Parameterized Power Vertex Cover

Eric Angel, Evripidis Bampis, Bruno Escoffier, Michael Lampis

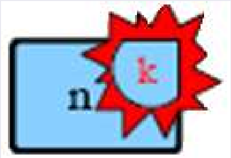
Universities in Paris

WG 2016

Overview

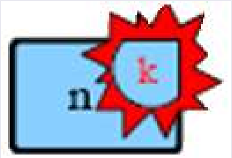
Parameterized Power Vertex Cover

Parameterized Power Vertex Cover

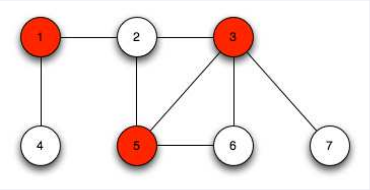


- Parameterized
  - Dealing with NP-hard problem
  - Goal: Algorithm exponential in some parameter FPT

Parameterized Power Vertex Cover

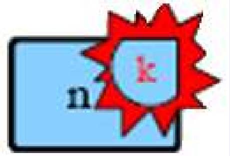


- Parameterized
  - Dealing with NP-hard problem
  - Goal: Algorithm exponential in some parameter FPT

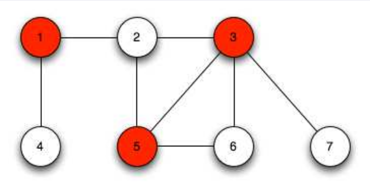


- Vertex Cover
  - Given graph G, find minimum set of vertices that hit all edges
  - Standard NP-hard problem

Parameterized Power Vertex Cover

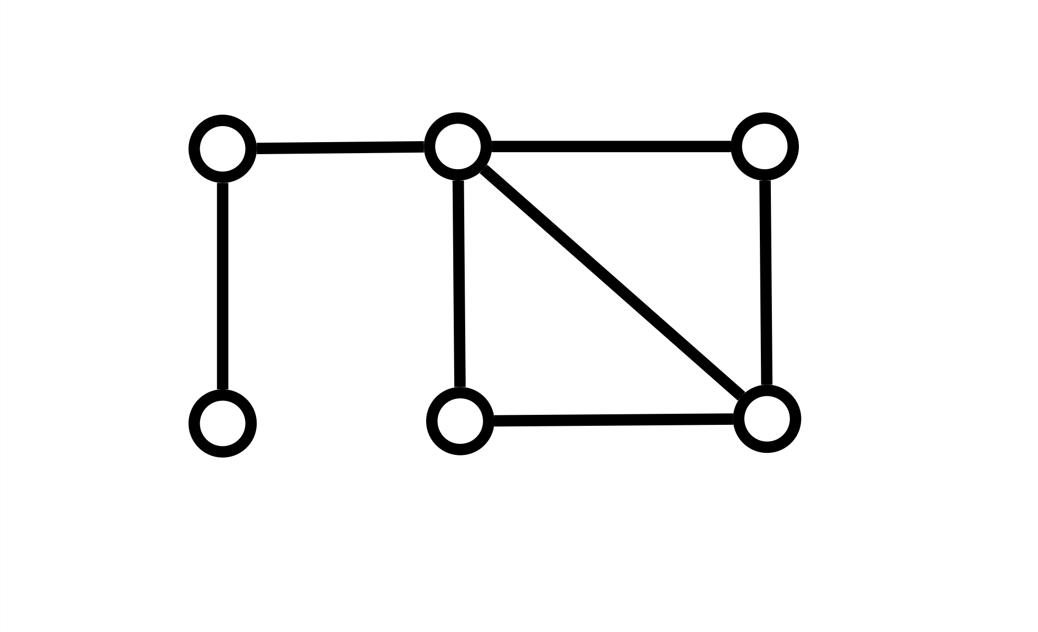


- Parameterized
  - Dealing with NP-hard problem
  - Goal: Algorithm exponential in some parameter FPT



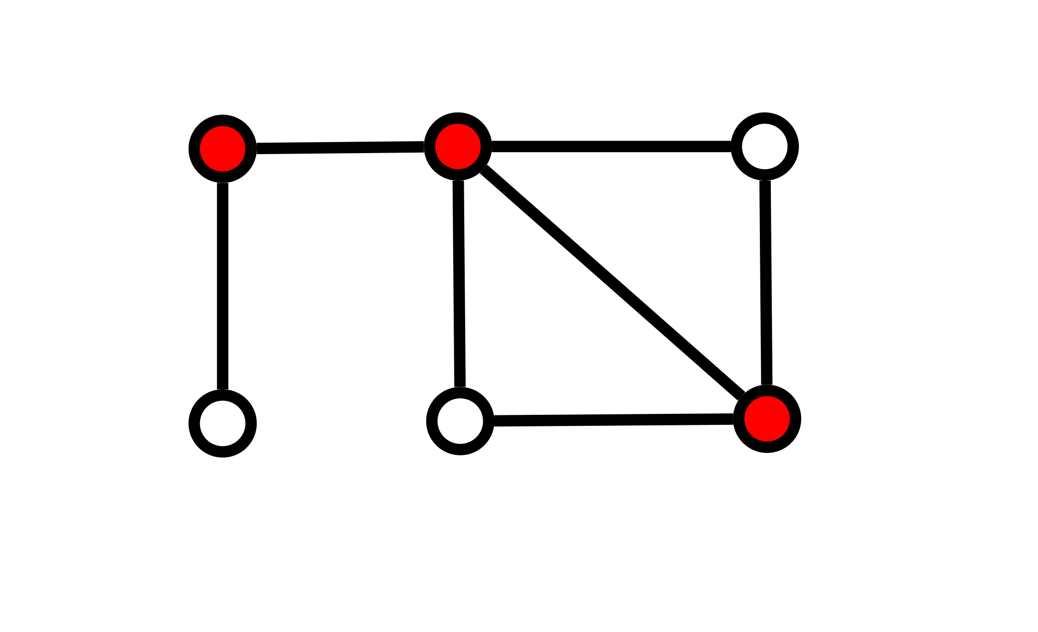
- Vertex Cover
  - Given graph G, hit all edges
  - Standard NP-ha
- Power?

Power Vertex Cover



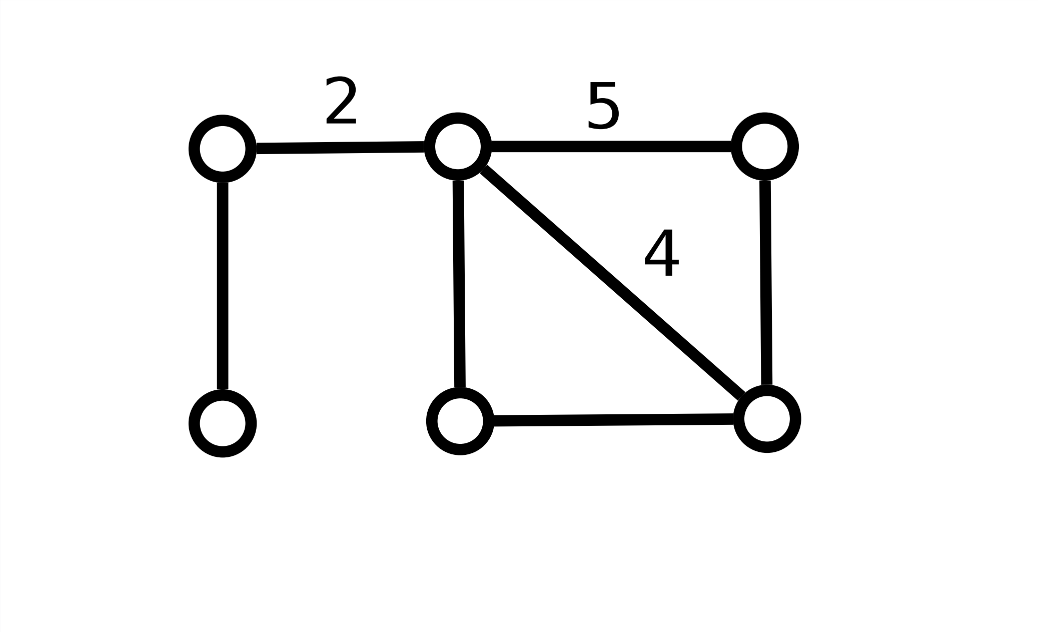
Vertex Cover: Select vertices that touch all edges

Power Vertex Cover



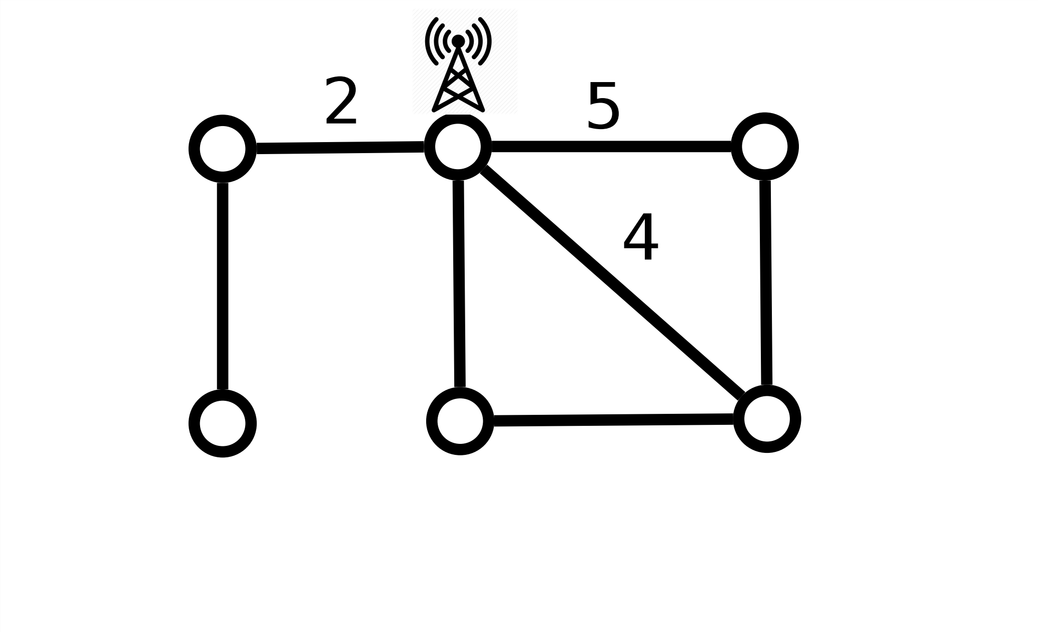
Vertex Cover: Select vertices that touch all edges

Power Vertex Cover



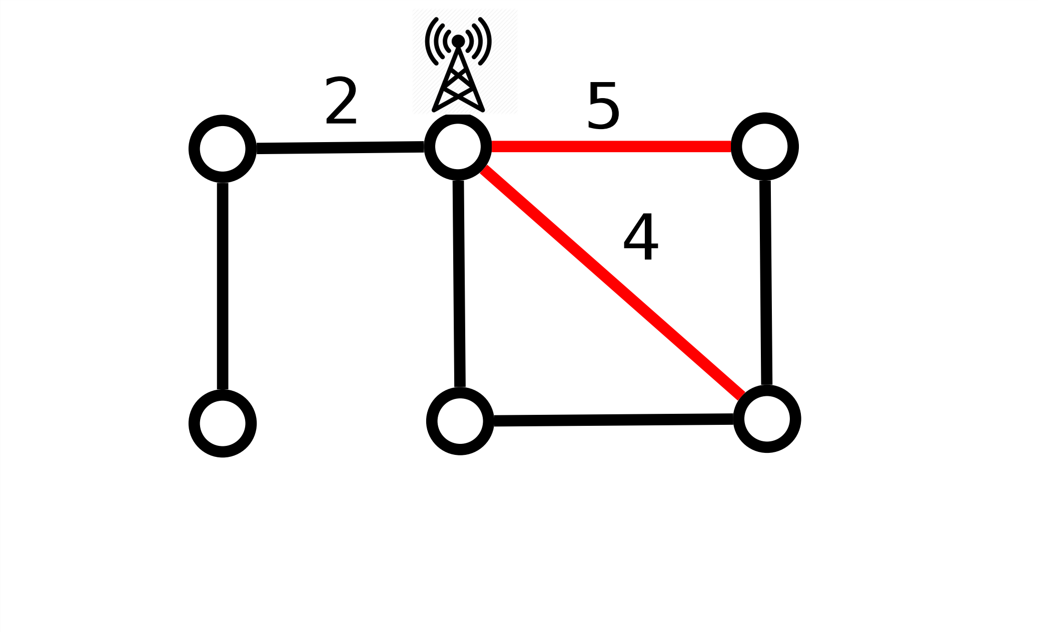
Power: Some edges **demand more power** to be covered

Power Vertex Cover



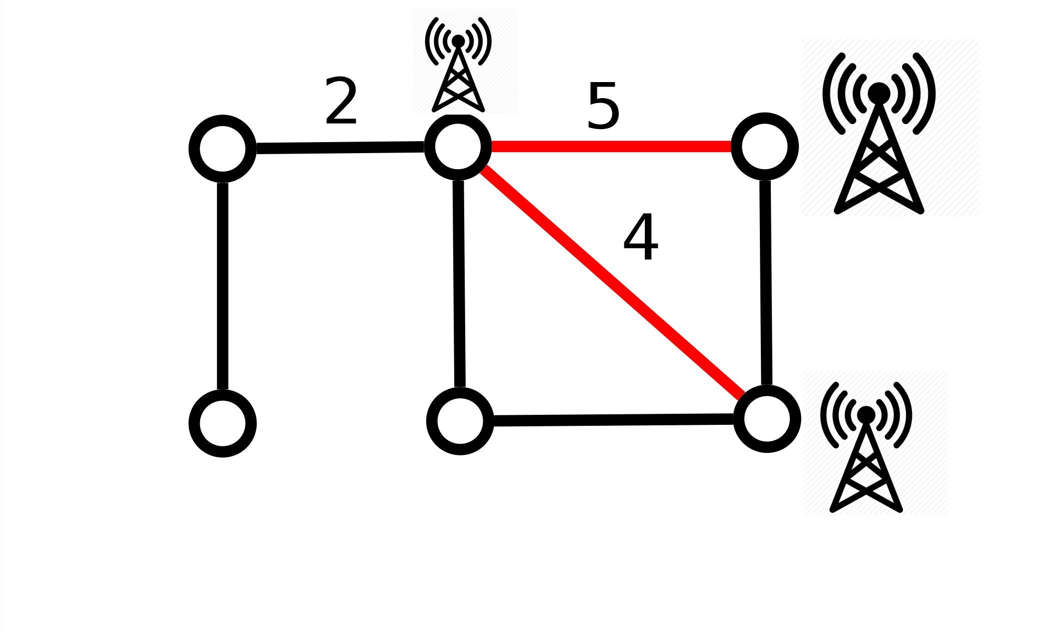
Power: Some edges **demand more power** to be covered

Power Vertex Cover



Power: Some edges **demand more power** to be covered

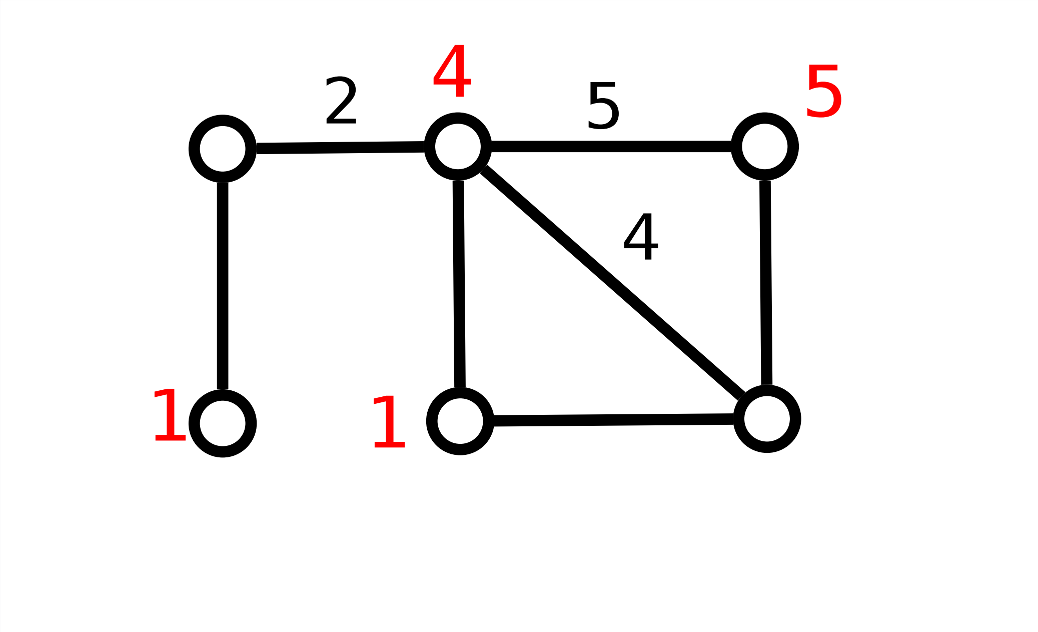
Power Vertex Cover



Power Vertex Cover: Must decide which vertices get power

. . . **and how much**

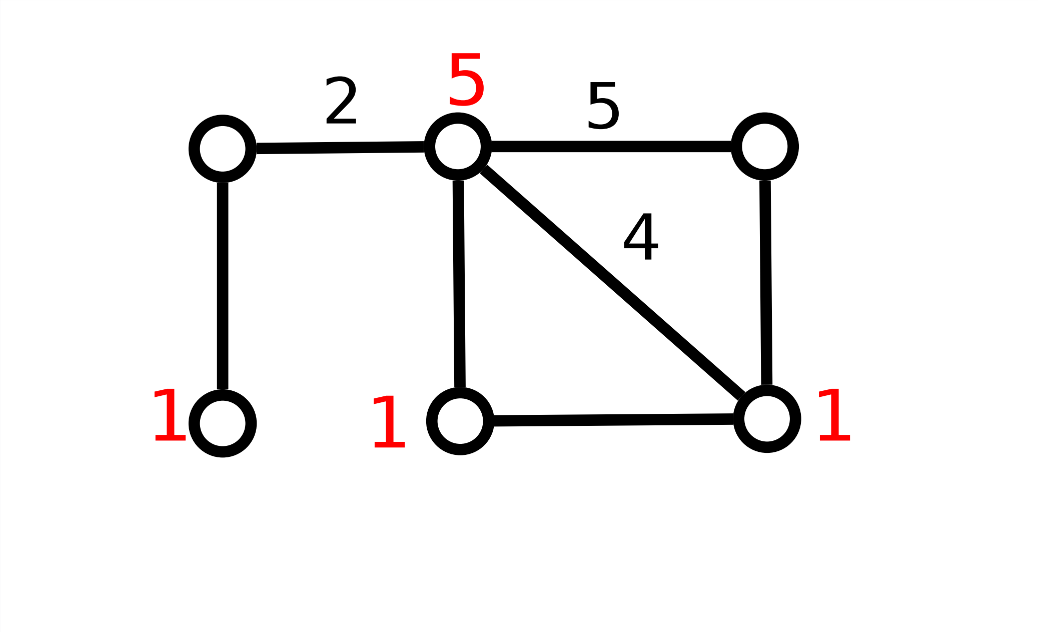
Power Vertex Cover



Power Vertex Cover: Must decide which vertices get power

. . . **and how much**

Power Vertex Cover



Formal Definition:

$$\min \sum p(v)$$

$$\max\{p(u), p(v)\} \geq d((u,v)) \ \ \forall (u,v) \in E$$

Motivation

- Applications to communication networks

Motivation

- Applications to communication networks ??

Motivation

- Applications to communication networks ??
- Interesting Generalization of Vertex Cover
  - Note: added **non-linear** constraint $\max\{p(u), p(v)\} \geq d((u, v)) \;\; \forall (u, v) \in E$
    - Compare: $p(u) + p(v) \geq d((u, v))$
  - Is this problem really different/harder from Vertex Cover?
    - Admits 2 approximation
    - In P for bipartite graphs [Angel et al. ISAAC '15]

Motivation

- Applications to communication networks ??
- Interesting Generalization of Vertex Cover
  - Note: added **non-linear** constraint
    $\max\{p(u), p(v)\} \geq d((u, v)) \;\; \forall (u, v) \in E$
    - Compare: $p(u) + p(v) \geq d((u, v))$
  - Is this problem really different/harder from Vertex Cover?
    - Admits 2 approximation
    - In P for bipartite graphs [Angel et al. ISAAC '15]
  - What about Parameterized algorithms?
    - Vertex Cover is **flagship** problem
    - Compare: Weighted VC, Capacitated VC, Connected VC, . . .

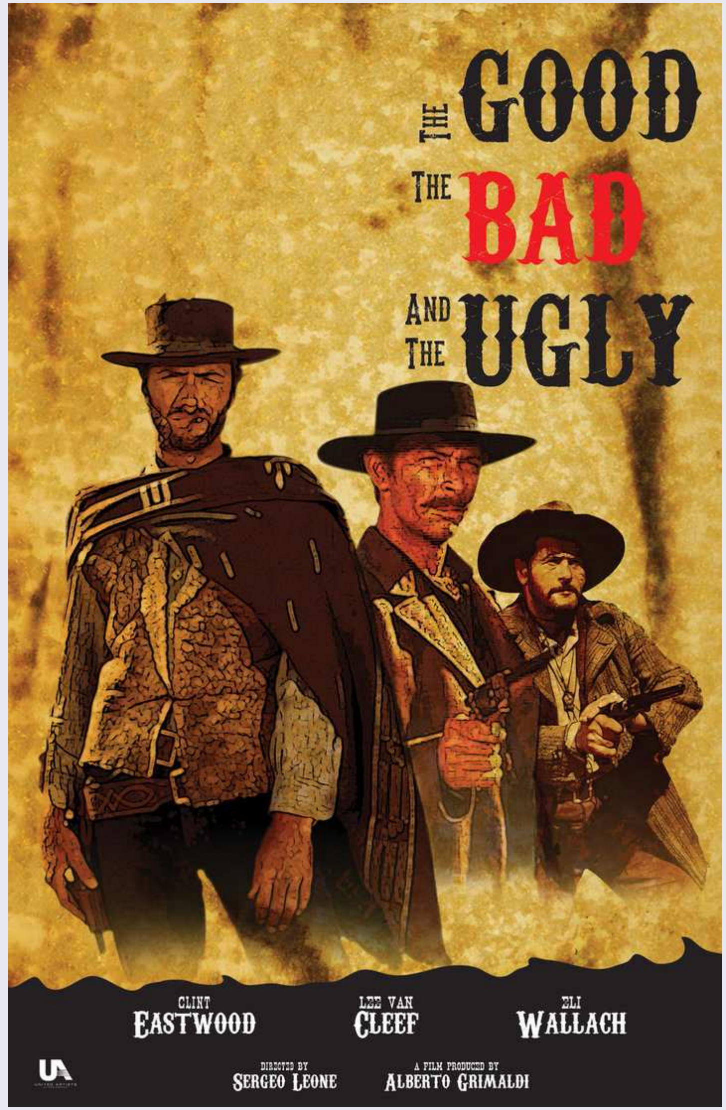
Motivation

- Applications to communication networks ??
- Interesting Generalization of Vertex Cover
  - Note: added **non-linear** constraint $\max\{p(u), p(v)\} \geq d((u,v)) \;\; \forall (u,v) \in E$
    - Compare: $p(u) + p(v) \geq d((u,v))$
  - Is this problem really different/harder from Vertex Cover?
    - Admits 2 approximation
    - In P for bipartite graphs [Angel et al. ISAAC '15]
  - What about Parameterized algorithms?
    - Vertex Cover is **flagship** problem
    - Compare: Weighted VC, Capacitated VC, Connected VC, ...

Bottom line: Natural and interesting generalization of VC

Results

Results


THE GOOD
THE BAD
AND THE UGLY
CLINT EASTWOOD
LEE VAN CLEEF
ELI WALLACH

Results

- Good
  - FPT parameterized by budget
    - Same complexity as VC!
  - FPT parameterized by used vertices

Results

- Good
  - FPT parameterized by budget
    - Same complexity as VC!
  - FPT parameterized by used vertices
- Bad
  - W-hard parameterized by treewidth!

Results

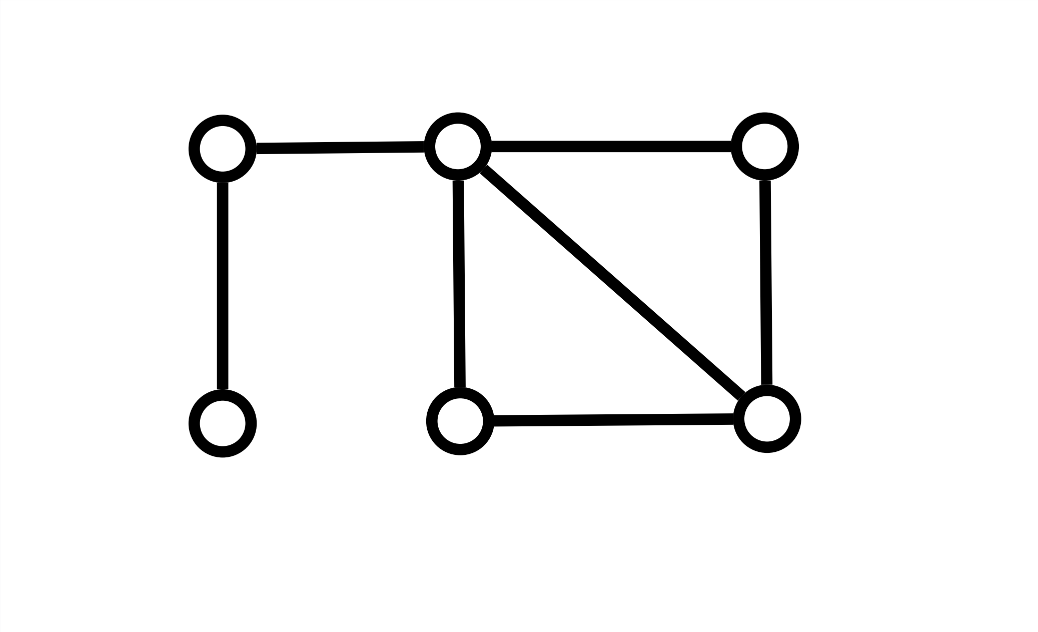
- Good
  - FPT parameterized by budget
    - Same complexity as VC!
  - FPT parameterized by used vertices
  - FPT $(1+\epsilon)$-approximation for treewidth time $(\log n/\epsilon)^{tw}$
- Bad
  - W-hard parameterized by treewidth!

Results

- Good
  - FPT parameterized by budget
    - Same complexity as VC!
  - FPT parameterized by used vertices
  - FPT $(1+\epsilon)$-approximation for treewidth time $(\log n/\epsilon)^{tw}$
- Bad
  - W-hard parameterized by treewidth!
- Ugly
  - Quadratic (bi)-kernel
    - Linear **kernel**?
  - k^k for asymmetric case
    - c^k? c^n?

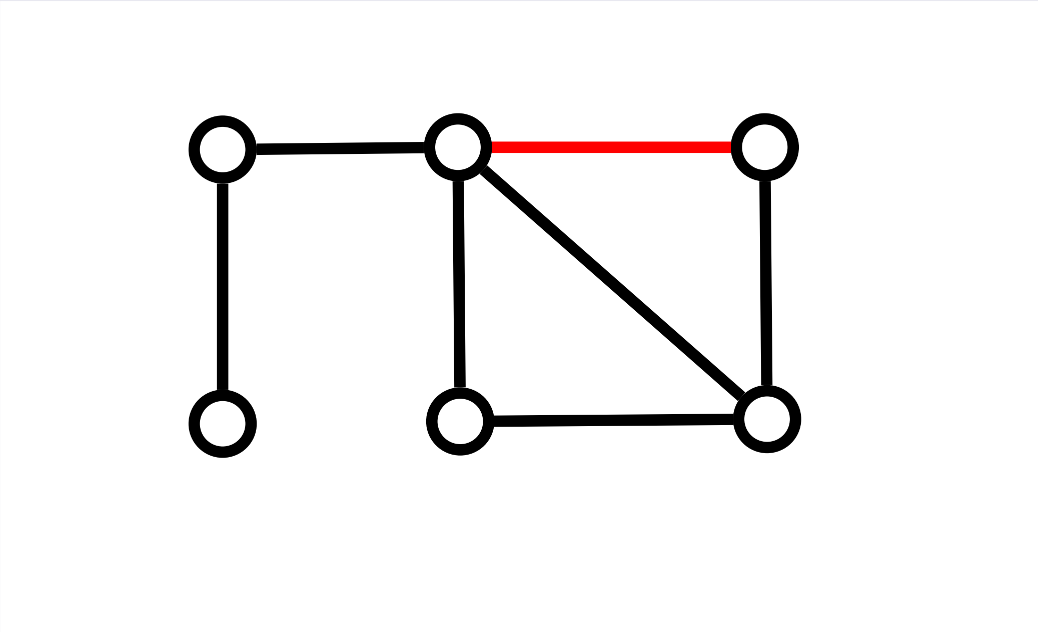
Things you (almost) already know

Basic FPT Algorithm



Basic Branching Algorithm for Vertex Cover

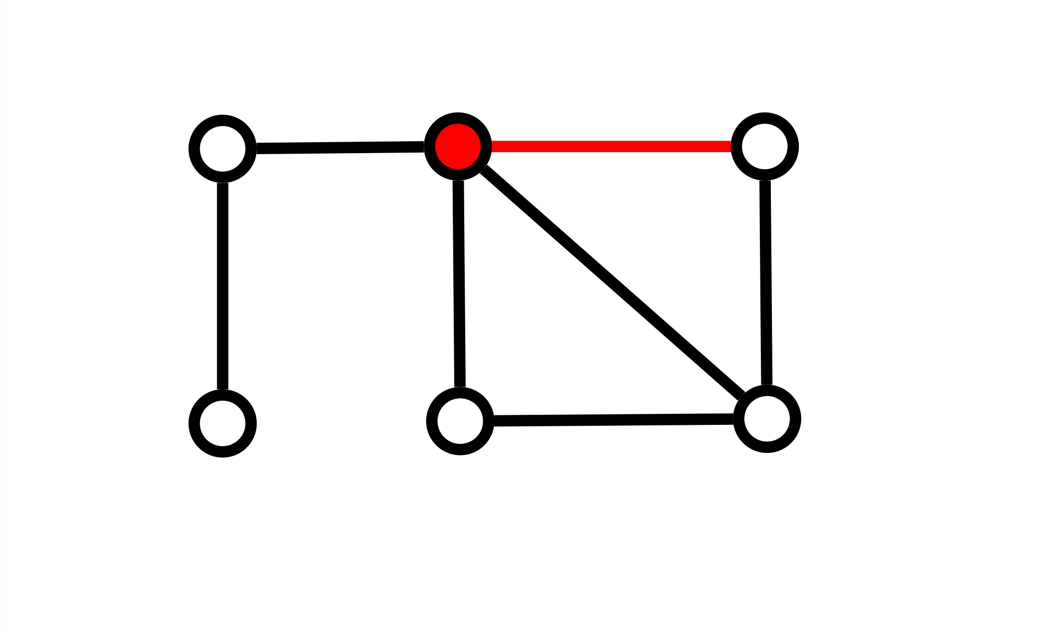
Basic FPT Algorithm



Basic Branching Algorithm for Vertex Cover

– Pick an uncovered edge

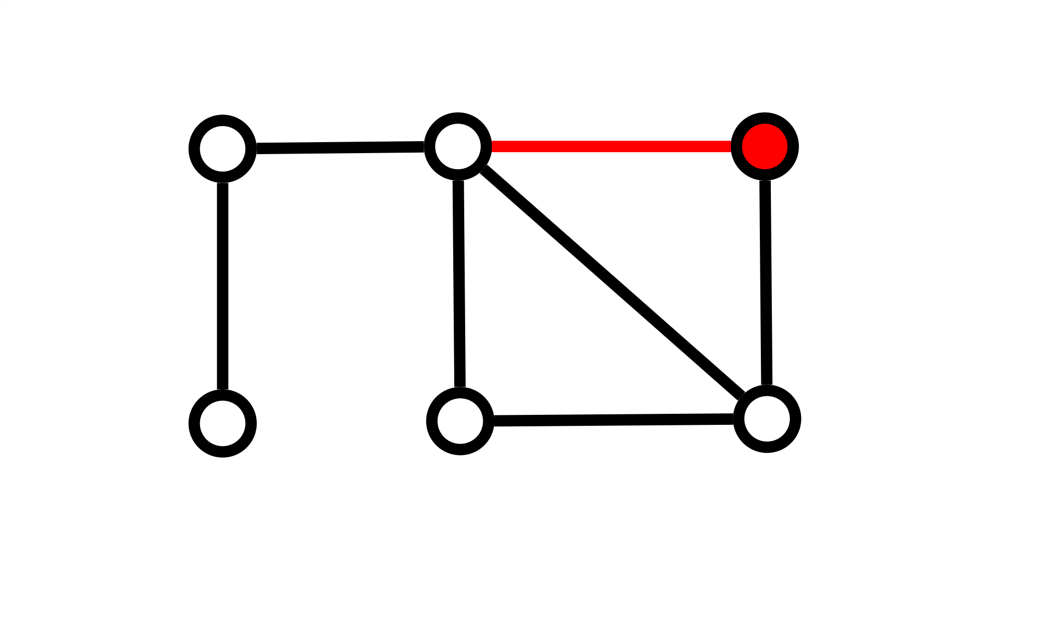
Basic FPT Algorithm



Basic Branching Algorithm for Vertex Cover

– Pick an uncovered edge
– Pick one of its endpoints (Branch)

Basic FPT Algorithm

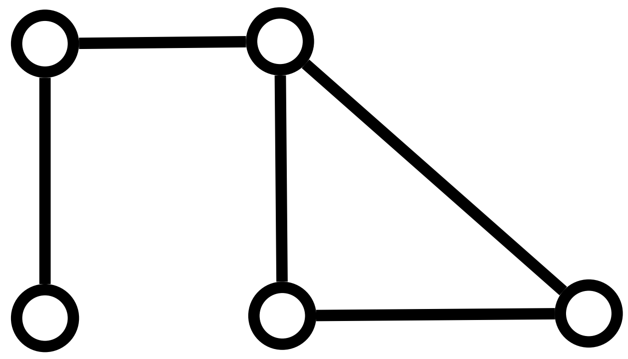


Basic Branching Algorithm for Vertex Cover

– Pick an uncovered edge

– Pick one of its endpoints (Branch)

Basic FPT Algorithm

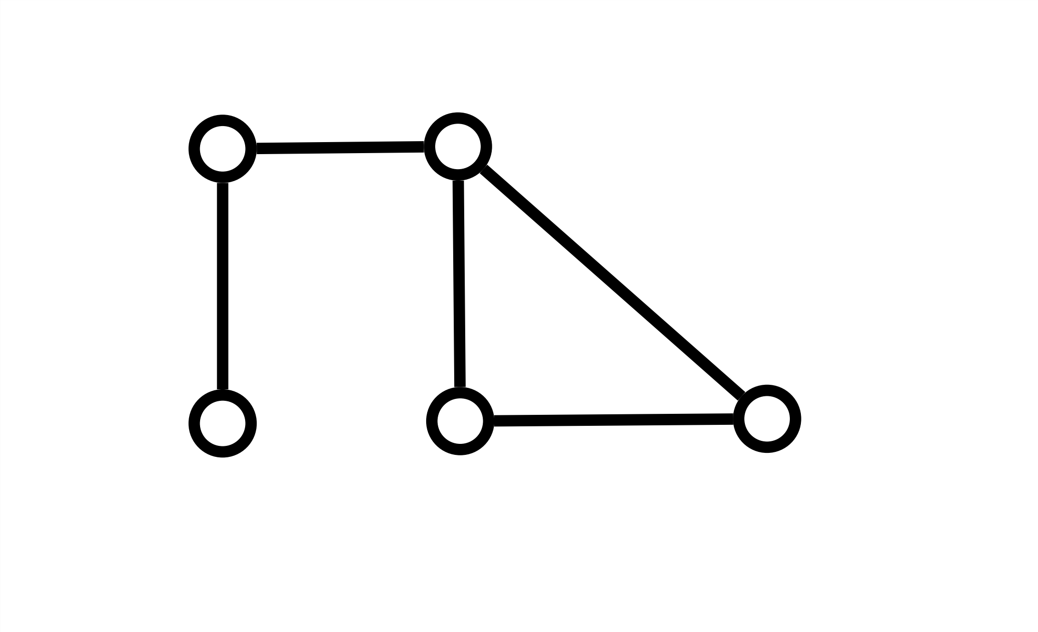


Basic Branching Algorithm for Vertex Cover

– Pick an uncovered edge
– Pick one of its endpoints (Branch)
– Remove endpoint, decrease budget by 1

Running time: 2^k

Basic FPT Algorithm



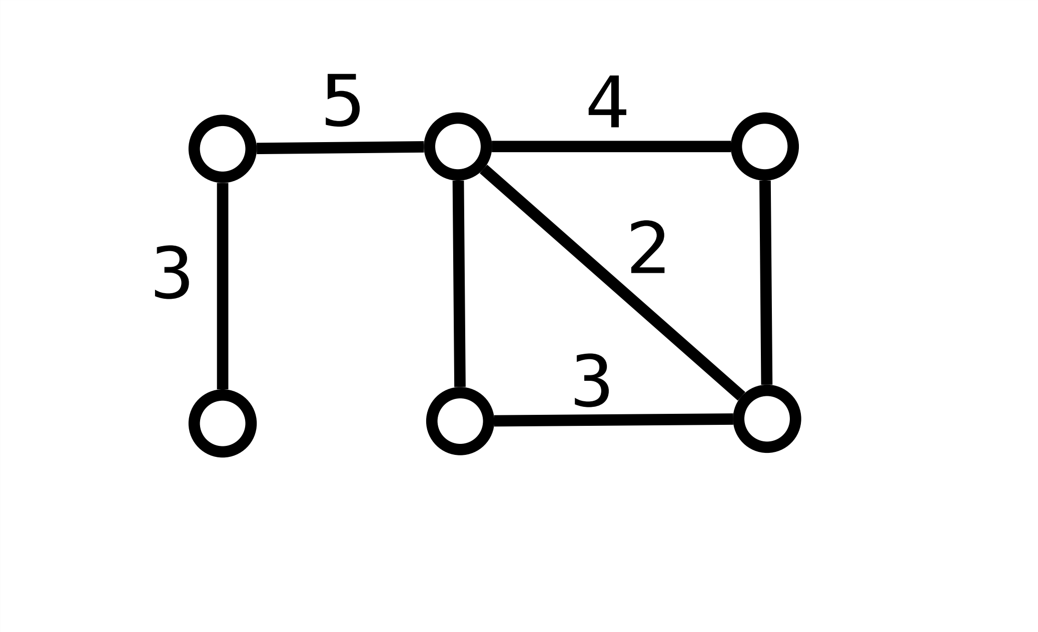
Basic Branching Algorithm for Vertex Cover

– Pick an uncovered edge
– Pick one of its endpoints (Branch)
– Remove endpoint, decrease budget by 1

Running time: 2^k

...Can be improved to 1.28^k with smarter branching

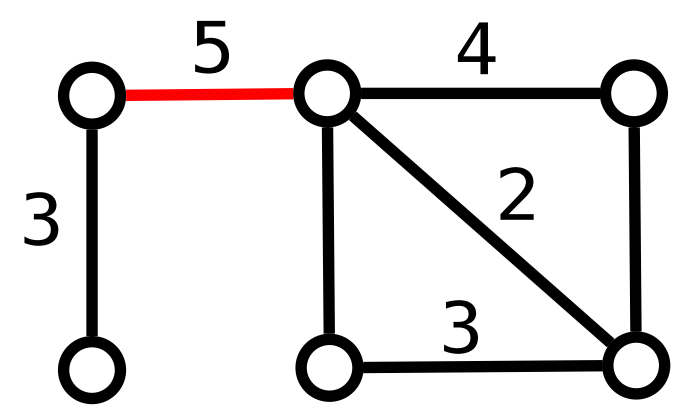
Basic FPT Algorithm



Power Vertex Cover

Parameter: Total Budget P

Basic FPT Algorithm



Power Vertex Cover

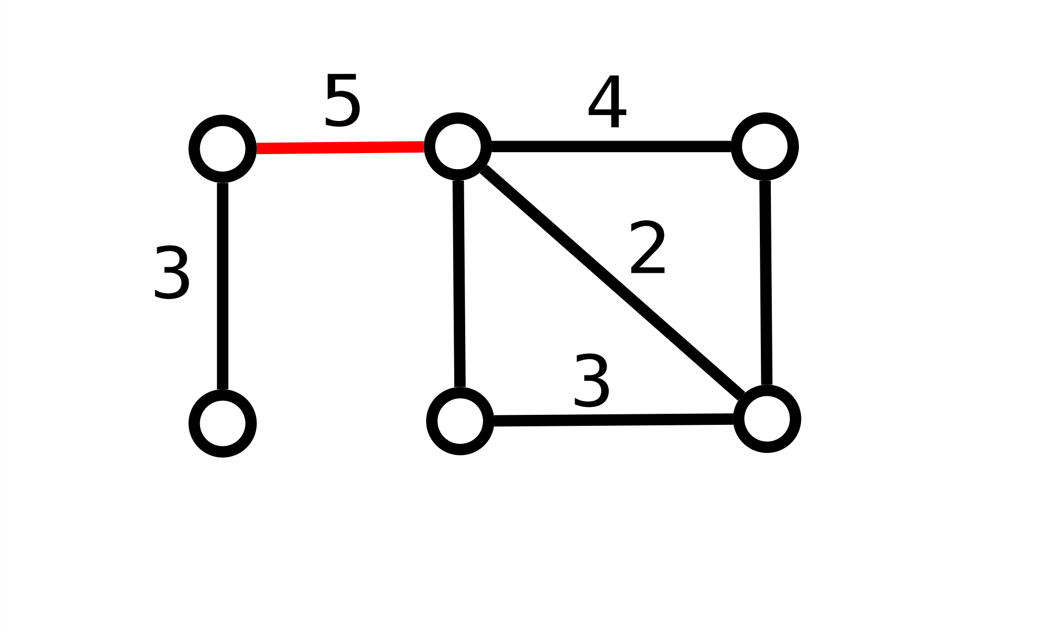
Parameter: Total Budget P

Basic Branching Algorithm

– Pick **The heaviest edge** to branch on

– If unweighted call VC algorithm

Basic FPT Algorithm



Power Vertex Cover

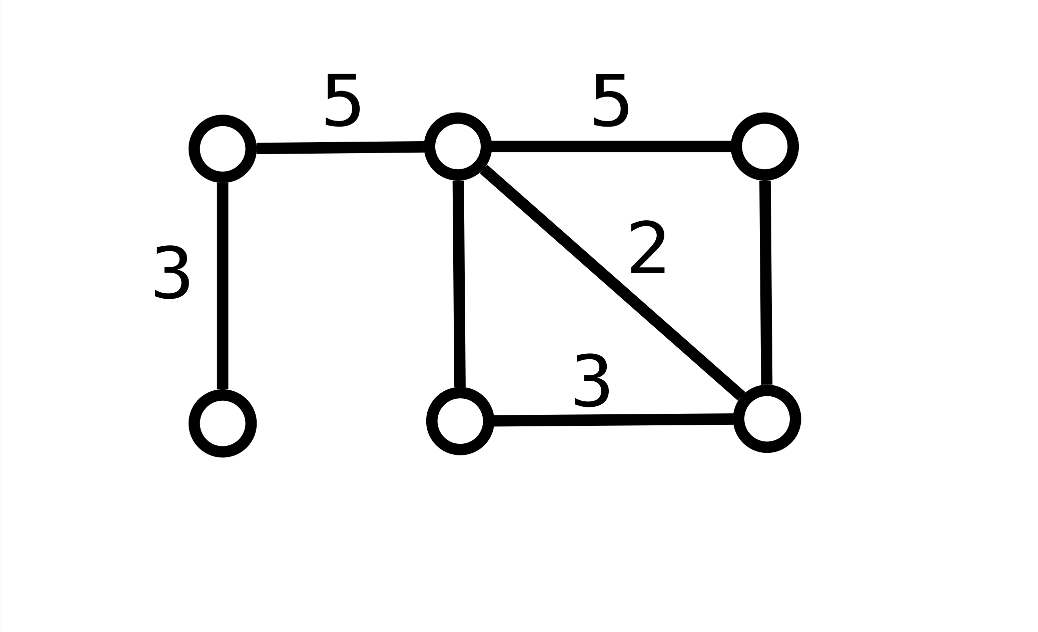
Parameter: Total Budget P

Basic Branching Algorithm

- Pick **The heaviest edge** to branch on
- If unweighted call VC algorithm

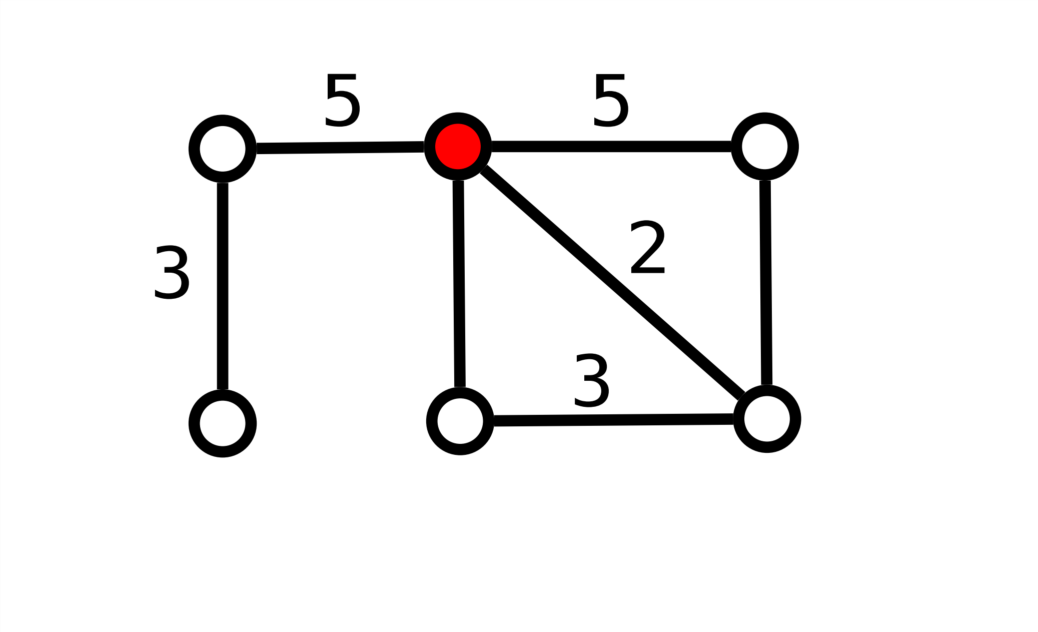
Almost as good as best VC algorithm

Basic FPT Algorithm



Power Vertex Cover
Parameter: Total Budget P
Better Branching Algorithm

– If two heaviest edges share vertex branch there



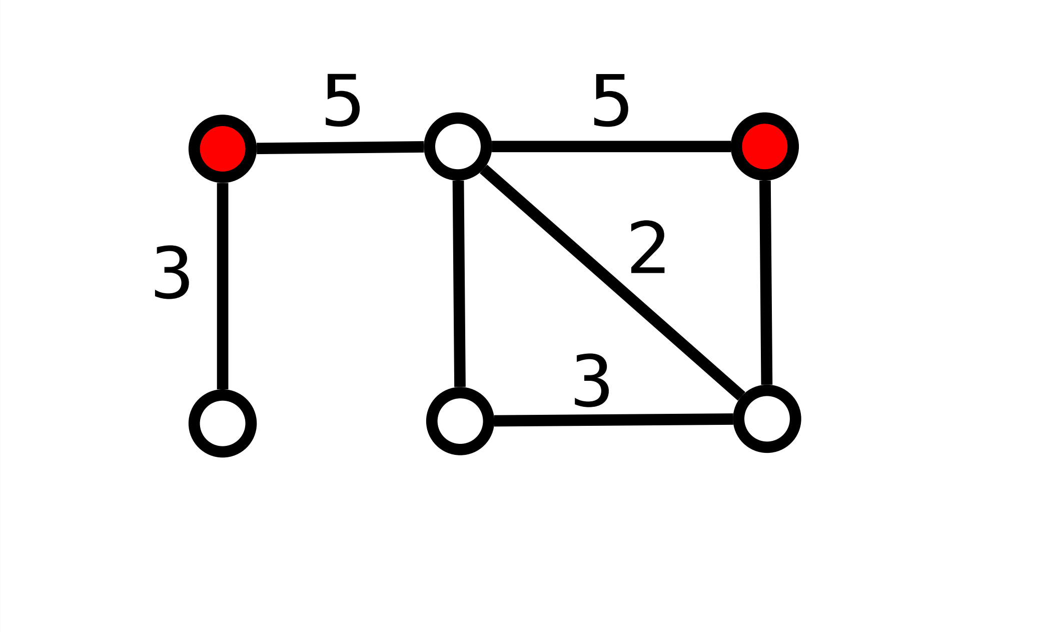
Power Vertex Cover

Parameter: Total Budget P

Better Branching Algorithm

– If two heaviest edges share vertex branch there

Basic FPT Algorithm



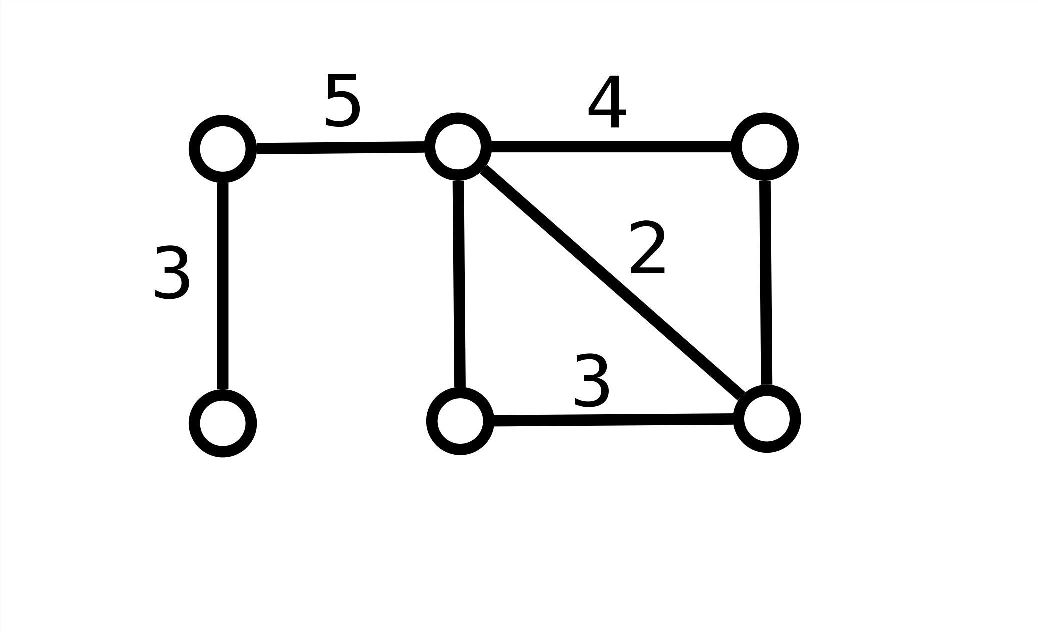
Power Vertex Cover

Parameter: Total Budget P

Better Branching Algorithm

– If two heaviest edges share vertex branch there

Basic FPT Algorithm



Power Vertex Cover

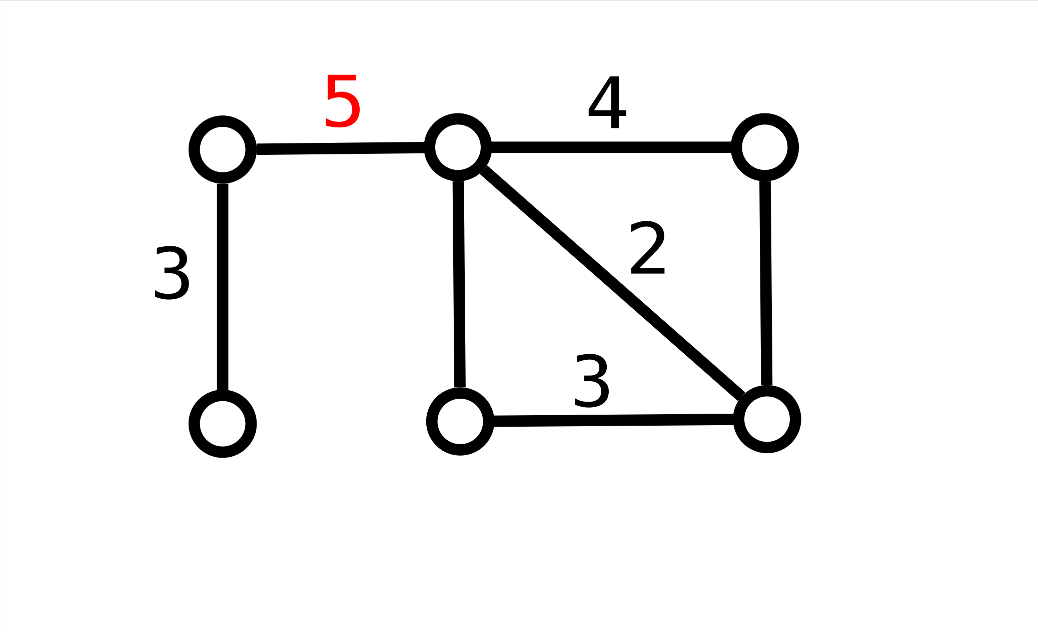
Parameter: Total Budget P

Better Branching Algorithm

– If two heaviest edges share vertex branch there

– If not decrease weight of heaviest edge and budget by 1

Basic FPT Algorithm



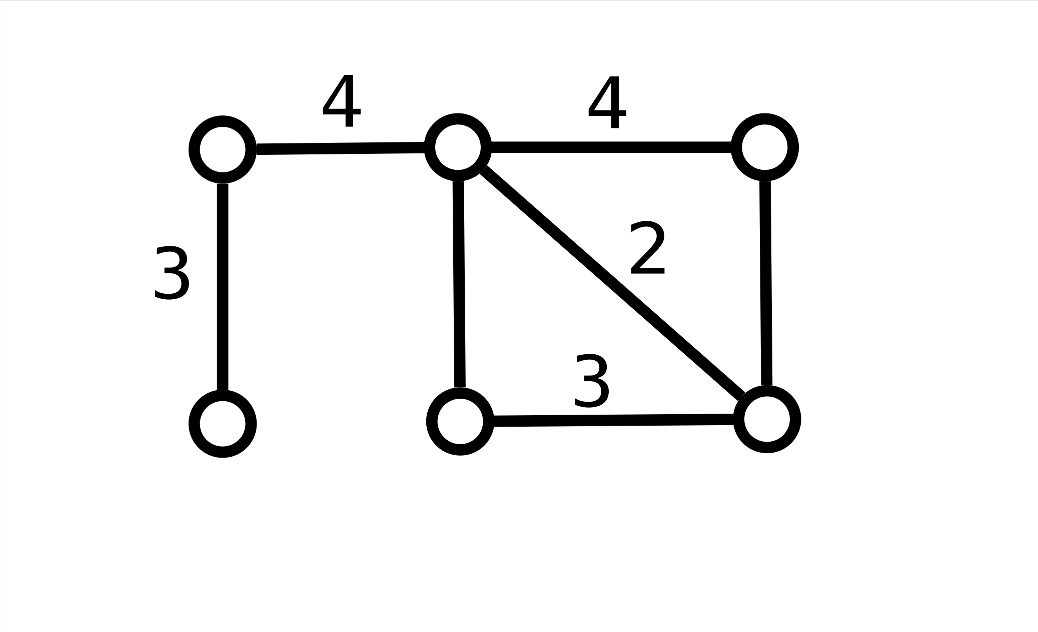
Power Vertex Cover

Parameter: Total Budget P

Better Branching Algorithm

– If two heaviest edges share vertex branch there

– If not decrease weight of heaviest edge and budget by 1



Power Vertex Cover

Parameter: Total Budget P

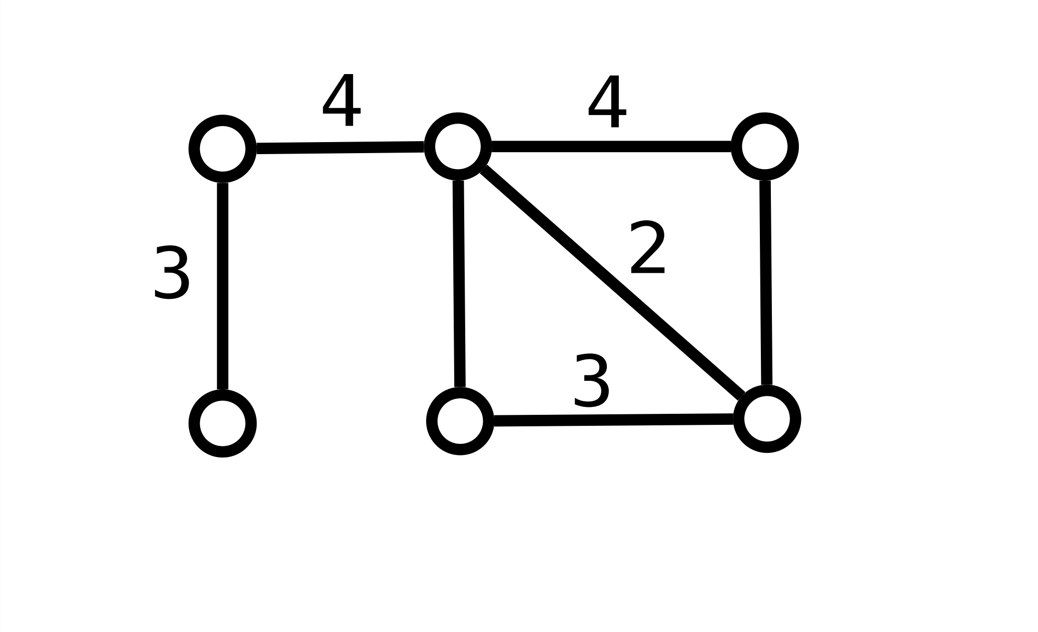
Better Branching Algorithm

– If two heaviest edges share vertex branch there

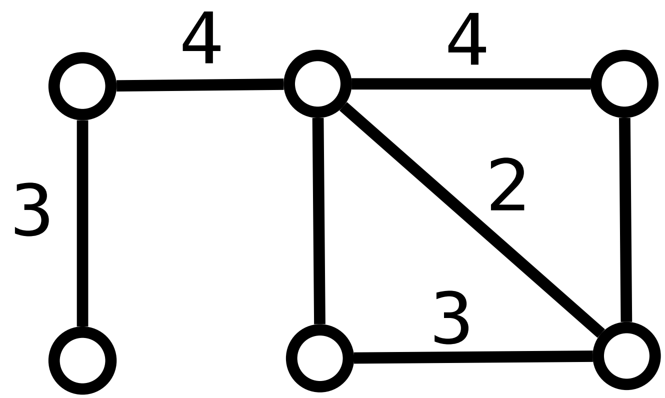
– If not decrease weight of heaviest edge and budget by 1

As fast as best VC algorithm! (1.28^P)

Basic FPT Algorithm



Power Vertex Cover
Parameter: Total Budget P
Parameter 2: Number of selected vertices k



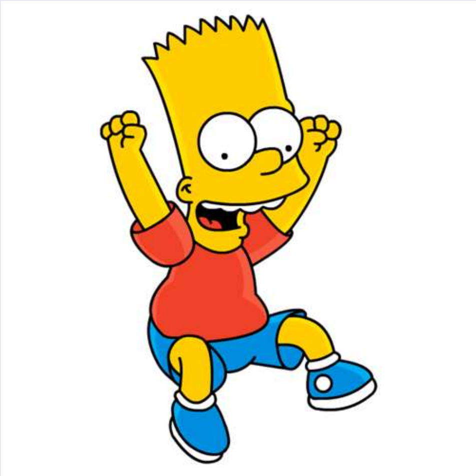
Power Vertex Cover
Parameter: Total Budget P
Parameter 2: Number of selected vertices k

Same algorithm gives 1.41^k

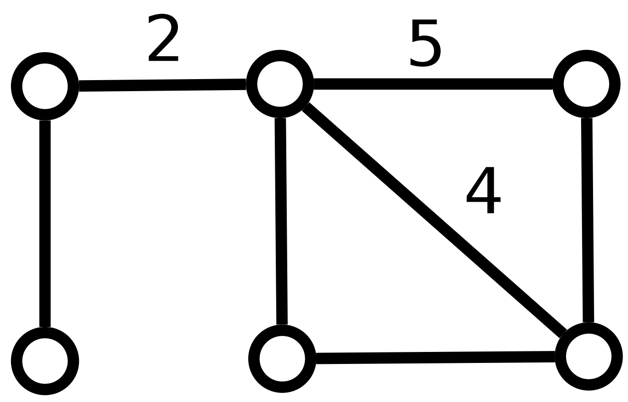
Note: $k < P$ so this is a harder problem
Q: Can we do as fast as VC here?

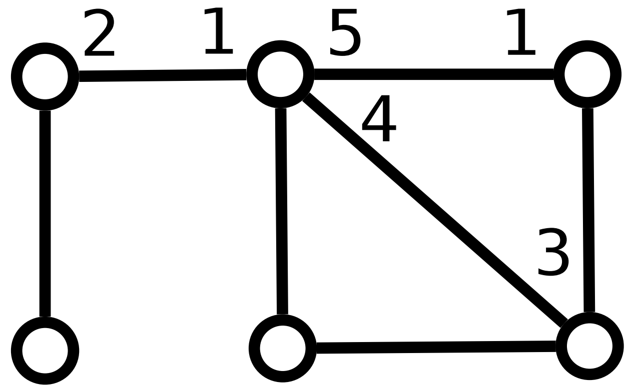
This is too easy!

Let's make things more interesting!

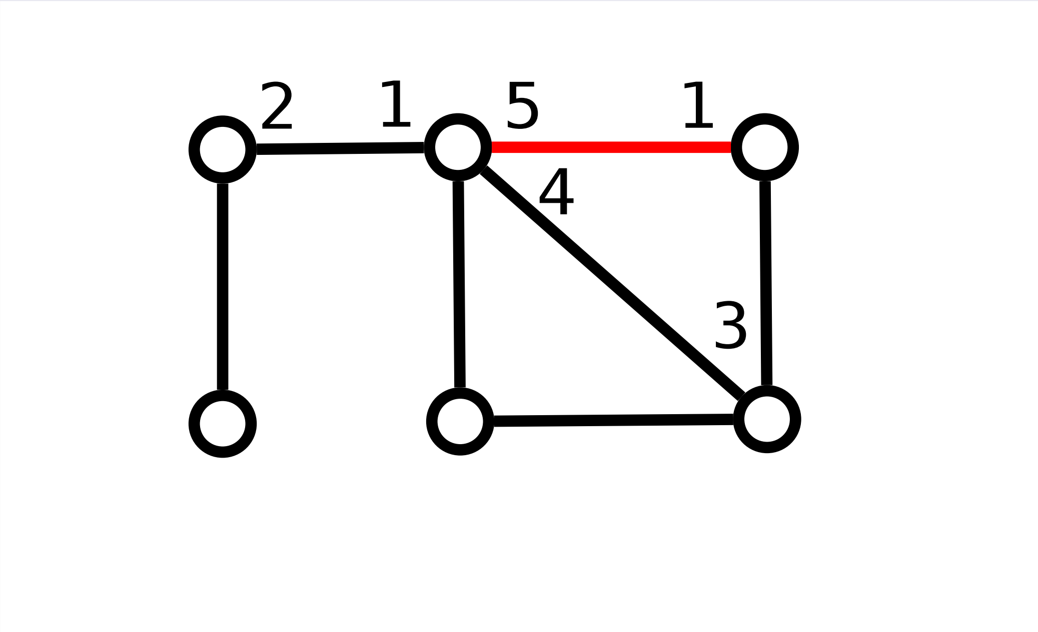


The Asymmetric Case





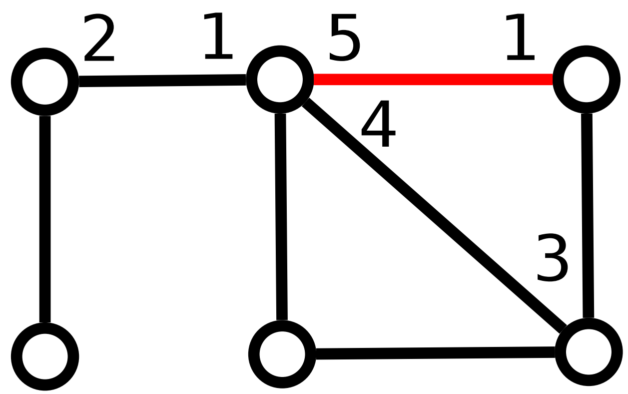
Asymmetric Power Vertex Cover:
Each edge has a different demand for each endpoint



Asymmetric Power Vertex Cover:
Each edge has a different demand for each endpoint

- Problem: what is a “heaviest” edge?
- Branching not guaranteed to be fast

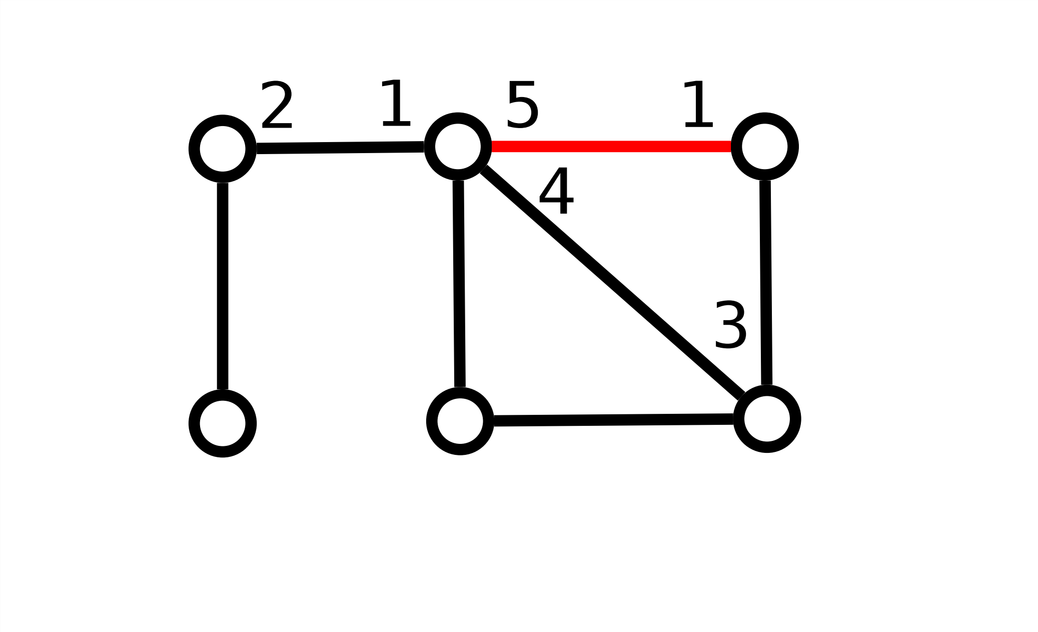
The Asymmetric Case



Asymmetric Power Vertex Cover:
Each edge has a different demand for each endpoint

- Problem: what is a “heaviest” edge?
- Branching not guaranteed to be fast
- Result: 1.325^P algorithm with case analysis

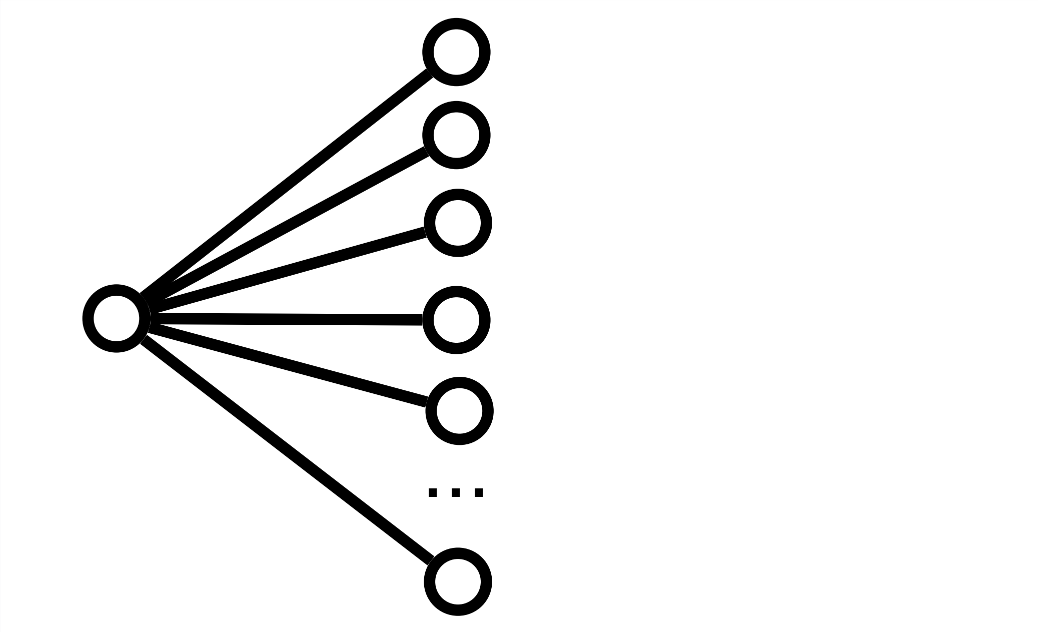
The Asymmetric Case



Asymmetric Power Vertex Cover:
Each edge has a different demand for each endpoint

- Problem: what is a “heaviest” edge?
- Branching not guaranteed to be fast
- Result: 1.325^P algorithm with case analysis
- What about parameter k?

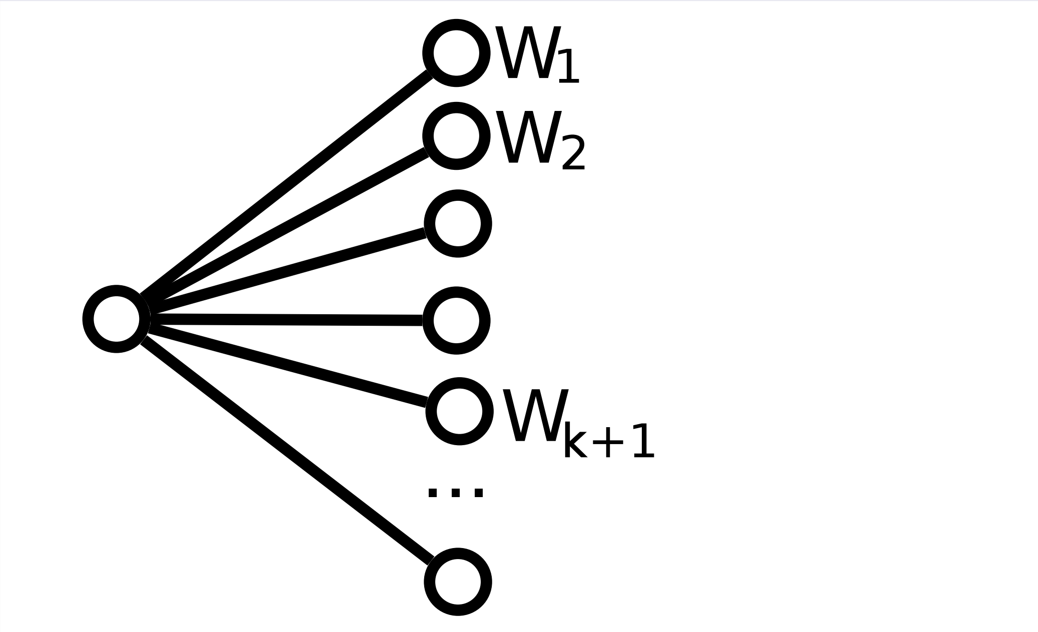
A simple kernel for the Asymmetric case



A simple kernel for parameter k

- Consider a vertex withe degree $> k$

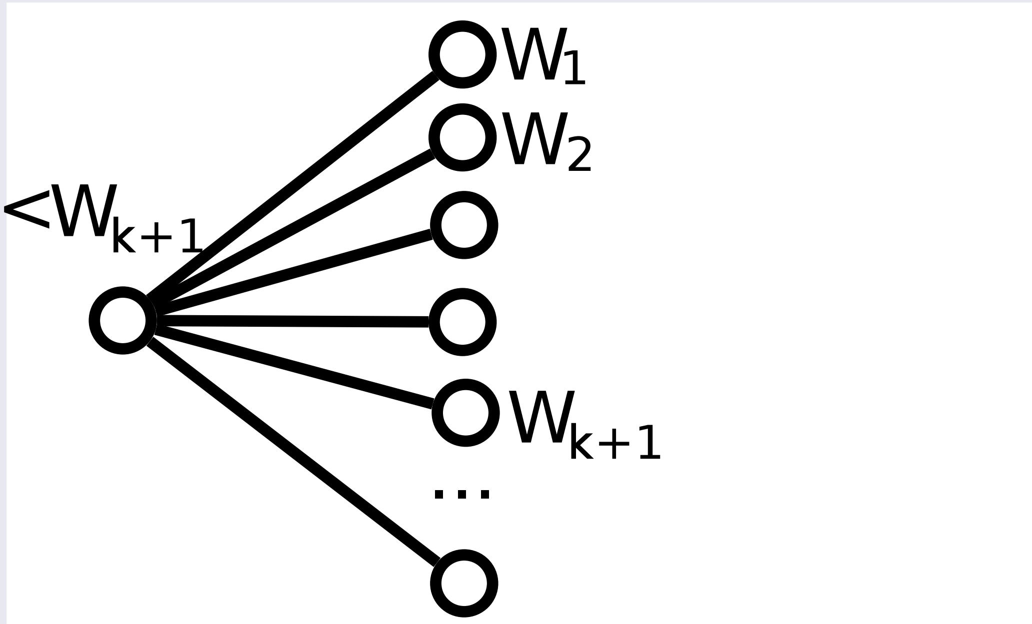
A simple kernel for the Asymmetric case



A simple kernel for parameter k

- Consider a vertex withe degree $> k$
- Order its incident edges by demand

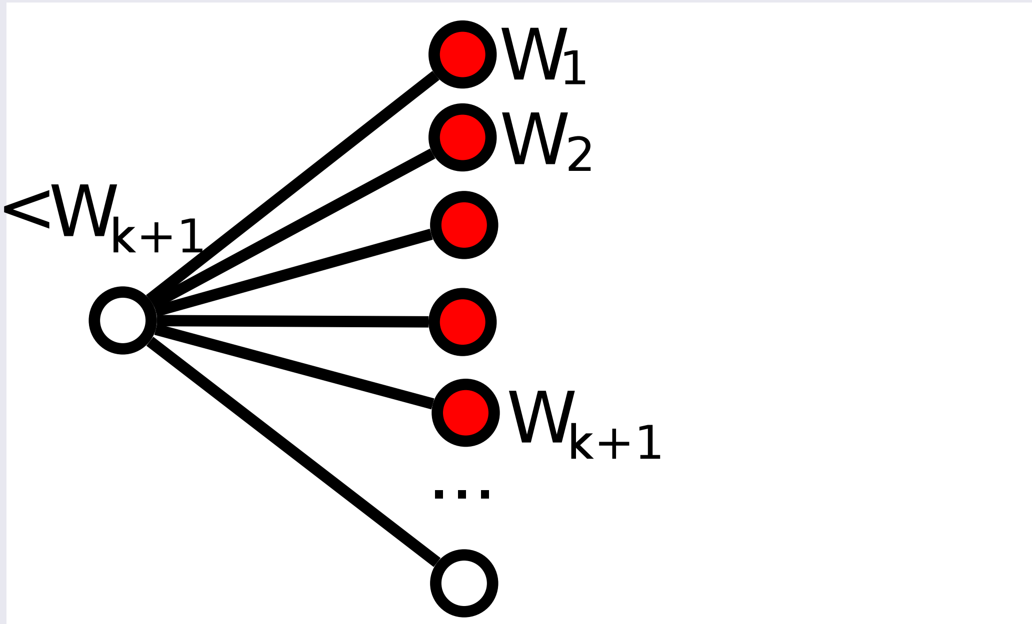
A simple kernel for the Asymmetric case



A simple kernel for parameter k

- Consider a vertex withe degree $> k$
- Order its incident edges by demand
- If the vertex gets power lower than the $k+1$-th cost...

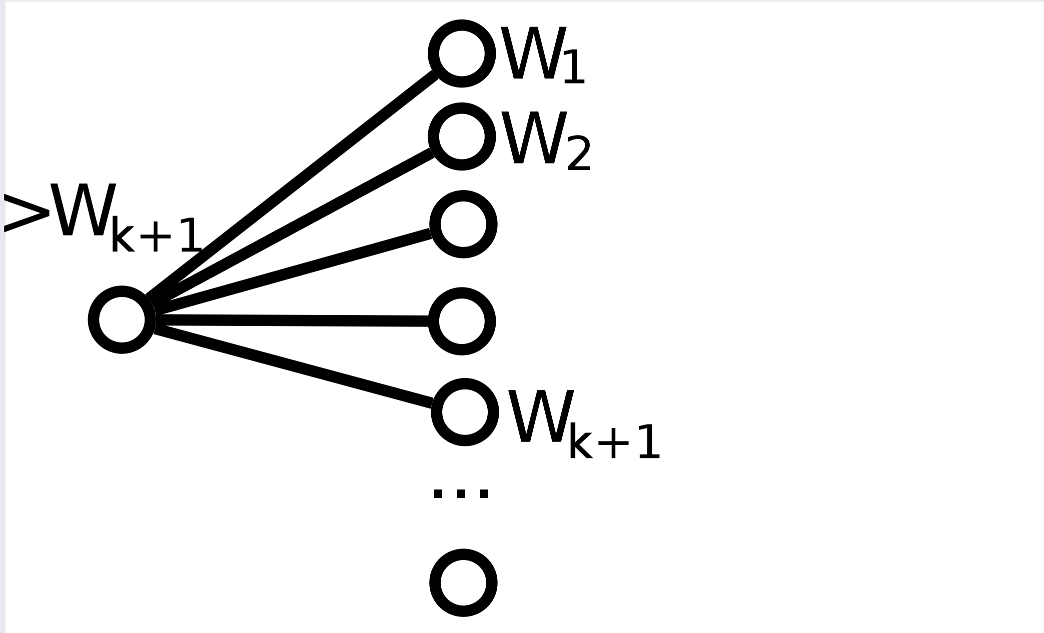
A simple kernel for the Asymmetric case



A simple kernel for parameter k

- Consider a vertex withe degree $> k$
- Order its incident edges by demand
- If the vertex gets power lower than the $k+1$-th cost. . .
- we need to use $> k$ vertices

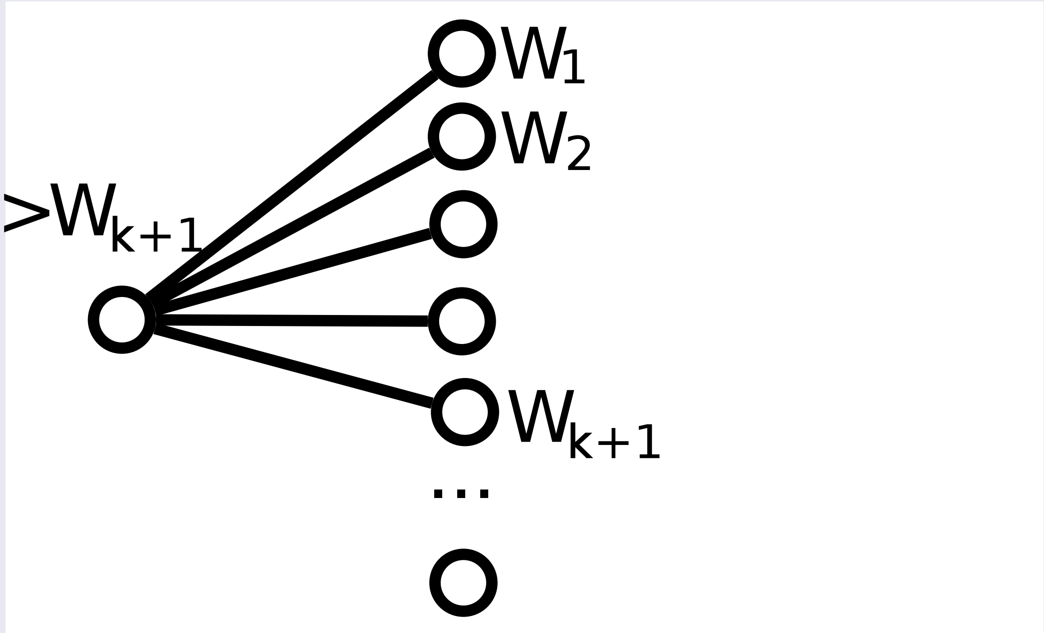
A simple kernel for the Asymmetric case



A simple kernel for parameter k

- Consider a vertex withe degree $> k$
- Order its incident edges by demand
- If the vertex gets power lower than the $k+1$-th cost...
- we need to use $> k$ vertices
- We can therefore give it power W_{k+1}, which covers the lower cost edges

A simple kernel for the Asymmetric case



A simple kernel for parameter k

- In the end graph has $O(k^2)$ edges left.
- **Q:** Running time of FPT algorithm?
- **Q:** Kernel inherently asymmetric?
- **Q:** Linear (order) kernel?

Things which are different

W-hard for treewidth

Reminder:

- Treewidth is most basic graph width
- Vertex Cover solvable in $2^{tw}n$ time

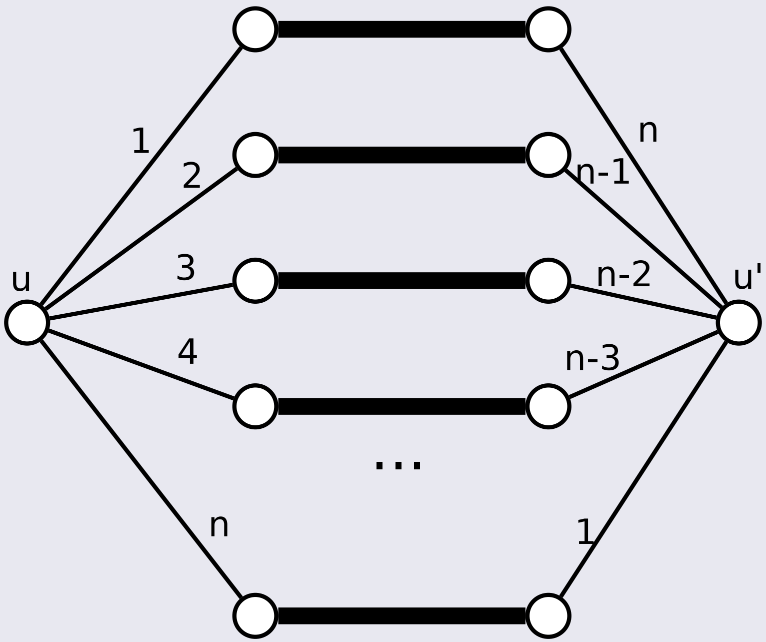
W-hard for treewidth

Theorem: There is no $n^{o(t)}$ algorithm for PVC (under ETH)

W-hard for treewidth

Theorem: There is no $n^{o(t)}$ algorithm for PVC (under ETH)
Proof: Reduction from Multi-Colored Clique

Theorem: There is no $n^{o(t)}$ algorithm for PVC (under ETH)
Proof: Reduction from Multi-Colored Clique

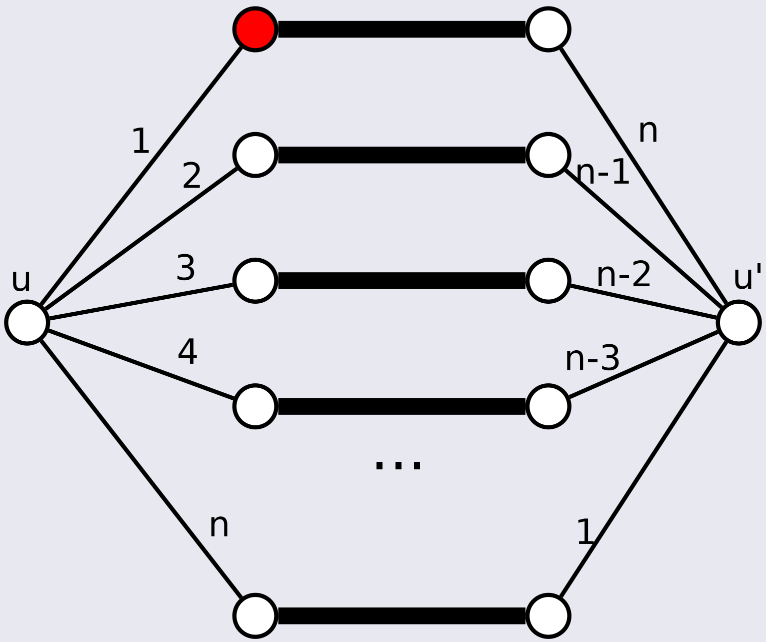


Vertex Selection Gadget:

- Thick edges have weight n

W-hard for treewidth

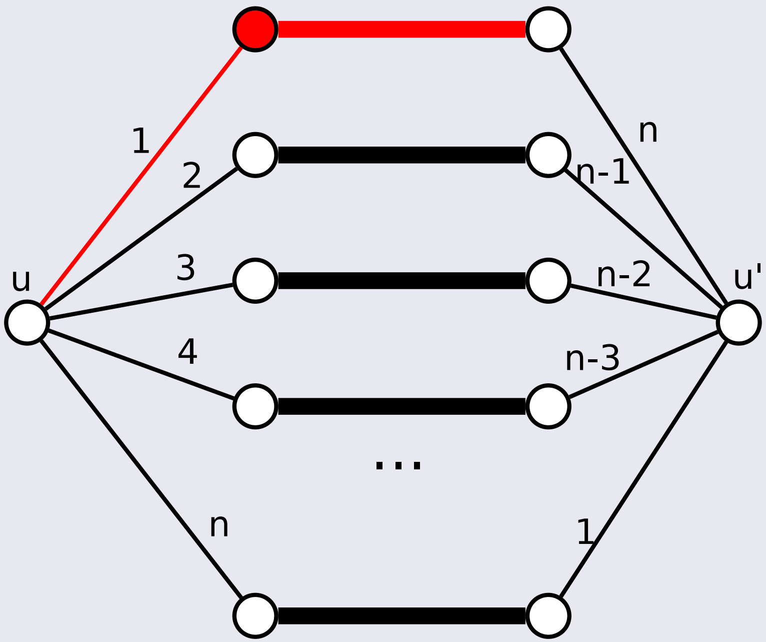
Theorem: There is no $n^{o(t)}$ algorithm for PVC (under ETH)
Proof: Reduction from Multi-Colored Clique



Vertex Selection Gadget:

- Thick edges have weight n
- At least one internal vertex must get power n

Theorem: There is no $n^{o(t)}$ algorithm for PVC (under ETH)
Proof: Reduction from Multi-Colored Clique

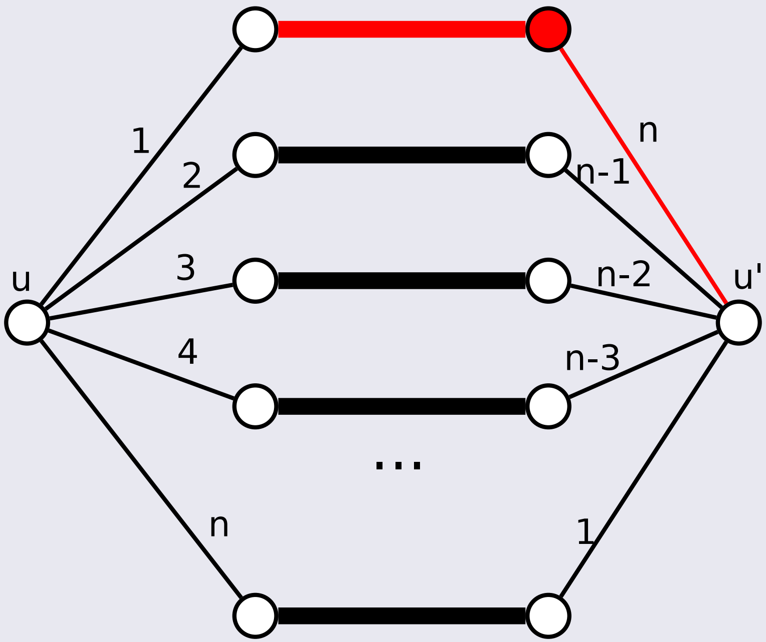


Vertex Selection Gadget:

- Thick edges have weight n
- At least one internal vertex must get power n

W-hard for treewidth

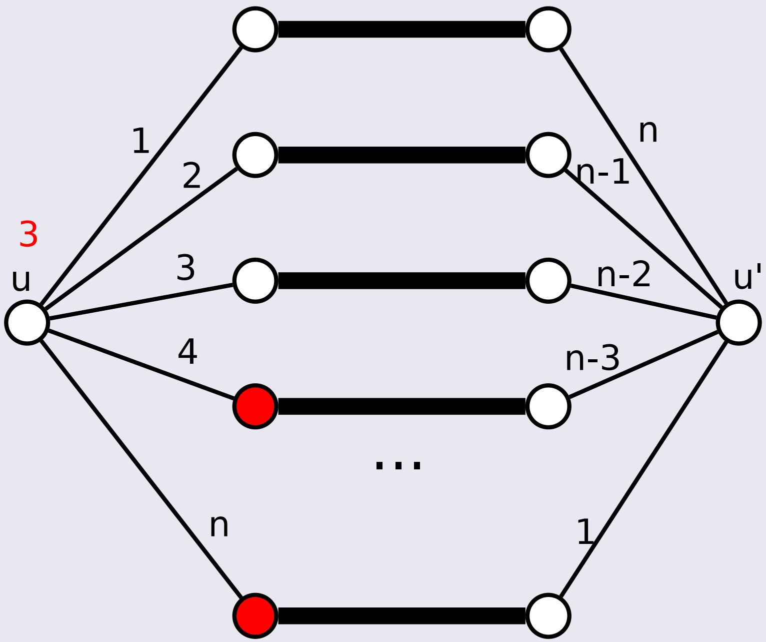
Theorem: There is no $n^{o(t)}$ algorithm for PVC (under ETH)
Proof: Reduction from Multi-Colored Clique



Vertex Selection Gadget:

- Thick edges have weight n
- At least one internal vertex must get power n

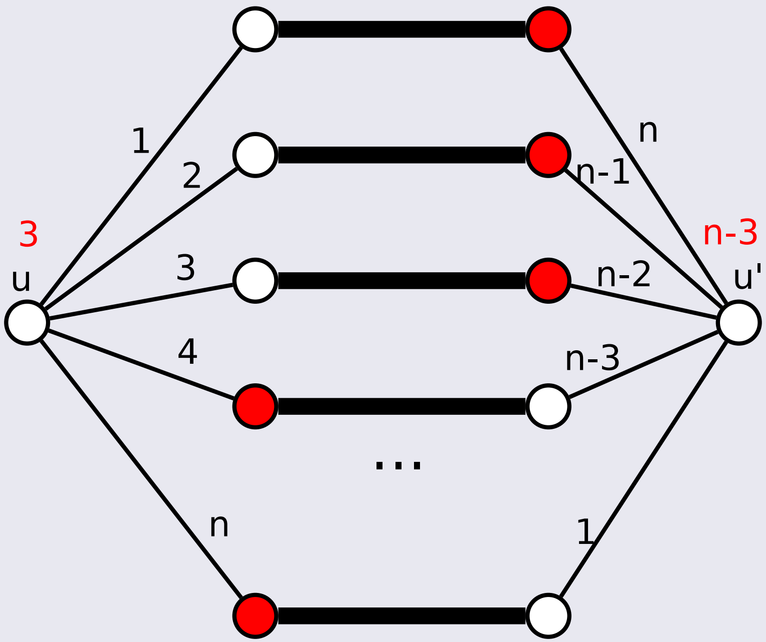
Theorem: There is no $n^{o(t)}$ algorithm for PVC (under ETH)
Proof: Reduction from Multi-Colored Clique



Vertex Selection Gadget:

- Thick edges have weight n
- At least one internal vertex must get power n
- Main claim: Optimal power gives i to u and $n-i$ to u'

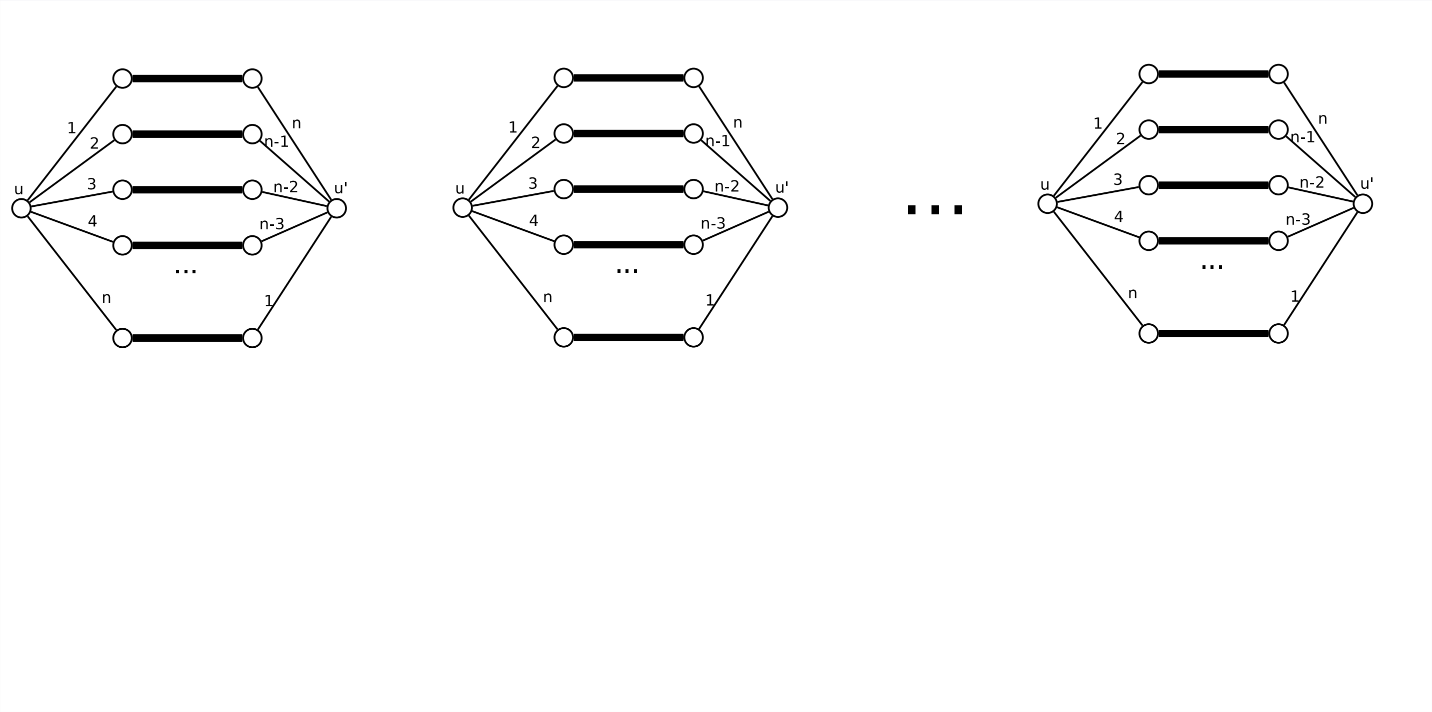
Theorem: There is no $n^{o(t)}$ algorithm for PVC (under ETH)
Proof: Reduction from Multi-Colored Clique



Vertex Selection Gadget:

- Thick edges have weight n
- At least one internal vertex must get power n
- Main claim: Optimal power gives i to u and $n-i$ to u'
- Encode vertex selection by power level for u

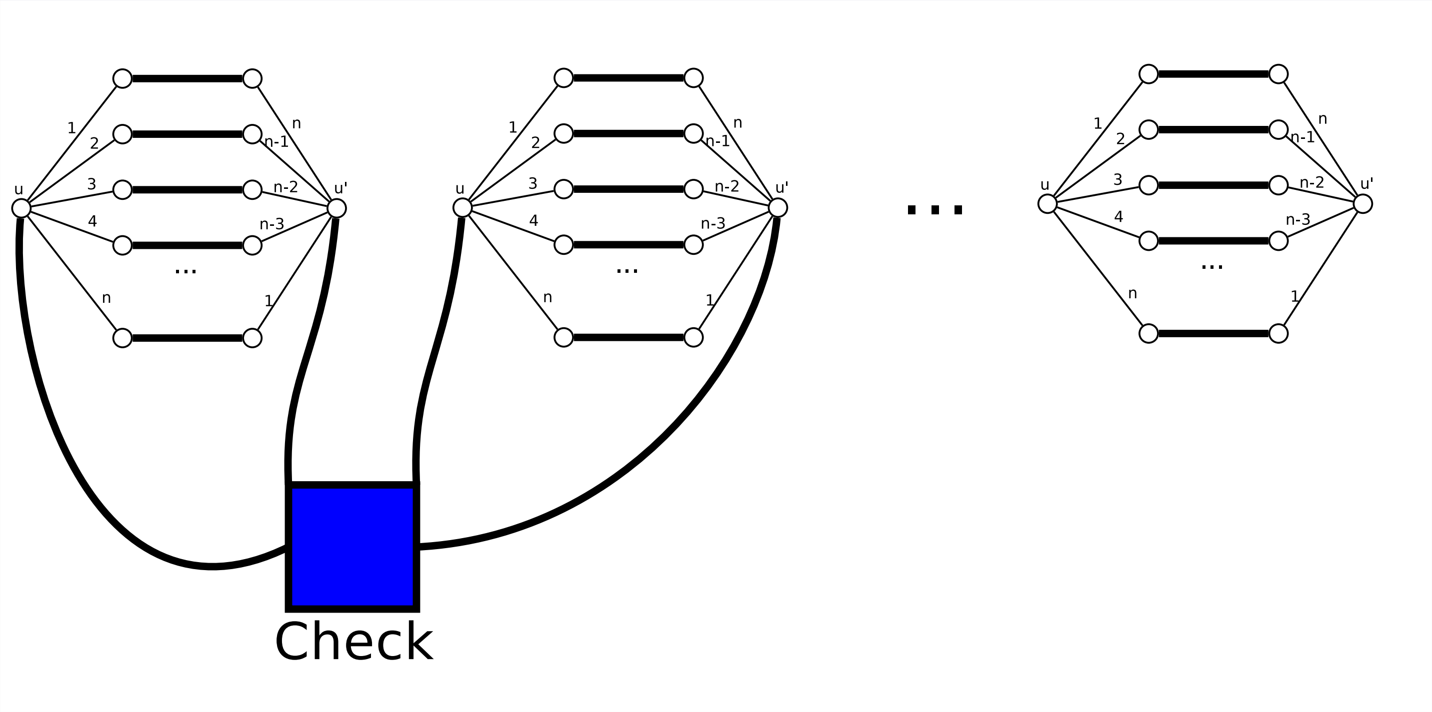
Theorem: There is no $n^{o(t)}$ algorithm for PVC (under ETH)
Proof: Reduction from Multi-Colored Clique



- Take k copies of previous gadget

W-hard for treewidth

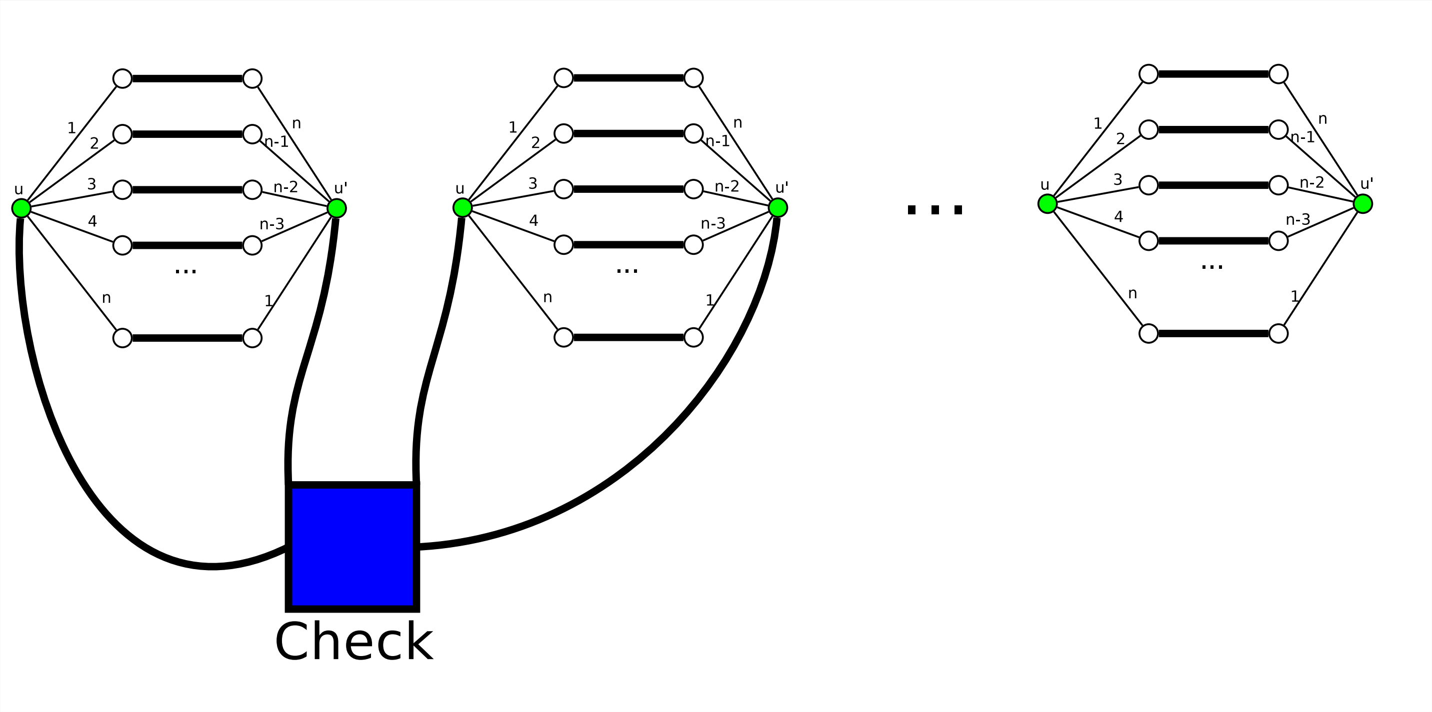
Theorem: There is no $n^{o(t)}$ algorithm for PVC (under ETH)
Proof: Reduction from Multi-Colored Clique



- Take k copies of previous gadget
- Add a (small) check gadget for each non-edge of original graph

W-hard for treewidth

Theorem: There is no $n^{o(t)}$ algorithm for PVC (under ETH)
Proof: Reduction from Multi-Colored Clique

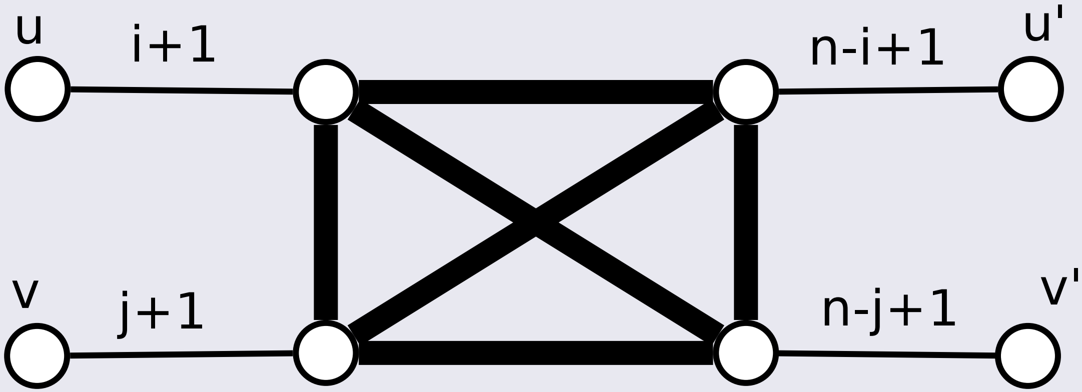


- Take k copies of previous gadget
- Add a (small) check gadget for each non-edge of original graph
- Whole graph has treewidth $O(k)$

W-hard for treewidth

Theorem: There is no $n^{o(t)}$ algorithm for PVC (under ETH)
Proof: Reduction from Multi-Colored Clique

Check gadget:



Meaning: not (i and j)

Treewidth Algorithms

Treewidth doesn't work!

Treewidth doesn't work!

Actually it's not so bad. . .

Treewidth Algorithms

Easy **Exact** Algorithms

- $(\Delta + 1)^{tw} n$ time
- $(M + 1)^{tw} n$ time (M=maximum weight)

Main observation: Each vertex has limited number of reasonable power values.

(These running times are optimal)

Treewidth Algorithms

Easy **Exact** Algorithms

- $(\Delta + 1)^{tw} n$ time
- $(M + 1)^{tw} n$ time (M=maximum weight)

Main observation: Each vertex has limited number of reasonable power values.

(These running times are optimal)
Can we do better?

FPT **Approximation** Scheme

- $(M+1)^{tw}n$ time to solve exactly

FPT **Approximation** Scheme

- $(M+1)^{tw} n$ time to solve exactly
- Main idea: **Rounding**
  - Instead of power value p for each vertex store $\lfloor \log_{1+\epsilon}(p) \rfloor$
  - At most $\log M / \log(1+\epsilon)$ possible values
  - At most a $(1+\epsilon)$ factor from correct value
  - If $M = n^{O(1)}$ running time $(\log n/\epsilon)^{tw}$
  - (If not, easy: think Knapsack)

FPT **Approximation** Scheme

- $(M+1)^{tw}n$ time to solve exactly
- Main idea: **Rounding**
  - Instead of power value p for each vertex store $\lfloor \log_{1+\epsilon}(p) \rfloor$
  - At most $\log M / \log(1+\epsilon)$ possible values
  - At most a $(1+\epsilon)$ factor from correct value
  - If $M = n^{O(1)}$ running time $(\log n/\epsilon)^{tw}$
  - (If not, easy: think Knapsack)

Bottom line: Fast FPT algorithm for W-hard problem, only $(1+\epsilon)$ error!
(This is part of a more general technique [L. ICALP '14])

Things we don’t understand

Linear (bi)-kernel?

- Recall: $O(k^2)$ kernel for (Asymmetric) PVC
- Can we do better?
- Using LP perhaps?

Linear (bi)-kernel?

- Recall: $O(k^2)$ kernel for (Asymmetric) PVC
- Can we do better?
- Using LP perhaps?
- Recall: for VC we have if LP says $v(x) = 0$, we should not take x

Linear (bi)-kernel?

- Recall: $O(k^2)$ kernel for (Asymmetric) PVC
- Can we do better?
- Using LP perhaps?
- Recall: for VC we have if LP says $v(x) = 0$, we should not take x
- **Theorem:** Given an instance of PVC and an optimal fractional LP solution that sets $p(x) = 0$ it is NP-hard to decide whether to take x.

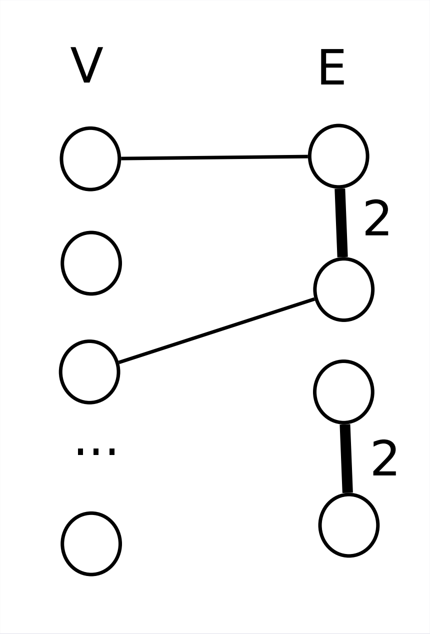
LPs don't help

Theorem: Given an instance of PVC and an optimal fractional LP solution that sets $p(x) = 0$ it is NP-hard to decide whether to take x.

LPs don't help

Theorem: Given an instance of PVC and an optimal fractional LP solution that sets $p(x) = 0$ it is NP-hard to decide whether to take x.

Reduction from VC



- Left side contains vertices, right edges
- Incidence encoded with weight 1 edges
- Optimal fractional solution: weight 1 to all right vertices

Conclusions

- Interesting generalization of Vertex Cover
- W-hard for treewidth
- But approximable!

Open questions:

- Linear kernel?
- c^k for asymmetric?
- FPT for feedback vertex set?

Thank you!