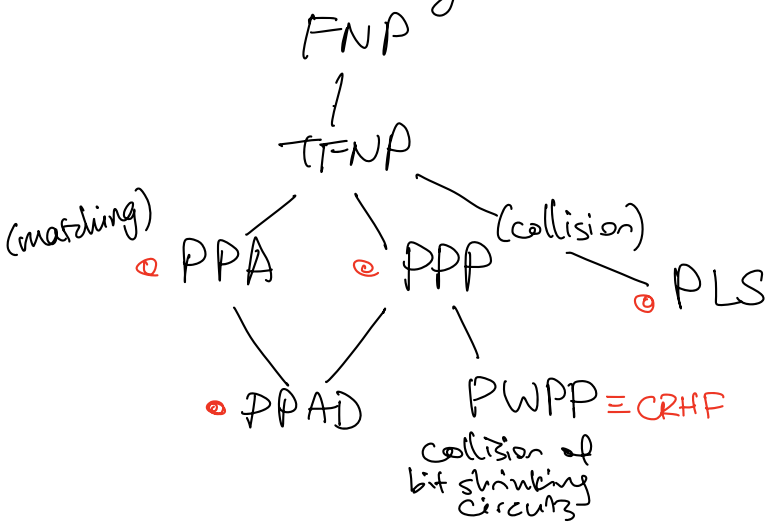
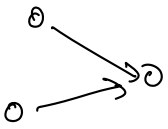
Feb 29th, 2024: Background for PPAD + hardness from crypto assumptions

TFNP Hierarchy



principle behind many problems:
graph with no cycle and one leaf has another

ways to enforce acyclicity ⇒ different problems

- directed graph & a certificate of  ⇒ PPP

END-OF-LINE (PPAD), directed graph on $V = \{0,1\}^n$

given $S : V \to V$ represents forward edges

$P : V \to V$ (predecessor)

$\exists e = (\overrightarrow{u,v})$ if $S(u) = v$ and $P(v) = u$

this forces: all nodes have out-deg ≤ 1
in-deg ≤ 1

Convention: if $S(v) = v$ and $P(v) = v$ consider this v as not part of the graph

lastly, given s s.t. $S(s) \neq s$ and $P(s) = s$

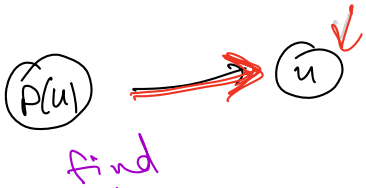
s is a source

seed problem for PPAD is:

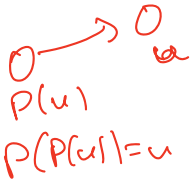
find another source $s' \neq s$ s.t. $S(s') \neq s'$
but $P(s') = s'$
or $S(P(s')) \neq s'$

or

(interesting sol) sink u s.t. u has no out edge but $P(u) \neq u$, and $S(P(u)) = u$



(eg $S(u)=u$ or $S(u)=w$ but $P(w) \neq u$)

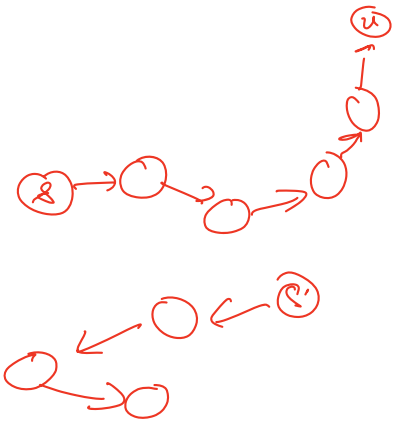


PPAD: another source or sink

PPADS: find a sink

sink

PPADS ⊆ PPAD



convention / way to interpret is: (u,v) if $S(u)=v$, $P(v)=u$

SINK-OF-DAG $V = \{0,1\}^n$

$S : V \to V$

C, cost function $C : V \to \{0,1\}^m$

and $e = (u,v)$ exists if $S(u) = v$
and $C(v) > C(u)$

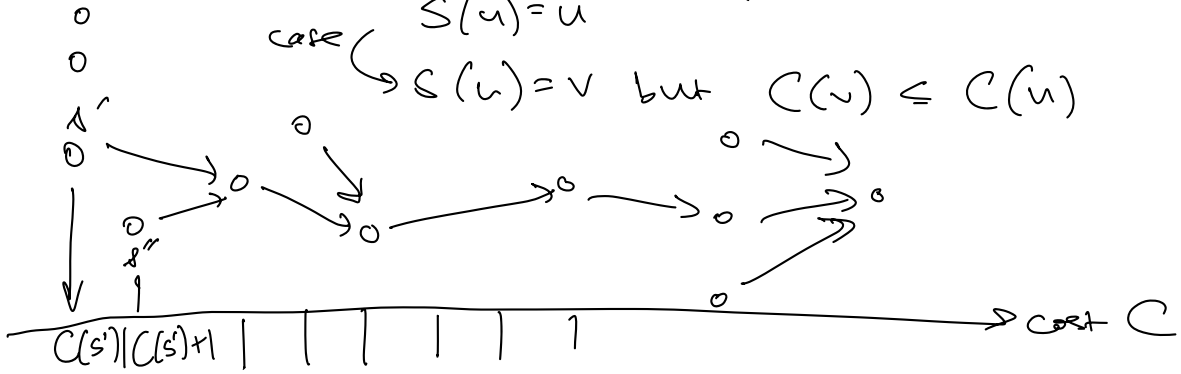
and given a source s' $S(s') \neq s'$

$C(S(s')) > C(s')$

find a sink: find a u with no out edge

$S(u) = u$

case $S(u) = v$ but $C(v) \leq C(u)$



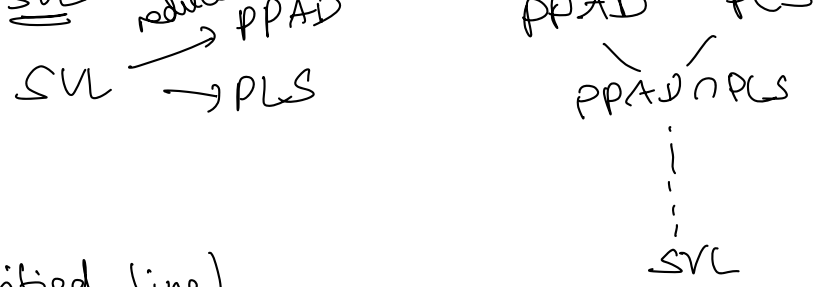
SINK OF DAG: PLS-complete (1st lecture, defined PLS via "ITER", easy exercise ITER $\leq$ SINK-OF-DAG)

easy.

we've seen (hard) constructions of other TFNP subclasses from number theory / crypto: PPA, PPP, PWPP

can we build hard instances of PPAD (S,P) or PLS (S,C)

we only know (for now) 1 way: thru an intermediate problem called SVL



SVL (sink of verified line)

given two circuits:

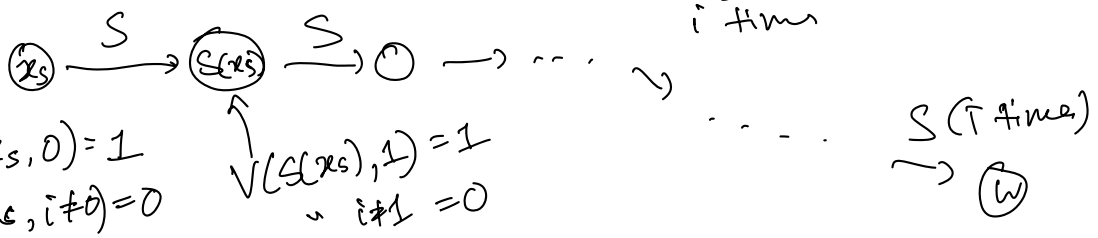
$S : \{0,1\}^n \to \{0,1\}^n$ successor circuit

verifier $V : \{0,1\}^n \times [T] \to \{0,1\}$

given $x_s \in \{0,1\}^n$

and S, V, x_s satisfy the following promise:

$$V(w,i) = 1 \iff w = S^i(x_s) = S(S(\cdots S(x_s)\cdots))$$ (i times)



one last input: number T

search problem: find w s.t. $V(w,T) = 1$

promise problem: SVL $\notin$ TFNP $\iff$ $S^T(x_s) = w$

why? S, V, T (T: target)

head

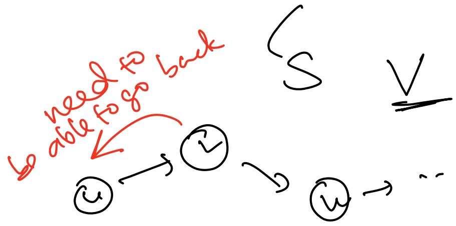
(no way to c c that V is valid
S, V fulfill the promise
maybe ∄ w st $V(w,T) = 1$

a valid (promise-true) instance of SVL reduces to PPAD
(PPAD ∩ PLS)

tons of crypto → hard promise-true instances of SVL → hard PPAD ∩ PLS

Goal:
Promise true instance of SVL reduce to PPAD & PLS.

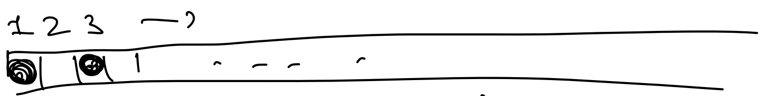
(S V



V's promise is true
⇒ line

"moral" having a verifier for being at a certain point in the line will make S reversible

Pebbling games tape



cells have 0 or 1 pebbles

some number n of pebbles
can place pebbles only using valid moves:

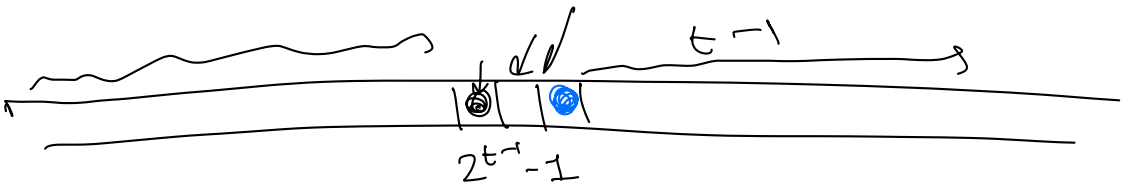
each move valid is reversible

1. place + remove a pebble in position 1
2. place a pebble in cell i if $(i-1)$ is occupied
3. remove a pebble in cell i if $(i-1)$ is occupied.

If I don't care about n, "get" to cell n w/ n pebbles in n moves

Claim can place a pebble on cell $2^t - 1$ using t (n) pebbles.
recursion: $t = 1$, easy, use ①

assume I can do it for $2^{t-1}-1$ using $t-1$ pebbles



first time I place pebble on $2^{t-1}-1$, $t-1$

next move is place a pebble on 2^{t-1} $t+1$

next undo all moves up to $2^{t-1}-1$

$\Rightarrow$ free up $t-1$ pebbles

redo the sequence using $t-1$ pebbles starting from 2^{t-1} (as the "0" position)

$2^{t-1}+2^{t-1}-1 = 2^t-1$ ✓

to place a pebble in 2^t-1 := steps(2^t-1)

$= 3 \cdot \text{steps}(2^{t-1}-1)$

$= 3^t + t = O(3^t)$

use pebble game to reduce SVL $\leq$ END-OF-LINE

SVL : (S, V, x_s, T) $\quad t = \lceil \log T \rceil$

S: nodes of the END-OF-LINE instance

$x' \in \{0,1\}^{n'}$ ←

$n' = t \cdot (n+t)$

$x' = (u_1, \ldots, u_t)$

u_i: tell you where the i^{th} pebble ($i \in [t]$) is in the pebbling game

$u_i = (j, x)$ $\quad x \in \{0,1\}^n$ $\quad i \in [T]$

x' is valid if $\forall i$, if $u_i = (j, x)$ ← i^{th} pebble is in cell j + need "proof": x

so can check $V(x,j)=1$

$V(j, x) = 1$ (that is $x = S^j(x_s)$)

pebble unused: $u_i := (0, x_s)$

$S'(x')$ = next config in the pebbling game where you're trying to place pebble in spot $\hat{T}$ ← $x = S^{\hat{j}}(x_s)$

$x' \leftarrow u_{i_2}(j, x)$

if it frees pebble #i replace it, set $u_i = (0, x_s)$

if it places pebble i on cell j, then spot $j-1$ occupied

S' looks through pebbles, ie $x' = (u_1, \dots, u_t)$

finds the $k \in [t]$ st $u_k = (j-1, \tilde{x})$

must be valid, $\tilde{x} = S^{j-1}(x_s)$

set $u_i := (j, \hat{x})$

$\hat{x} = S^j(x_s)$

$\hat{x} := S(\tilde{x})$

P: next config in pebbling game if you reverse last move

Obs: Based off the positions of pebbles can compute exact step in the pebbling game

$x' = u_1, \dots, u_t$

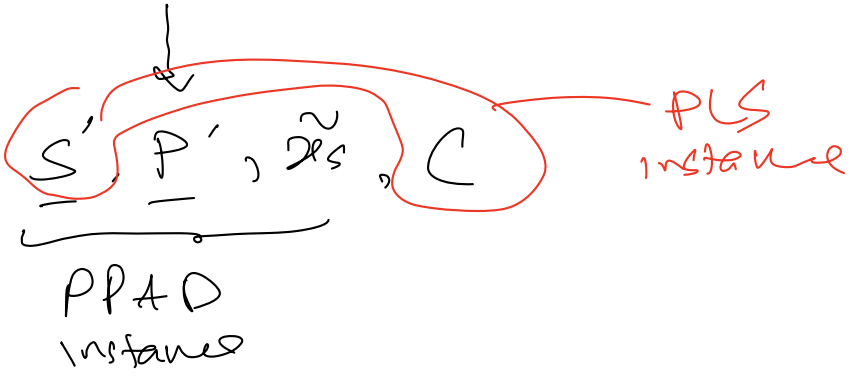
$(j_1, x_1) \cdots (j_t, x_t)$

$\Rightarrow$ which step in $O(3^t)$ steps

very easy to define C (cost circuit)

$x' \longrightarrow [3^t]$

S, V, x_s, T



promise-true SVL instance ⇒ PPAD instance, PLS instance, unique solution (no other sources, only 1 sink)

SVL reduces to UEOPL (end of unique potential line)

(S, P, C, V, target T, x_S)

$V(x, i) = 1$ iff $x = S^i(x_S)$

Sol: find sink

but also allow "V error sols"

→ $x \neq x'$ s.t. $V(i, x) = V(i, x') = 1$