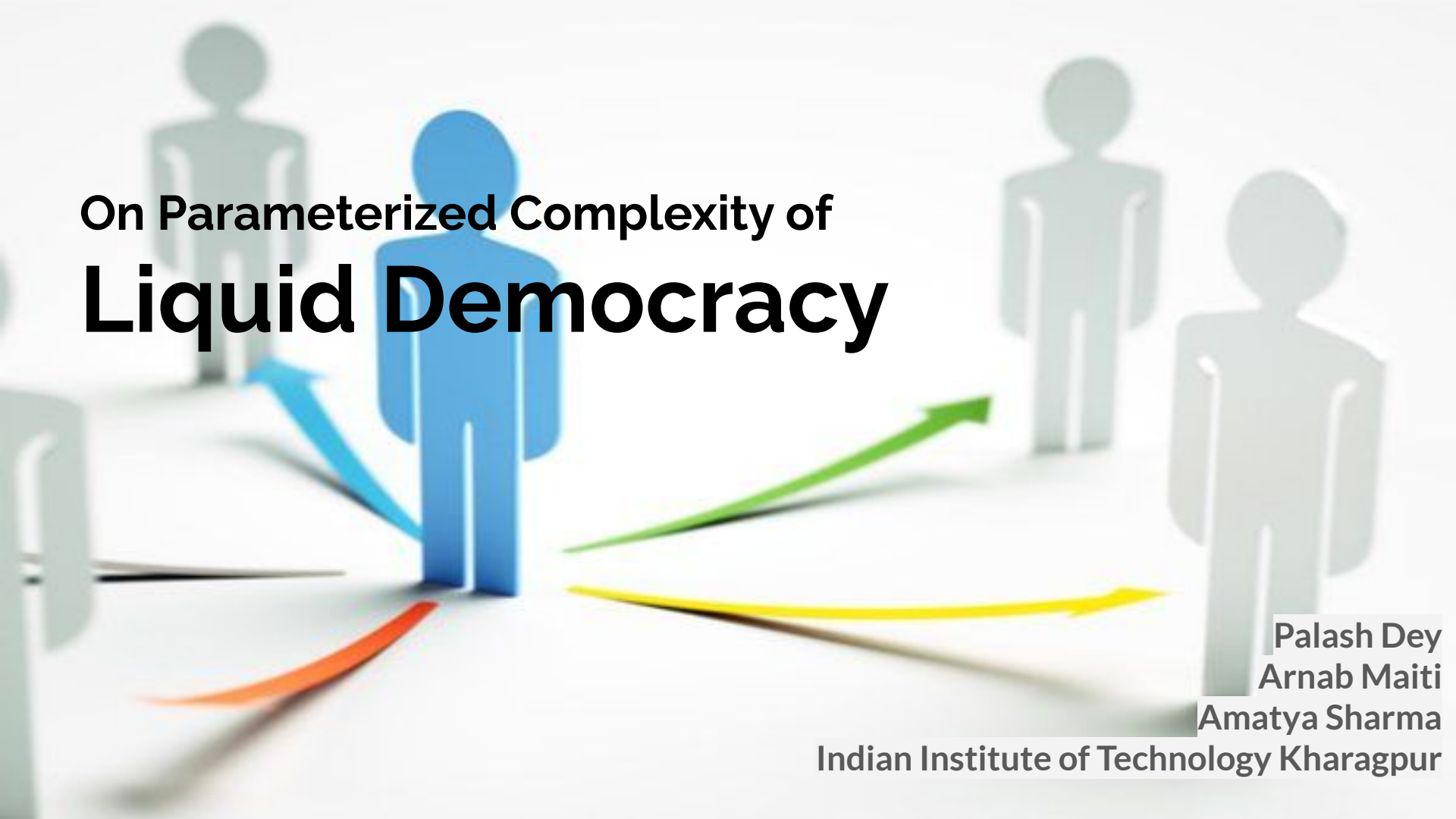
On Parameterized Complexity of

Liquid Democracy



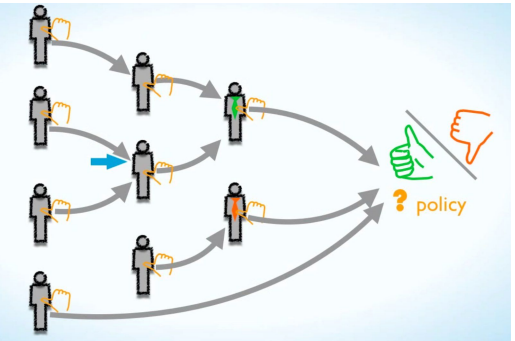
Palash Dey
Arnab Maiti
Amatya Sharma
Indian Institute of Technology Kharagpur

Problem Definition

Liquid Democracy

Voters can delegate their votes to other agents who can vote on their behalf

Disruptive approach to democratic voting system



Any consequences of LD?

Consequence??
Super Voters!!

[Kling, Kunegis, Hartmann, Strohmaier, Staab]

- “PowerFul” Sink Nodes
- Have a lot of voting power
- May turn up to be unfair

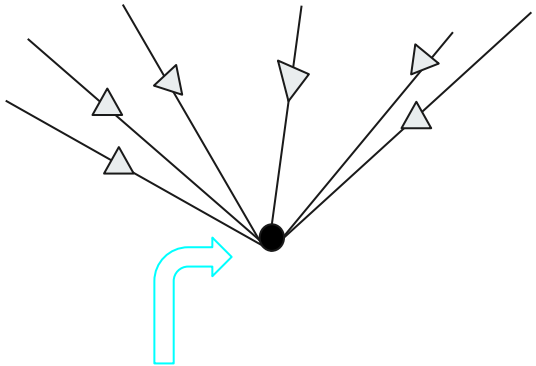
To Deal with this :

- Central mechanism ensures that no super-voter has a lot of voting power
- Allow Multiple Delegations

Sink Nodes

Nodes with 0-outdegree

‘Sinks’ of the Graph.

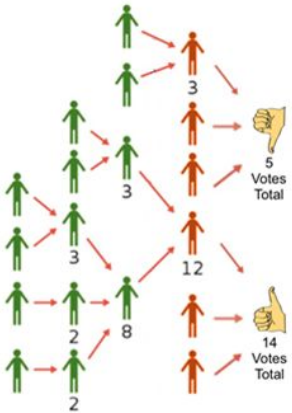


Super Voter with weight (power) 7

Delegation Graph

- Vertices = Voters
- Directed edge from vertex i to vertex j iff voter i delegates her vote to voter j
- May contain cycles

Find a acyclic subgraph of the delegation graph such that no super-voter has a lot of voting power



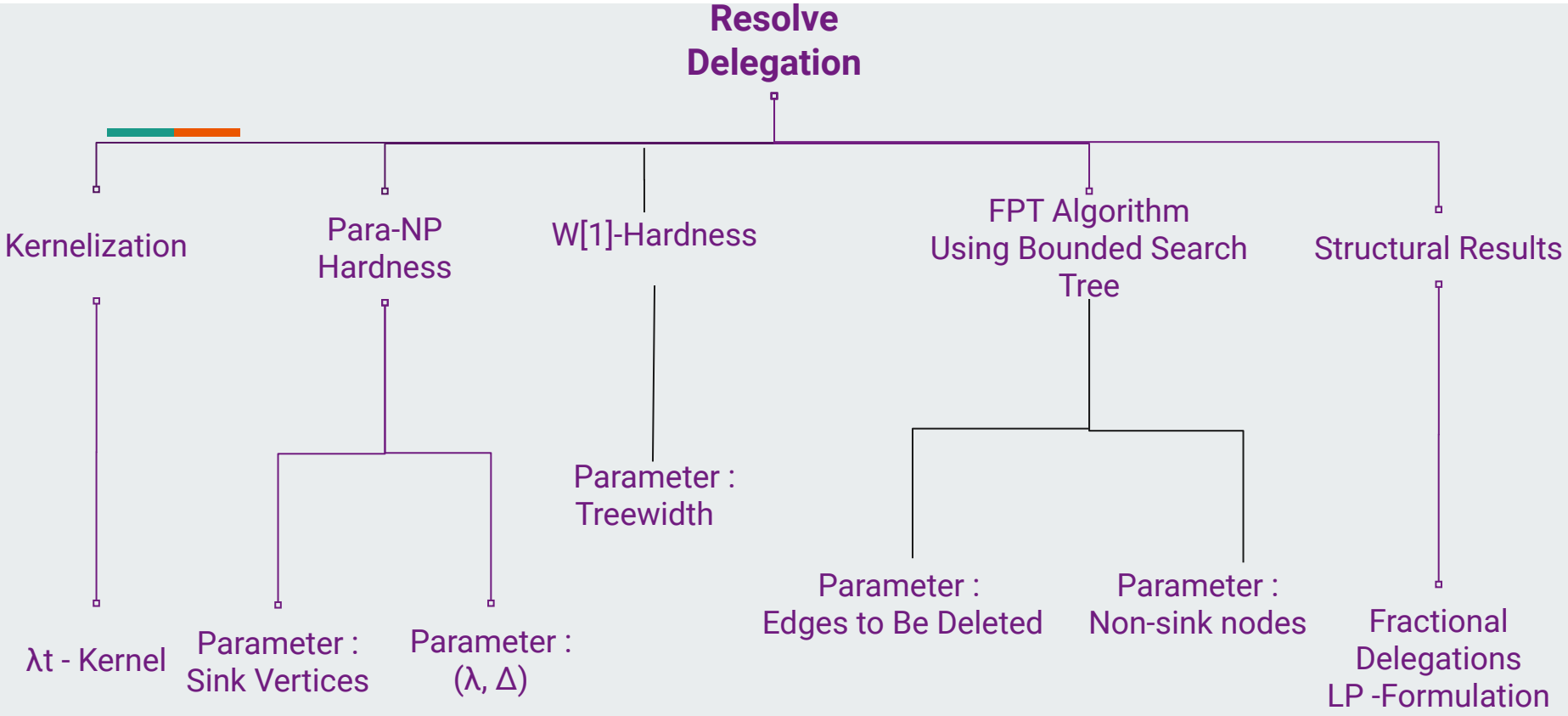
Resolve Delegation Problem

- Input :
  - Directed Delegation graph G (V, E)
  - Sink set T ⊆ V
  - Max Weight Limit λ
- Goal :
  - Decide if there exists a spanning subgraph H ⊆ G such that
    - H is a forest with roots at T
    - Weight of each tree is at most λ

Thinking of Approximation?
Ω(log n)
lower bound on approx-factor

Isolated vertices??

Our Results

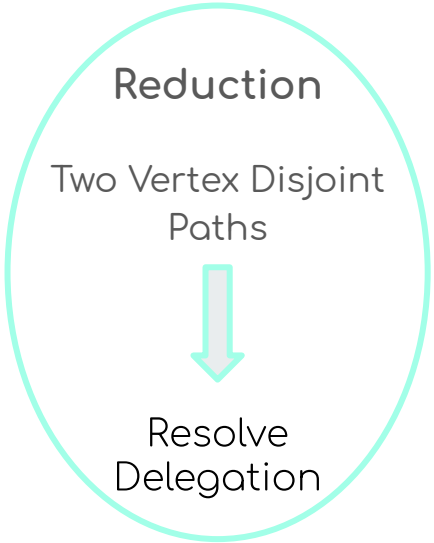

Resolve Delegation
Kernelization
Para-NP Hardness
W[1]-Hardness
FPT Algorithm Using Bounded Search Tree
Structural Results
λt - Kernel
Parameter : Sink Vertices
Parameter : (λ, Δ)
Parameter : Treewidth
Parameter : Edges to Be Deleted
Parameter : Non-sink nodes
Fractional Delegations LP -Formulation

Result 1 : Para-NP-Hardness (t)

Thm. Resolve Delegation problem is **NP-complete even if we have only 3 sink vertices**. In particular, Resolve Delegation is **para-NP-hard** with respect to the **parameter t.**

Black Box

Two Vertex Disjoint Paths is NP-complete



Resolve Delegation problem is **NP-complete even if we have only 3 sink vertices.**

Two Vertex Disjoint Paths

Input : (G, s1 , t1 , s2 , t2)

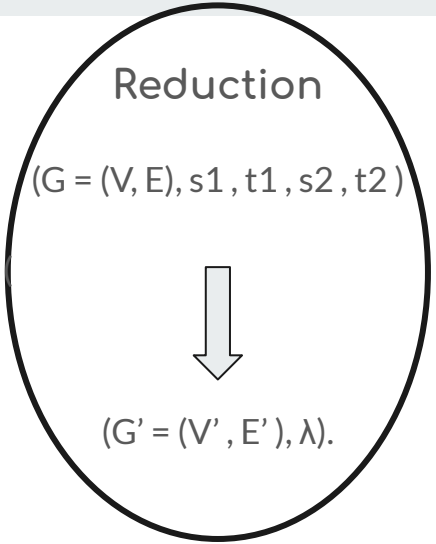
- Directed graph G = (V, E)
- Two disjoint vertex pairs (s1 , t1) and (s2 , t2)

Goal : Decide if there exists two vertex disjoint paths P1 and P2
Pi is a path from si to ti for i ∈ [2]

Black Box

Two Vertex Disjoint Paths is NP-C

Main Reduction



- $V' = \{a_v : v \in V\} \cup D_1 \cup D_1' \cup D_2 \cup D_2' \cup D_3'$

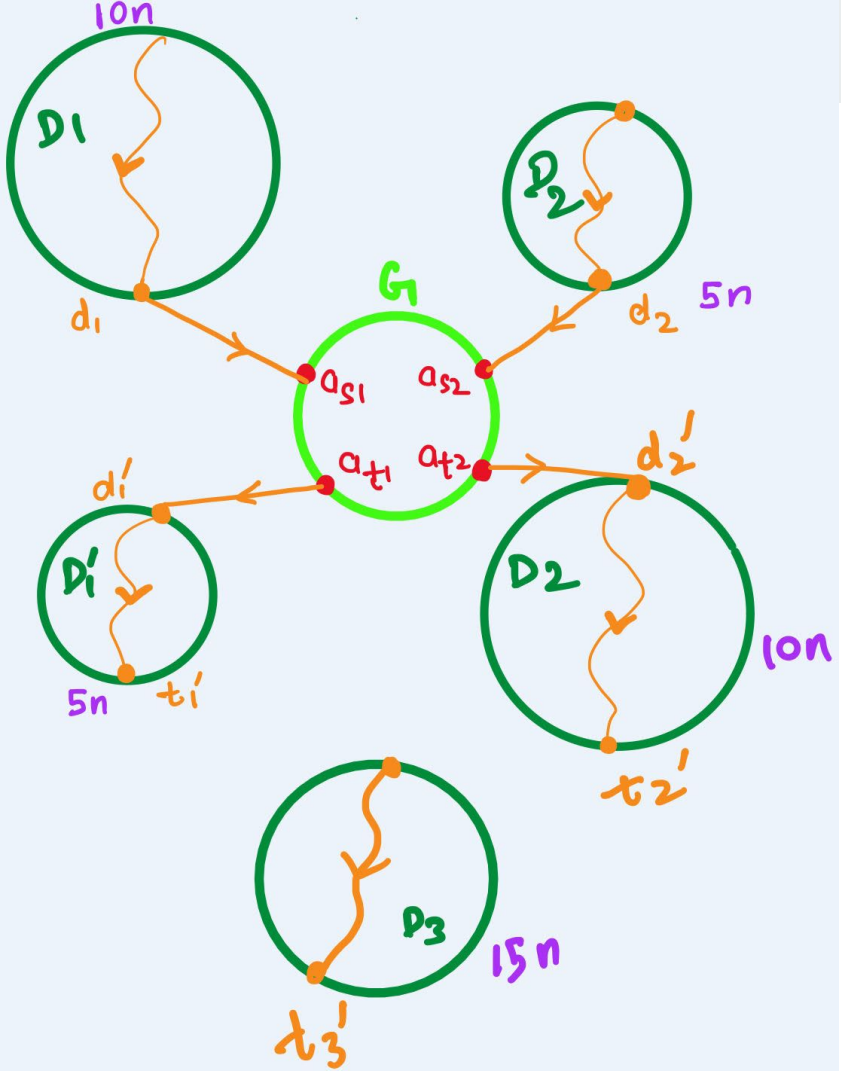
  where

  $|D_1| = |D_2'| = 10n$

  $|D_1'| = |D_2| = 5n$

  $|D_3'| = 15n$

- $E' = \{(a_u , a_v) : (u, v) \in E\} \cup F$
- $\lambda = 17n$



Result1 : Para-NP-Hardness (t)...

Sinks : t'_1, t'_2, t'_3 Sinks in G?

F

- each **D** is a path
- $\{(d_i , a_{si}),\ (a_{ti} , d_i')\ i=1,2\}$
- $\{(a_v , t_3')\mid \forall v \in V\}$

d_i , t_i = end, start vertices D_i
d_i', t_i' = start, end vertices D_i'

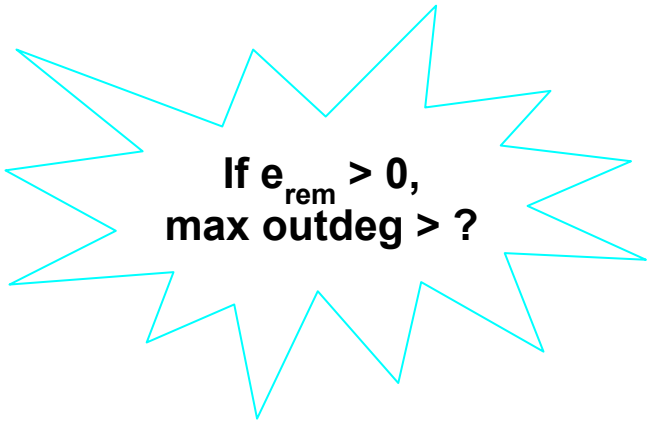
Result 2 : FPT w.r.t e_{rem}

Thm. Resolve Delegation problem has a FPT with respect to the parameter e_{rem} (number of edges to be deleted).

Branching

Note: = e_{rem} = |E| − (|V| -|T|)

1. Consider vertex with maximum outdegree ($d_{out}(v)$)
2. Delete one of the two groups of edges :
   $\{1, \ldots, \lfloor d_{out}(v) \rfloor\}$ and $\{\lceil d_{out}(v) \rceil, \ldots, d_{out}(v)\}$
3. Solve recursively



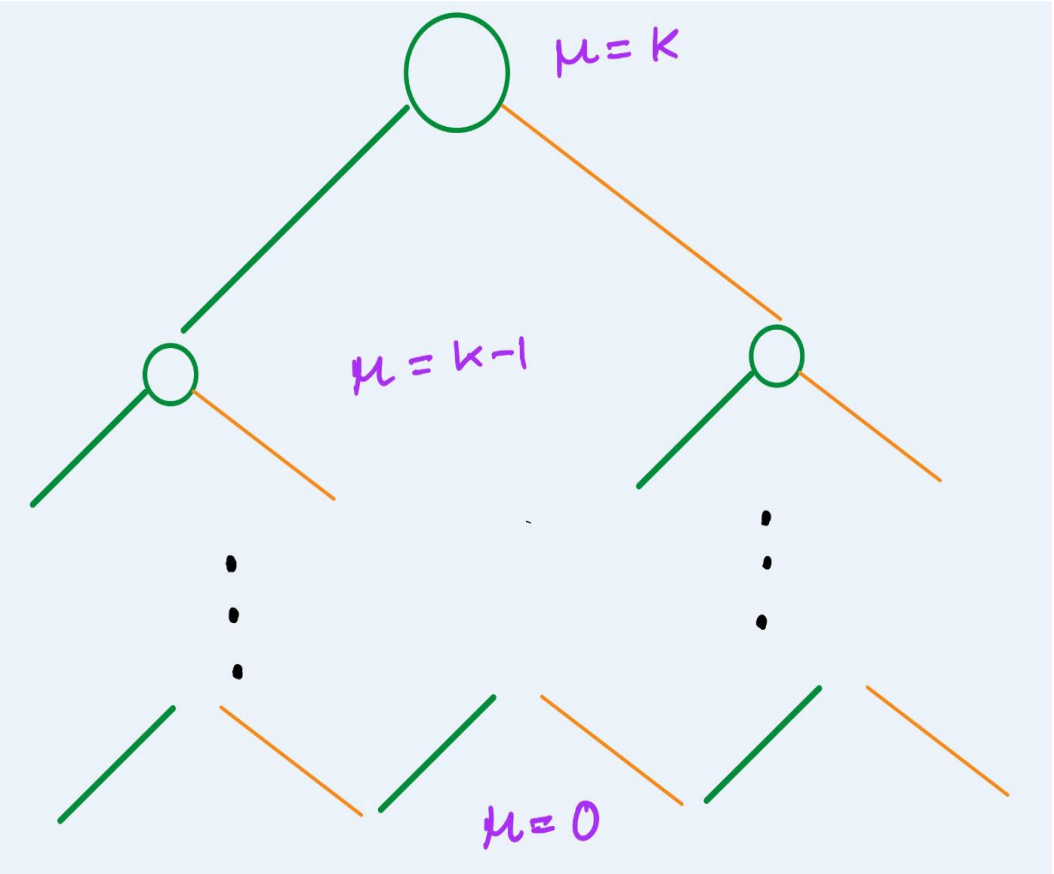
Branching : Proof And Analysis

Obsv. If k > 0 and only the sink nodes have outdegree 0, then there is a non-sink node with outdegree at least 2

Branching

1. Consider max out deg vertex v
2. Delete one of : $\{1, \ldots, \lfloor d_{out}(v) \rfloor\}$ $\{\lceil d_{out}(v) \rceil, \ldots, d_{out}(v)\}$ edge sets
3. Solve recursively

Result3 : FPT w.r.t e_{rem}...



$T(\mu) = T(\mu_1) + T(\mu_2) + T'(node)$

$T'(node) = n^{O(1)}$

$T(\mu_i) \leq T(\mu - 1)$ [T is non-dec]

$T(\mu) \leq 2.T(\mu-1) + T'(node)$

$T(k) = 2^k.n^{O(1)}$ $[k=e_{rem}]$

Result 3 : Fractional Delegations

Thm. There exists a linear programming formulation for the optimization version of Resolve Delegation where fractional delegation of votes is allowed.

Thus the fractional variant is solvable in polynomial time.

- Similar to the LP formulation of flow-problems (e.g. Max-FLow-Min-Cut etc)
- $x_{u,v}$:= weight to every edge of solution graph (0 means no edge)
- $x_{u,v}$:= points towards 'fractional' power delegated by u to v

- Constraints for
  - Flow Conservation (Vote Conservation)
  - Non-Negative Delegations

The Linear Program

$$\text{minimize } z$$

$$z \geqslant \sum_{v \in \mathcal{V} \setminus \mathcal{T}} x_{v,t} + 1, \ \forall t \in \mathcal{T}$$

$$\sum_{u \in \mathcal{V} \setminus \mathcal{T}} x_{u,s} + 1 = \sum_{v \in \mathcal{V}} x_{s,v} \ , \ \forall s \in \mathcal{V} \setminus \mathcal{T}$$

$$x_{u,v} \geqslant 0 \ , \forall (u,v) \in \mathcal{E}[\mathcal{G}]$$

$$x_{u,v} = 0 \ , \forall (u,v) \notin \mathcal{E}[\mathcal{G}]$$

Objective Function

Vote/FLow Conservation

> Non-Negative Delegation
> Ignoring edges not in input

Result 4 : A λt - Kernel

Thm. There is a kernel for Resolve Delegation consisting of at most **λt vertices.**

FPT algorithm for the Resolve Delegation problem parameterized by (λ, t).

- If the n > λt, then instance is a **NO** instance.

Hint : **Pigeonhole Principle!!**

Sinks = Holes
Non-Sinks = Pigeons

Result 5 : Para-NP-Hardness (λ, Δ)

Thm. Resolve Delegation problem is **NP-complete** even if we have **λ = 3** and both the **out-degree** and **in-degree** of every vertex is at **most 3.**

Resolve Delegation is para-NP-hard with respect to the parameter (λ, Δ)

Black Box

(3,B2)-SAT
is NP-complete

(3, B2)-SAT

Input : Instance (X,C)

where

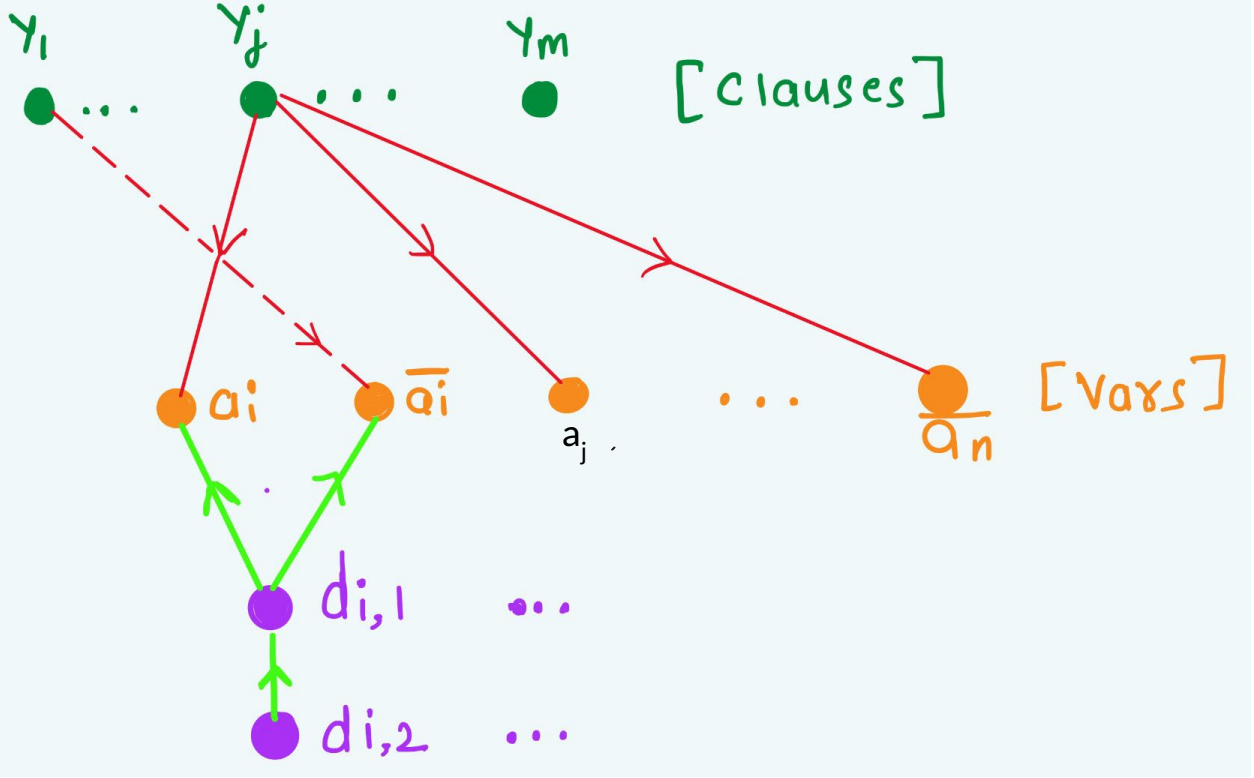
- Variables : $X = \{x_i : i \in [n]\}$
- 3-CNF Clauses : $C = \{C_j : j \in [m]\}$
- x_i and $\bar{x}_i$ each appear in exactly 2 clauses [for all $i \in [n]$]

Goal : Decide if there is an assignment satisfying all clauses

Black Box

(3,B2)-SAT
is
NP-C

Result5 : Para-NP-Hardness (λ, Δ)...

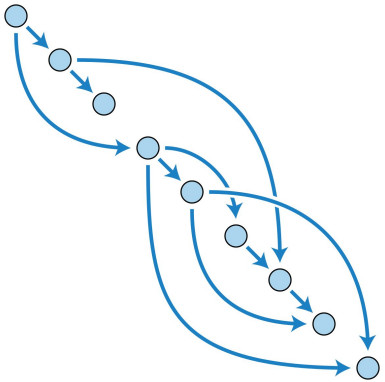


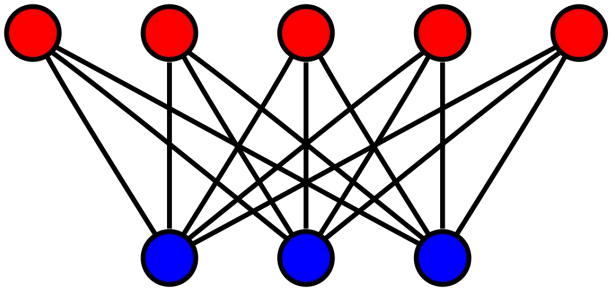
- Vertices = Clauses
- Sinks = Literals
- Dummies to enforce to pick exactly one of x, ~x
- λ = 3

The delegation graph in reduction is Directed Acyclic

An Extension to Bipartite Graphs and DAG

- The delegation graph in reduction is Directed Acyclic
- Bipartite as well!!

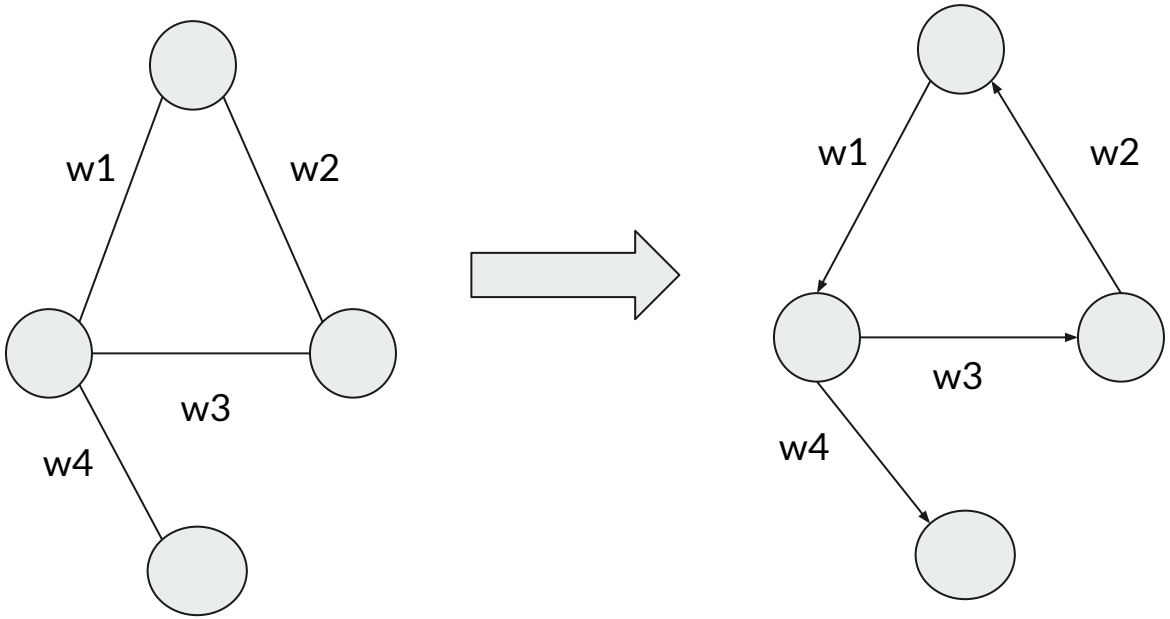




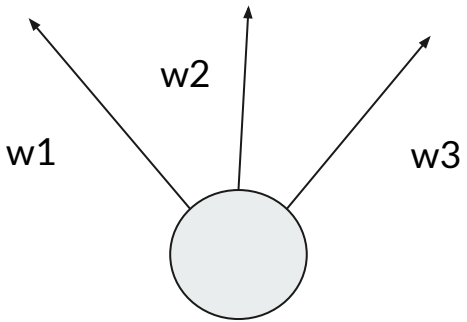
Result 6 : W[1]-Hardness w.r.t tree-width

Thm. Resolve Delegation is W[1]-hard when parameterized by the treewidth of the instance graph.

Minimum Maximum Outdegree



$$\Sigma w_i \leq r$$

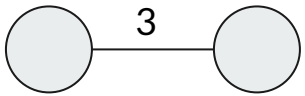


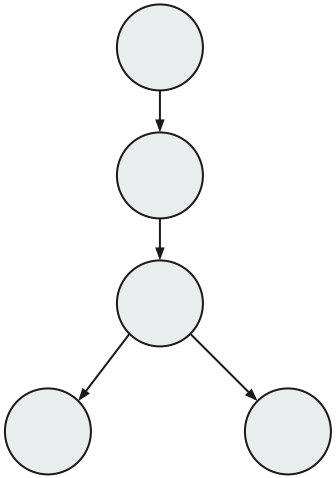
Input instance: (G, w, r)

NOTE: w is represented in Unary

Reduction

(G, w, r)

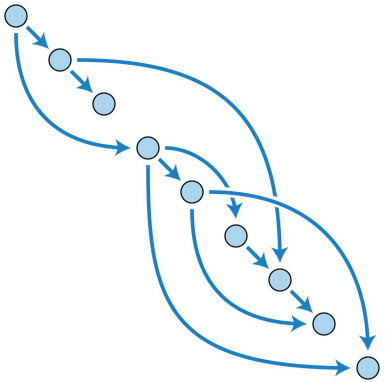


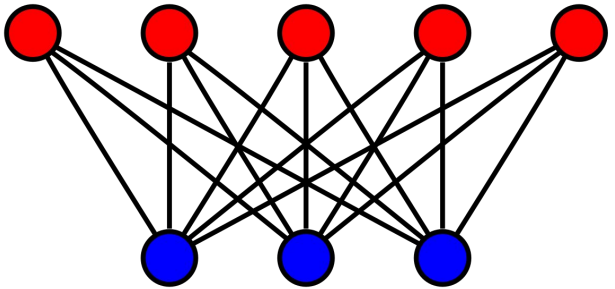


(H, λ=r+1)

An Extension to Bipartite Graphs and DAG

- The delegation graph in reduction is Directed Acyclic
- Bipartite as well!!

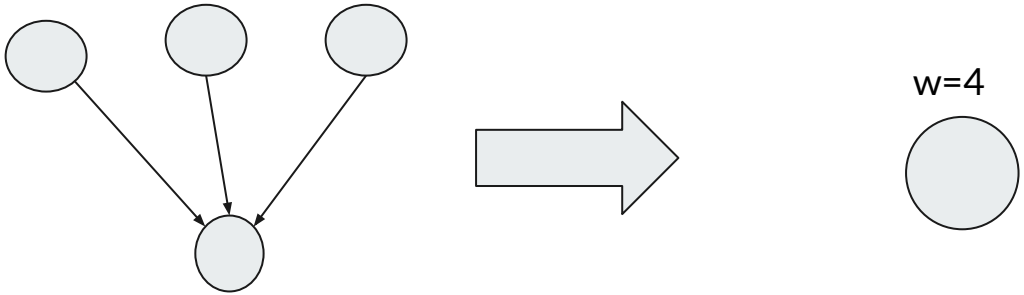




Result 7 : FPT w.r.t no. of non-sink nodes

Thm. The Resolve Delegation problem has a FPT with respect to the parameter k which is the number of non-sink nodes in G (delegation graph).

Notation of weight of vertex

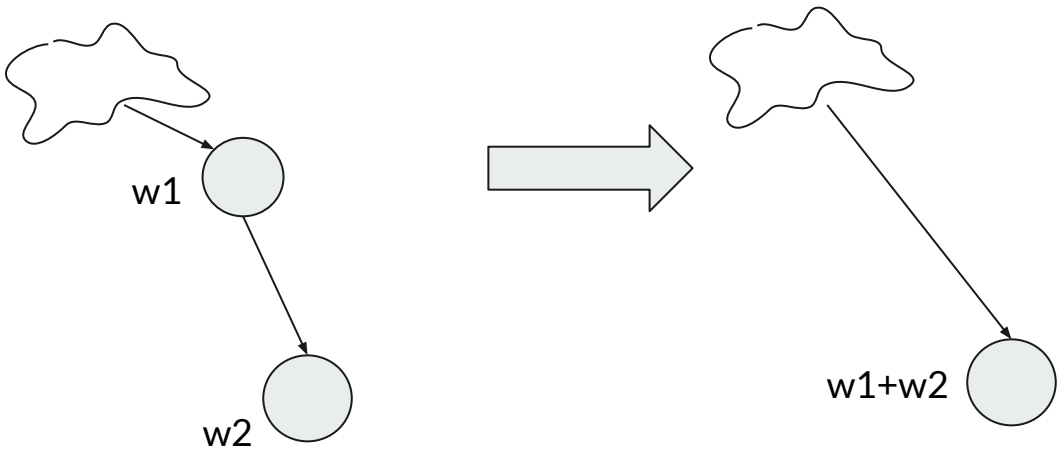


Points to remember

1) Weight of each node initially is 1
2) Weight of a tree rooted at a sink node is now sum of the weights of node in the tree
3) Problem instance is denoted by (G, λ, k) where k is our parameter: non-sink nodes

Bounded search tree based algorithm

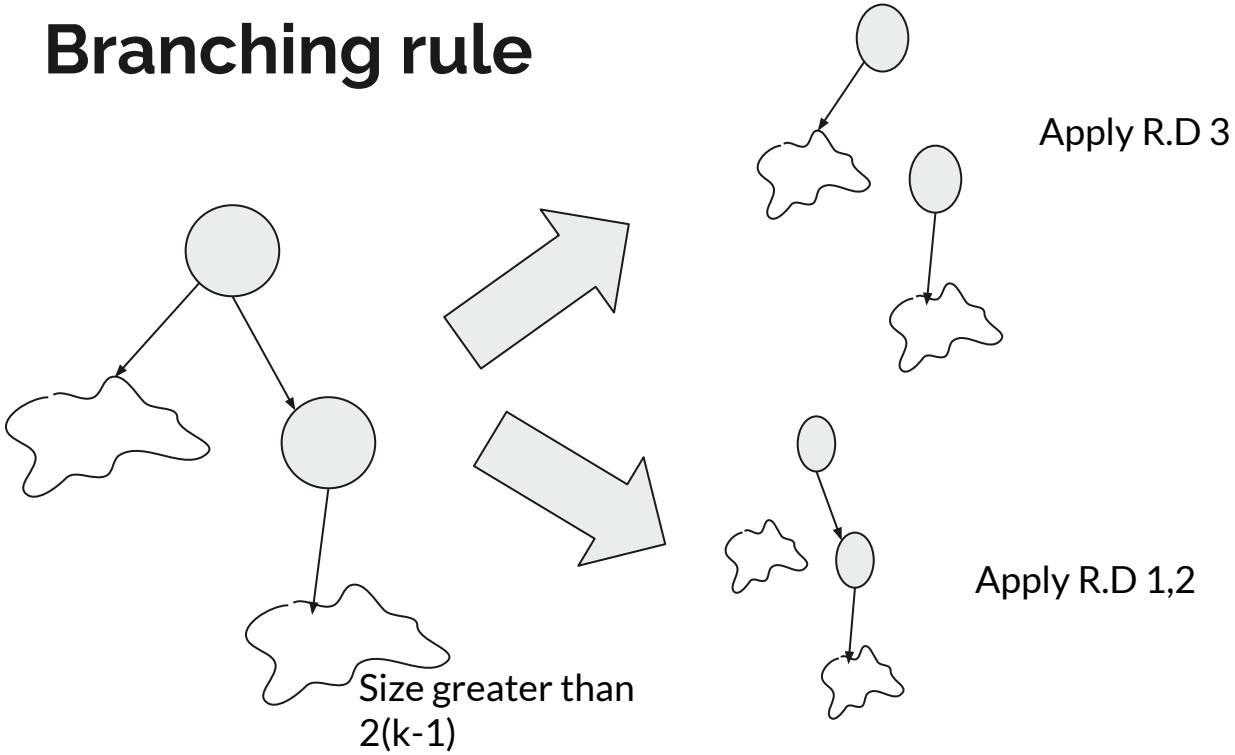
Reduction 1:



Reduction 2: Remove self loops or multiple edges

Reduction 3: If G contains a non-sink node v with outdegree more than $2(k-1)$ and indegree 0, delete v from G. The new instance is $(G - v, \lambda, k-1)$

Branching rule



Future Directions

- What if the underlying undirected graph of the input graph is a tree?
- Kernalizations with respect to:
  - no. of non-sink nodes
  - no. of edges to be deleted
- FPT-Approximation :
  - Ω(log n) lower bound on the approximation factor
  - Is there o(logn) FPT-Approximation??

Practical Implications

Go Vote

Google experimented liquid democracy. Internal social network system Google Votes.

Civicracy

Experimental form of liquid democracy tested at the Vienna University of Technology [2012]

The Industrial Workers of the World

International labor union
Uses liquid democracy using self-developed apps

LiquidFeedback
The Democracy Software

Software used for political opinion formation and decision making

Thanks
Q&A

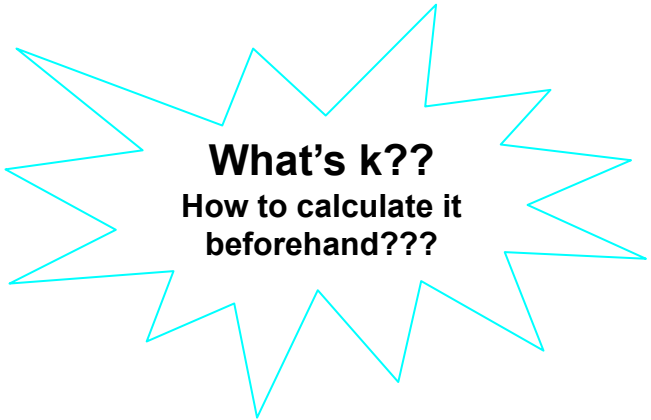
Preliminaries

Parameterized Algorithms

Parameterized problem : Language $L \subseteq \Sigma^* \times N$, where Σ is a fixed, finite alphabet. For an instance $(x, k) \in \Sigma* \times N$, k is called the parameter.

Fixed Parameter Tractable [FPT] : Running time $f(k)n^c$
[for some constant c>0]

Slice-wise Polynomial [XP] : Running time $f(k)n^{g(k)}$



Para-NP-Hardness

Para-NP [Flum and Grohe]:

Class of parameterized problems solvable in time $\mathbf{f(k).|x|^{O(1)}}$ by a nondeterministic TM [f is computable]

Para-NP-Hard :
NP-Hard for a constant value of parameter

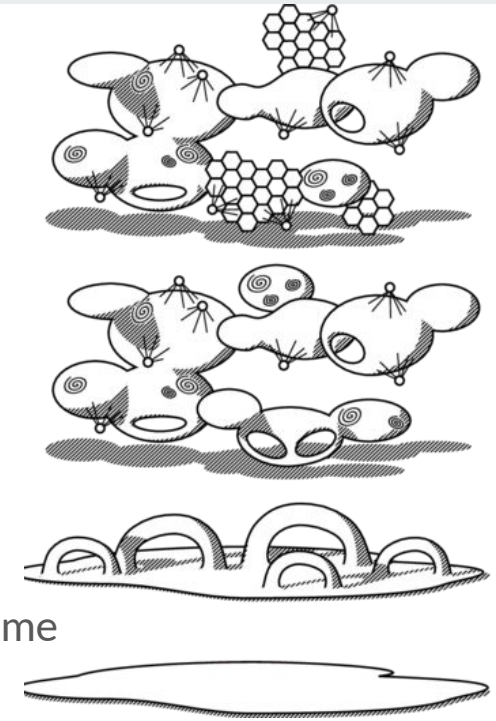
NP-Hard for a "Slice" of the parameter

Kernelization aka Kernel

Kernelization algorithm given an instance (I,k) of problem Q -

- Runs in polynomial time
- Returns an equivalent instance (I', k') of Q
- $A(k) \leq g(k)$ for some computable function $g : N \rightarrow N$.

Reduction rule : Polynomial (in |I| and k) time computable function that maps an instance (I,k) to an equivalent instance (I' , k') of the same problem.



Bounded Search Trees aka Branching

- Origin from idea of backtracking
- Tries to build a feasible solution to the problem by a sequence of decisions on its branching
- Search Tree
- Traverse till a solution leaf node

What's Bounded??

- Depth of tree
- Node Processing Time
- Branching/Children at all levels

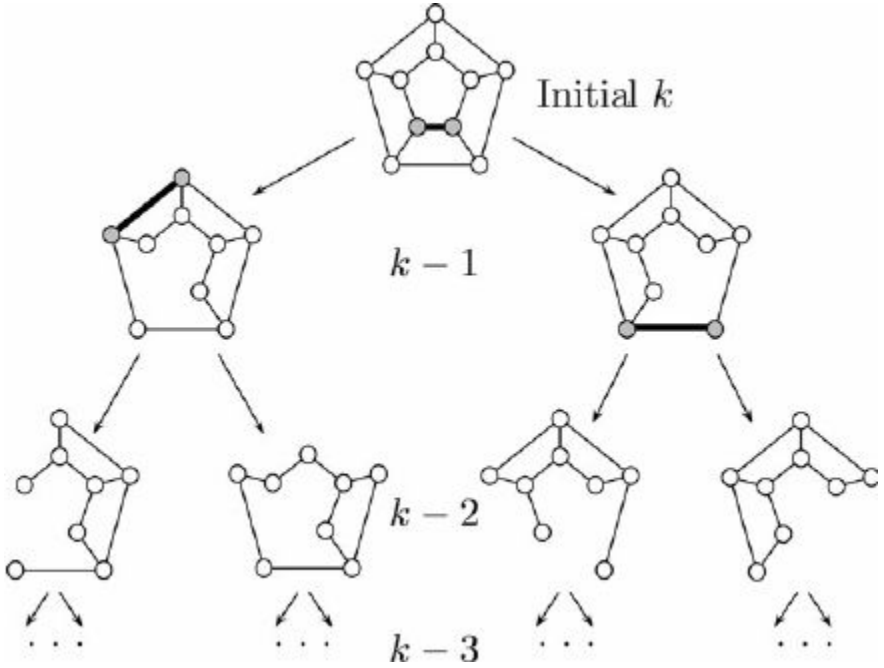


Fig. A search tree for vertex cover

Parameters Under Consideration

We study the problem w.r.t following parameters :

- λ := max weight
- t = |T| := number of sink nodes
- e_{rem} := Number of edges to be removed to form solution
- k := number of non-sink nodes
- Δ := maxdegree [max (max indeg, max outdeg)]