

# Polynomial Time Algorithms for Tracking Paths

---

Pratibha Choudhary

Indian Institute of Technology Jodhpur

\*Institute of Mathematical Sciences Chennai

IWOCA 2020

# Contents

# Contents

❖ Introduction

# Contents

- ❖ Introduction
- ❖ Characterization of Tracking Set

# Contents

- ❖ Introduction
- ❖ Characterization of Tracking Set
- ❖ Bounded Degree Graphs

# Contents

- ❖ Introduction
- ❖ Characterization of Tracking Set
- ❖ Bounded Degree Graphs
- ❖ Chordal Graphs

# Contents

- ❖ Introduction
- ❖ Characterization of Tracking Set
- ❖ Bounded Degree Graphs
- ❖ Chordal Graphs
- ❖ Tracking Using Edges

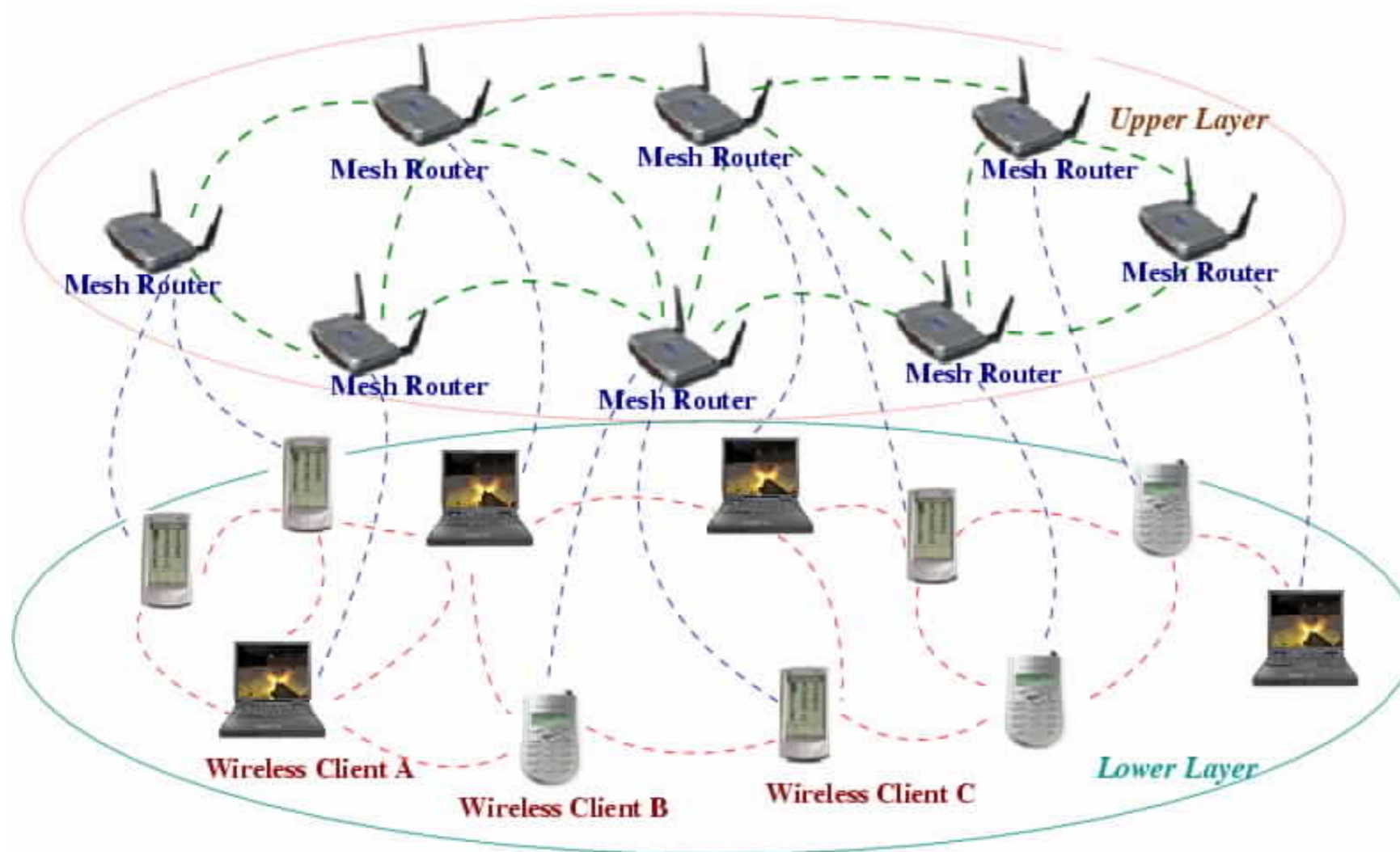
# Contents

- ❖ Introduction
- ❖ Characterization of Tracking Set
- ❖ Bounded Degree Graphs
- ❖ Chordal Graphs
- ❖ Tracking Using Edges
- ❖ Conclusion



# Motivation

- Network Data Tracking



# Motivation

- Network Data Tracking
- Secure Facility Movement Tracking



# Motivation

- Network Data Tracking
- Secure Facility Movement Tracking
- Traffic Monitoring



# Aim — Design a system to Track Paths



# Aim — Design a system to Track Paths

- Find a **method** to identify the path traversed by entities in a network



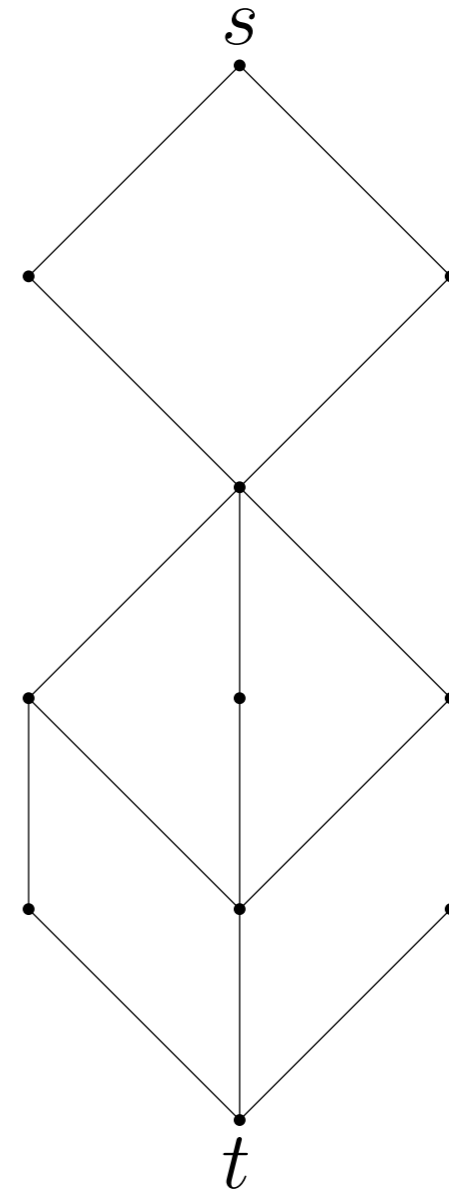
# Aim — Design a system to Track Paths

- Find a **method** to identify the path traversed by entities in a network
- Using minimal resources



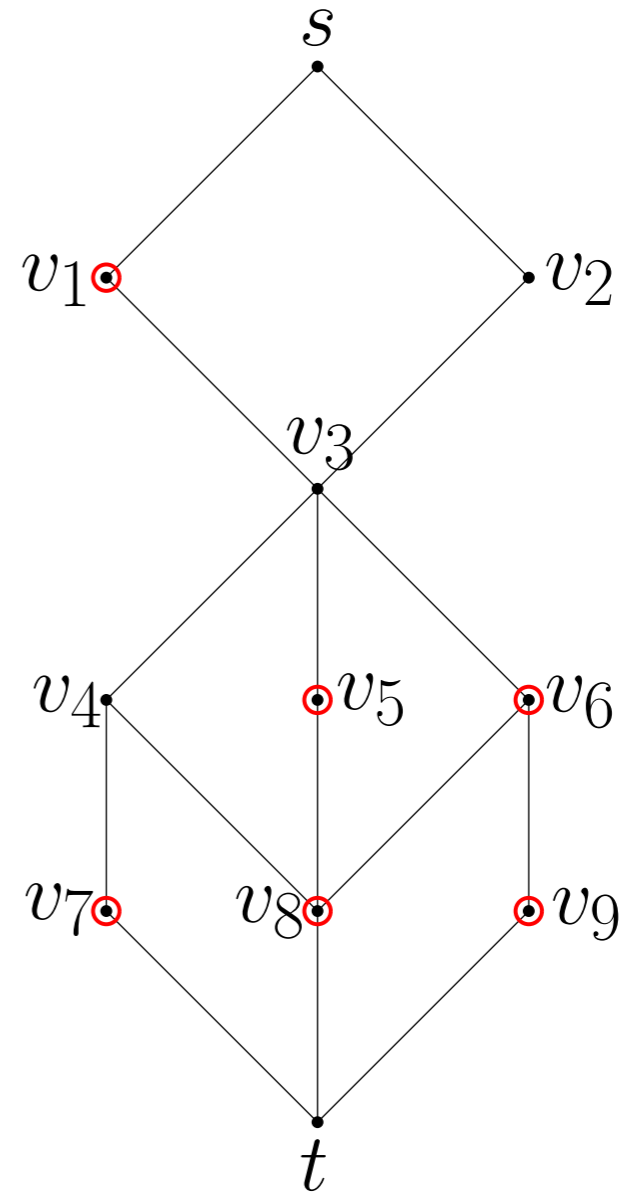
# Problem Definition

- $G=(V,E)$  be the map of secure facility



# Problem Definition

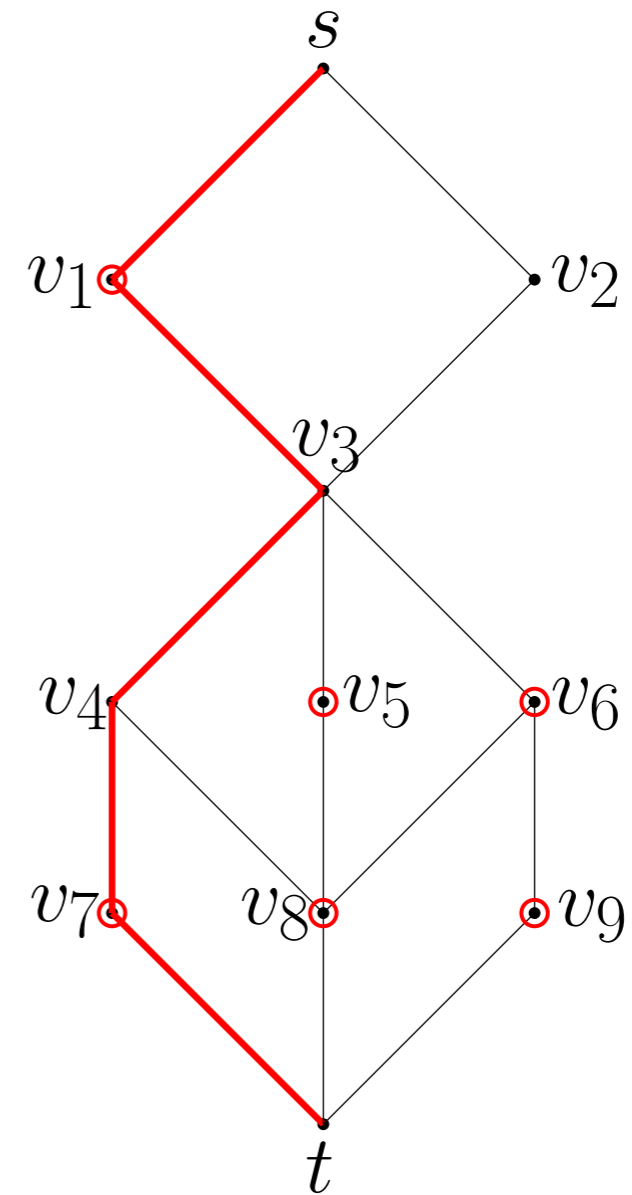
- $G=(V,E)$  be the map of secure facility
- *Objective:* Place trackers in order to track all s-t paths





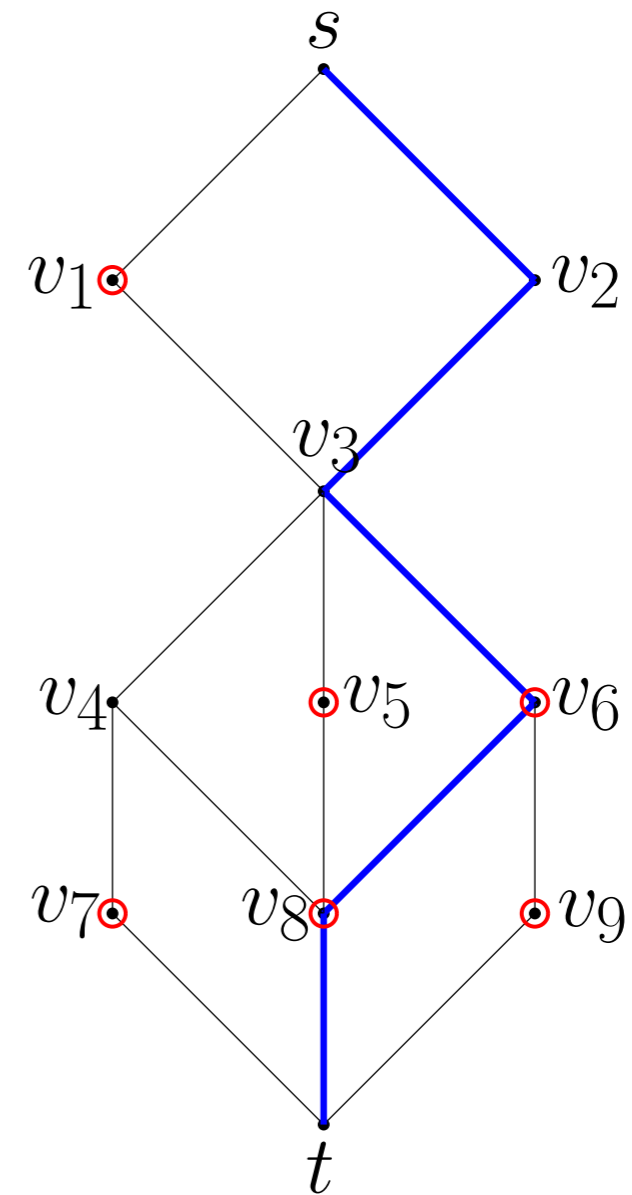
# Problem Definition

- $G=(V,E)$  be the map of secure facility
- *Objective:* Place trackers in order to track all s-t paths
- *Condition:* For any 2 s-t paths  $P_1, P_2$ : trackers in  $P_1$  are **different** from trackers in  $P_2$



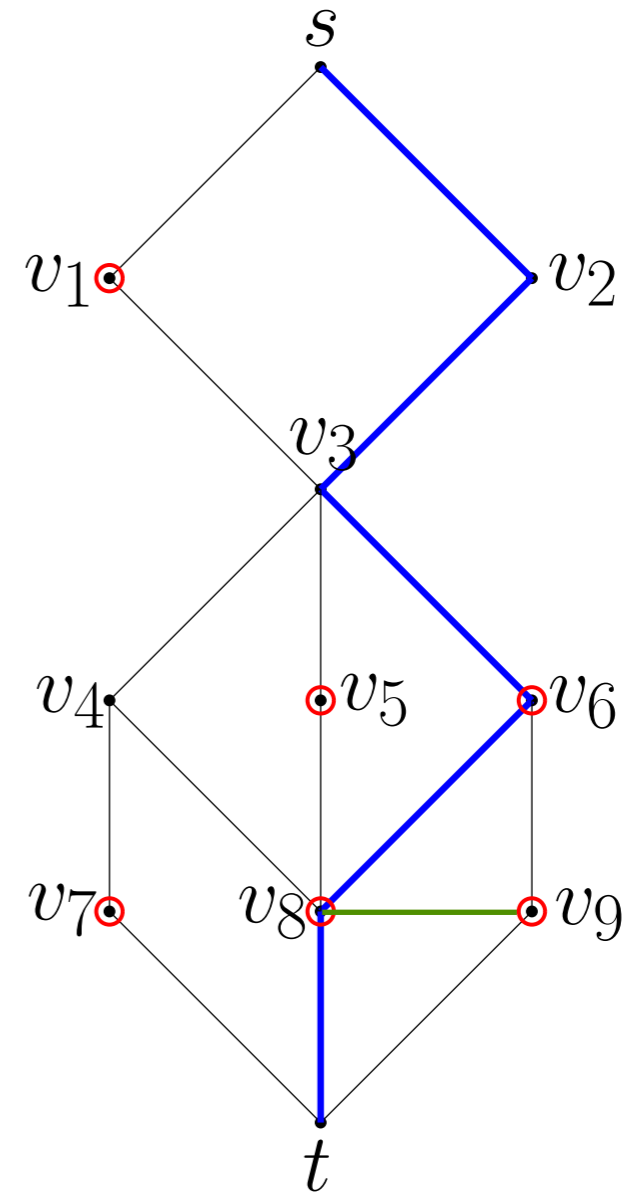
# Problem Definition

- $G=(V,E)$  be the map of secure facility
- *Objective:* Place trackers in order to track all s-t paths
- *Condition:* For any 2 s-t paths  $P_1, P_2$ : trackers in  $P_1$  are **different** from trackers in  $P_2$



# Problem Definition

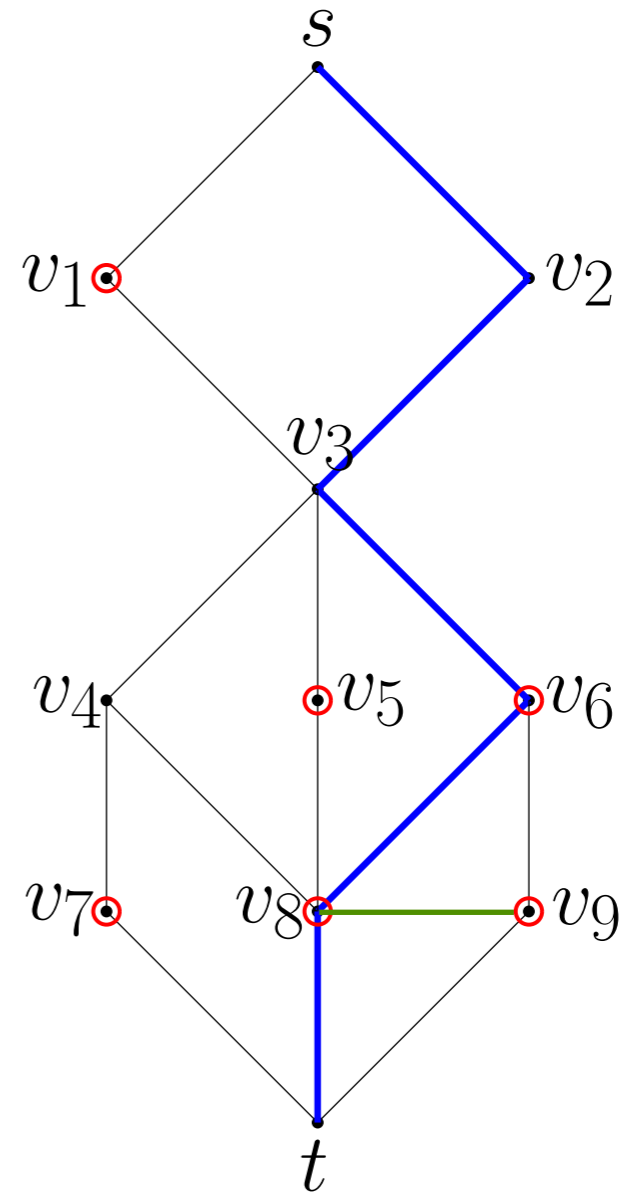
- $G=(V,E)$  be the map of secure facility
- *Objective:* Place trackers in order to track all s-t paths
- *Condition:* For any 2 s-t paths  $P_1, P_2$ : trackers in  $P_1$  are **different** from trackers in  $P_2$




# Problem Definition

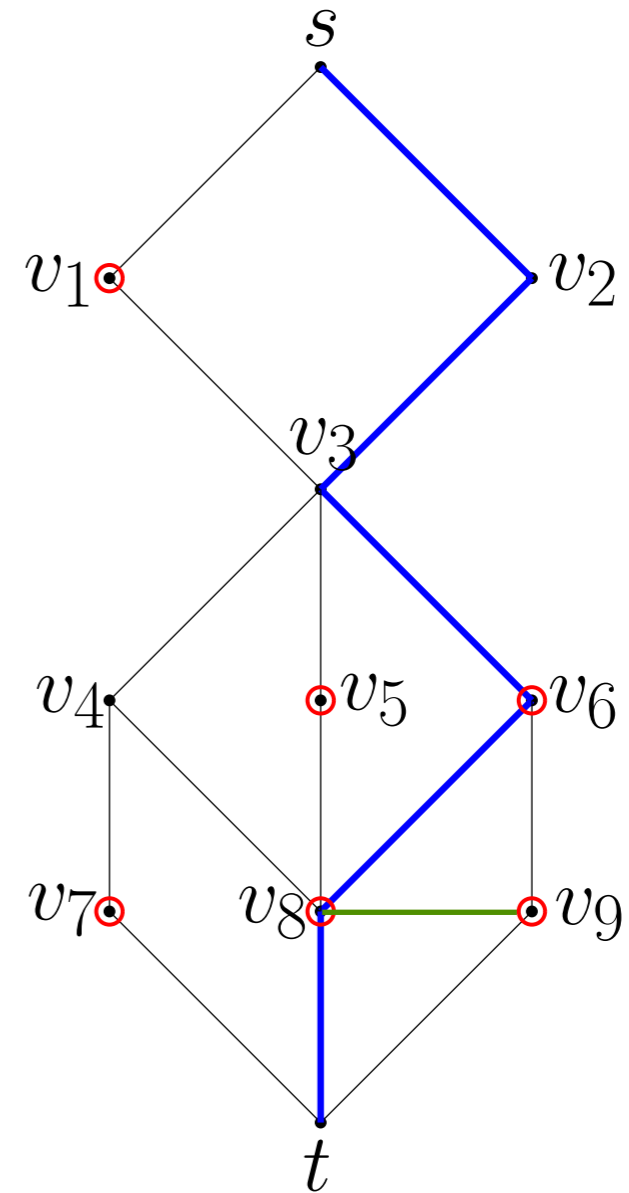
- $G=(V,E)$  be the map of secure facility
- *Objective:* Place trackers in order to track all s-t paths
- *Condition:* For any 2 s-t paths  $P_1, P_2$ : trackers in  $P_1$  are different from trackers in  $P_2$

sequence



# Problem Definition

- $G=(V,E)$  be the map of secure facility
- *Objective:* Place trackers in order to track all s-t paths
- *Condition:* For any 2 s-t paths  $P_1, P_2$ : trackers in  $P_1$  are **different** from trackers in  $P_2$   

- *Problem:* Find a minimum cardinality Tracking Set  
*Decision Version:* Decide if there exists a Tracking Set of size at most  $k$



**Is Tracking Paths problem hard to solve ?**

# Is Tracking Paths problem hard to solve ?

▶ NP-hard

# Is Tracking Paths problem hard to solve ?

▶ NP-hard





# Many doors to success



# Many doors to success



# Many doors to success

Parameterized  
Analysis



# Many doors to success

Parameterized  
Analysis

Approximation



# Many doors to success

Parameterized  
Analysis

Approximation

Restricted  
Versions



# Previous Work

Parameterized  
Analysis



# Previous Work

Find associated parameters for which problem can be solved in polynomial time\*

Parameterized  
Analysis



# Previous Work

Find associated parameters for which problem can be solved in polynomial time\*

Parameterized  
Analysis



☀ *Tracking Shortest Paths:* FPT: exponential kernel (CALDAM'18 with Aritra Banik)



# Previous Work

Find associated parameters for which problem can be solved in polynomial time\*

Parameterized  
Analysis

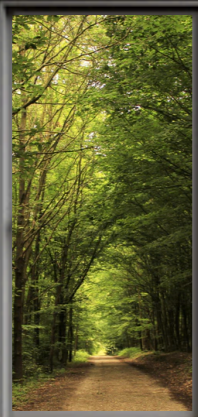


- ☀ *Tracking Shortest Paths*: FPT: exponential kernel (CALDAM'18 with Aritra Banik)
- ☀ FPT:  $\text{par} = \text{max. no. of vertices equidistant from source}$  (SIROCCO'19 Biló et al.)

# Previous Work

Find associated parameters for which problem can be solved in polynomial time\*

Parameterized  
Analysis



- ☀ **Tracking Shortest Paths:** FPT: exponential kernel (*CALDAM'18 with Aritra Banik*)
- ☀ FPT: par = max. no. of vertices equidistant from source (*SIROCCO'19 Biló et al.*)
- ☀ **Tracking Paths:** FPT - polynomial kernel (*LATIN'18 with Aritra Banik, Daniel Lokshantov, Venkatesh Raman, Saket Saurabh*)

# Previous Work

Find associated parameters for which problem can be solved in polynomial time\*

Parameterized  
Analysis



- ✿ **Tracking Shortest Paths:** FPT: exponential kernel (*CALDAM'18 with Aritra Banik*)
- ✿ FPT: par = max. no. of vertices equidistant from source (*SIROCCO'19 Biló et al.*)
- ✿ **Tracking Paths:** FPT - polynomial kernel (*LATIN'18 with Aritra Banik, Daniel Lokshantov, Venkatesh Raman, Saket Saurabh*)
- ✿ Quadratic Kernel for general, linear kernel for planar graphs,  
Tracking Set of size  $(n-k)$  is  $W[1]$ -hard (*Under Review, with Venkatesh Raman*)

# Previous Work

Find associated parameters for which problem can be solved in polynomial time\*

Parameterized  
Analysis



- ✿ **Tracking Shortest Paths:** FPT: exponential kernel (*CALDAM'18 with Aritra Banik*)
- ✿ FPT: par = max. no. of vertices equidistant from source (*SIROCCO'19 Biló et al.*)
- ✿ **Tracking Paths:** FPT - polynomial kernel (*LATIN'18 with Aritra Banik, Daniel Lokshantov, Venkatesh Raman, Saket Saurabh*)
- ✿ Quadratic Kernel for general, linear kernel for planar graphs,  
Tracking Set of size  $(n-k)$  is  $W[1]$ -hard (*Under Review, with Venkatesh Raman*)

# Previous Work



Approximation

# Previous Work


Find solutions that are close to an optimum solution in polynomial time

Approximation



# Previous Work

Find solutions that are close to an optimum solution in polynomial time



Approximation

✻ *Tracking Shortest Paths:* APX-hard, 2-approx for planar graphs (CIAC'17 Banik et al.)

# Previous Work

Find solutions that are close to an optimum solution in polynomial time

Approximation

☀️ *Tracking Shortest Paths:* APX-hard, 2-approx for planar graphs (CIAC'17 Banik et al.)

First  
Work



# Previous Work

Find solutions that are close to an optimum solution in polynomial time

Approximation



☀ *Tracking Shortest Paths*: APX-hard, 2-approx for planar graphs (CIAC'17 Banik et al.)

☀  $\tilde{O}(\sqrt{n})$ -approx (SIROCCO'19 with Bilo et al.)



First  
Work

# Previous Work

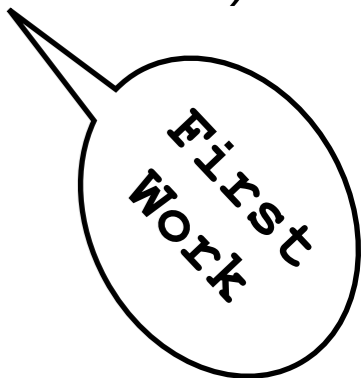
Find solutions that are close to an optimum solution in polynomial time

Approximation



☀ *Tracking Shortest Paths*: APX-hard, 2-approx for planar graphs (CIAC'17 Banik et al.)

☀  $\tilde{O}(\sqrt{n})$  - approx (SIROCCO'19 with Bilo et al.) multiple s-t



# Previous Work

Find solutions that are close to an optimum solution in polynomial time

Approximation



- ☀ *Tracking Shortest Paths*: APX-hard, 2-approx for planar graphs (CIAC'17 Banik et al.)
- ☀  $\tilde{O}(\sqrt{n})$ -approx (SIROCCO'19 with Bilo et al.) multiple s-t
- ☀ *Tracking Paths*: 4-approx for planar graphs (ISAAC'19 Eppstein et al.)



First  
Work

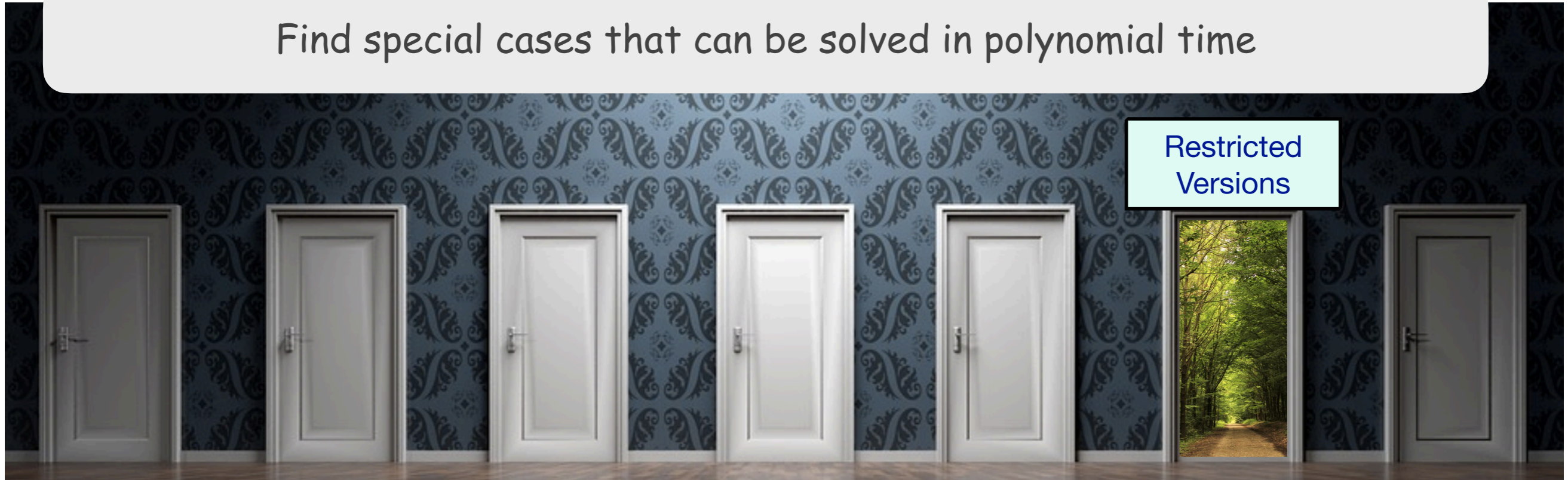
# Previous Work



# Previous Work

Find special cases that can be solved in polynomial time

Restricted  
Versions



# Previous Work

Find special cases that can be solved in polynomial time

Restricted  
Versions

✻ *Tracking Paths*: Linear time: bounded clique-width graphs (*ISAAC'19 Eppstein et al.*)

# This Paper

New



# This Paper

New

Approximation





# This Paper

New

Approximation

Restricted  
Versions



# Assumptions

# Assumptions

- ▶ Unique Source "s" and Unique Destination "t"

# Assumptions

- ▶ Unique Source "s" and Unique Destination "t"
- ▶ Only simple s-t paths are considered (also no self loops, parallel edges)

# Assumptions

- ▶ Unique Source "s" and Unique Destination "t"
- ▶ Only simple s-t paths are considered (also no self loops, parallel edges)
- ▶ s-t path: path between source and destination

# Assumptions

- ▶ Unique Source "s" and Unique Destination "t"
- ▶ Only simple s-t paths are considered (also no self loops, parallel edges)
- ▶ s-t path: path between source and destination
- ▶ TS: Tracking Set (solution)

# Assumptions

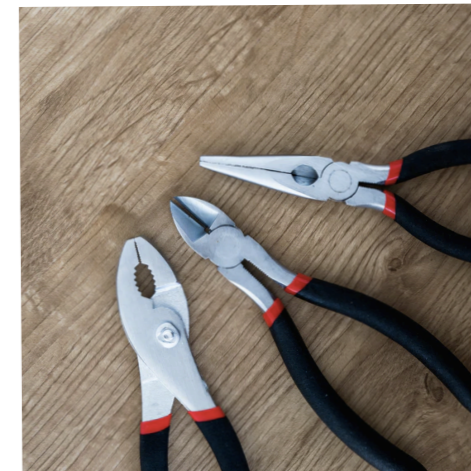
- ▶ Unique Source "s" and Unique Destination "t"
- ▶ Only simple s-t paths are considered (also no self loops, parallel edges)
- ▶ s-t path: path between source and destination
- ▶ TS: Tracking Set (solution)
- ▶ FVS: Feedback Vertex Set (set of vertices whose removal makes the graph acyclic)

# Assumptions

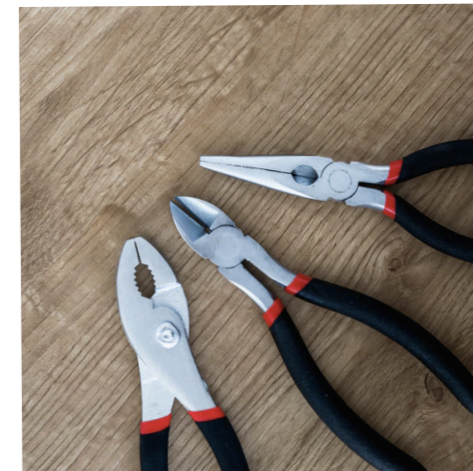
- ▶ Unique Source "s" and Unique Destination "t"
- ▶ Only simple s-t paths are considered (also no self loops, parallel edges)
- ▶ s-t path: path between source and destination
- ▶ TS: Tracking Set (solution)
- ▶ FVS: Feedback Vertex Set (set of vertices whose removal makes the graph acyclic)
- ▶ FES: Feedback Edge Set (set of edges whose removal makes the graph acyclic)



# Characterization of Tracking Set

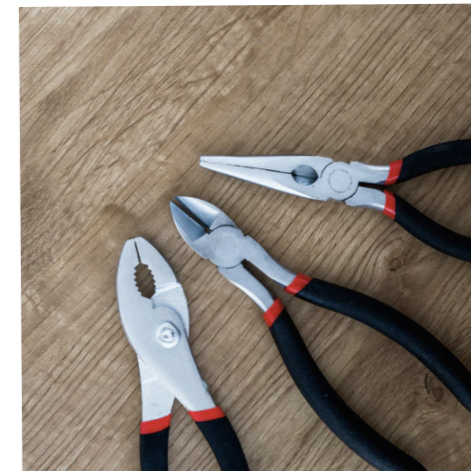


# Characterization of Tracking Set



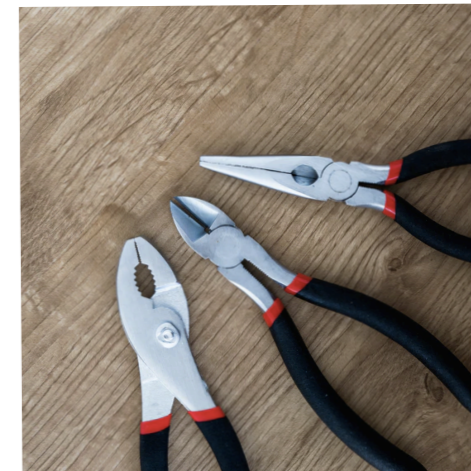
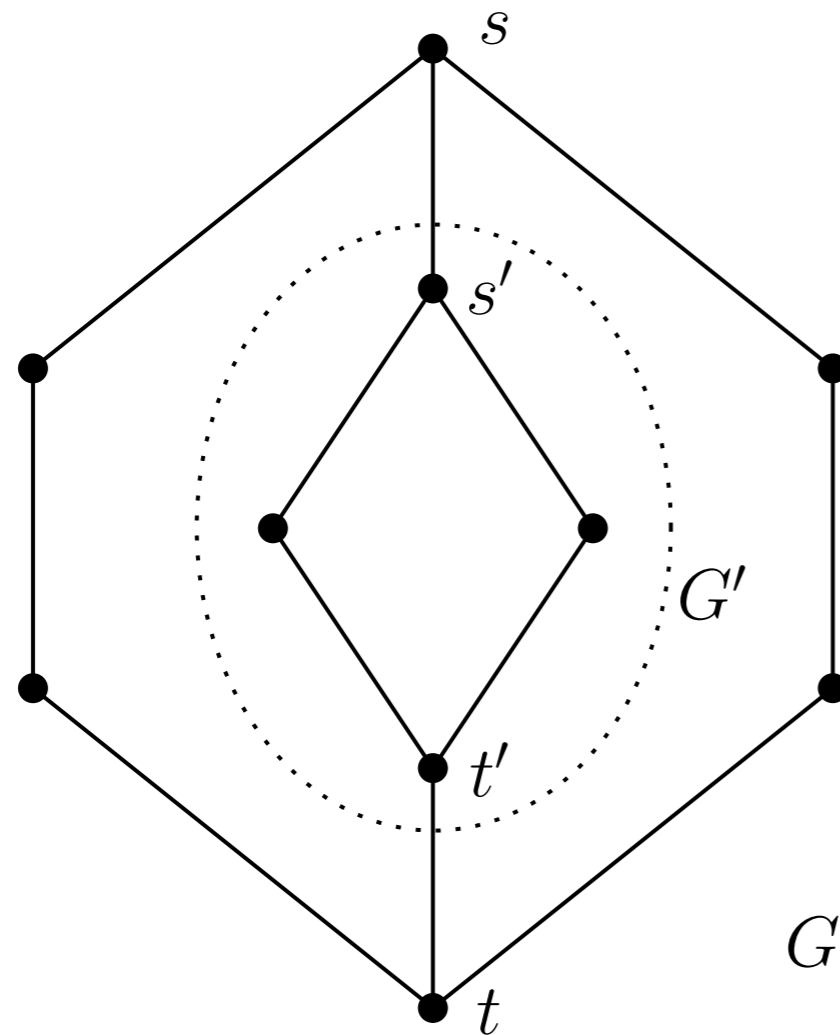
# Characterization of Tracking Set

*Local source - destination*



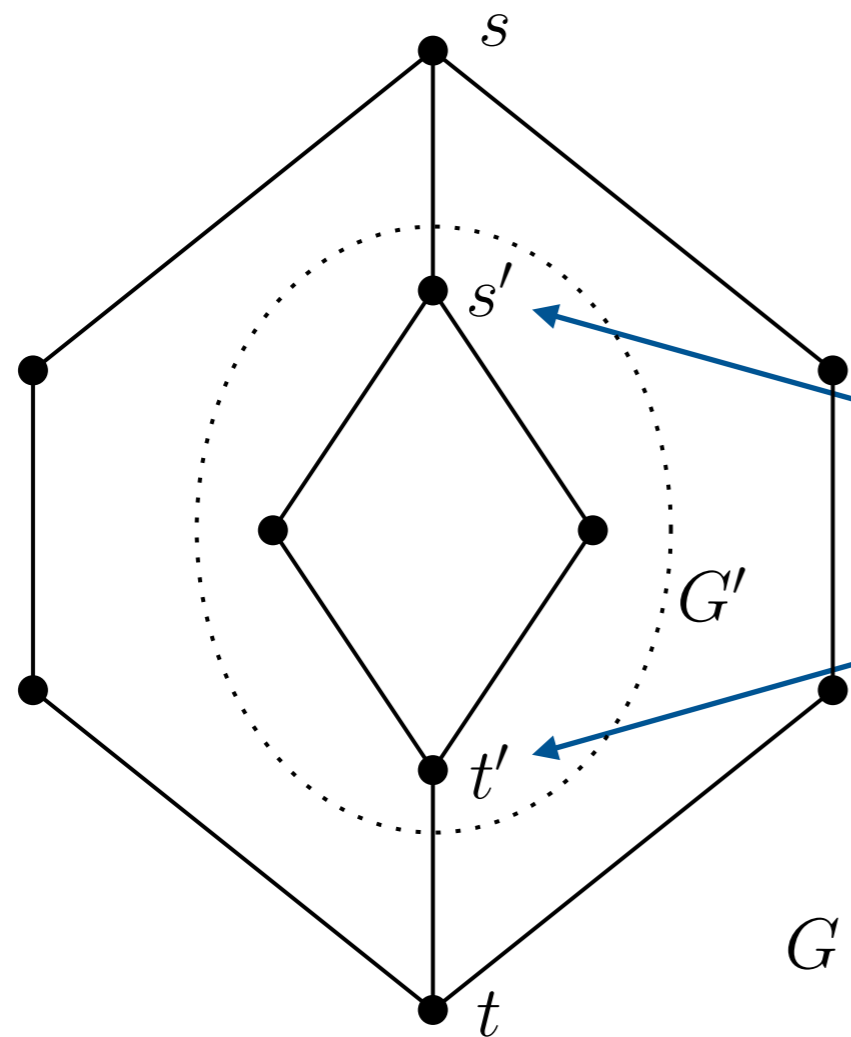
# Characterization of Tracking Set

Local source - destination

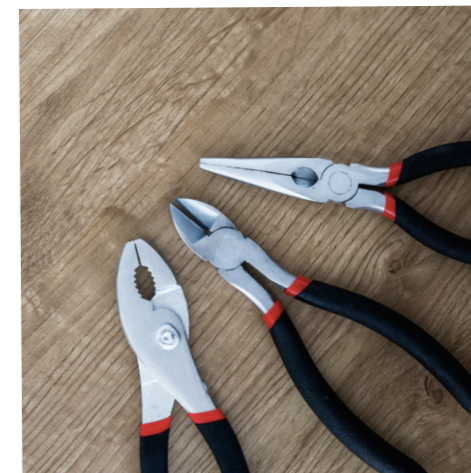


# Characterization of Tracking Set

## Local source - destination

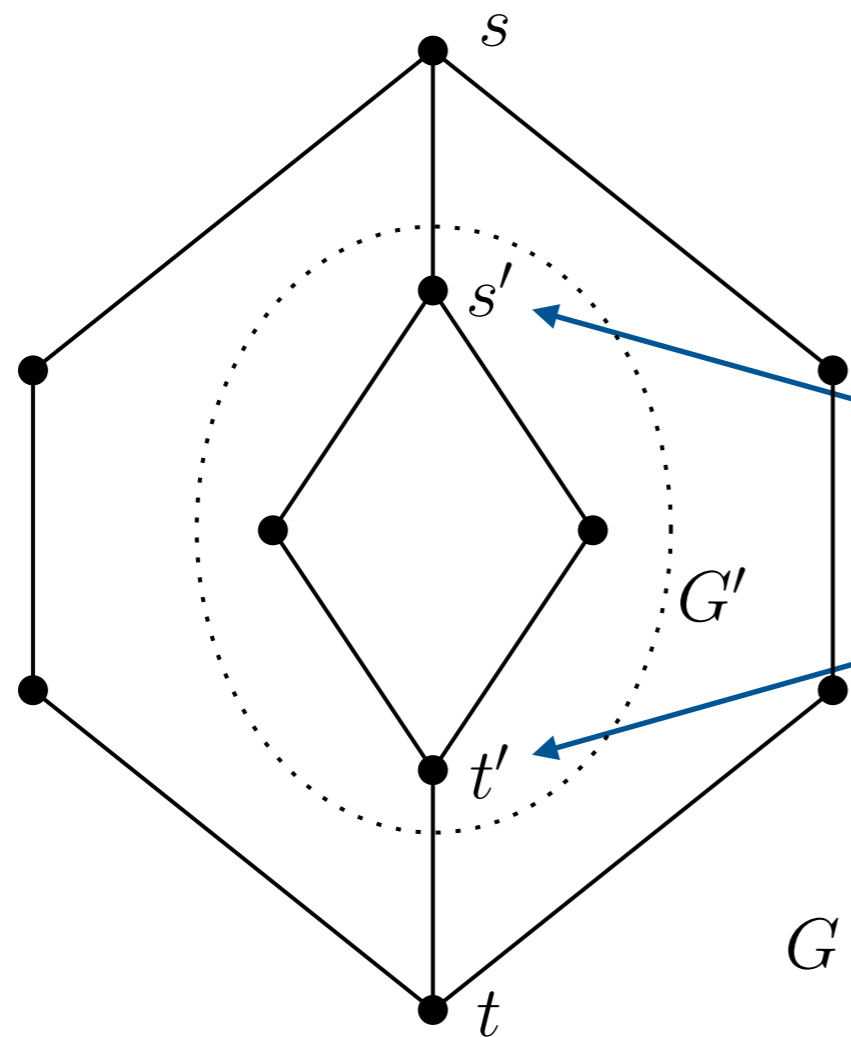


Local source - destination pair for  $G'$



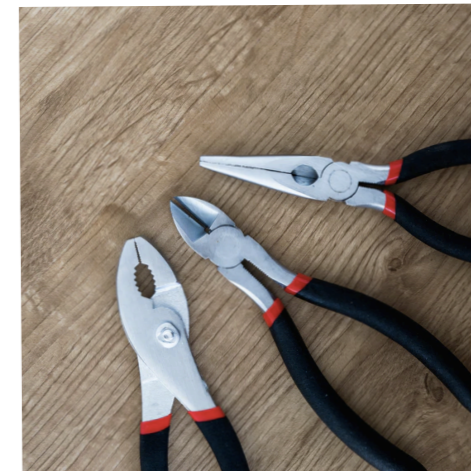
# Characterization of Tracking Set

## Local source - destination



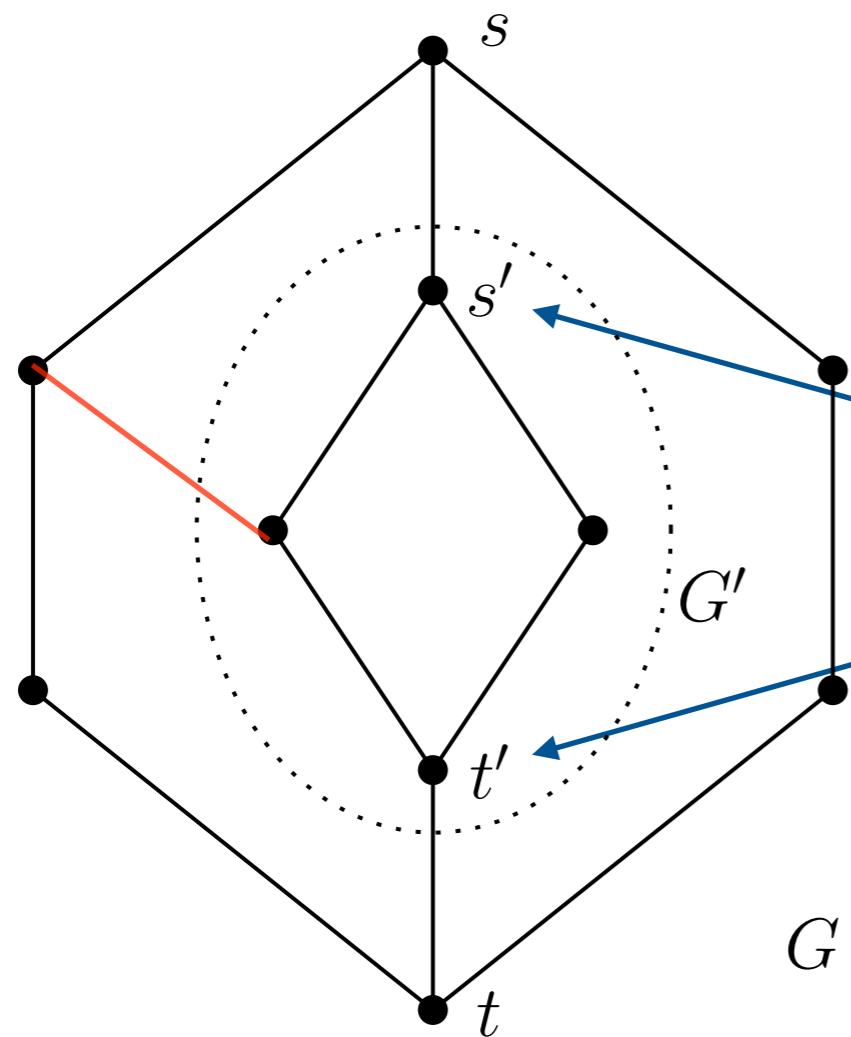
Local source - destination pair for  $G'$

$s', t'$  can be considered as source, destination for  $G'$



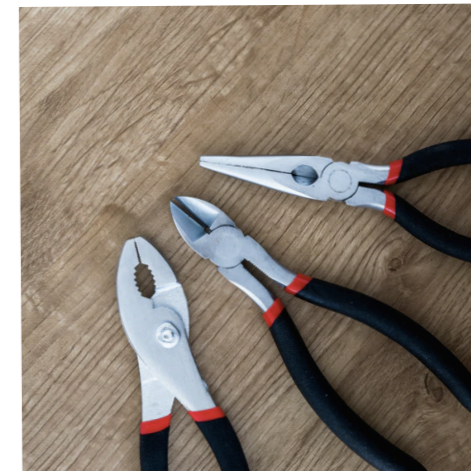
# Characterization of Tracking Set

## Local source - destination



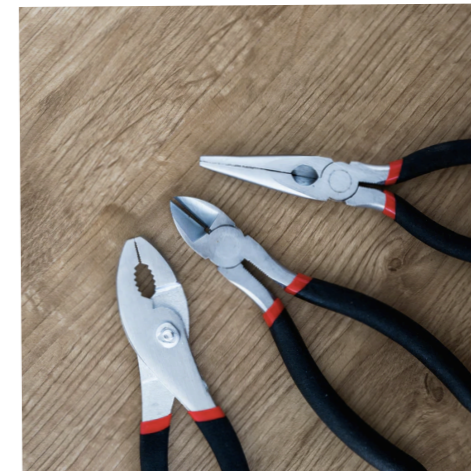
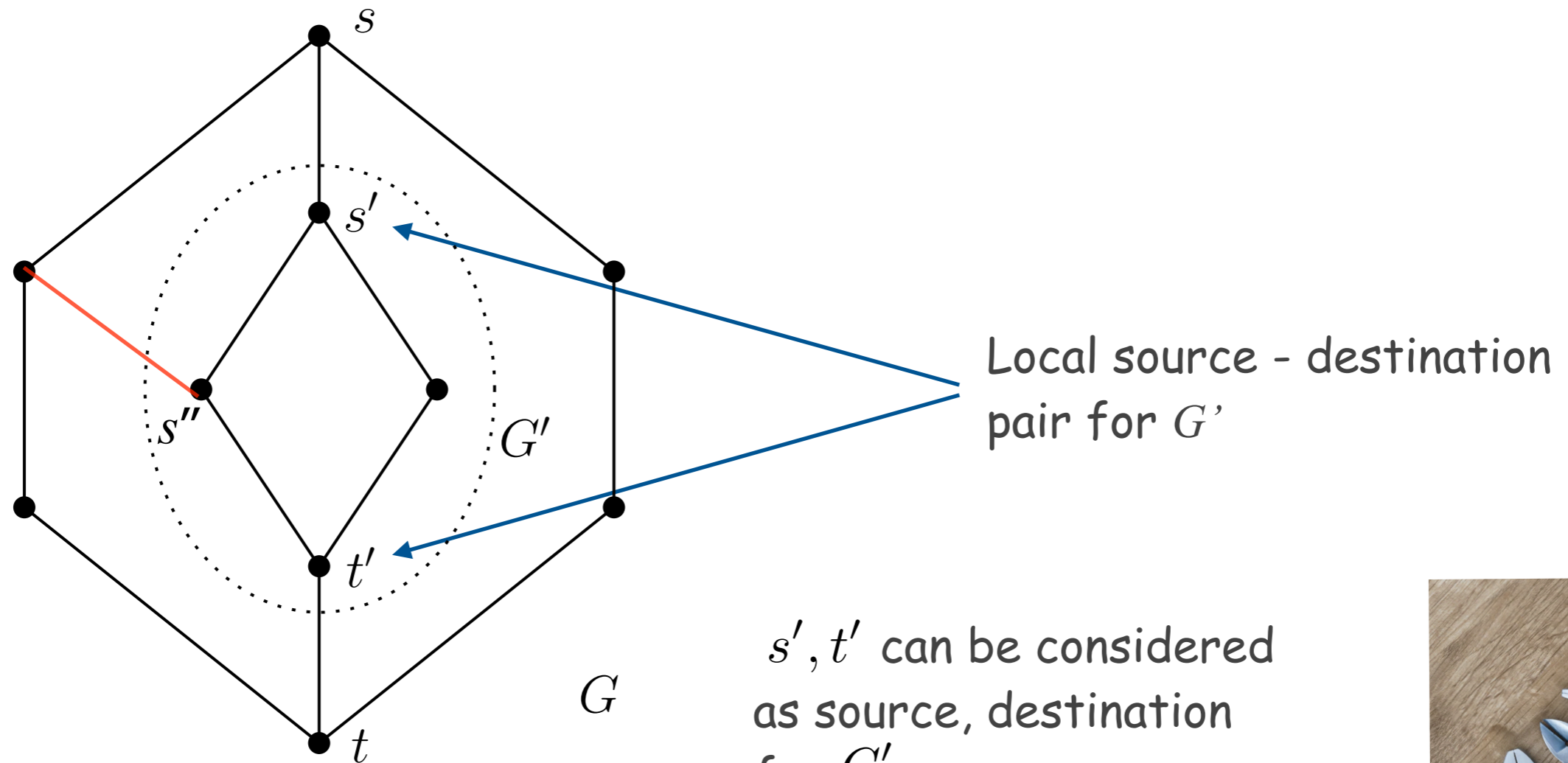
Local source - destination pair for  $G'$

$s', t'$  can be considered as source, destination for  $G'$



# Characterization of Tracking Set

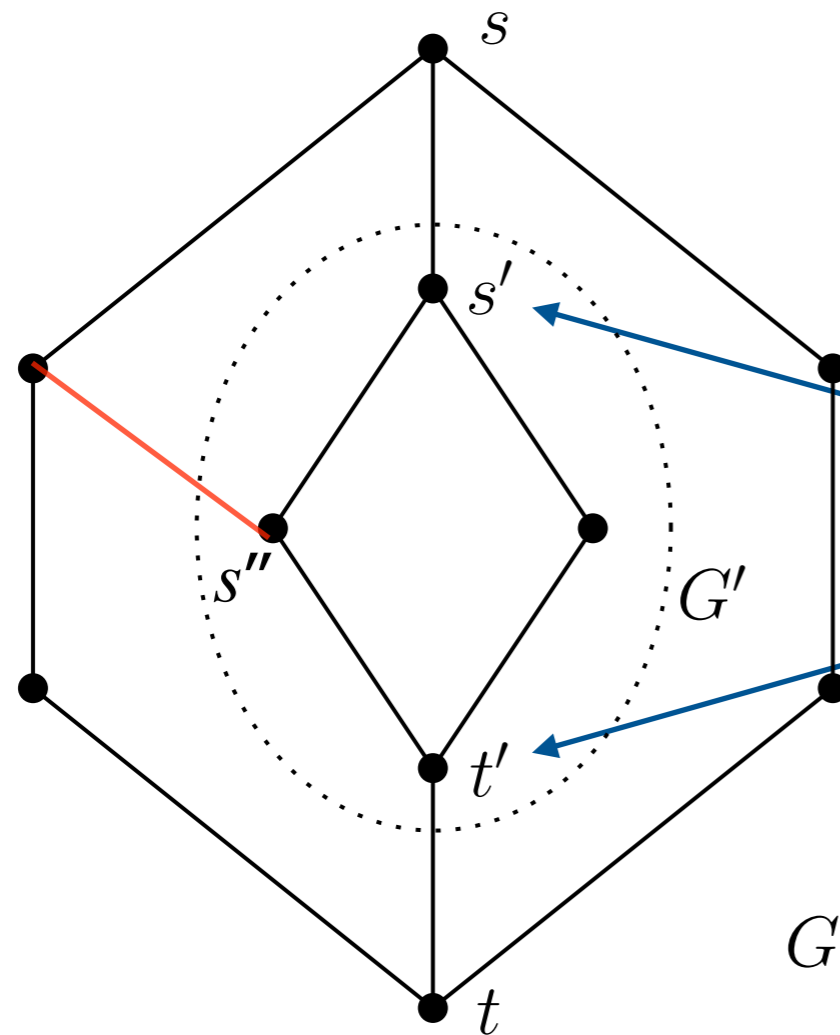
## Local source - destination





# Characterization of Tracking Set

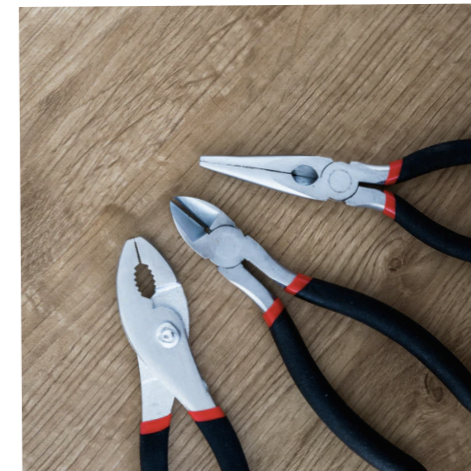
## Local source - destination



there can be more than one pair in a subgraph

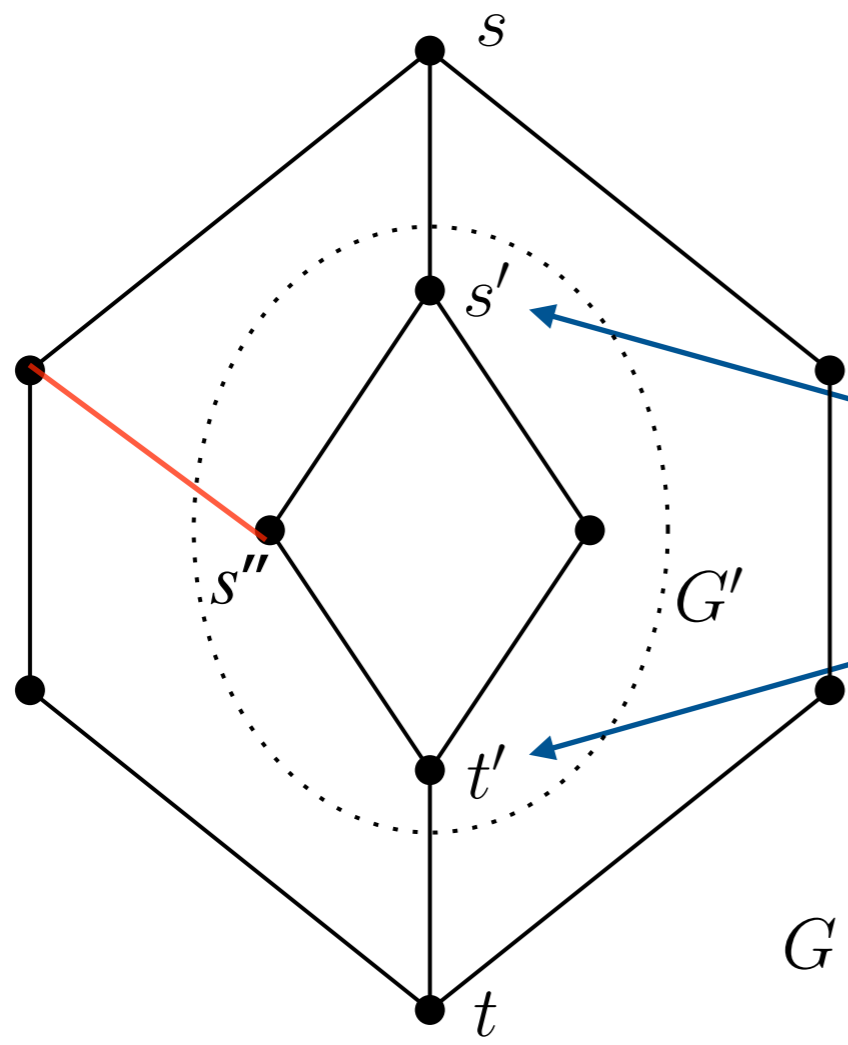
Local source - destination pair for  $G'$

$s', t'$  can be considered as source, destination for  $G'$



# Characterization of Tracking Set

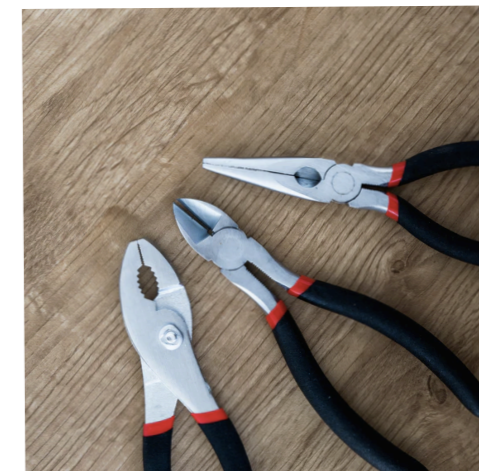
## Local source - destination



there can be more than one pair in a subgraph  $s'', t'$

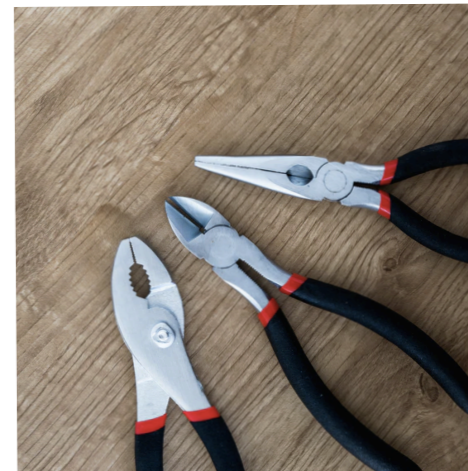
Local source - destination pair for  $G'$

$s', t'$  can be considered as source, destination for  $G'$



# Characterization of Tracking Set

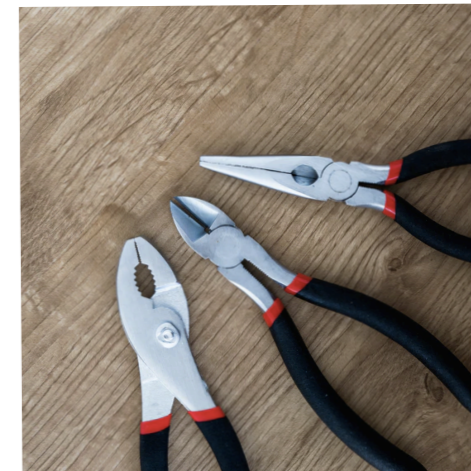
*Local source - destination*



# Characterization of Tracking Set

*Local source - destination*

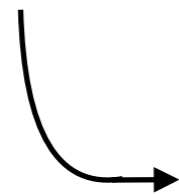
★ Each subgraph consisting of at least one edge



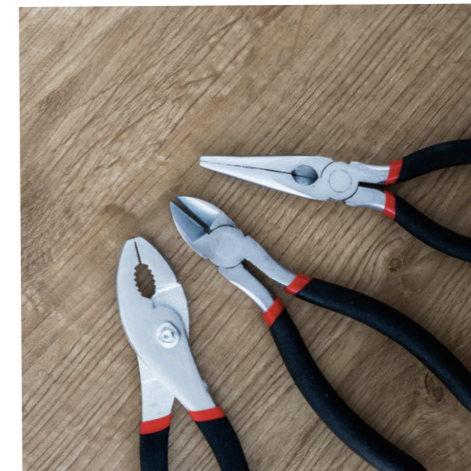
# Characterization of Tracking Set

## Local source - destination

★ Each subgraph consisting of at least one edge



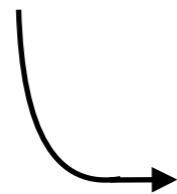
Contains at least one local s-t



# Characterization of Tracking Set

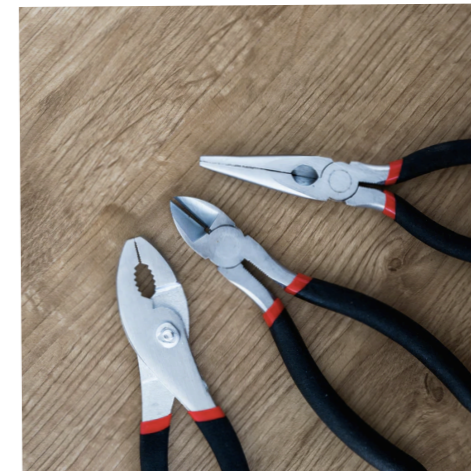
## Local source - destination

★ Each subgraph consisting of at least one edge



Contains at least one local s-t

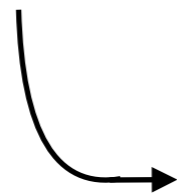
★ *Tracking Paths* is hereditary



# Characterization of Tracking Set

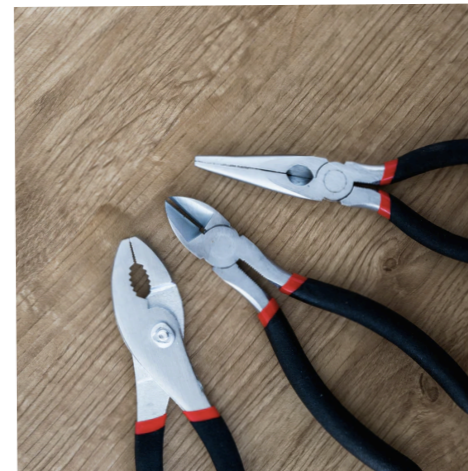
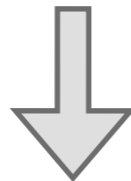
## Local source - destination

★ Each subgraph consisting of at least one edge



Contains at least one local s-t

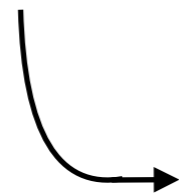
★ *Tracking Paths* is hereditary



# Characterization of Tracking Set

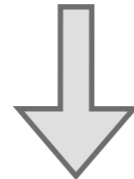
## Local source - destination

★ Each subgraph consisting of at least one edge

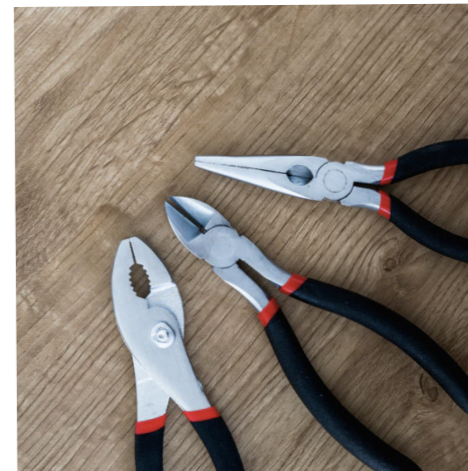


Contains at least one local s-t

★ *Tracking Paths* is hereditary

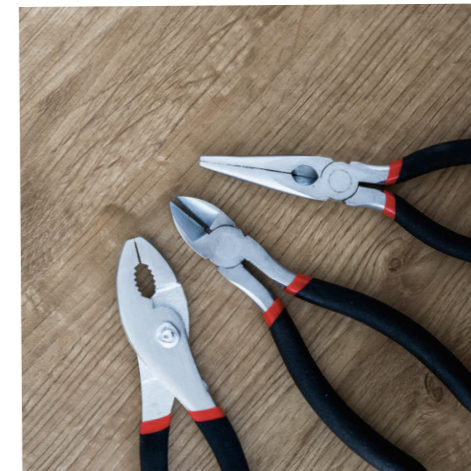


If all paths between a local s-t in a subgraph are not tracked, then all s-t paths in a graph are not tracked



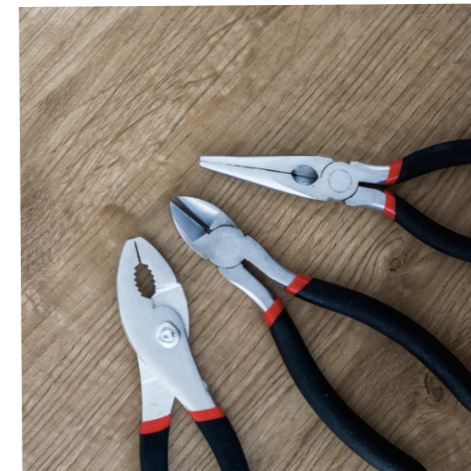


# Characterization of Tracking Set



# Characterization of Tracking Set

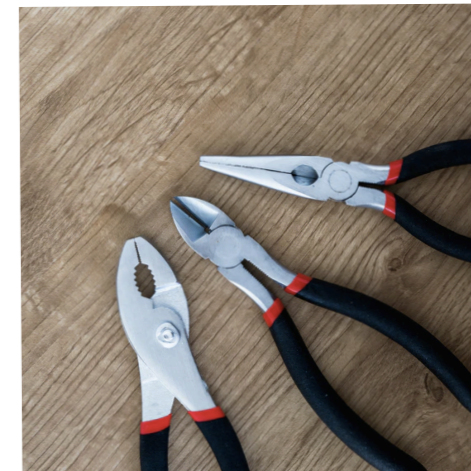
*Tracking Set Condition*



# Characterization of Tracking Set

## Tracking Set Condition

Graph  $G = (V, E)$

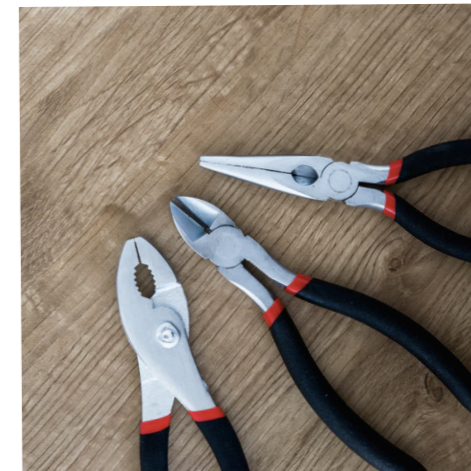


# Characterization of Tracking Set

## Tracking Set Condition

Graph  $G = (V, E)$

$T \subseteq V$



# Characterization of Tracking Set

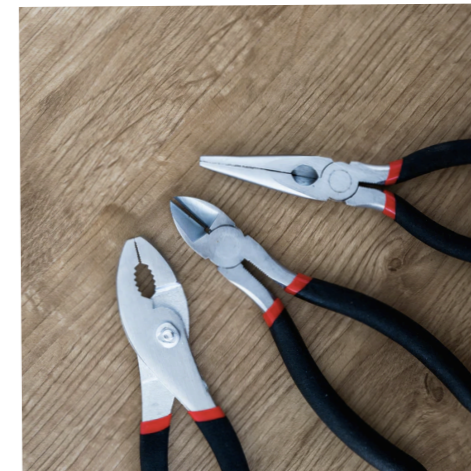
## Tracking Set Condition

Graph  $G = (V, E)$

$T \subseteq V$

$\nexists u, v \in V$

in  $G \setminus T \cup \{u, v\}$



# Characterization of Tracking Set

## Tracking Set Condition

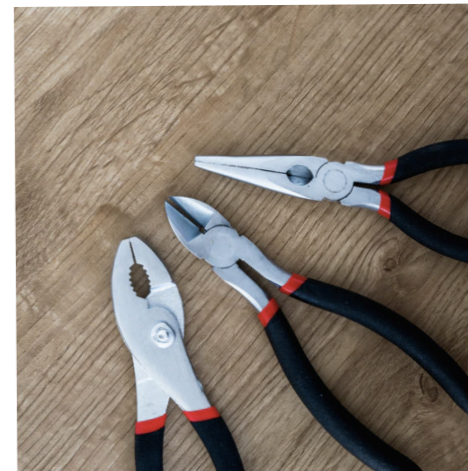
Graph  $G = (V, E)$

$T \subseteq V$

$\nexists u, v \in V$

in  $G \setminus T \cup \{u, v\}$

$\exists$  two distinct paths  $P_1, P_2$  between  $u$  and  $v$



# Characterization of Tracking Set

## Tracking Set Condition

Graph  $G = (V, E)$

$T \subseteq V$

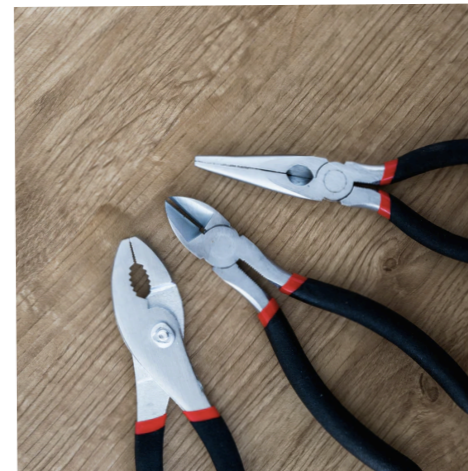
$\nexists u, v \in V$

in  $G \setminus T \cup \{u, v\}$

$\exists$  two distinct paths  $P_1, P_2$  between  $u$  and  $v$

where

$u$  and  $v$  are local source-destination pair



# Characterization of Tracking Set

## Tracking Set Condition

Graph  $G = (V, E)$

$T \subseteq V$

$\nexists u, v \in V$

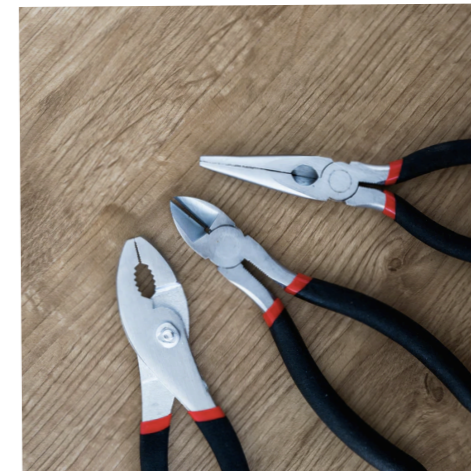
in  $G \setminus T \cup \{u, v\}$

$\exists$  two distinct paths  $P_1, P_2$  between  $u$  and  $v$

where

$u$  and  $v$  are local source-destination pair

$\Rightarrow T$  satisfies *tracking set condition*





# Characterization of Tracking Set

## Tracking Set Condition

Graph  $G = (V, E)$

$T \subseteq V$

$\nexists u, v \in V$

in  $G \setminus T \cup \{u, v\}$

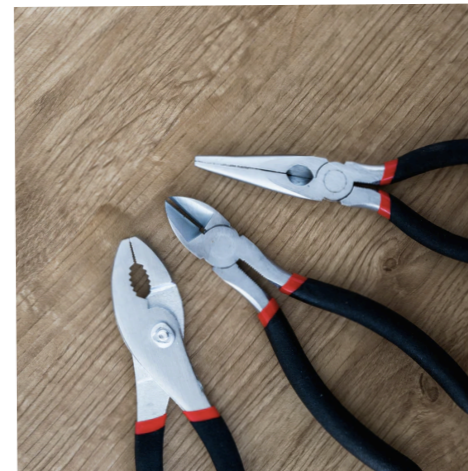
$\exists$  two distinct paths  $P_1, P_2$  between  $u$  and  $v$

where

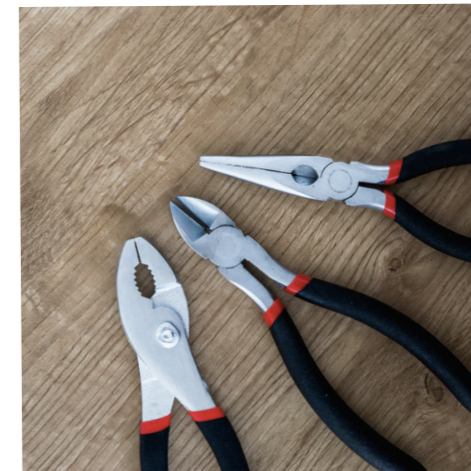
$u$  and  $v$  are local source-destination pair

$\Rightarrow T$  satisfies *tracking set condition*

Helps verify a tracking set in polynomial time

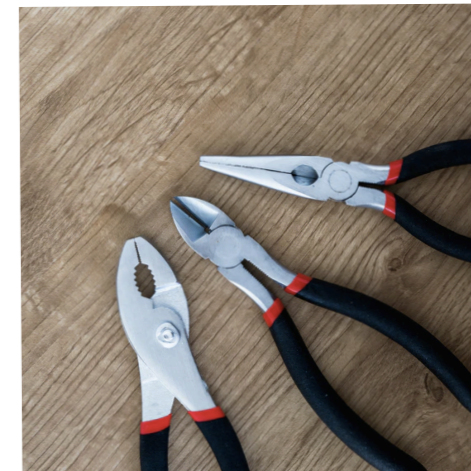
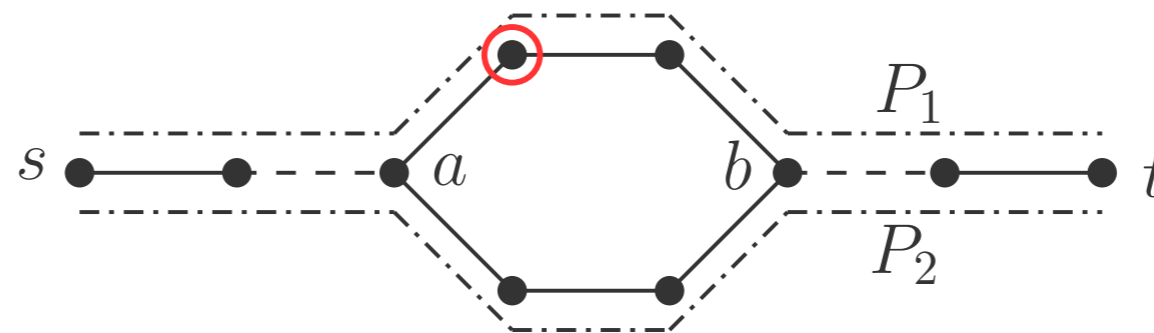


# Characterization of Tracking Set



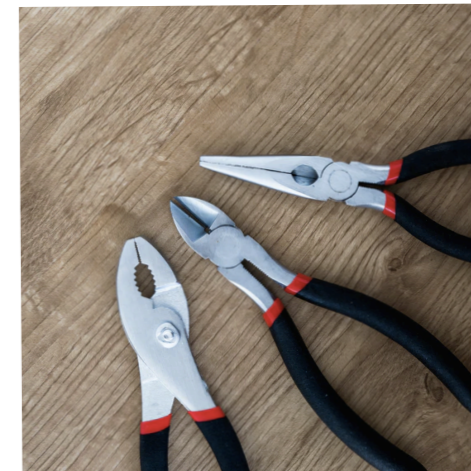
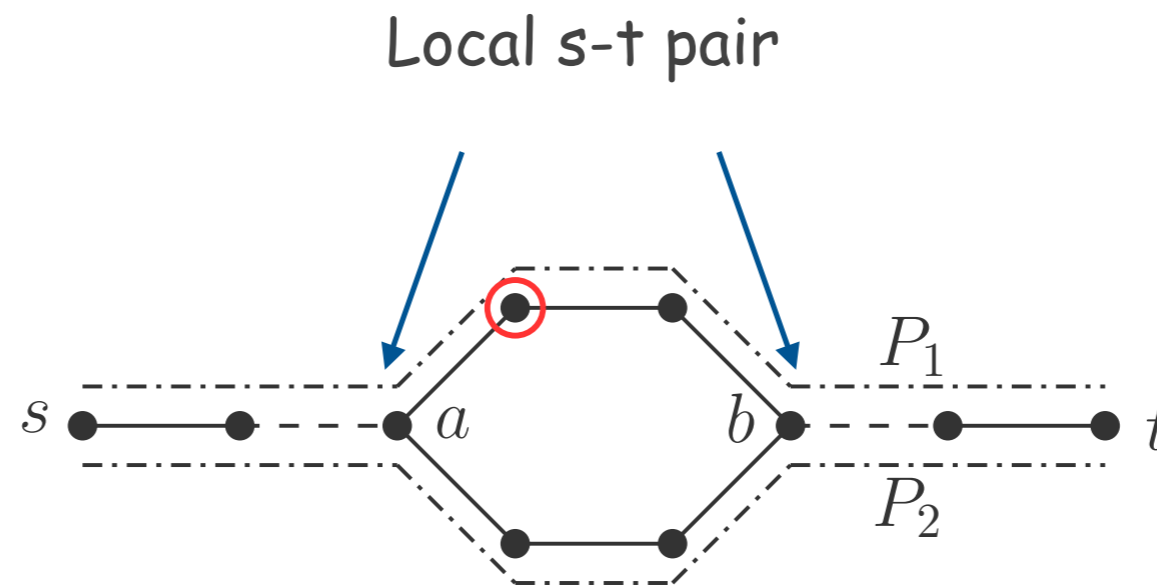
# Characterization of Tracking Set

Graph **satisfying**  
tracking  
set condition



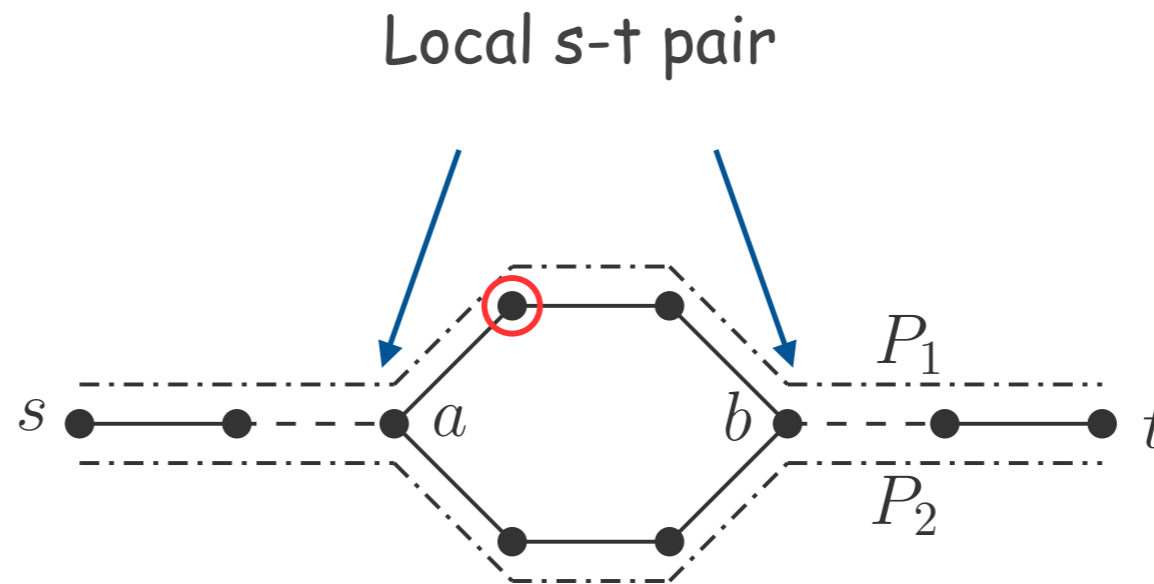
# Characterization of Tracking Set

Graph **satisfying**  
tracking  
set condition

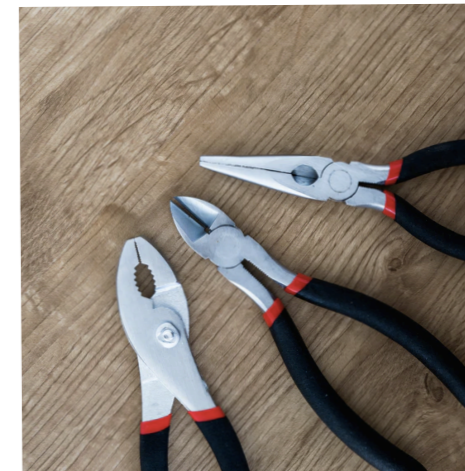
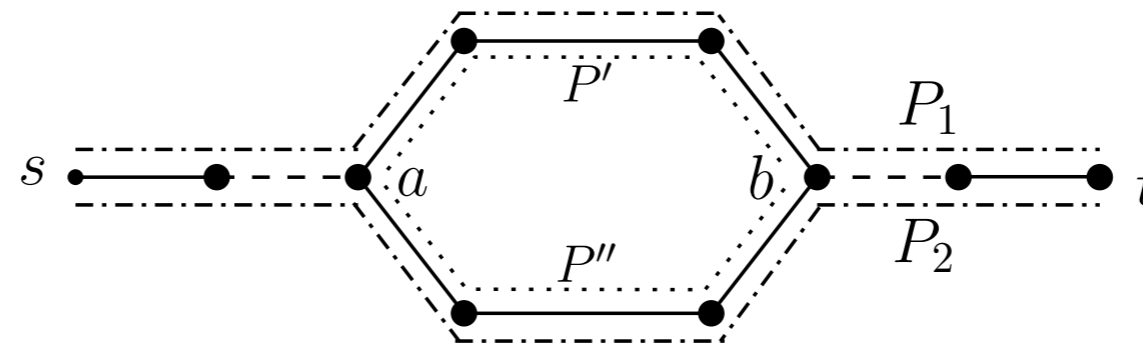


# Characterization of Tracking Set

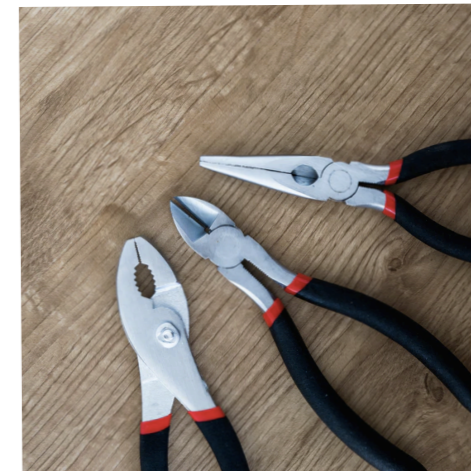
Graph **satisfying** tracking set condition



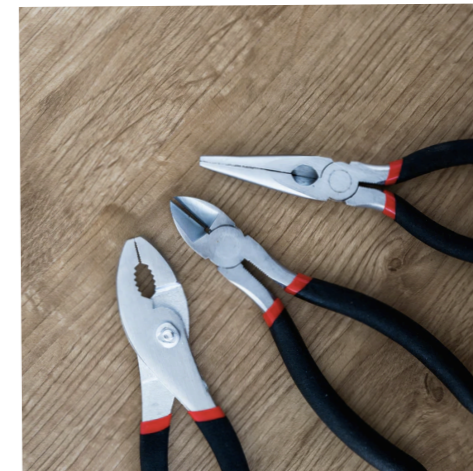
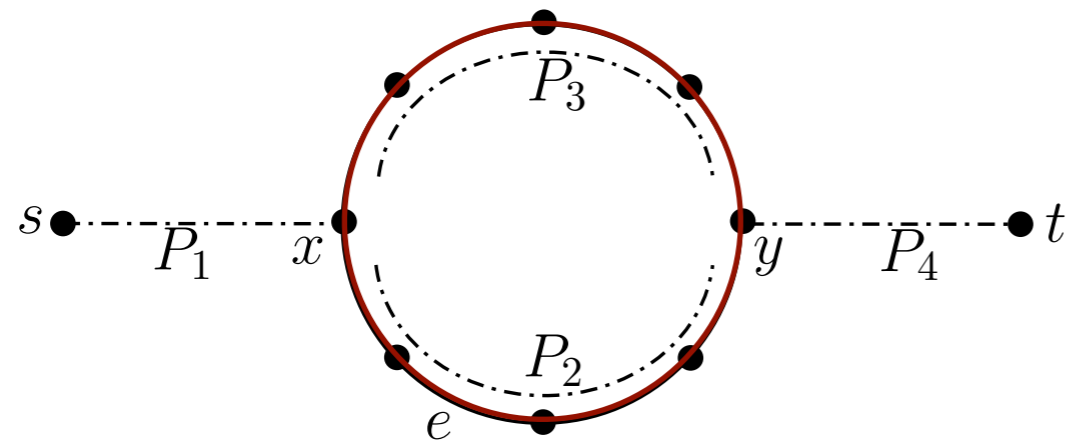
Graph **not satisfying** tracking set condition



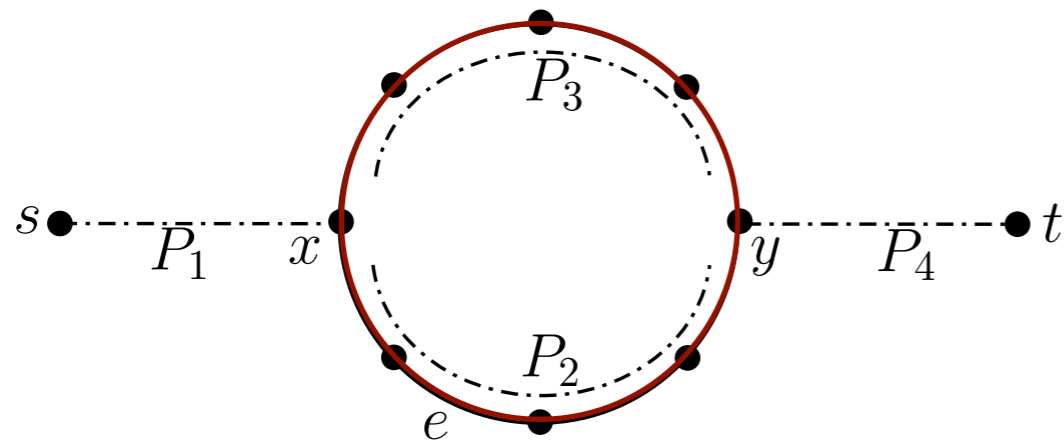
# The FVS connection



# The FVS connection

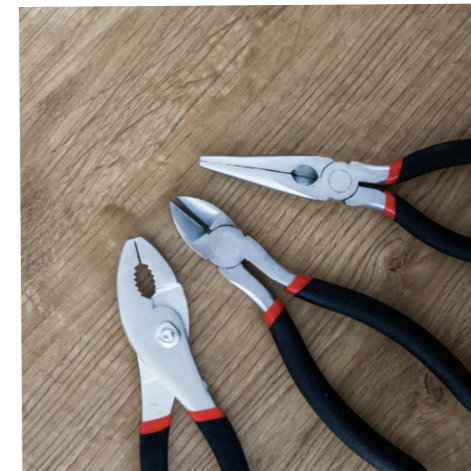


# The FVS connection



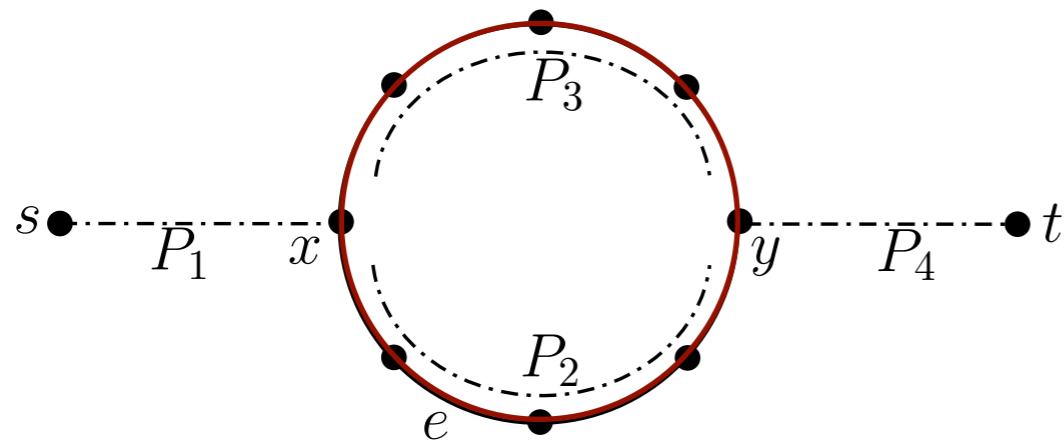
$$P' = P_1 \cdot P_2 \cdot P_4$$

$$P'' = P_1 \cdot P_3 \cdot P_4$$





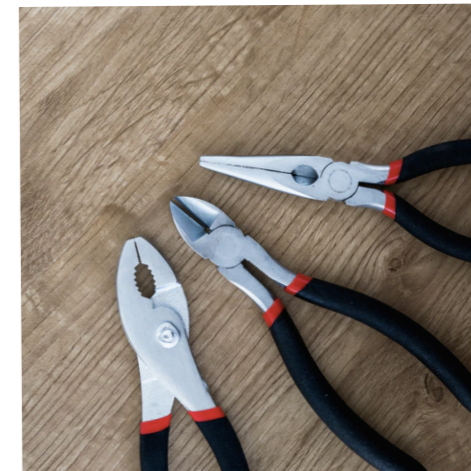
# The FVS connection



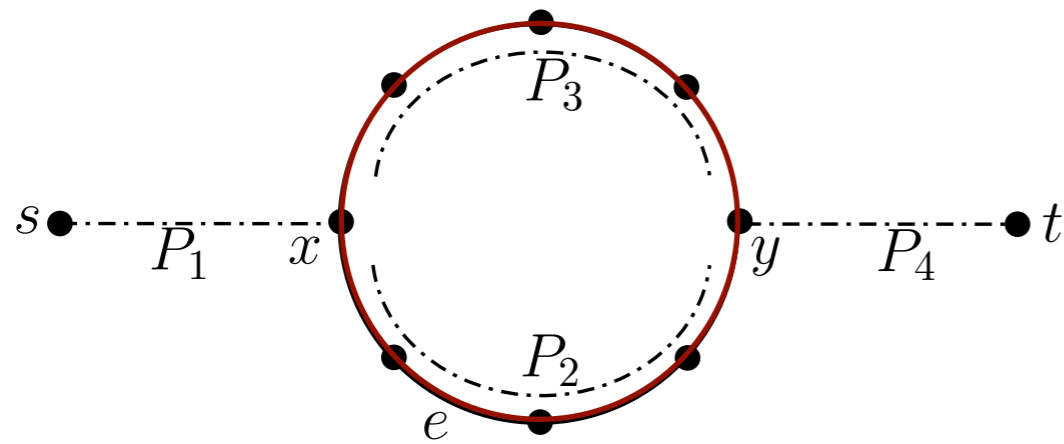
$$P' = P_1 \cdot P_2 \cdot P_4$$

$$P'' = P_1 \cdot P_3 \cdot P_4$$

► We need a tracker in the cycle



# The FVS connection

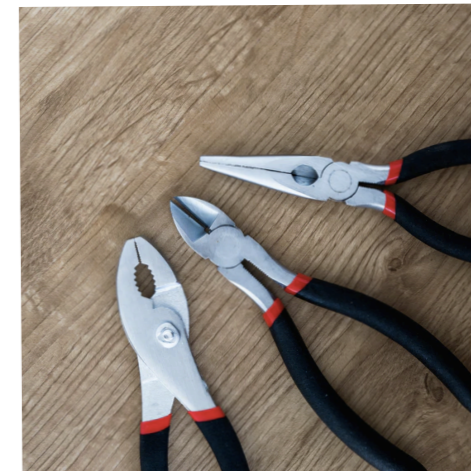


Each cycle must have a tracker

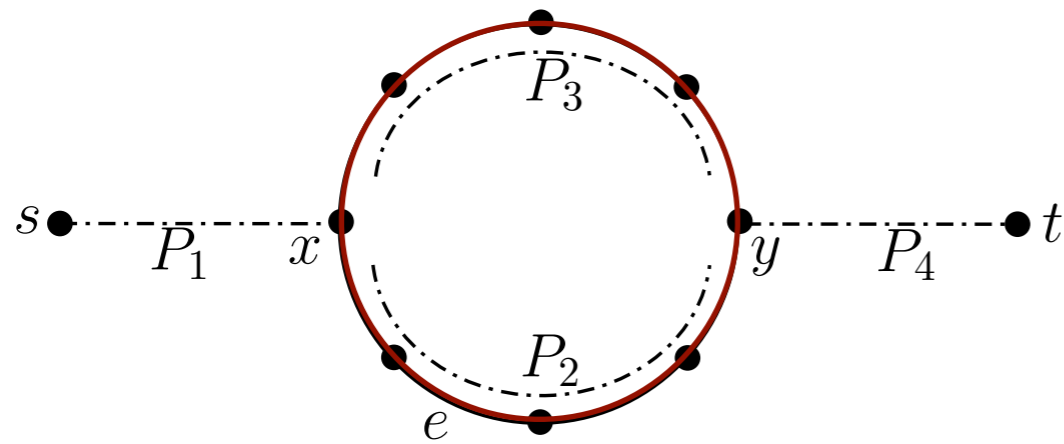
$$P' = P_1 \cdot P_2 \cdot P_4$$

$$P'' = P_1 \cdot P_3 \cdot P_4$$

► We need a tracker in the cycle



# The FVS connection



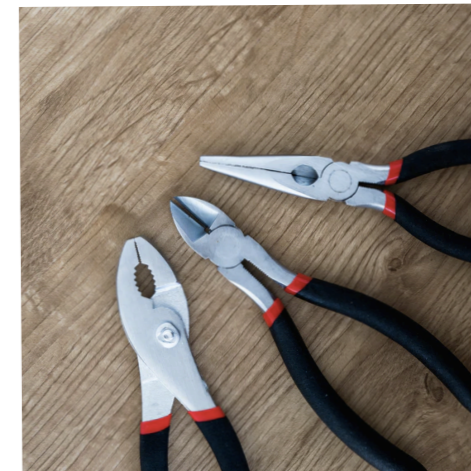
Each cycle must have a tracker

\* Cant we just use an FVS as a TS... ?

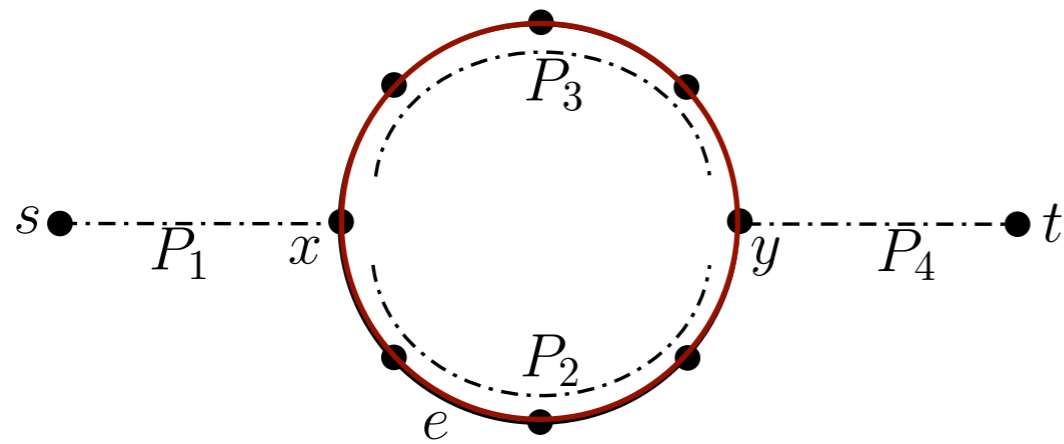
$$P' = P_1 \cdot P_2 \cdot P_4$$

$$P'' = P_1 \cdot P_3 \cdot P_4$$

► We need a tracker in the cycle



# The FVS connection



Each cycle must have a tracker

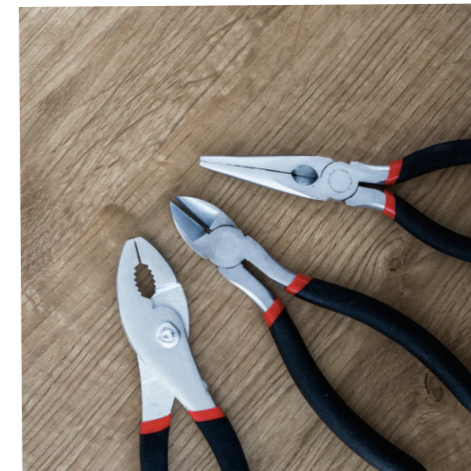
\* Cant we just use an FVS as a TS... ?

No

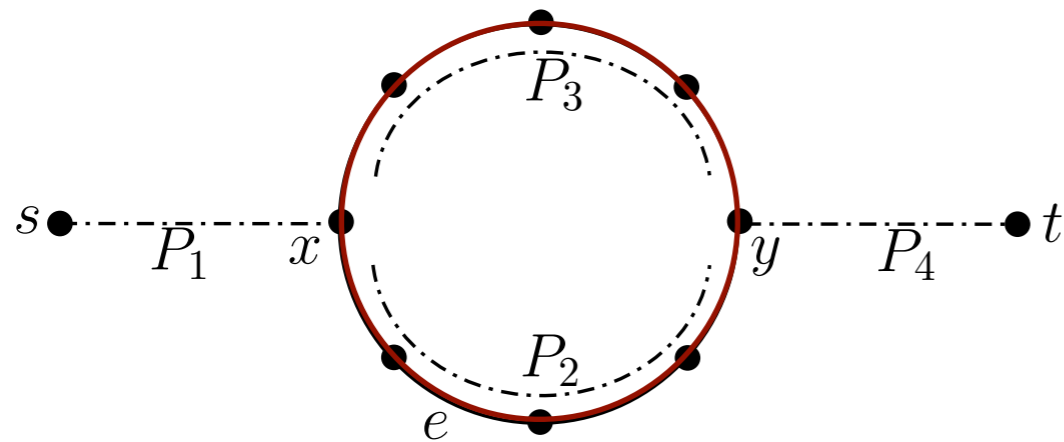
$$P' = P_1 \cdot P_2 \cdot P_4$$

$$P'' = P_1 \cdot P_3 \cdot P_4$$

► We need a tracker in the cycle



# The FVS connection



Each cycle must have a tracker

$$P' = P_1 \cdot P_2 \cdot P_4$$

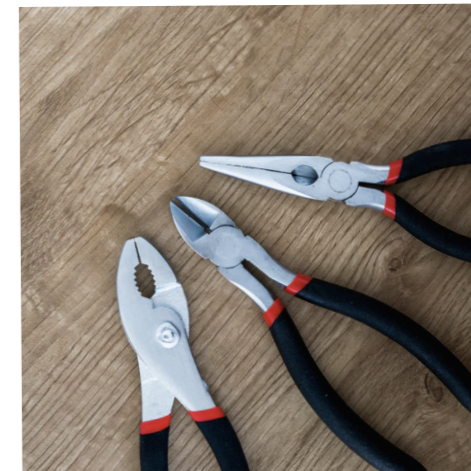
$$P'' = P_1 \cdot P_3 \cdot P_4$$

► We need a tracker in the cycle

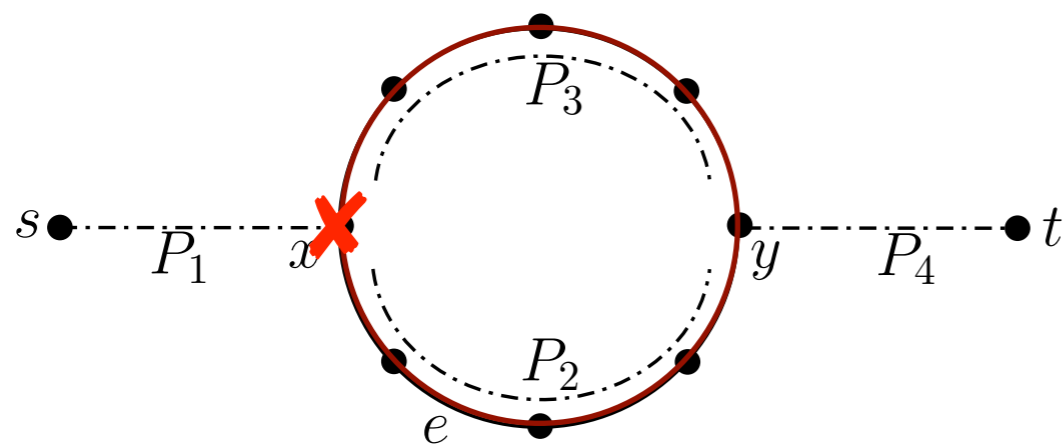
\* Cant we just use an FVS as a TS... ?

No

-> An FVS might just pick a local  $s$  or  $t$



# The FVS connection



Each cycle must have a tracker

\* Cant we just use an FVS as a TS... ?

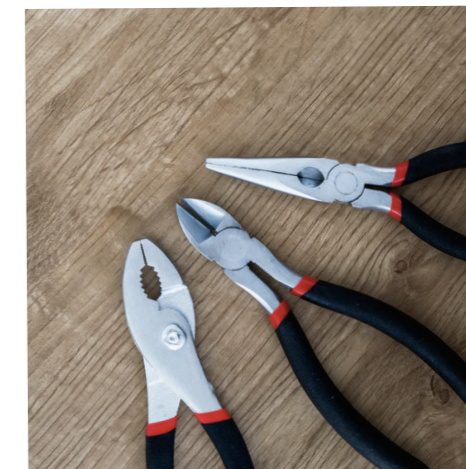
No

-> An FVS might just pick a local  $s$  or  $t$

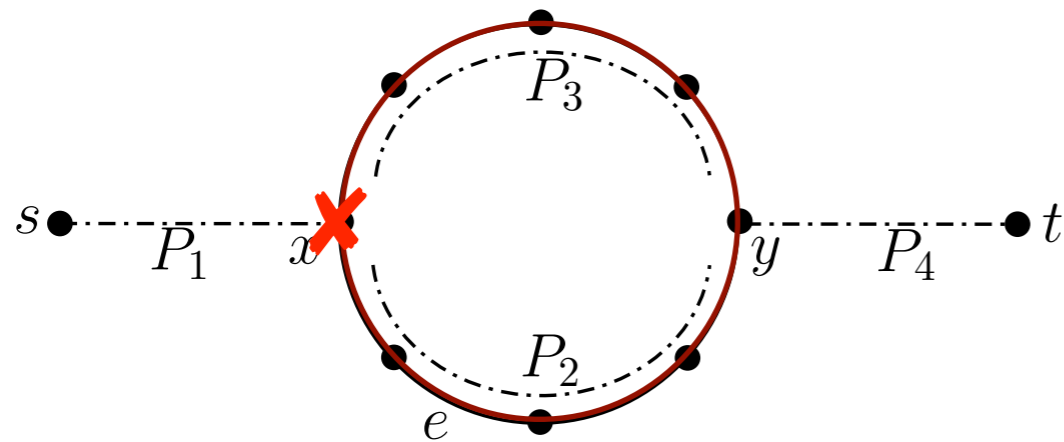
$$P' = P_1 \cdot P_2 \cdot P_4$$

$$P'' = P_1 \cdot P_3 \cdot P_4$$

► We need a tracker in the cycle



# The FVS connection



Each cycle must have a tracker

\* Cant we just use an FVS as a TS... ?

No

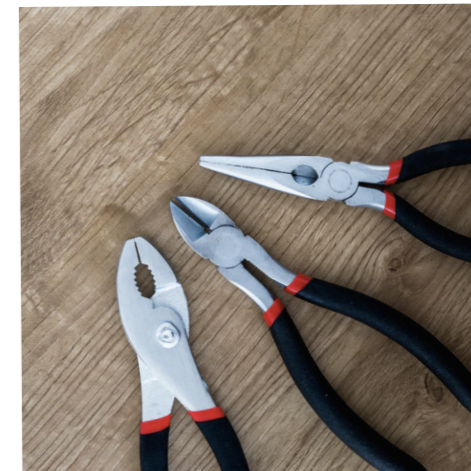
-> An FVS might just pick a local  $s$  or  $t$

$$P' = P_1 \cdot P_2 \cdot P_4$$

$$P'' = P_1 \cdot P_3 \cdot P_4$$

► We need a tracker in the cycle

$$|TS| \geq |FVS|$$





**Approximation**



# Bounded Degree Graphs

# Bounded Degree Graphs

- ▶ *Tracking Paths* is NP-hard in bounded degree graphs: By reduction from *Vertex Cover* for

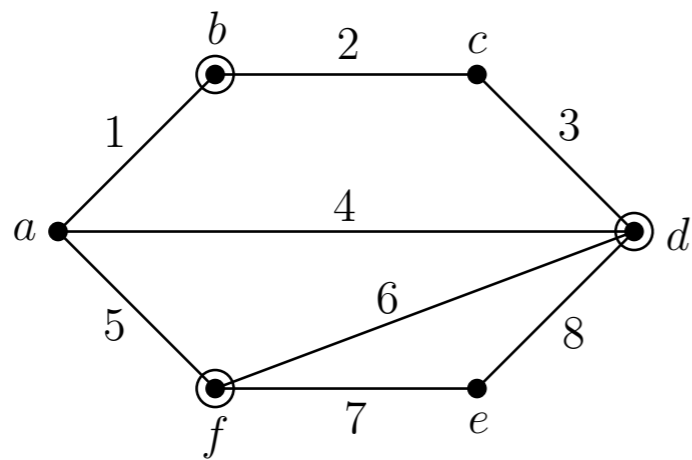
# Bounded Degree Graphs

- ▶ *Tracking Paths* is NP-hard in bounded degree graphs: By reduction from *Vertex Cover* for  $\delta \geq 6$

# Bounded Degree Graphs

- *Tracking Paths* is NP-hard in bounded degree graphs: By reduction from *Vertex Cover*

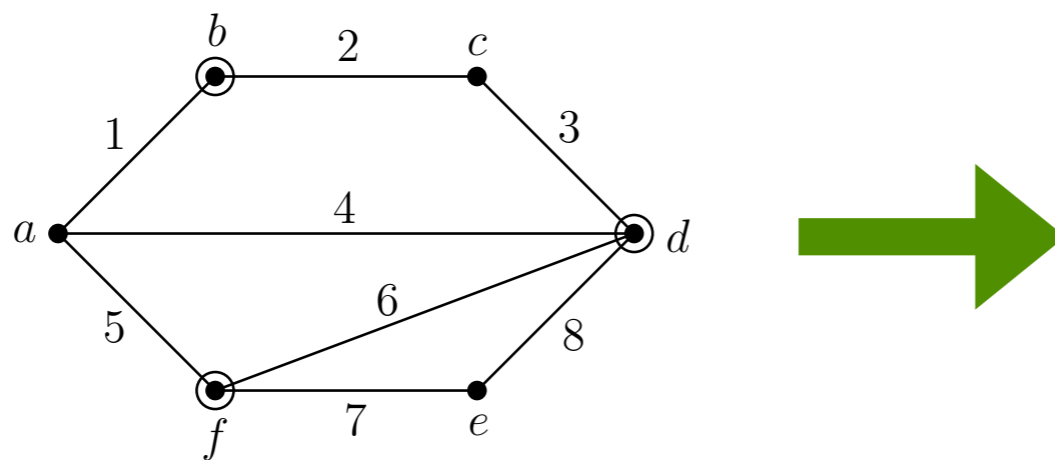
for  $\delta \geq 6$



# Bounded Degree Graphs

- *Tracking Paths* is NP-hard in bounded degree graphs: By reduction from *Vertex Cover*

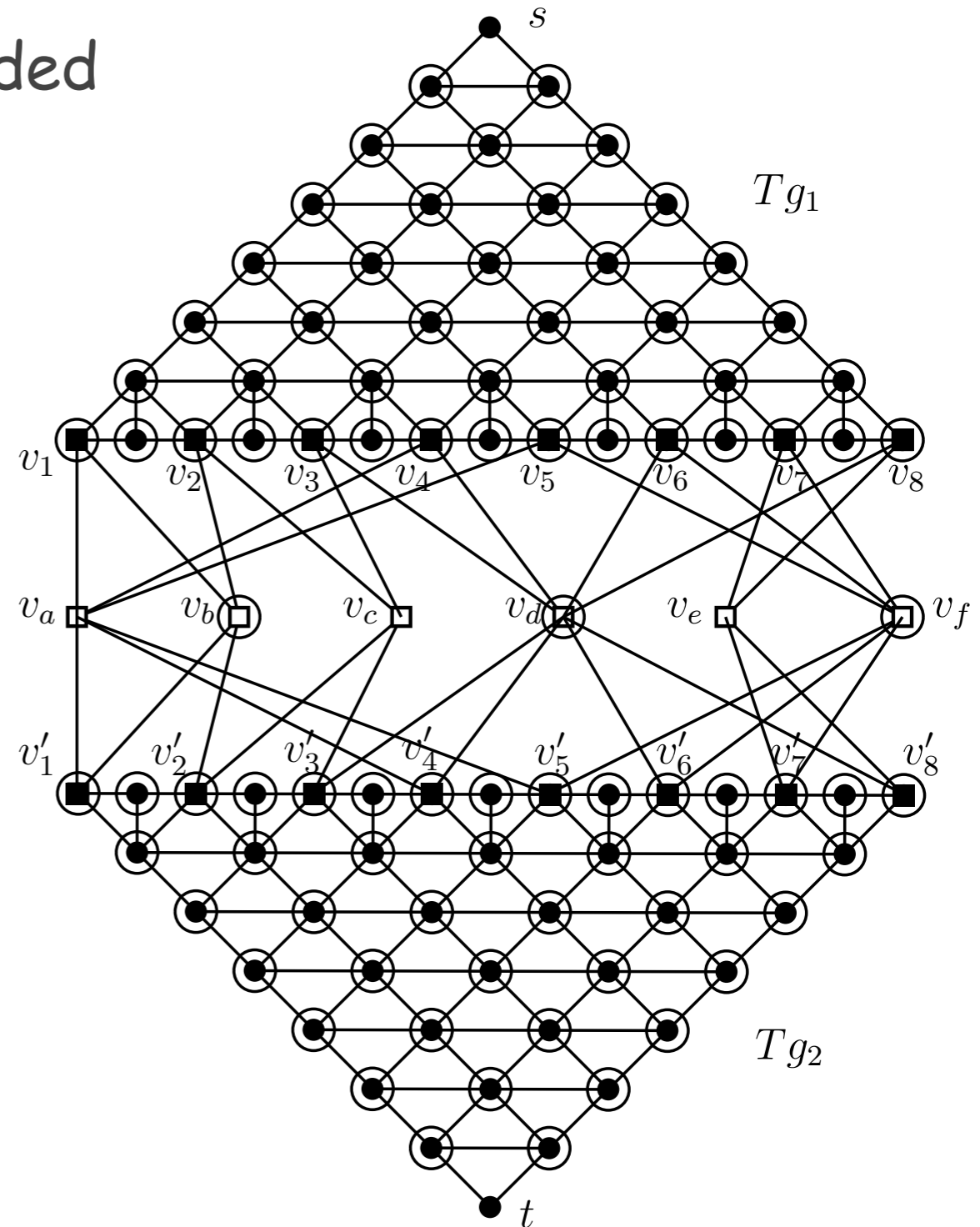
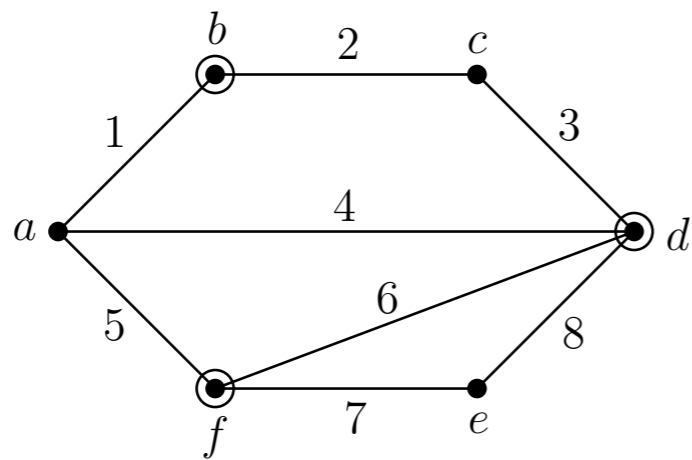
for  $\delta \geq 6$



# Bounded Degree Graphs

► *Tracking Paths* is NP-hard in bounded degree graphs: By reduction from *Vertex Cover*

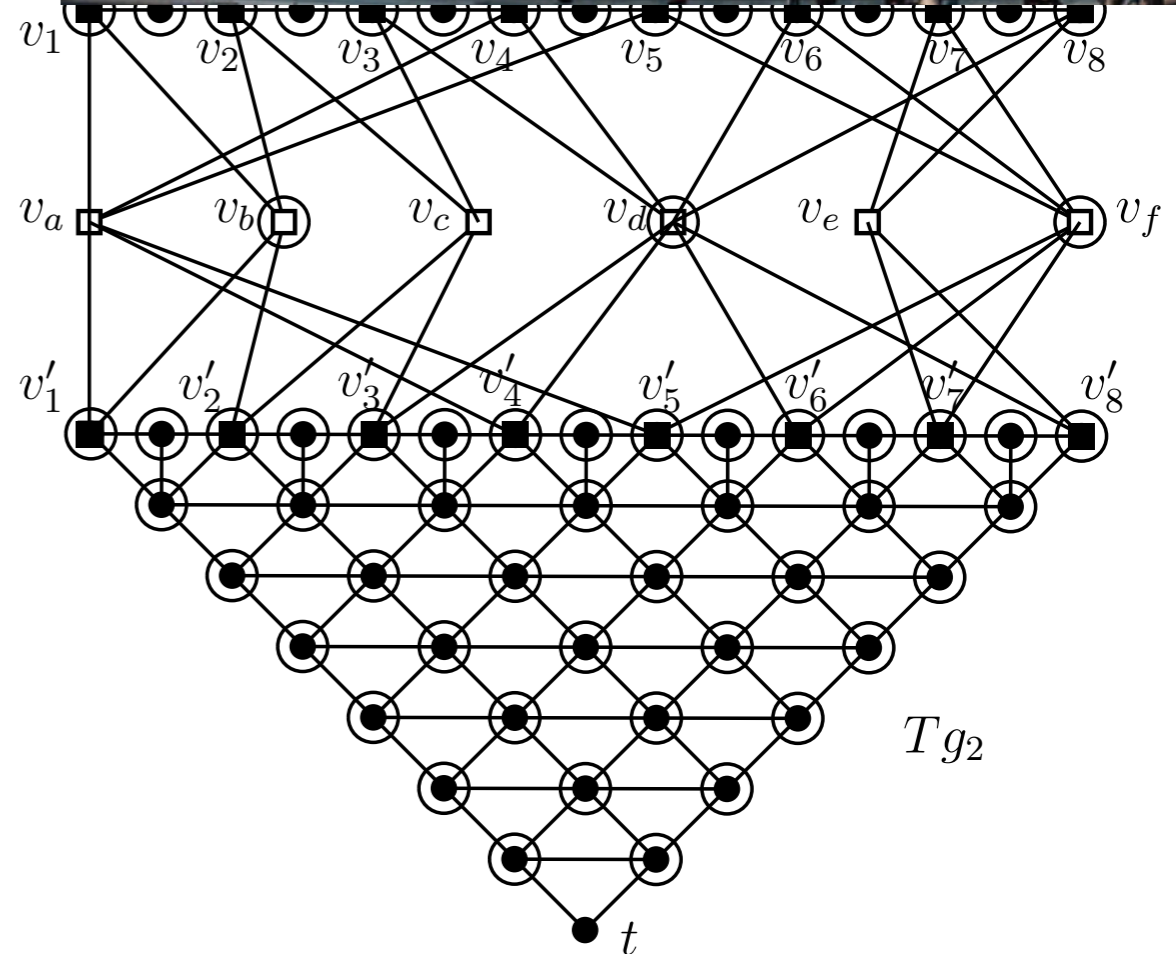
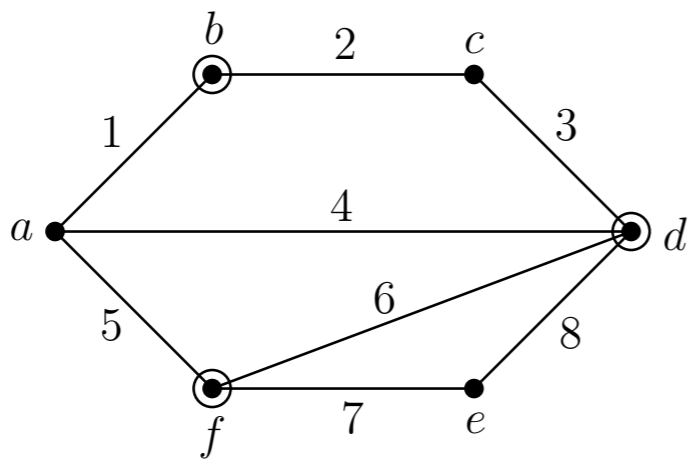
for  $\delta \geq 6$



# Bounded Degree Graphs

- *Tracking Paths* is NP-hard in bounded degree graphs: By reduction from *Vertex Cover*

for  $\delta \geq 6$

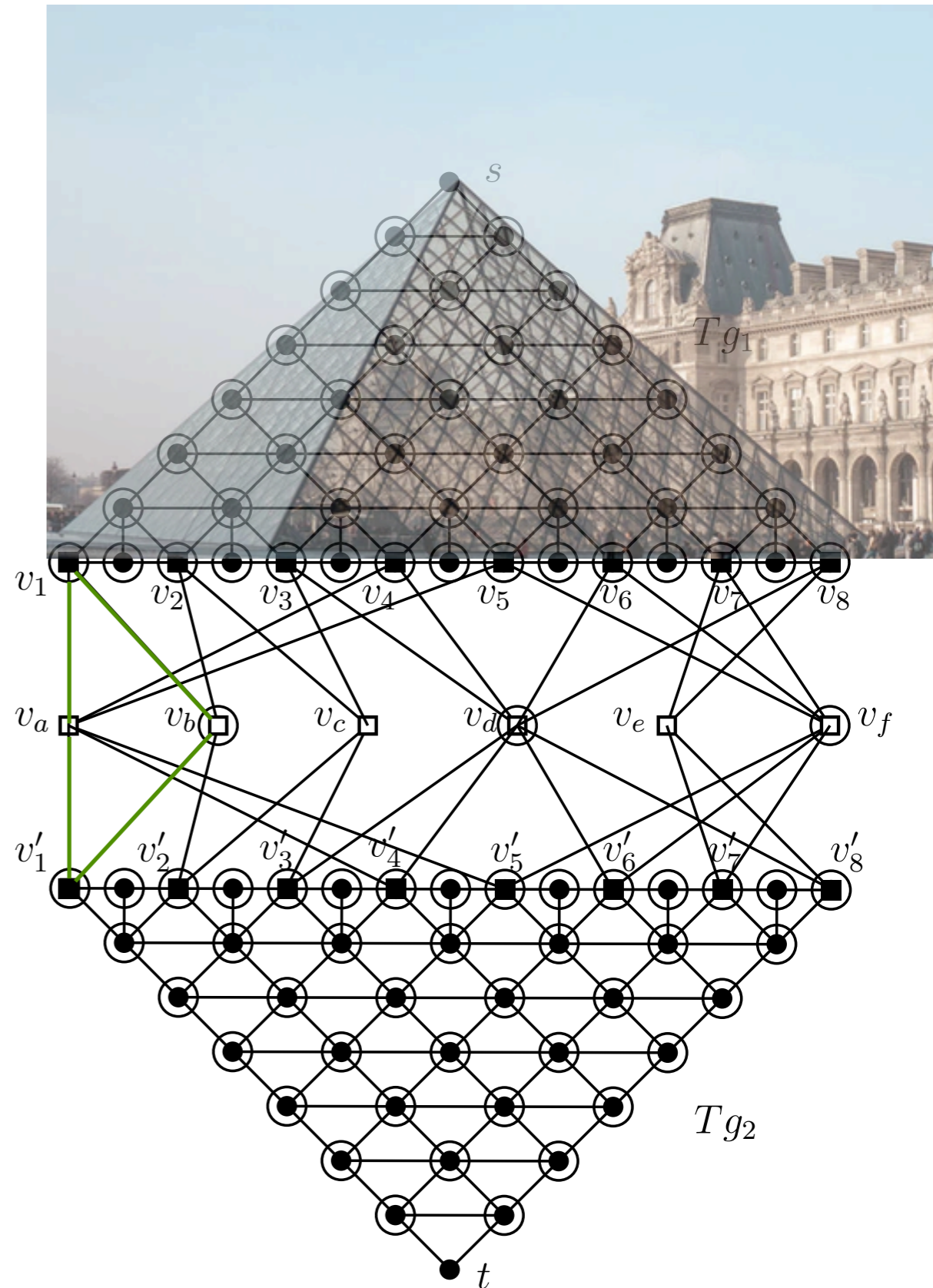
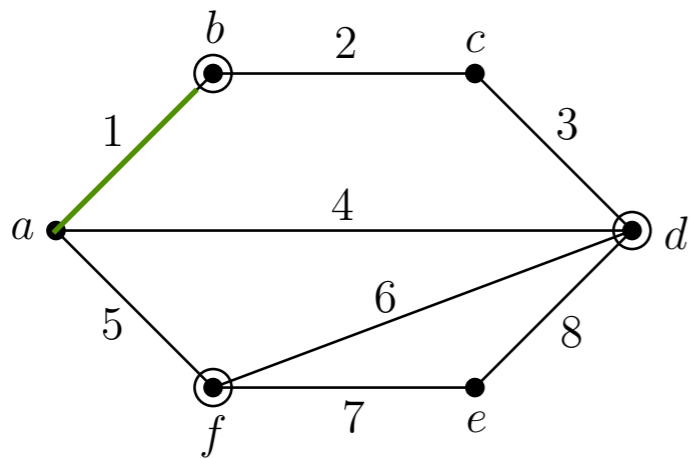


# Bounded Degree Graphs

► NP-hard: By reduction from

*Vertex Cover*

for  $\delta \geq 6$



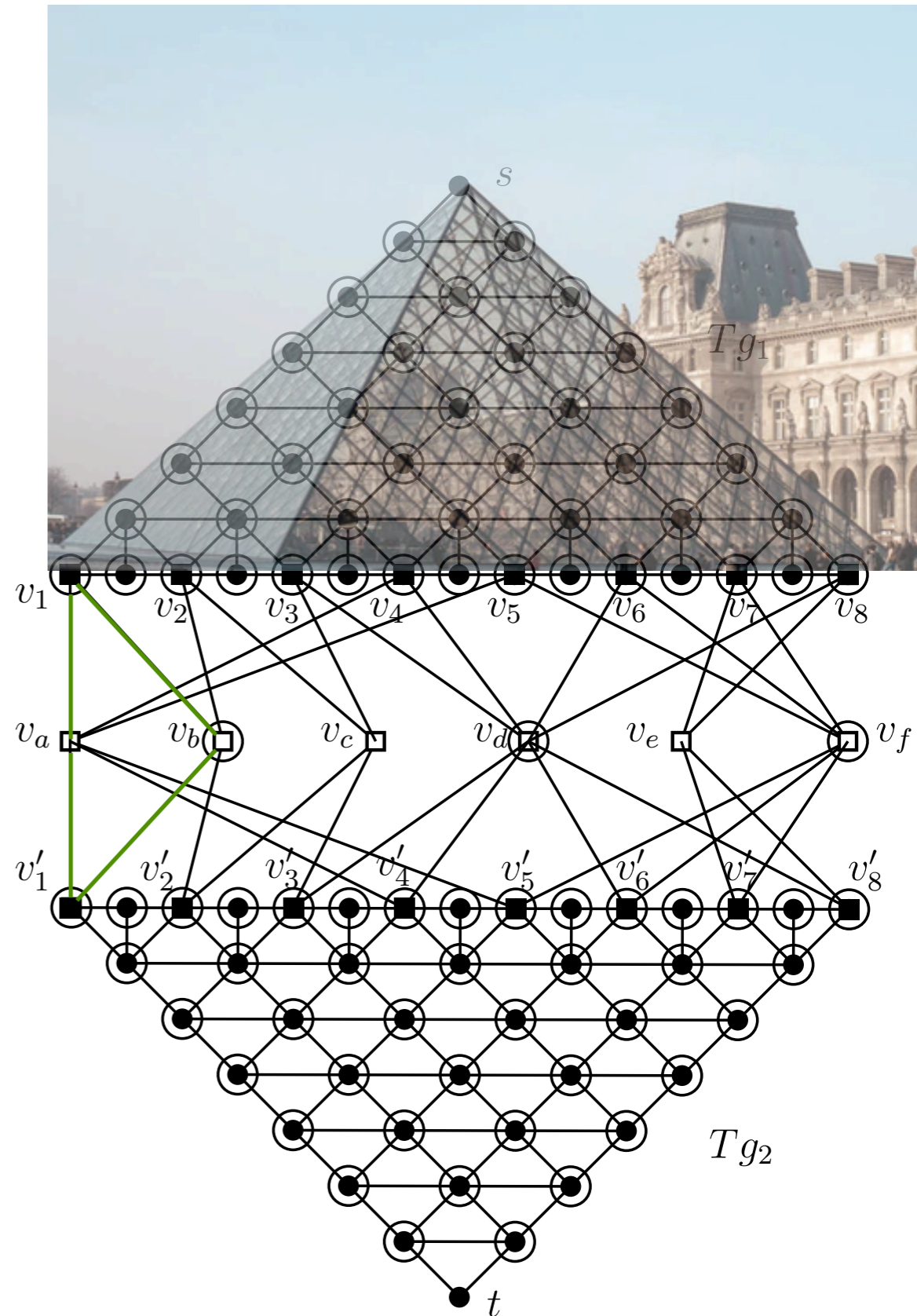
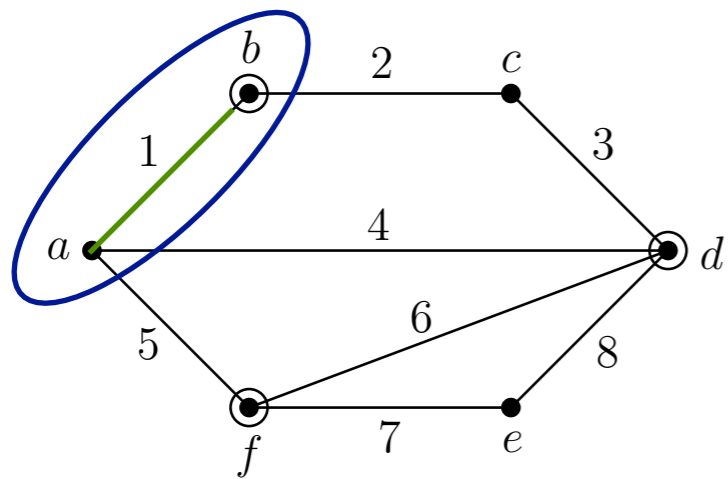


# Bounded Degree Graphs

► NP-hard: By reduction from

*Vertex Cover*

for  $\delta \geq 6$

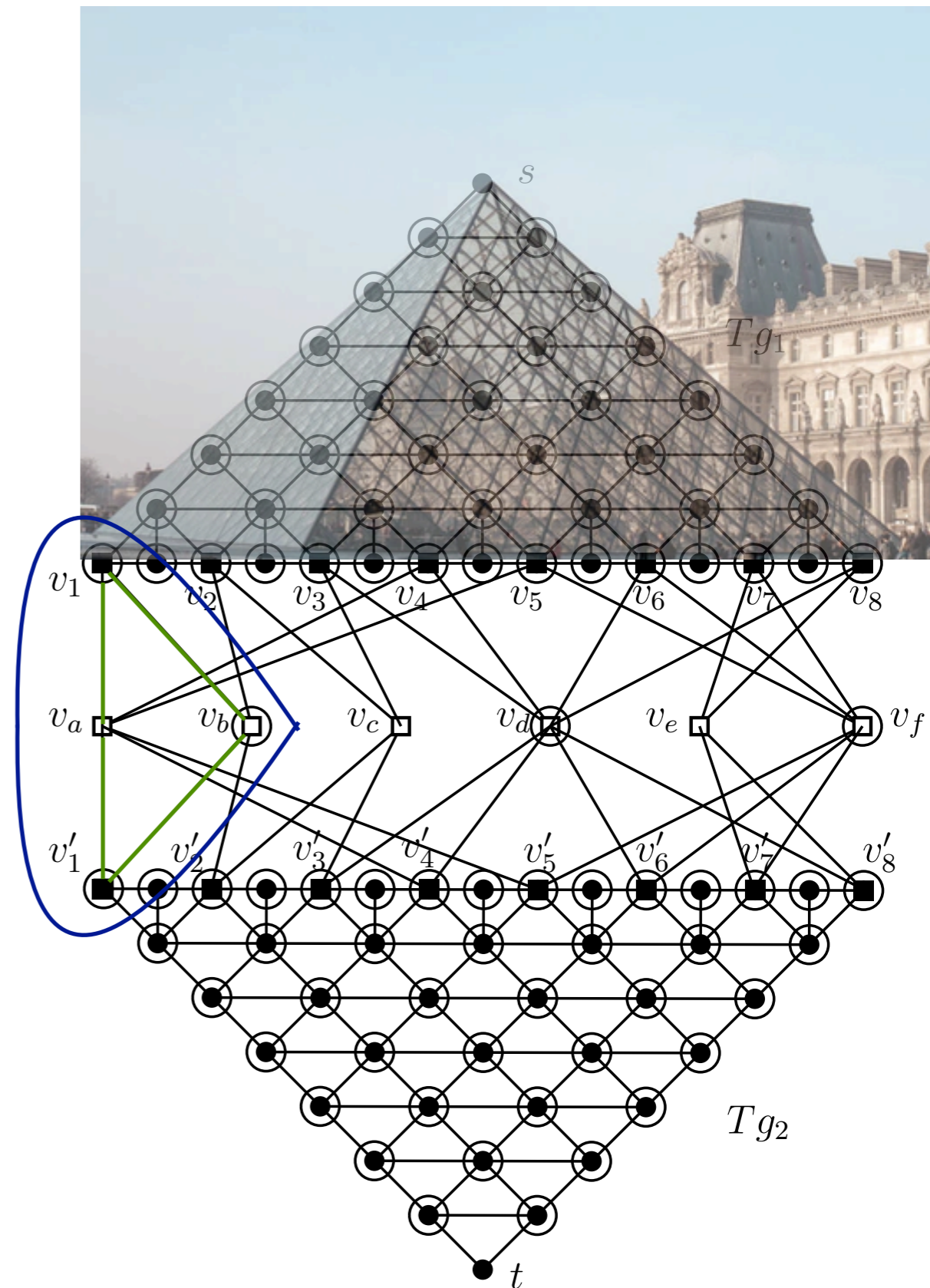
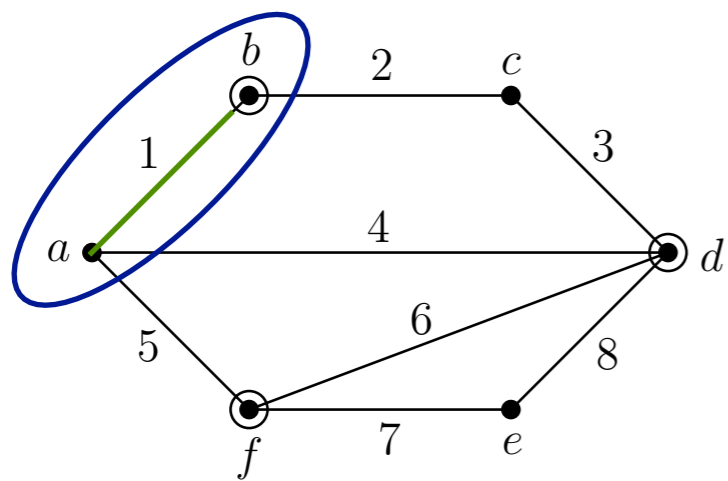


# Bounded Degree Graphs

► NP-hard: By reduction from

*Vertex Cover*

for  $\delta \geq 6$

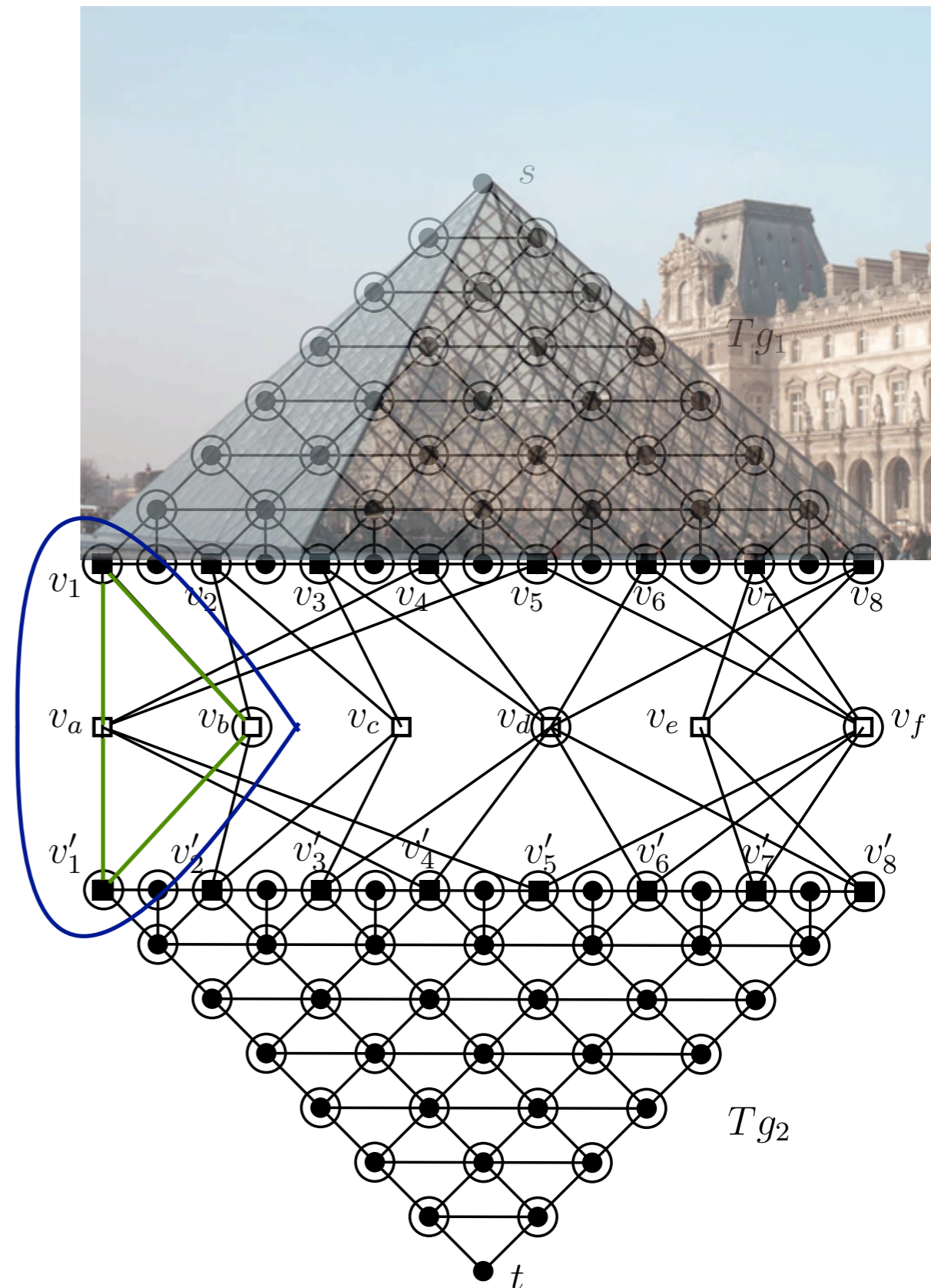
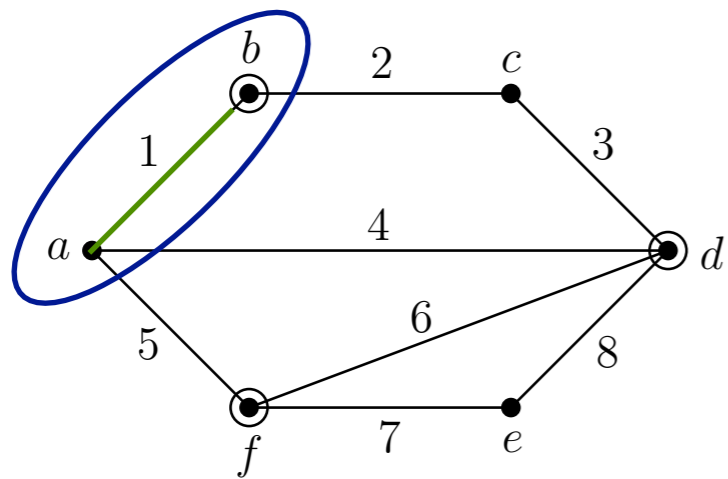


# Bounded Degree Graphs

► NP-hard: By reduction from

*Vertex Cover*

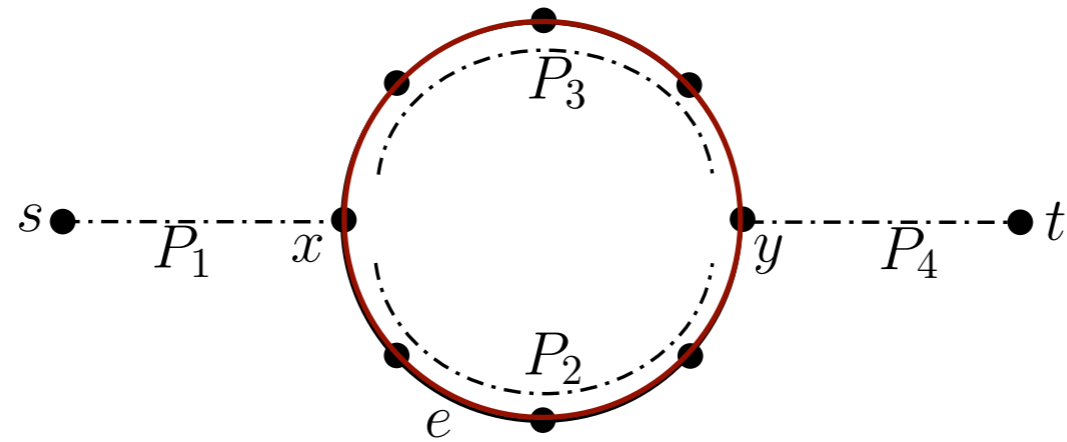
for  $\delta \geq 6$



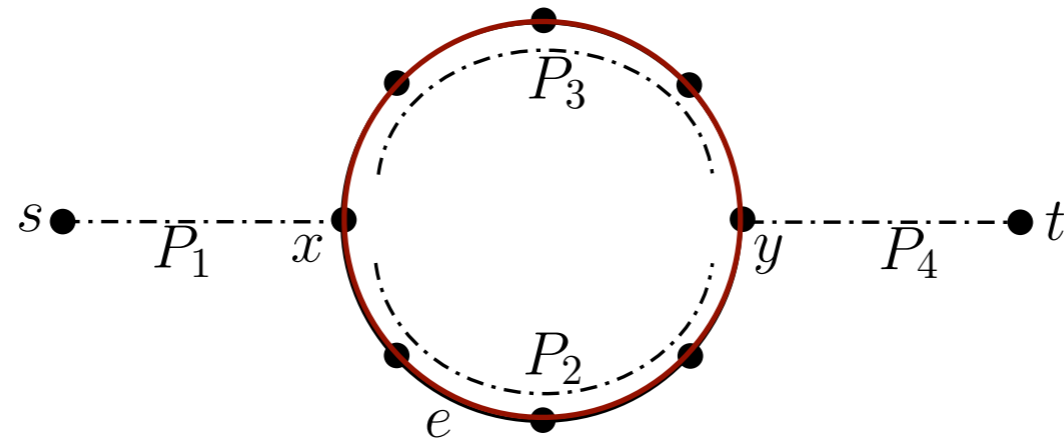
Complexity for  
 $3 \leq \delta \leq 5$  ?

# Bounded Degree Graphs

# Bounded Degree Graphs

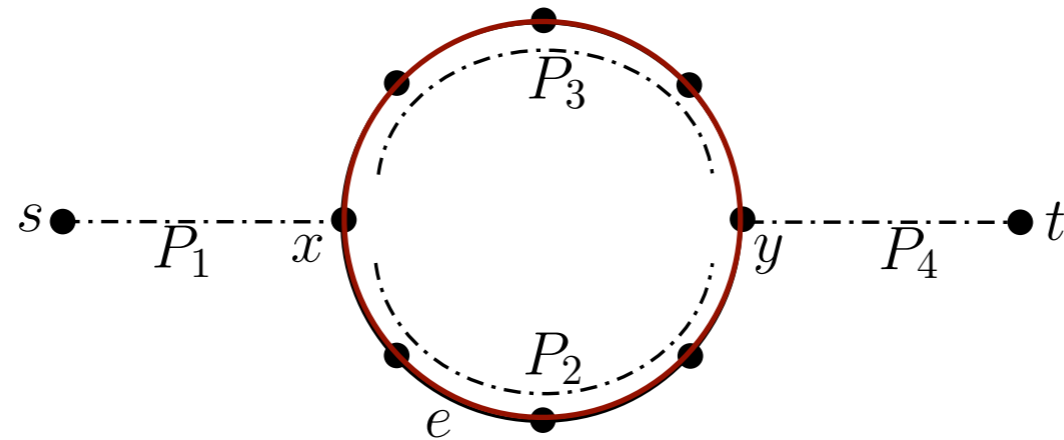


# Bounded Degree Graphs



Algorithm:

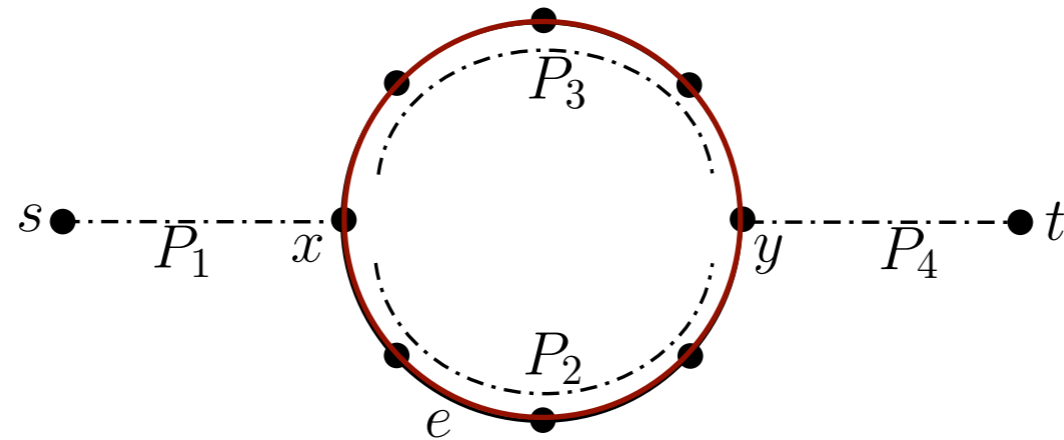
# Bounded Degree Graphs



Algorithm:

- ▶ start with a 2-approx FVS

# Bounded Degree Graphs

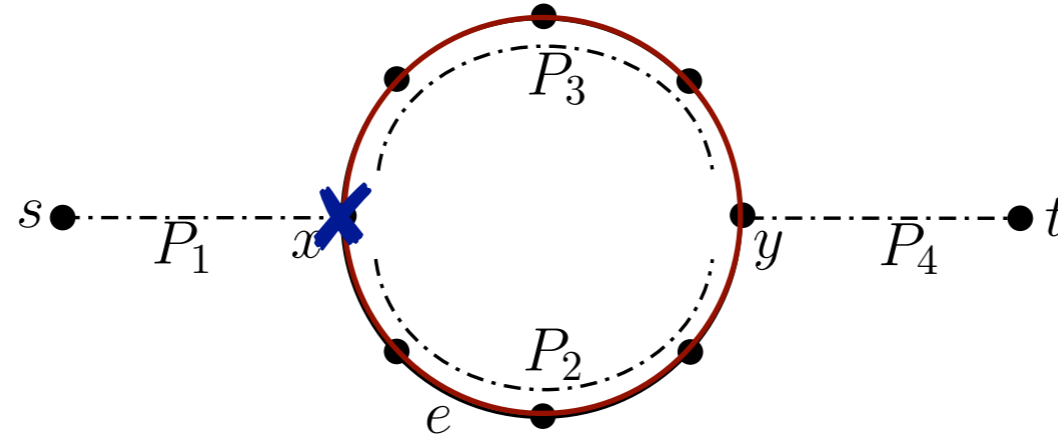


## Algorithm:

- ▶ start with a 2-approx FVS
- ▶ for each vertex in FVS  $\rightarrow$  Mark it and all its neighbours as trackers



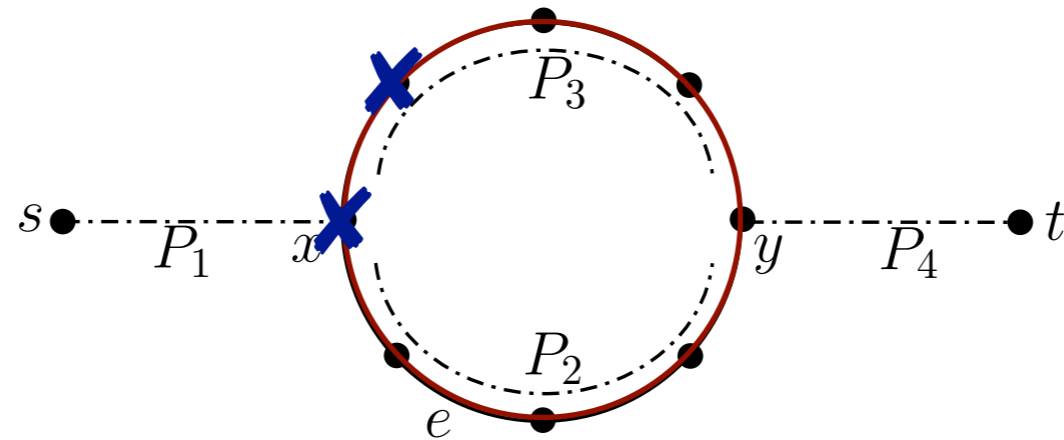
# Bounded Degree Graphs



## Algorithm:

- ▶ start with a 2-approx FVS
- ▶ for each vertex in FVS  $\rightarrow$  Mark it and all its neighbours as trackers

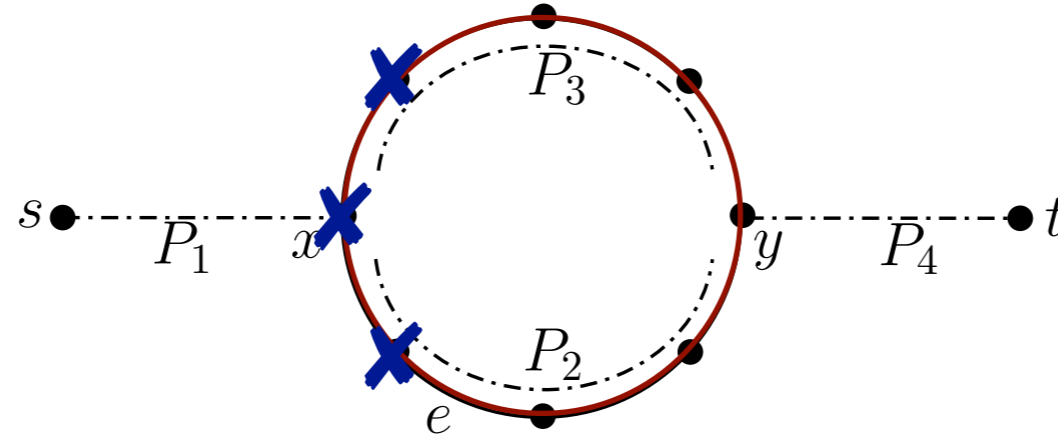
# Bounded Degree Graphs



## Algorithm:

- ▶ start with a 2-approx FVS
- ▶ for each vertex in FVS  $\rightarrow$  Mark it and all its neighbours as trackers

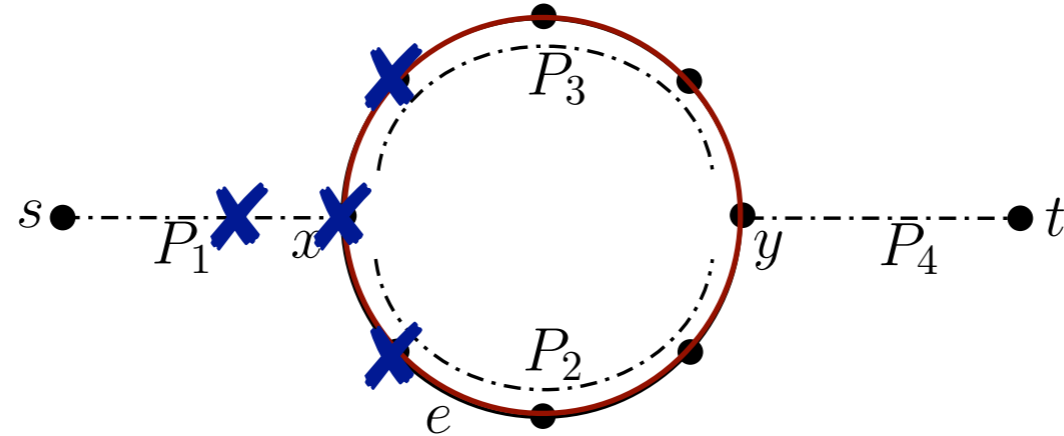
# Bounded Degree Graphs



## Algorithm:

- ▶ start with a 2-approx FVS
- ▶ for each vertex in FVS  $\rightarrow$  Mark it and all its neighbours as trackers

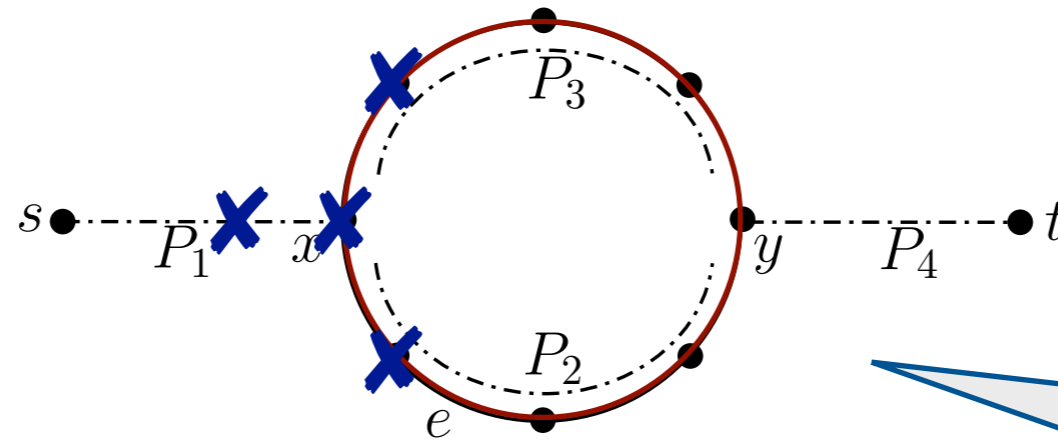
# Bounded Degree Graphs



## Algorithm:

- ▶ start with a 2-approx FVS
- ▶ for each vertex in FVS  $\rightarrow$  Mark it and all its neighbours as trackers

# Bounded Degree Graphs

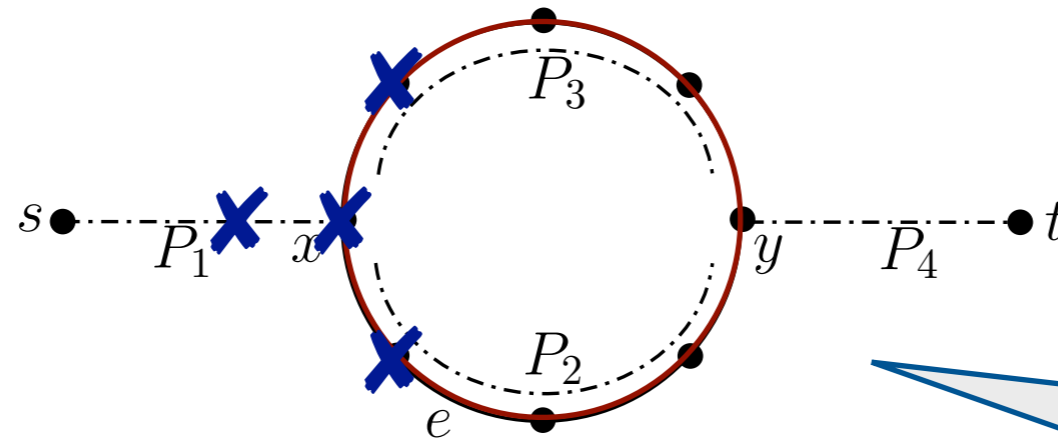


Each cycle will have a tracker (other than local s-t)

## Algorithm:

- ▶ start with a 2-approx FVS
- ▶ for each vertex in FVS  $\rightarrow$  Mark it and all its neighbours as trackers

# Bounded Degree Graphs



Each cycle will have a tracker (other than local s-t)

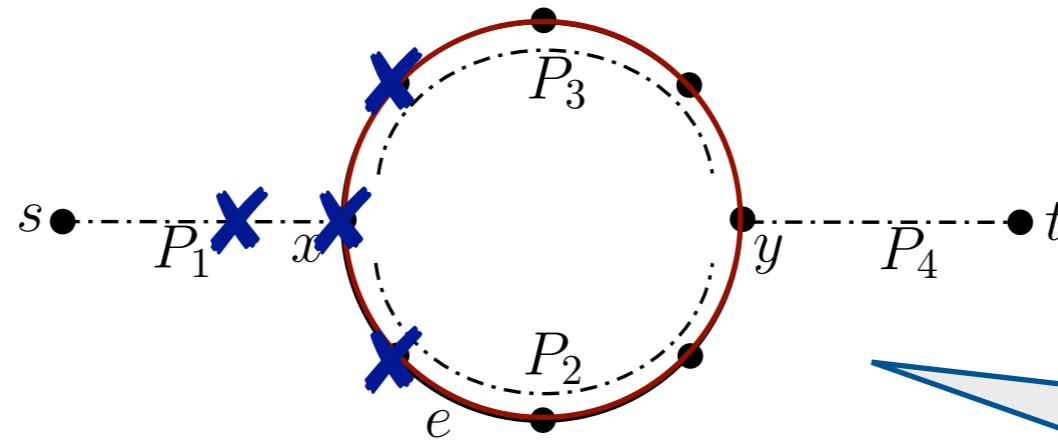
## Algorithm:

- ▶ start with a 2-approx FVS
- ▶ for each vertex in FVS  $\rightarrow$  Mark it and all its neighbours as trackers

*Recall*

$$|TS| \geq |FVS|$$

# Bounded Degree Graphs



Each cycle will have a tracker (other than local s-t)

## Algorithm:

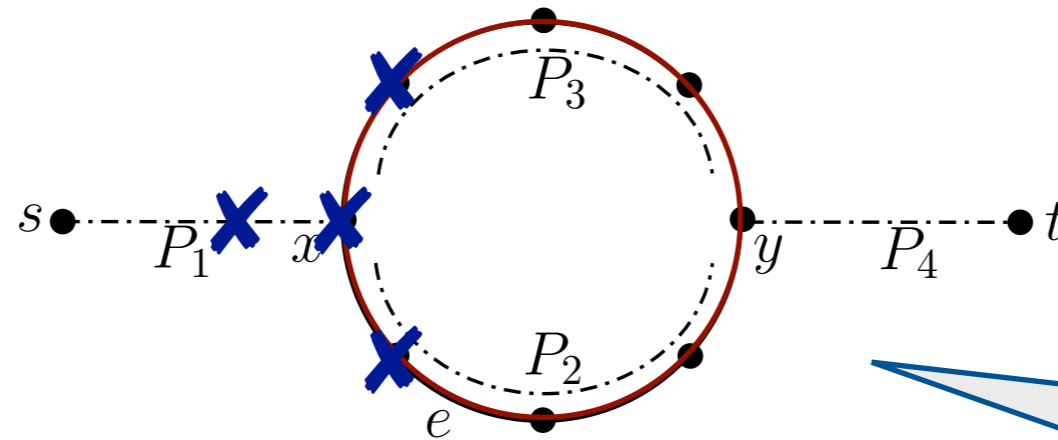
- ▶ start with a 2-approx FVS
- ▶ for each vertex in FVS  $\rightarrow$  Mark it and all its neighbours as trackers

$$\delta = \text{max degree of vertices}$$

*Recall*

$$|TS| \geq |FVS|$$

# Bounded Degree Graphs



## Algorithm:

- ▶ start with a 2-approx FVS
- ▶ for each vertex in FVS  $\rightarrow$  Mark it and all its neighbours as trackers

$\delta = \text{max degree of vertices}$

*Recall*

$$|TS| \geq |FVS|$$

$$2(\delta + 1) - \text{Approximate algorithm}$$







**Restricted Versions**

# Restricted Versions

# Restricted Versions

▶ *for which the problem lies in complexity class....*

# Restricted Versions

▶ *for which the problem lies in complexity class....*



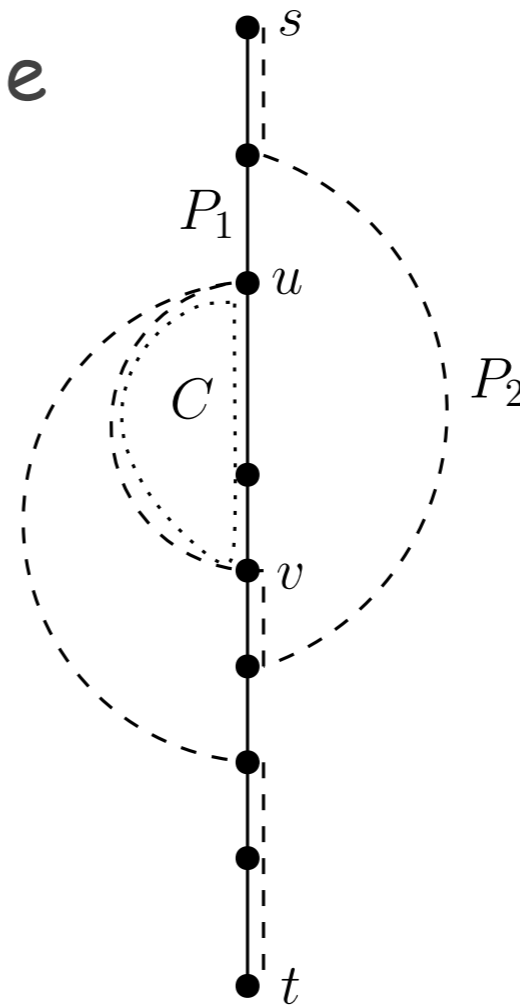
# Chordal Graphs

# Chordal Graphs

- ▶ Two distinct  $s$ - $t$  paths form a cycle

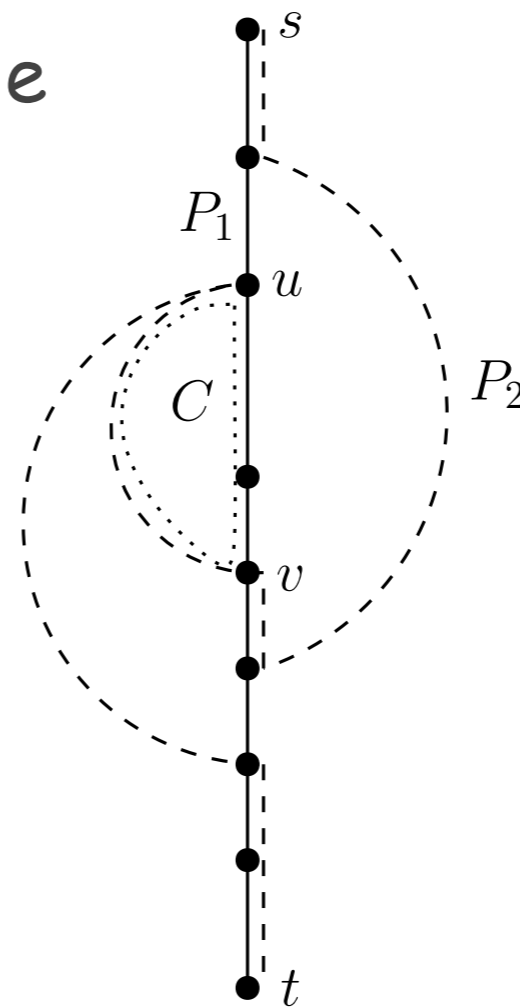
# Chordal Graphs

- ▶ Two distinct  $s$ - $t$  paths form a cycle



# Chordal Graphs

- ▶ Two distinct  $s$ - $t$  paths form a cycle
- ▶ Cycles need trackers





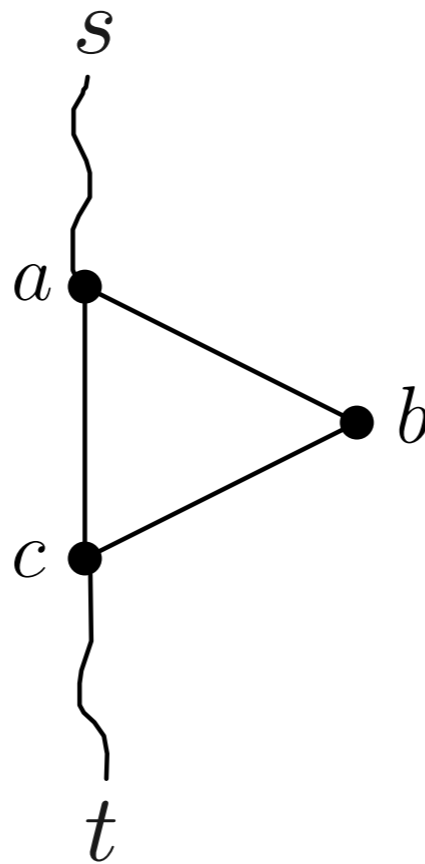
# Chordal Graphs

# Chordal Graphs

- ▶ All cycles in chordal graphs contain triangles

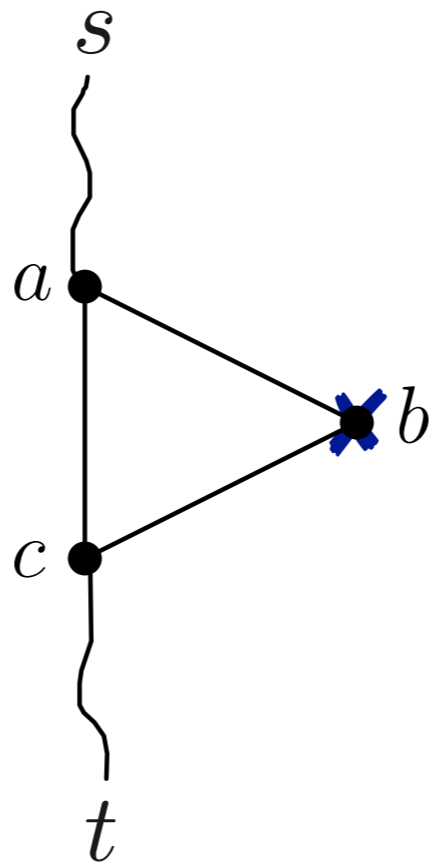
# Chordal Graphs

- ▶ All cycles in chordal graphs contain triangles



# Chordal Graphs

- ▶ All cycles in chordal graphs contain triangles



# Chordal Graphs

# Chordal Graphs

Algorithm:

# Chordal Graphs

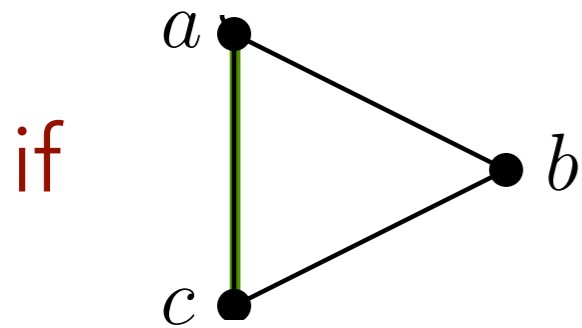
if



Algorithm:

► For each edge  $(a,c)$ :

# Chordal Graphs



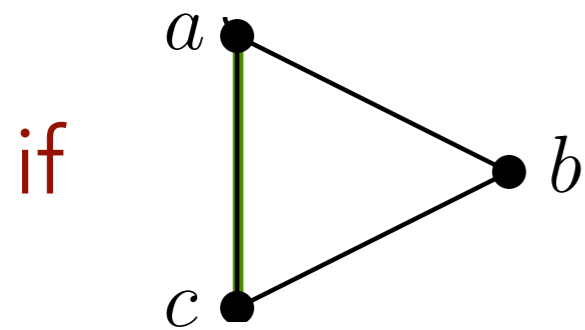
## Algorithm:

► For each edge  $(a,c)$ :

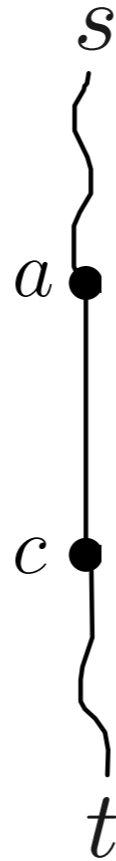
► For each vertex  $b$  adjacent to both endpoints of the edge



# Chordal Graphs



and

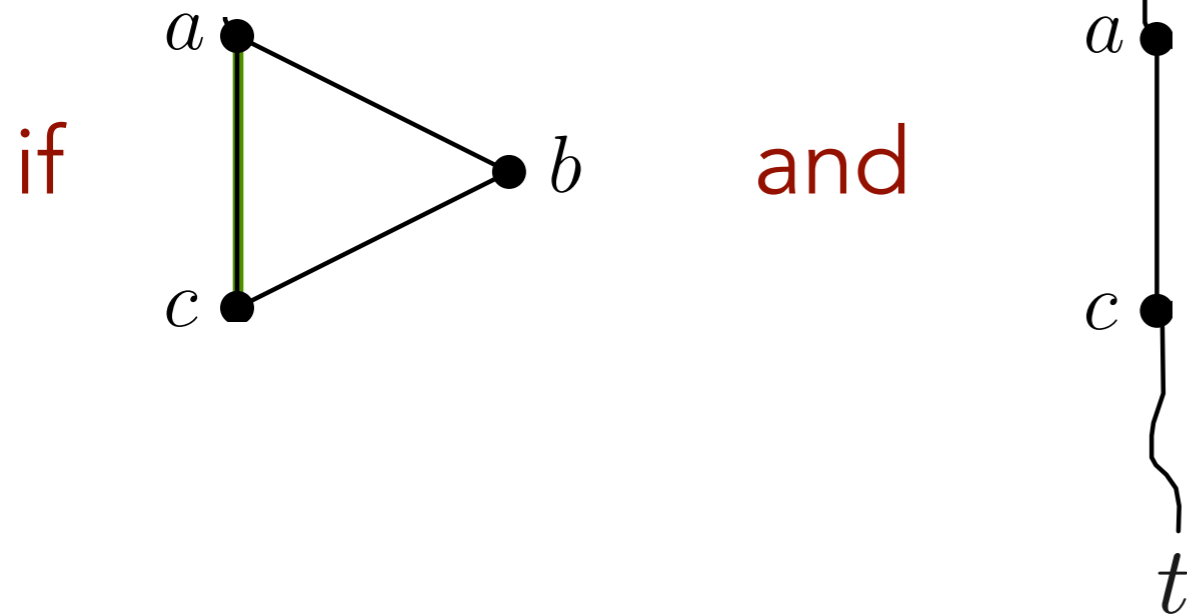


## Algorithm:

► For each edge  $(a, c)$ :

► For each vertex  $b$  adjacent to both endpoints of the edge

# Chordal Graphs

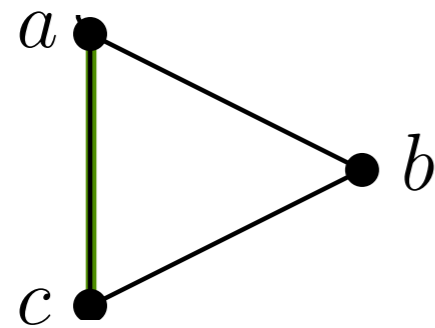


## Algorithm:

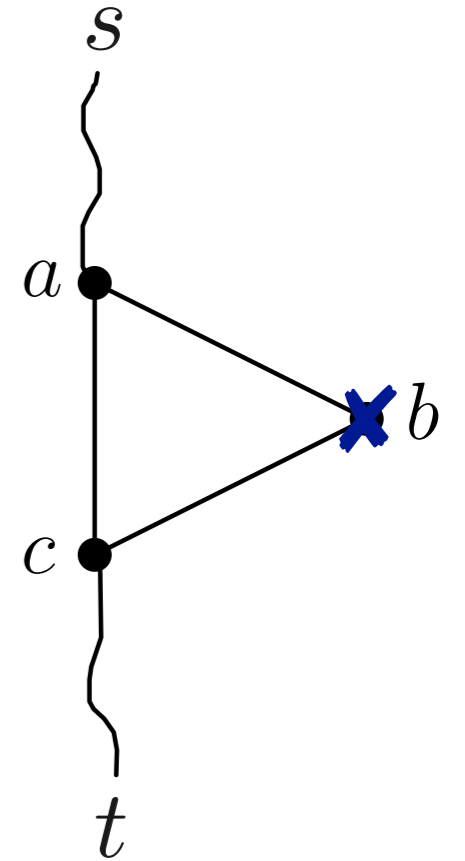
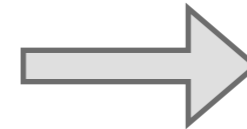
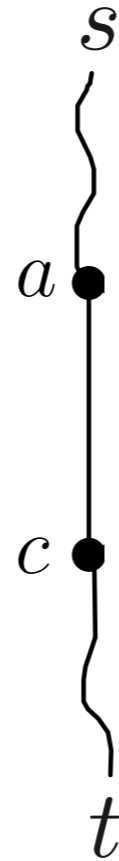
- ▶ For each edge  $(a,c)$ :
  - ▶ For each vertex  $b$  adjacent to both endpoints of the edge
  - ▶ If the edge participates in an  $s$ - $t$  path without  $b$  -> Mark  $b$  as a tracker

# Chordal Graphs

if



and



## Algorithm:

- ▶ For each edge  $(a, c)$ :
  - ▶ For each vertex  $b$  adjacent to both endpoints of the edge
  - ▶ If the edge participates in an  $s$ - $t$  path without  $b$  -> Mark  $b$  as a tracker

# Chordal Graphs

Proof by  
Contradiction

# Chordal Graphs

Proof by  
Contradiction

▶ Suppose not

# Chordal Graphs

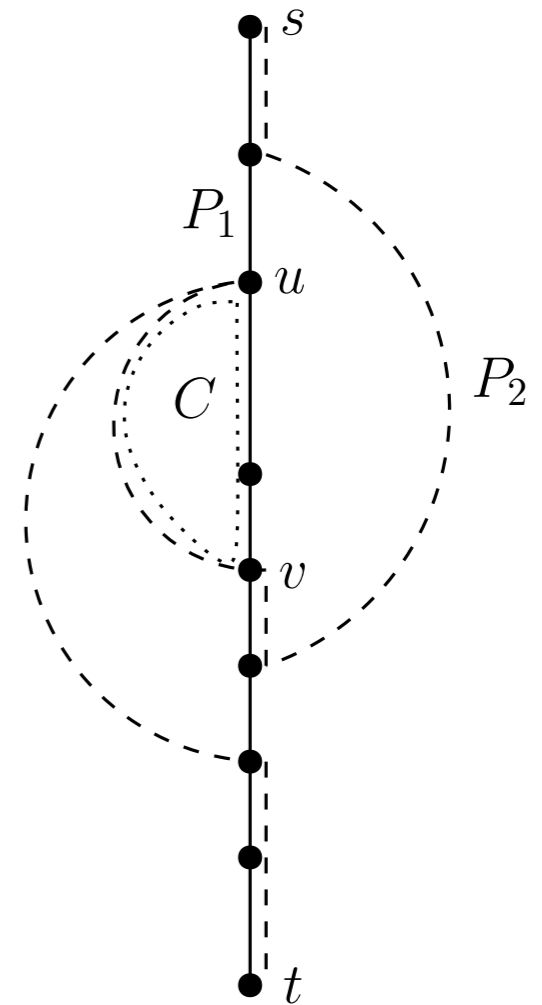
Proof by  
Contradiction

- ▶ Suppose not
- ▶ There will be a cycle with no trackers

# Chordal Graphs

Proof by  
Contradiction

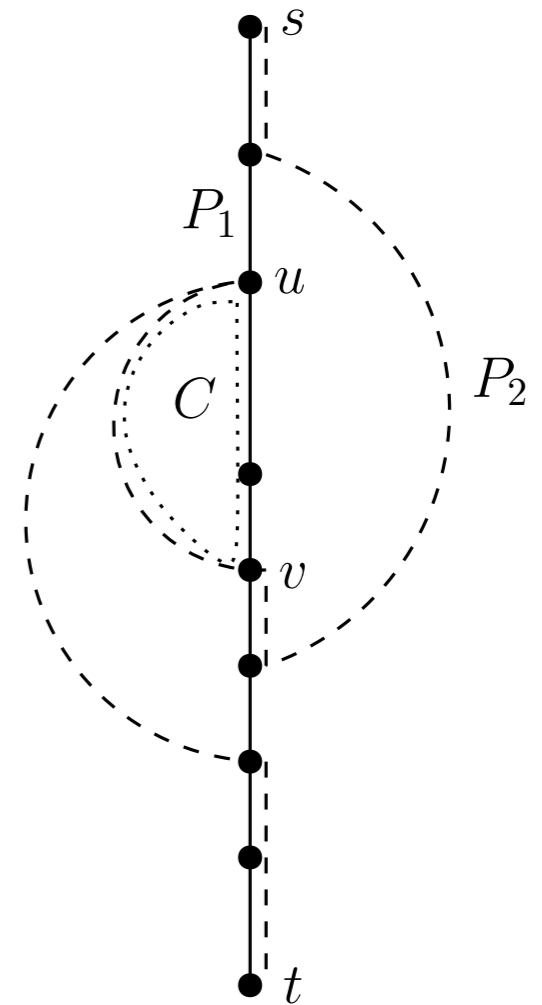
- ▶ Suppose not
- ▶ There will be a cycle with no trackers



# Chordal Graphs

Proof by  
Contradiction

- ▶ Suppose not
- ▶ There will be a cycle with no trackers
- ▶ Cycle will have a triangle

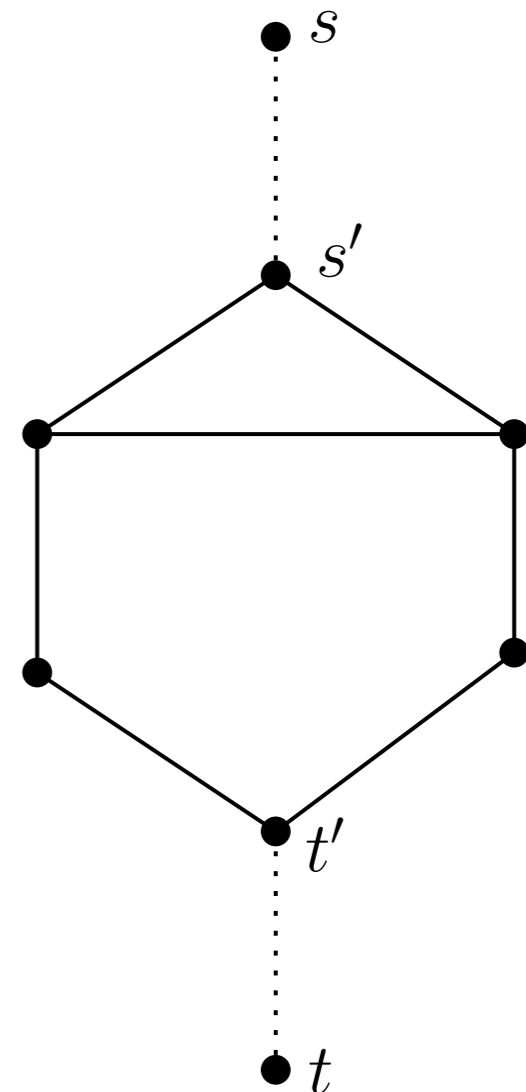




# Chordal Graphs

Proof by  
Contradiction

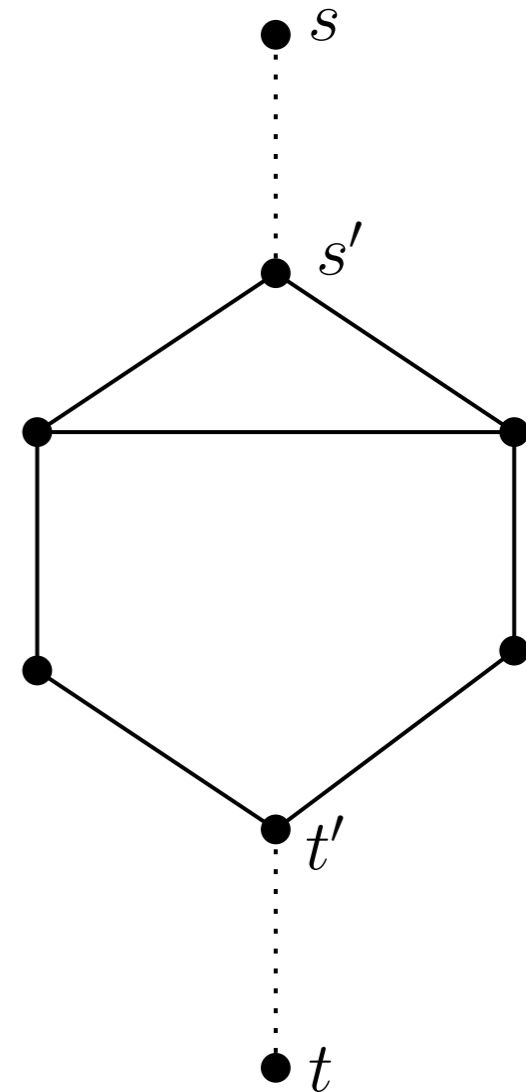
- ▶ Suppose not
- ▶ There will be a cycle with no trackers
- ▶ Cycle will have a triangle



# Chordal Graphs

Proof by  
Contradiction

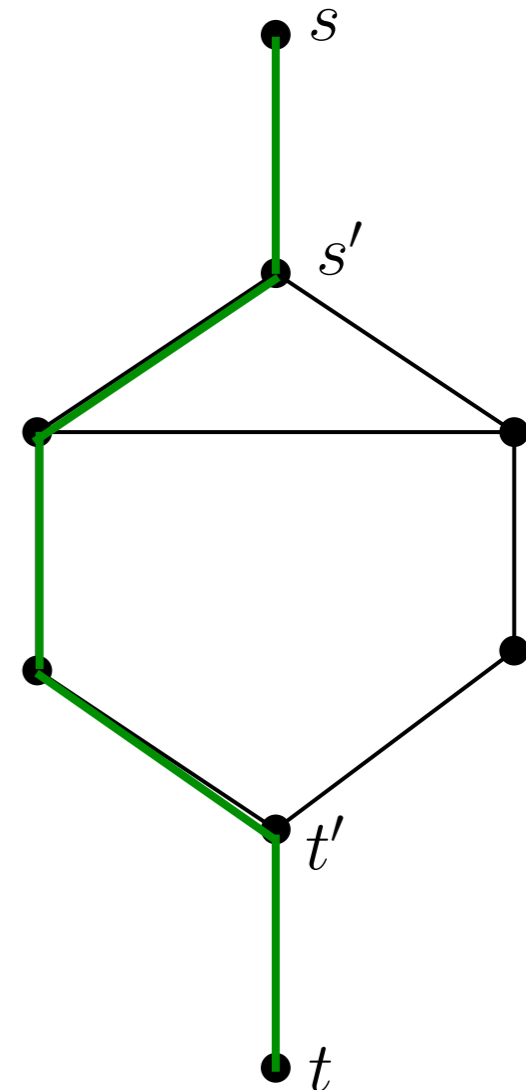
- ▶ Suppose not
- ▶ There will be a cycle with no trackers
- ▶ Cycle will have a triangle
- ▶ Triangle would have had a tracker



# Chordal Graphs

Proof by  
Contradiction

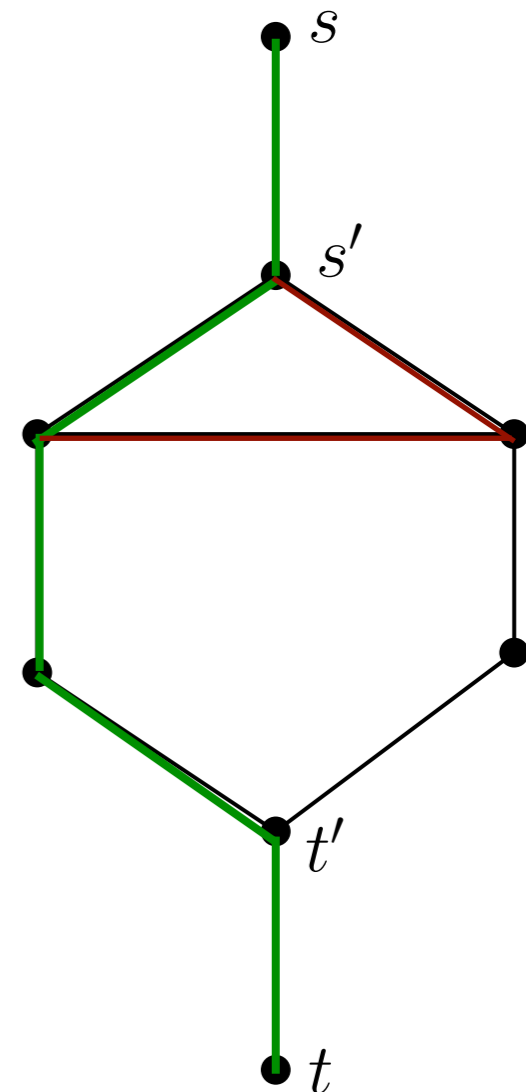
- ▶ Suppose not
- ▶ There will be a cycle with no trackers
- ▶ Cycle will have a triangle
- ▶ Triangle would have had a tracker



# Chordal Graphs

Proof by  
Contradiction

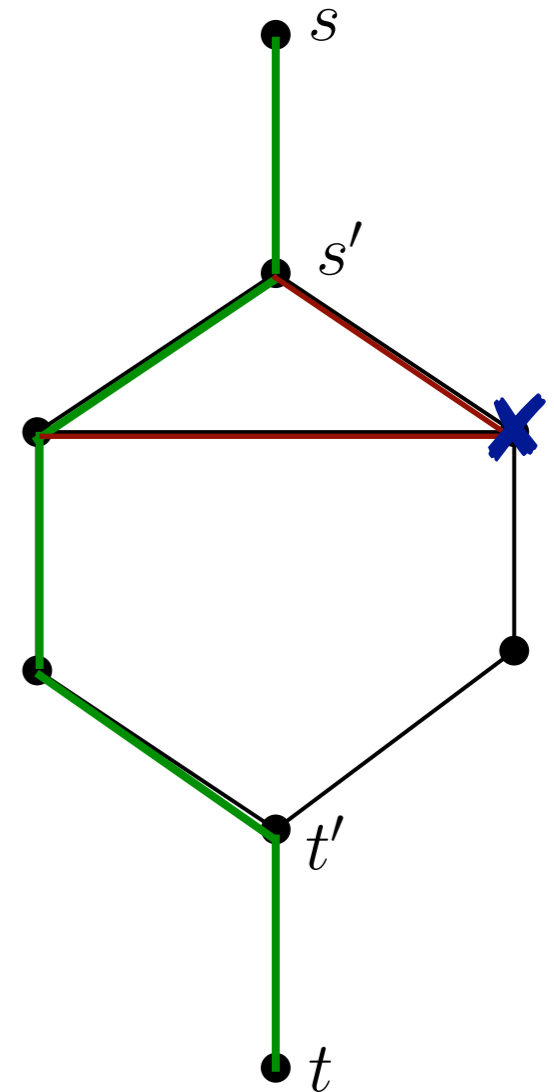
- ▶ Suppose not
- ▶ There will be a cycle with no trackers
- ▶ Cycle will have a triangle
- ▶ Triangle would have had a tracker



# Chordal Graphs

Proof by  
Contradiction

- ▶ Suppose not
- ▶ There will be a cycle with no trackers
- ▶ Cycle will have a triangle
- ▶ Triangle would have had a tracker

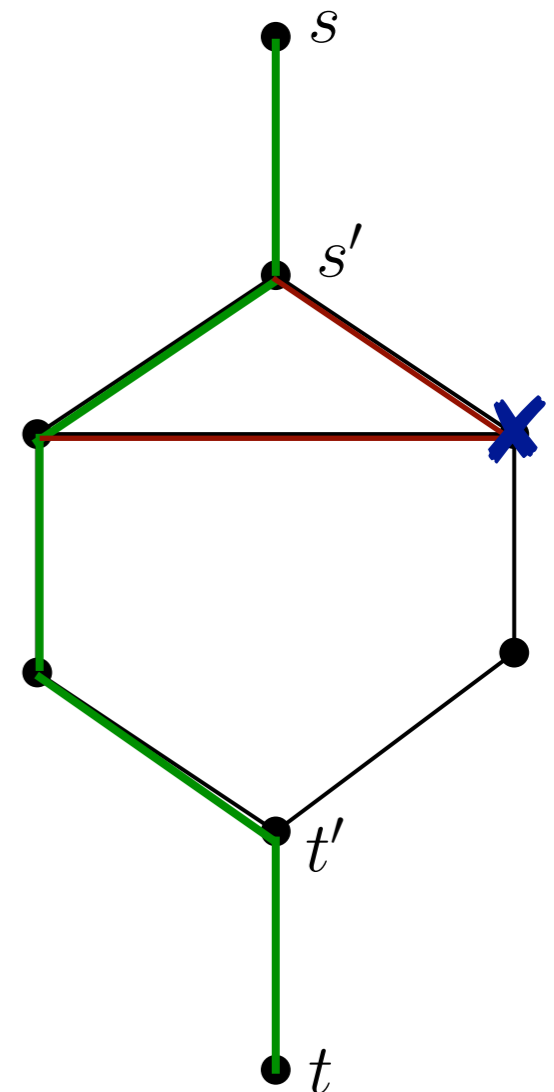


# Chordal Graphs

Proof by  
Contradiction

- ▶ Suppose not
- ▶ There will be a cycle with no trackers
- ▶ Cycle will have a triangle
- ▶ Triangle would have had a tracker

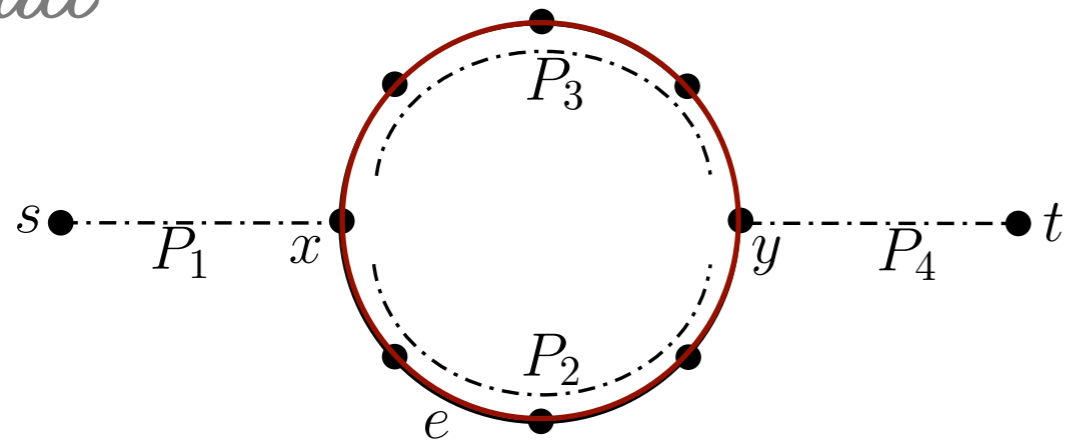
**Tracking Paths** is polynomial time solvable for chordal graphs



# Tracking Paths .... but with Edges

# Tracking Paths .... but with Edges

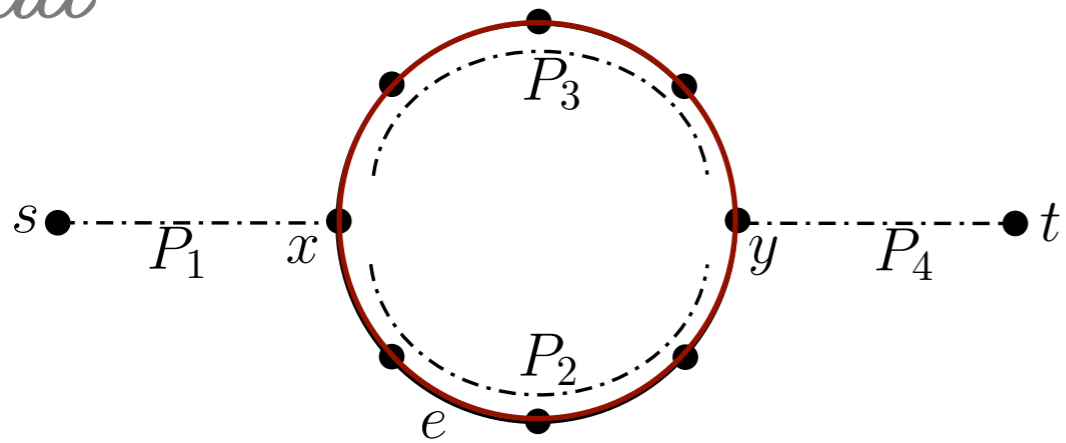
*Recall*





# Tracking Paths .... but with Edges

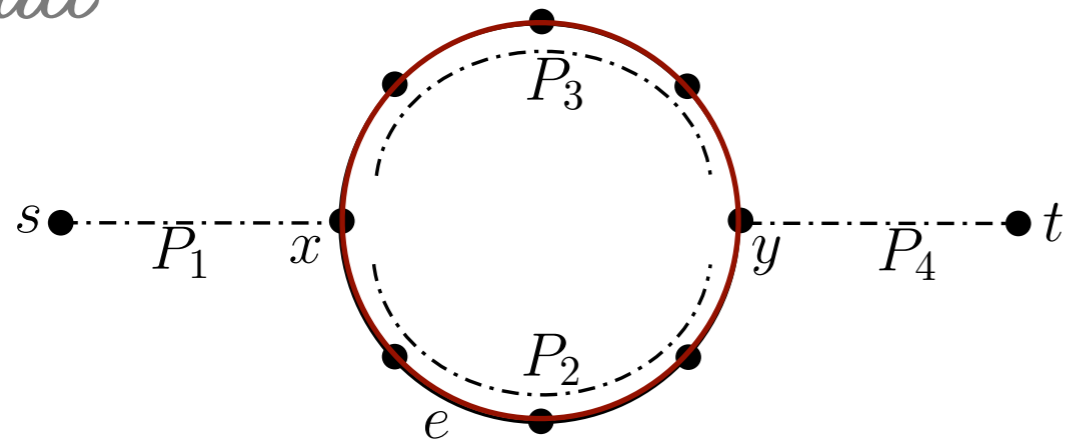
*Recall*



Each cycle must have a tracker

# Tracking Paths .... but with Edges

*Recall*

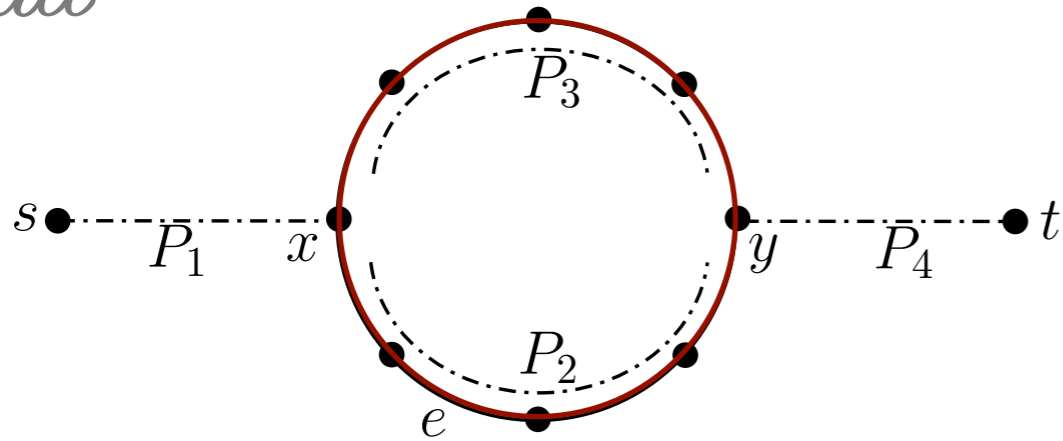


Each cycle must have a tracker

► An FVS might pick only a local  $s$  or  $t$

# Tracking Paths .... but with Edges

*Recall*

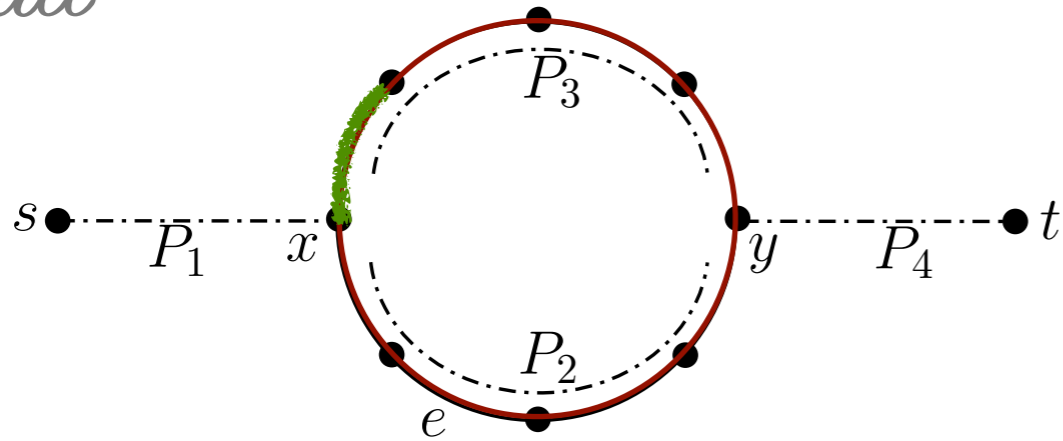


Each cycle must have a tracker

- ▶ An FVS might pick only a local  $s$  or  $t$
- ▶ But what if we use edges as trackers... ?

# Tracking Paths .... but with Edges

*Recall*

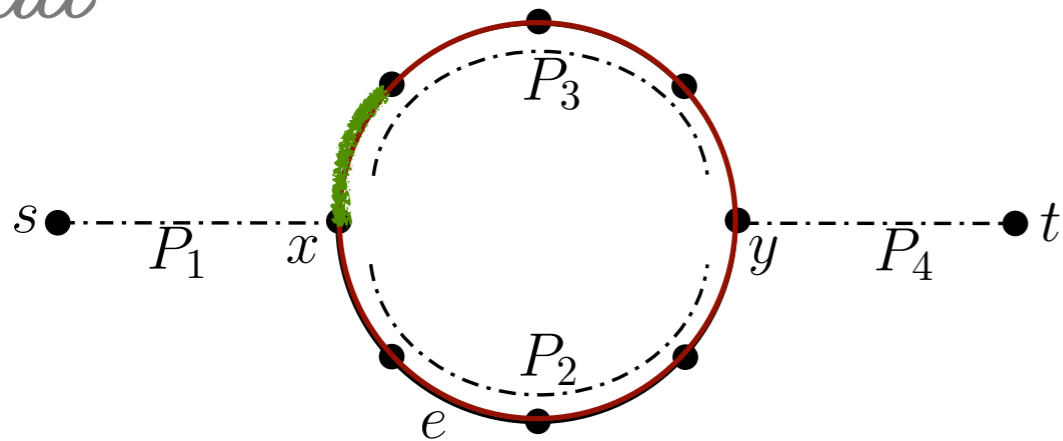


Each cycle must have a tracker

- ▶ An FVS might pick only a local  $s$  or  $t$
- ▶ But what if we use edges as trackers... ?

# Tracking Paths .... but with Edges

*Recall*

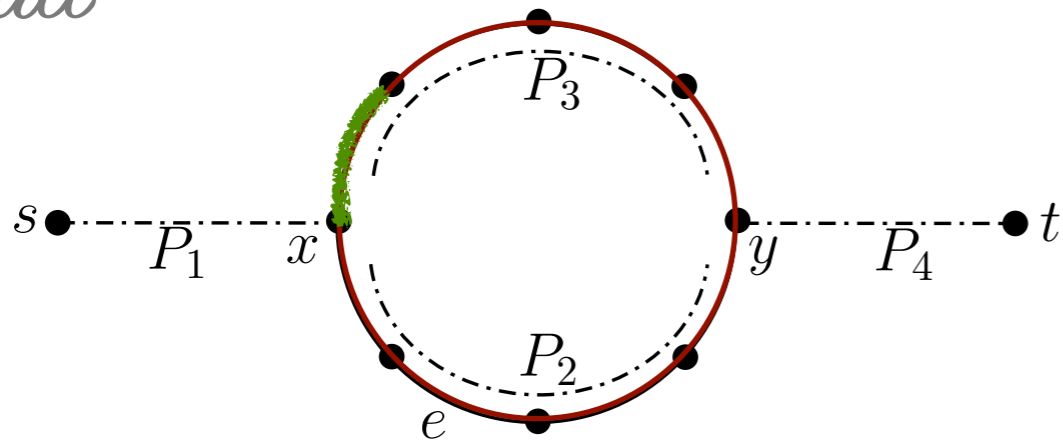


Each cycle must have a tracker

- ▶ An FVS might pick only a local  $s$  or  $t$
- ▶ But what if we use edges as trackers... ?
- ▶ An edge in a cycle can not be a local  $s$  or  $t$

# Tracking Paths .... but with Edges

*Recall*



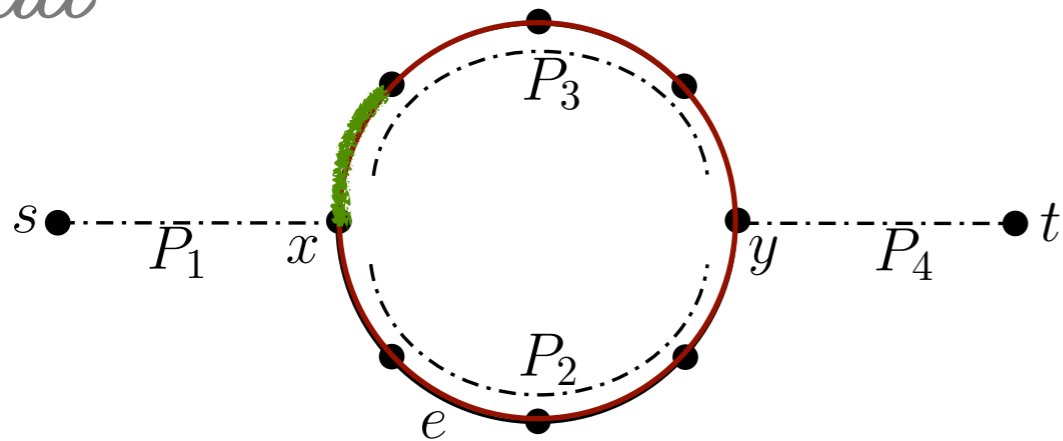
Each cycle must have a tracker

- ▶ An FVS might pick only a local  $s$  or  $t$
- ▶ But what if we use edges as trackers... ?
- ▶ An edge in a cycle can not be a local  $s$  or  $t$

Problem reduces to finding a minimum Feedback Edge Set

# Tracking Paths .... but with Edges

*Recall*



Each cycle must have a tracker

- ▶ An FVS might pick only a local  $s$  or  $t$
- ▶ But what if we use edges as trackers... ?
- ▶ An edge in a cycle can not be a local  $s$  or  $t$

Problem reduces to finding a minimum Feedback Edge Set

**Tracking Paths** is polynomial time solvable when trackers are edges



# Conclusion and Future Scope



# Conclusion and Future Scope

- *Approximation for general graphs*

# Conclusion and Future Scope

- *Approximation for general graphs*
- *Exploring more restricted graph classes*

# Conclusion and Future Scope

- *Approximation for general graphs*
- *Exploring more restricted graph classes*
- *Exploring graphs that are few vertices away from easy restricted cases*

# Conclusion and Future Scope

- *Approximation for general graphs*
- *Exploring more restricted graph classes*
- *Exploring graphs that are few vertices away from easy restricted cases*
- *Analysing the problem for general directed graphs*





*Thank  
You*