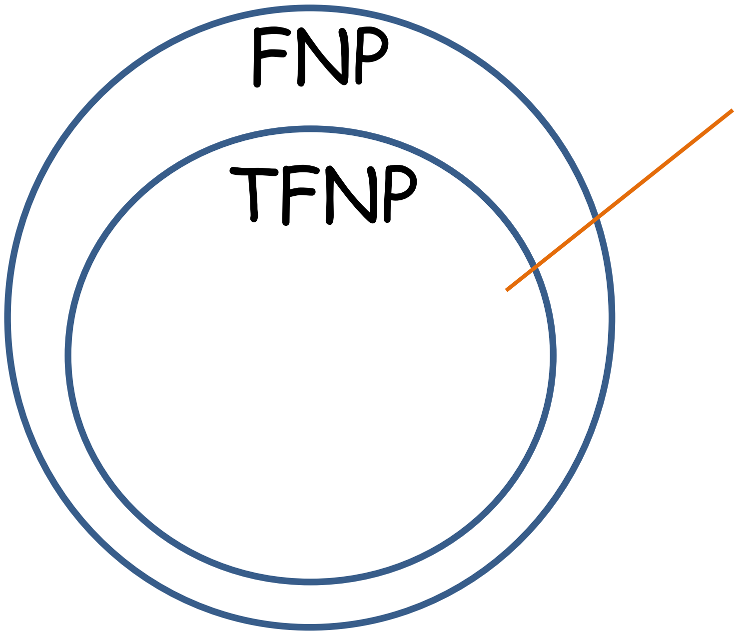
A poly-time verifier:
-compute $A_1(\phi)$
-verify that S is a NE of $A_1(\phi)$
-verify that $A_2(S)$ returns "no"

Note: we're using only the fact that every instance of CG has a NE

TFNP (total FNP): problems in FNP for which every instance has at least one witness.

Theorem

If a TFNP problem is FNP-complete then NP=coNP.

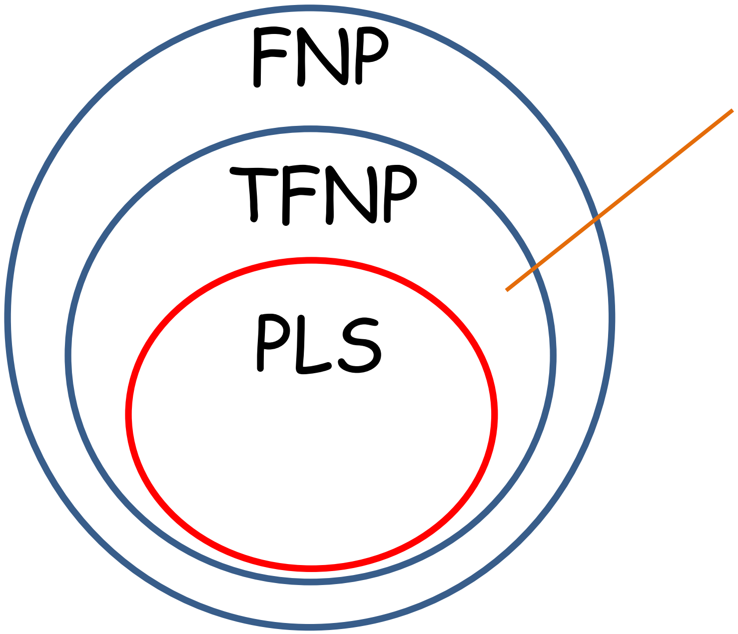


-CG-NE
-problem of finding a mixed-strategy NE for a finite game
-factoring
-...

can we prove that CG-NE is TFNP-complete?

no: no complete problem is known for TFNP
(and people think no one can exist)

Syntactic classes vs Semantic classes



-CG-NE
-problem of finding a mixed-strategy NE for a finite game
-factoring
-...

can we prove that CG-NE is TFNP-complete?

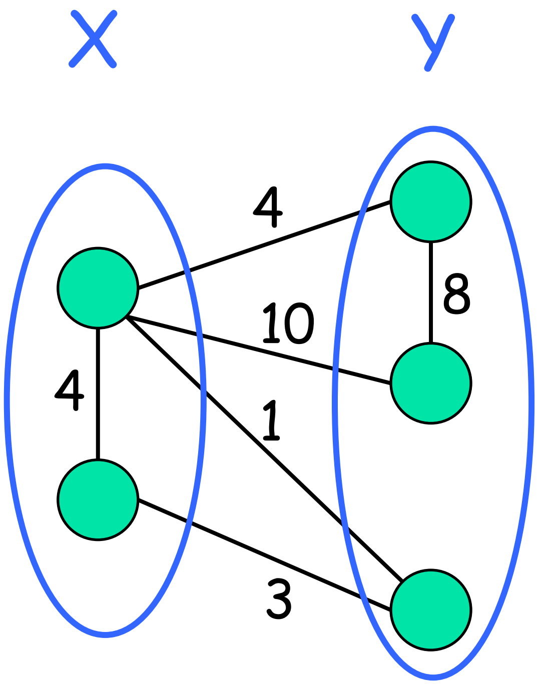
no: no complete problem is known for TFNP
(and people think no one can exist)

which is the right class for CG-NE?

PLS: abstract local search problems

Maximum Cut problem

- Input:
  - an undirected graph $G=(V,E,w)$ with non-negative edge weights
- Solution:
  - a cut (X,Y), where X and Y are a partition of V
- Measure (to maximize):
  - the weight of the cut, $\sum_{\substack{(x,y)\in E: \\ x\in X, y\in Y}} w(x,y)$



It is NP-hard

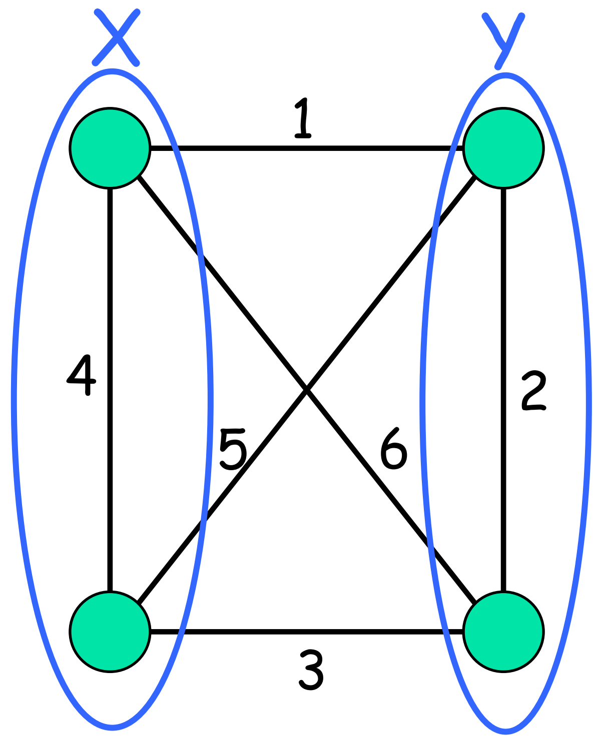
A natural heuristic: Local search algorithm

- initialize with an arbitrary cut (X,Y)
- **while** there is an improving local move **do**
  take an arbitrary such move

improving local move:
move a single vertex v from one side of the cut to the other side, if this improves the weight of the current cut.

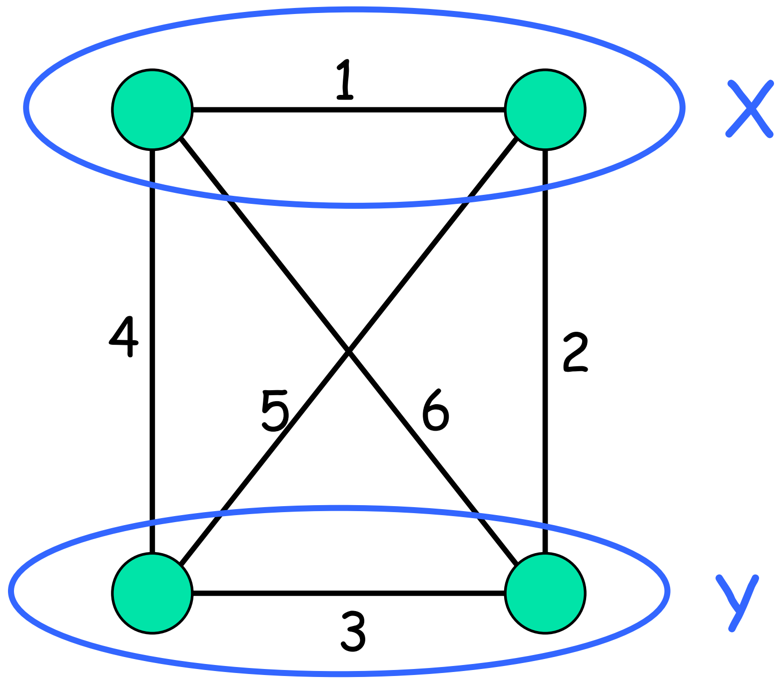
local optimum: cut with no improving local move available.

local optimum vs global optimum



local opt of weight 15

local optimum vs global optimum



global opt of weight 17

local optimum vs global optimum

is finding a local opt easier than finding a global opt?

sometimes strictly easier: unweighted graphs

- max cut is still NP-hard for unweighted graphs
- local search algorithm converges in poly-time

facts:

- no known poly-time local search alg for finding local opt for general weights
- no known poly-time alg for computing a local opt for general weights

local Max-Cut problem

Given an instance of Max Cut, find any local opt.

....this problem is PLS-complete.

Ingredients of an Abstract Local Search Problem

1. The first polynomial-time algorithm takes as input an instance and outputs an arbitrary feasible solution.

2. The second polynomial-time algorithm takes as input an instance and a feasible solution, and returns the objective function value of the solution.

3. The third polynomial-time algorithm takes as input an instance and a feasible solution, and either reports "locally optimal" or produces a solution with better objective function value.

A PLS reduction from L_1 to L_2

Two polynomial-time algorithms:

- A_1 mapping instances $x \in L_1$ to instances $A_1(x)$ of L_2
- A_2 mapping every local optimum of $A_1(x)$ to local optimum of x

Notice: if L_2 is solvable in poly-time then L_1 is solvable in poly-time as well.

Definition.

A problem L is PLS-complete if $L \in PLS$ and every problem in PLS reduces to it.

Theorem (Johnson, Papadimitriou, Yannakakis '85, Schaffer, Yannakakis 91)

Computing a local maximum of a maximum cut instance with general non-negative edge weights is a PLS-complete problem.

Theorem (Johnson, Papadimitriou, Yannakakis, '85, Schaffer, Yannakakis 91)

Computing a local maximum of a maximum cut instance with general non-negative edge weights using local search can require an exponential (in $|V|$) number of iterations, no matter how an improving local move is chosen in each iteration.

Theorem (Fabrikant, Papadimitriou, Talwar 2004)

CG-NE is PLS-complete.

proof

CG-NE $\in$ PLS

3 algorithms of the formal definition:

Alg 1: given the instance, returns any strategy profile S

Alg 2: given a strategy profile S, compute Φ(S)

Alg 3: given a strategy profile S, computes a better response for any player, if any, or report "S is a NE".

completeness: reduction from local MaxCut

proof

a player for each vertex v

two resources r_e and $\bar{r}_e$ for each edge e

two strategies for player v: $S_v = \{r_e : e \in \delta(v)\}$

$\bar{S}_v = \{\bar{r}_e : e \in \delta(v)\}$

cost of a resource $r \in \{r_e, \bar{r}_e\}$:

$$c_r(0)=c_r(1)=0 \quad \text{and } c_r(2)=w(e)$$

bijection between $2^{|V|}$ strategy profiles and cuts of the graph

cut corresponding to strategy profile S:

$$(X_S := \{v : v \text{ plays } S_v \text{ in } S\},\ Y_S := V \setminus X_S)$$

$$\Phi(S) = \sum_{r \in R} \sum_{i=0}^{k_r(S)} c_r(i) = W - W(X_S, Y_S)$$

$$W = \sum_{e \in E} w(e)$$

⇒ (X_S, Y_S) is a local maximum cut iff S local minimum for $\Phi(S)$ ■

what about the problem
of computing mixed Nash Equilibria?

MNE problem

Given an instance of a 2-player game in normal form (bimatrix game), find any mixed NE

Nash's theorem guarantees that a mixed NE always exists

MNE $\in$ TFNP

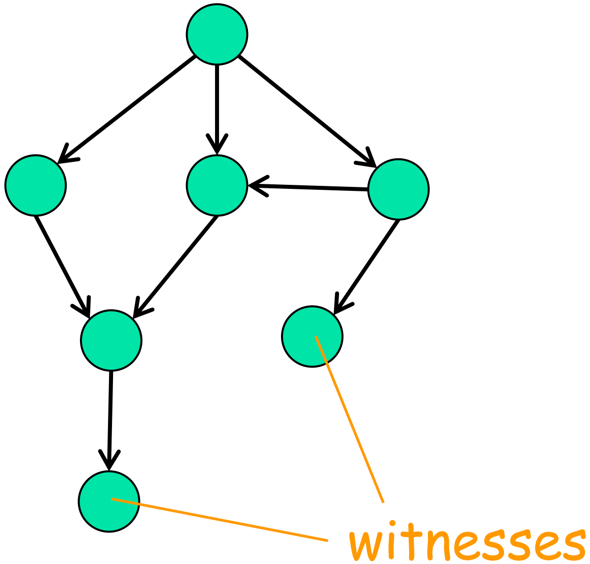
no polynomial time algorithm is known for MNE

what is the right class for MNE problem?

PLS: abstract local search problems

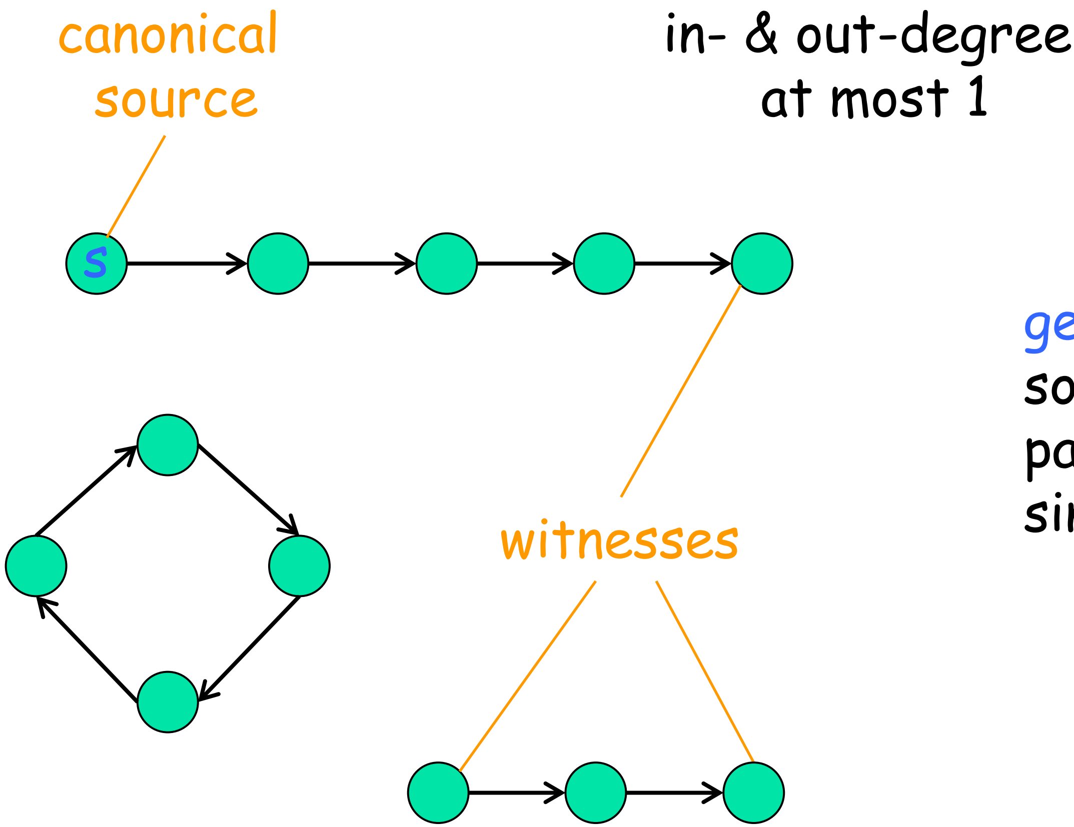
nodes: feasible solutions

edges: improving moves



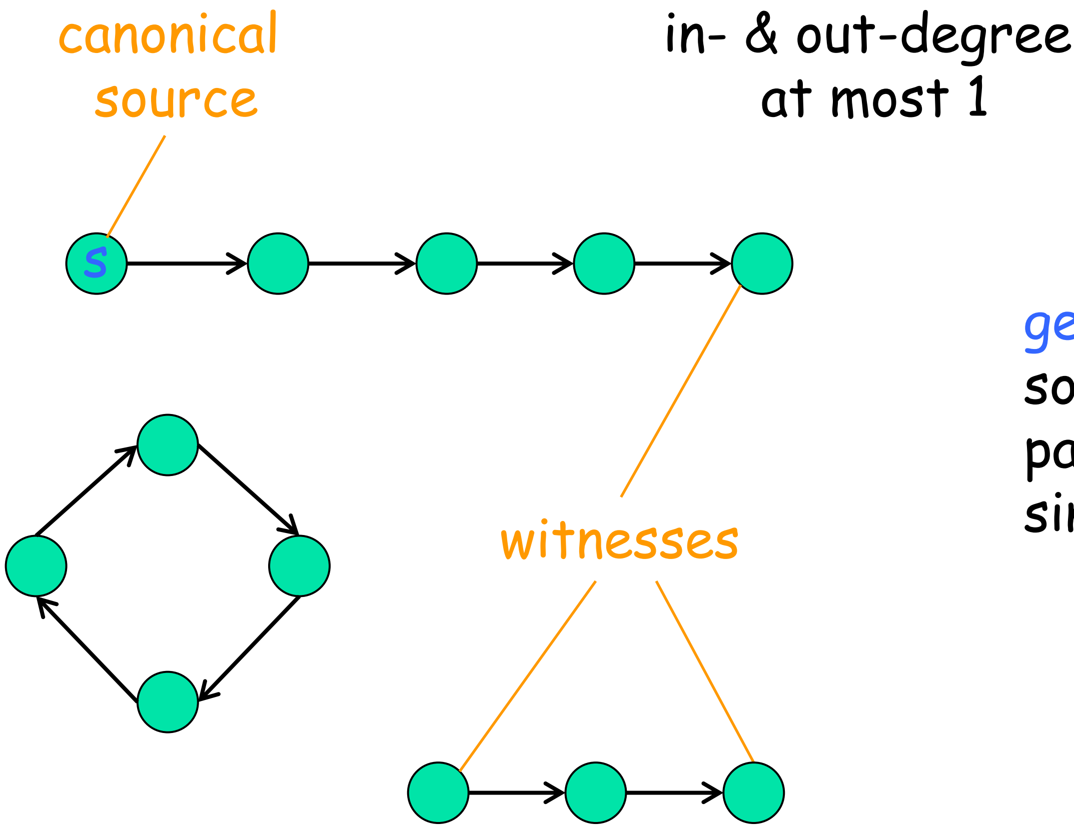
generic reason of membership:
solvable by local search, i.e. by following a directed path to a sink vertex.

PPAD



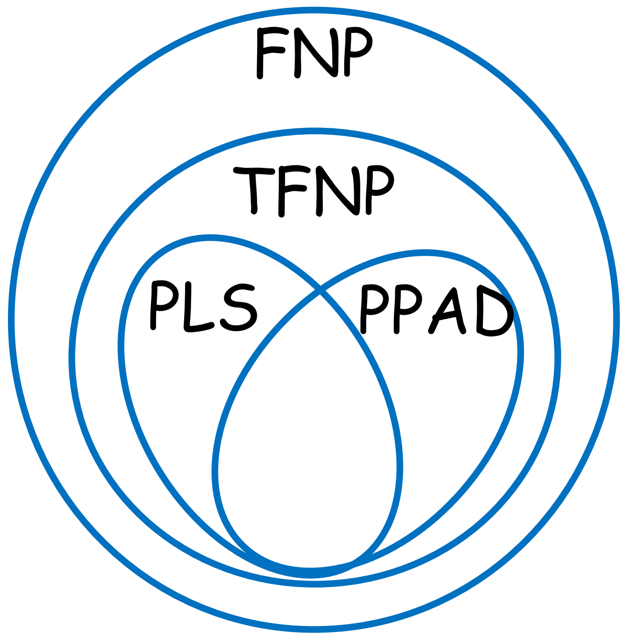
generic reason of membership:
solvable by following a directed path from the source to the sink vertex.

PPAD: polynomial parity argument in a directed graph



generic reason of membership:
solvable by following a directed path from the source to the sink vertex.

class PPAD introduced in 94
by Christos H. Papadimitriou



Theorem (Daskalakis, Godberg, Papadimitriou 06, Chen,Deng,Teng 06)

Computing any MNE of a bimatrix game is PPAD-complete