GROW 2022, Koper

# SEMI-PROPER INTERVAL GRAPHS

## AN IDEA IS COMING BACK TO KOPER. . .

Robert Scheffler

Brandenburg
University of Technology
Cottbus - Senftenberg

## ONCE UPON A TIME IN KOPER…

## ONCE UPON A TIME IN KOPER...

A **perfect elimination ordering (PEO)** of a graph

is a vertex ordering $(v_1, \ldots, v_n)$ such that

$v_i$ is simplicial in $G - \{v_{i+1}, \ldots, v_n\}$.

## ONCE UPON A TIME IN KOPER...

A **perfect elimination ordering (PEO)** of a graph

is a vertex ordering $(v_1, \ldots, v_n)$ such that

$v_i$ is simplicial in $G - \{v_{i+1}, \ldots, v_n\}$.

Let's look at **connected orderings**, i.e., $v_i v_{i+1} \in E(G)$ for any $i \in \{1, \ldots, n-1\}$.

## ONCE UPON A TIME IN KOPER...

A **perfect elimination ordering (PEO)** of a graph

is a vertex ordering $(v_1, \ldots, v_n)$ such that

$v_i$ is simplicial in $G - \{v_{i+1}, \ldots, v_n\}$.

Let's look at **connected orderings**, i.e., $v_i v_{i+1} \in E(G)$ for any $i \in \{1, \ldots, n-1\}$.

### Claim (of Koper)

*A graph has a* ***connected PEO*** *if and only if it is a connected* ***proper interval graph****.*

## IDEA OF “PROOF”

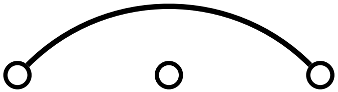
proper interval graphs have **proper interval orderings**.

## IDEA OF "PROOF"

proper interval graphs have **proper interval orderings**.

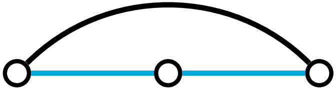
## IDEA OF “PROOF”

proper interval graphs have **proper interval orderings**.



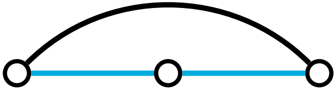
## IDEA OF “PROOF”

proper interval graphs have **proper interval orderings**.



## IDEA OF "PROOF"

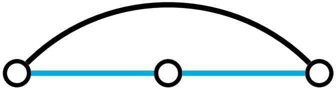
proper interval graphs have **proper interval orderings**.



Easy to show: Proper interval orderings of connected graphs are connected PEOs.

## IDEA OF "PROOF"

proper interval graphs have **proper interval orderings**.

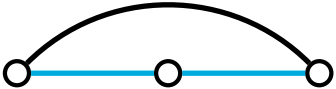


Easy to show: Proper interval orderings of connected graphs are connected PEOs.

For the other direction we consider the **forbidden subgraphs** of proper interval graphs:

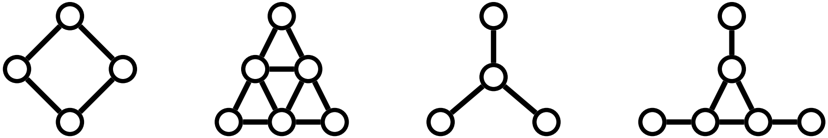
## IDEA OF “PROOF”

proper interval graphs have **proper interval orderings**.



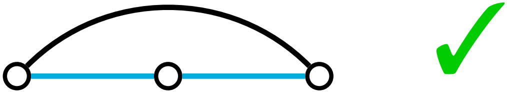
Easy to show: Proper interval orderings of connected graphs are connected PEOs.

For the other direction we consider the **forbidden subgraphs** of proper interval graphs:



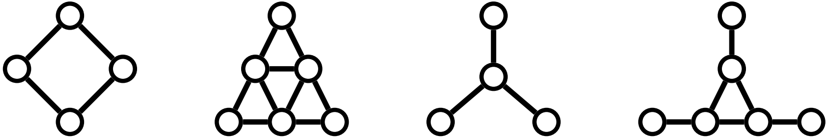
## IDEA OF “PROOF”

proper interval graphs have **proper interval orderings**.



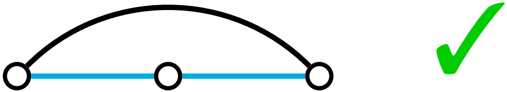
Easy to show: Proper interval orderings of connected graphs are connected PEOs.

For the other direction we consider the **forbidden subgraphs** of proper interval graphs:



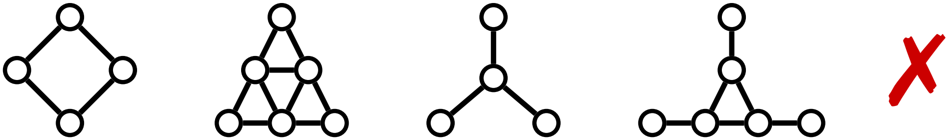
## IDEA OF “PROOF”

proper interval graphs have **proper interval orderings**.



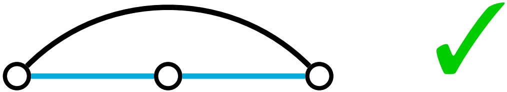
Easy to show: Proper interval orderings of connected graphs are connected PEOs.

For the other direction we consider the **forbidden subgraphs** of proper interval graphs:



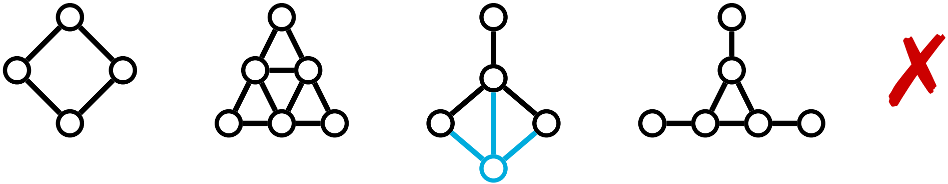
## IDEA OF “PROOF”

proper interval graphs have **proper interval orderings**.



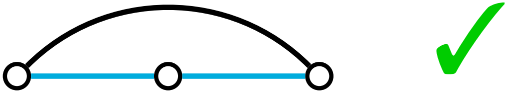
Easy to show: Proper interval orderings of connected graphs are connected PEOs.

For the other direction we consider the **forbidden subgraphs** of proper interval graphs:



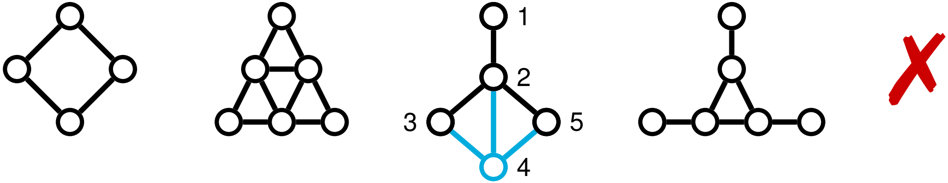
## IDEA OF “PROOF”

proper interval graphs have **proper interval orderings**.



Easy to show: Proper interval orderings of connected graphs are connected PEOs.

For the other direction we consider the **forbidden subgraphs** of proper interval graphs:



# WHAT ELSE. . . ?

## WHAT ELSE. . . ?

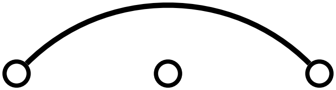
### Theorem

*A vertex ordering is a* ***connected PEO*** *if and only if it is a* ***connected interval ordering****.*

## WHAT ELSE. . . ?

### Theorem

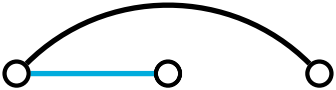
*A vertex ordering is a* ***connected PEO*** *if and only if it is a* ***connected interval ordering****.*



## WHAT ELSE. . . ?

### Theorem

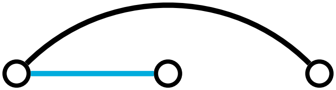
*A vertex ordering is a **connected PEO** if and only if it is a **connected interval ordering**.*



## WHAT ELSE. . . ?

### Theorem

*A vertex ordering is a* ***connected PEO*** *if and only if it is a* ***connected interval ordering****.*



### Corollary

*Graphs having a* ***connected PEO*** *are connected* ***interval graphs****.*

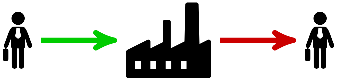
# INTERVAL MODEL

How does an **interval model** of a graph with a connected PEO look like?

## INTERVAL MODEL

How does an **interval model** of a graph with a connected PEO look like?

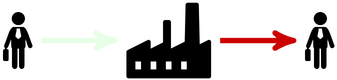
Consider personnel policies of **SP Inc.**



## INTERVAL MODEL

How does an **interval model** of a graph with a connected PEO look like?

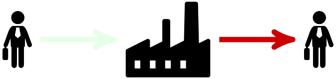
Consider personnel policies of **SP Inc.**



# INTERVAL MODEL

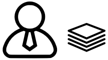
How does an **interval model** of a graph with a connected PEO look like?

Consider personnel policies of **SP Inc.**



employee needs

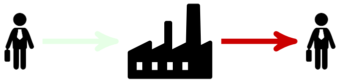
**replacement**



## INTERVAL MODEL

How does an **interval model** of a graph with a connected PEO look like?

Consider personnel policies of **SP Inc.**



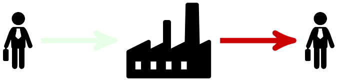
employee needs

**replacement**

# INTERVAL MODEL

How does an **interval model** of a graph with a connected PEO look like?

Consider personnel policies of **SP Inc.**



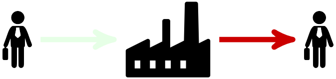
employee needs

**replacement**

## INTERVAL MODEL

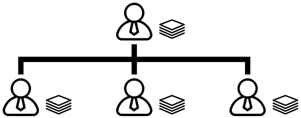
How does an **interval model** of a graph with a connected PEO look like?

Consider personnel policies of **SP Inc.**



employee needs **replacement**

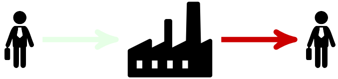
replacement must be **subordinate**



# INTERVAL MODEL

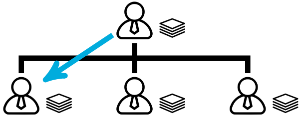
How does an **interval model** of a graph with a connected PEO look like?

Consider personnel policies of **SP Inc.**



employee needs **replacement**

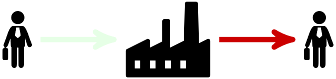
replacement must be **subordinate**



## INTERVAL MODEL

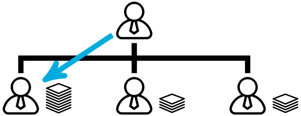
How does an **interval model** of a graph with a connected PEO look like?

Consider personnel policies of **SP Inc.**



employee needs **replacement**

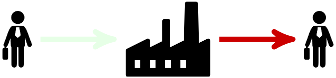
replacement must be **subordinate**



# INTERVAL MODEL

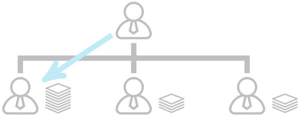
How does an **interval model** of a graph with a connected PEO look like?

Consider personnel policies of **SP Inc.**



employee needs **replacement**

replacement must be **subordinate**

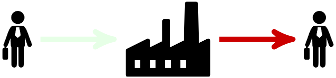


subordinates were **hired later**

## INTERVAL MODEL

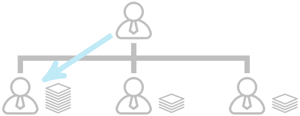
How does an **interval model** of a graph with a connected PEO look like?

Consider personnel policies of **SP Inc.**

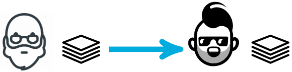


employee needs **replacement**

replacement must be **subordinate**



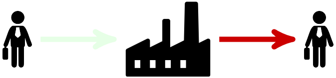
subordinates were **hired later**



# INTERVAL MODEL

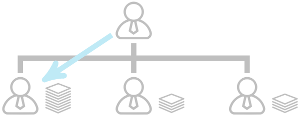
How does an **interval model** of a graph with a connected PEO look like?

Consider personnel policies of **SP Inc.**

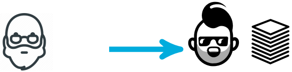


employee needs **replacement**

replacement must be **subordinate**



subordinates were **hired later**



# INTERVAL MODEL

How does an **interval model** of a graph with a connected PEO look like?

Consider personnel policies of **SP Inc.**

employee n

**replacem**

an employee can

**replace several other**

bordinates were

**hired later**

## INTERVAL MODEL

How does an **interval model** of a graph with a connected PEO look like?

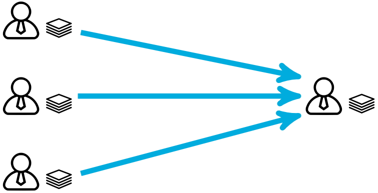
Consider personnel policies of **SP Inc.**

employee n

replacem

an employee can

**replace several other**



ordinates were

**hired later**

## INTERVAL MODEL

How does an **interval model** of a graph with a connected PEO look like?

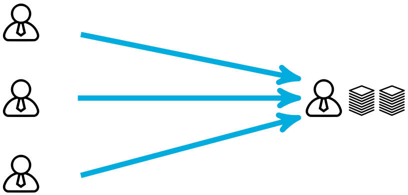
Consider personnel policies of **SP Inc.**

employee n

**replacem**

an employee can

**replace several other**



ordinates were

**hired later**

# INTERVAL MODEL

## Definition

An interval model $\mathcal{I} = \{I_v\}_{v \in V(G)}$ of a graph $G$ is called **semi-proper**

## INTERVAL MODEL

### Definition

An interval model $\mathcal{I} = \{I_v\}_{v \in V(G)}$ of a graph $G$ is called **semi-proper**

if for every vertex $v \in V(G)$ except from one

# INTERVAL MODEL

## Definition

An interval model $\mathcal{I} = \{I_v\}_{v \in V(G)}$ of a graph $G$ is called **semi-proper**

if for every vertex $v \in V(G)$ except from one

there is a vertex $w \in V(G)$ with

### Definition

An interval model $\mathcal{I} = \{I_v\}_{v \in V(G)}$ of a graph $G$ is called **semi-proper**

if for every vertex $v \in V(G)$ except from one

there is a vertex $w \in V(G)$ with

$\ell(I_v) \leq \ell(I_w) \leq r(I_v) < r(I_w)$.

## SEMI-PROPER INTERVAL GRAPHS

### Theorem

*The following conditions are equivalent for a graph G:*

1. *G has a **connected PEO**.*
2. *G has a **semi-proper interval model**.*

## Theorem

*The following conditions are equivalent for a graph $G$:*

1. *$G$ has a* ***connected PEO****.*
2. *$G$ has a* ***semi-proper interval model****.*

## Definition

A consecutive clique ordering $\Phi = (C_1, \ldots, C_k)$ of $G$ is a **semi-proper clique ordering**

## SEMI-PROPER INTERVAL GRAPHS

### Theorem

*The following conditions are equivalent for a graph $G$:*

1. *$G$ has a **connected PEO**.*
2. *$G$ has a **semi-proper interval model**.*

### Definition

A consecutive clique ordering $\Phi = (C_1, \ldots, C_k)$ of $G$ is a **semi-proper clique ordering**

if for any $i$ with $1 < i < k$ there is a vertex $v \in (C_i \cap C_{i+1}) \setminus C_{i-1}$.

## SEMI-PROPER INTERVAL GRAPHS

### Theorem

*The following conditions are equivalent for a graph $G$:*

1. *$G$ has a **connected PEO**.*
2. *$G$ has a **semi-proper interval model**.*
3. *$G$ has a **semi-proper clique ordering**.*

### Definition

A consecutive clique ordering $\Phi = (C_1, \ldots, C_k)$ of $G$ is a **semi-proper clique ordering**

if for any $i$ with $1 < i < k$ there is a vertex $v \in (C_i \cap C_{i+1}) \setminus C_{i-1}$.

# RECOGNITION

## Idea:

## RECOGNITION

**Idea:**

1. compute vertex ordering that is a **connected PEO** if such an ordering **exists**.

# RECOGNITION

## Idea:

1. compute vertex ordering that is a **connected PEO** if such an ordering **exists**.

2. **check** in linear time **whether** this ordering is a **connected PEO**.

## RECOGNITION

### Idea:

1. compute vertex ordering that is a **connected PEO** if such an ordering **exists**.

2. **check** in linear time **whether** this ordering is a **connected PEO**.

We use **multi-sweep** graph searches similar to:

## RECOGNITION

### Idea:

1. compute vertex ordering that is a **connected PEO** if such an ordering **exists**.

2. **check** in linear time **whether** this ordering is a **connected PEO**.

We use **multi-sweep** graph searches similar to:

- 1 sweep of LBFS gives PEO **[Rose, Tarjan, Lueker, 1976]**

## RECOGNITION

**Idea:**

1. compute vertex ordering that is a **connected PEO** if such an ordering **exists**.

2. **check** in linear time **whether** this ordering is a **connected PEO**.

We use **multi-sweep** graph searches similar to:

- 1 sweep of LBFS gives PEO **[Rose, Tarjan, Lueker, 1976]**
- *n* sweeps of LBFS give interval ordering **[Dusart, Habib, 2017]**

## RECOGNITION

### Idea:

1. compute vertex ordering that is a **connected PEO** if such an ordering **exists**.
2. **check** in linear time **whether** this ordering is a **connected PEO**.

We use **multi-sweep** graph searches similar to:

- 1 sweep of LBFS gives PEO **[Rose, Tarjan, Lueker, 1976]**
- $n$ sweeps of LBFS give interval ordering **[Dusart, Habib, 2017]**
- O(1) sweeps of LBFS (with special tie-breaking) give interval ordering **[Corneil, Olariu, Stewart, 2010] [Li, Wu, 2014] [Cao, 2021]**

# RECOGNITION

**Idea:**

1. compute vertex ordering that is a **connected PEO** if such an ordering **exists**.
2. **check** in linear time **whether** this ordering is a **connected PEO**.

We use **multi-sweep** graph searches similar to:

- 1 sweep of LBFS gives PEO **[Rose, Tarjan, Lueker, 1976]**
- *n* sweeps of LBFS give interval ordering **[Dusart, Habib, 2017]**
- O(1) sweeps of LBFS (with special tie-breaking) give interval ordering **[Corneil, Olariu, Stewart, 2010] [Li, Wu, 2014] [Cao, 2021]**
- 3 sweeps of LBFS give proper interval ordering **[Corneil, 2004]**

# GRAPH SEARCHES

## GRAPH SEARCHES

### Inclusion property:

If set of **visited neighbors** of vertex $x$ is **strict subset** of the visited neighbors of vertex $y$,

then $x$ **cannot** be visited next.

# GRAPH SEARCHES

## Inclusion property:

If set of **visited neighbors** of vertex $x$ is **strict subset** of the visited neighbors of vertex $y$,

then $x$ **cannot** be visited next.

## Tie property:

If sets of **visited neighbors** of vertices $x$ and $y$ are **equal**,

then **both or none** of them can be visited next.

## GRAPH SEARCHES

### Inclusion property:

If set of **visited neighbors** of vertex $x$ is **strict subset** of the visited neighbors of vertex $y$,

then $x$ **cannot** be visited next.

### Tie property:

If sets of **visited neighbors** of vertices $x$ and $y$ are **equal**,

then **both or none** of them can be visited next.

### Strictly monotone graph search:

Has inclusion property and tie property.

## GRAPH SEARCHES

**Inclusion property:**

If set of **visited neighbors** of vertex $x$ is **strict subset** of the visited neighbors of vertex $y$,

then $x$ **cannot** be visited next.

**Tie property:**

If sets of **visited neighbors** of vertices $x$ and $y$ are **equal**,

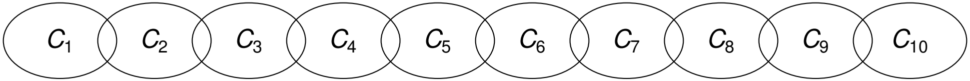
then **both or none** of them can be visited next.

**Strictly monotone graph search:**

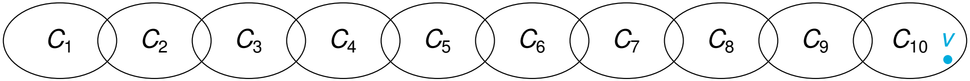
Has inclusion property and tie property.

**Examples** are LBFS, LDFS, MCS, and MNS.

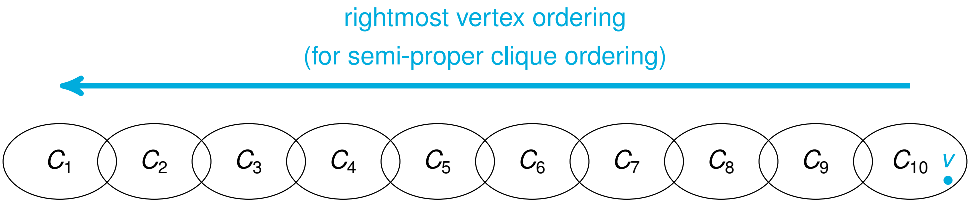
## THE ALGORITHM



## THE ALGORITHM



## THE ALGORITHM

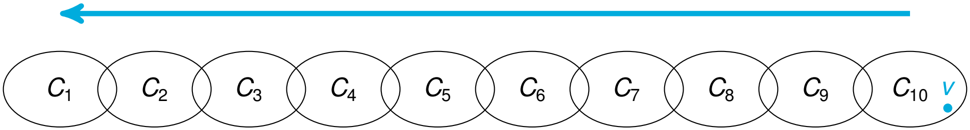


## THE ALGORITHM

if $r_\Phi(x) < r_\Phi(y)$, then $y \prec_\sigma x$

rightmost vertex ordering

(for semi-proper clique ordering)

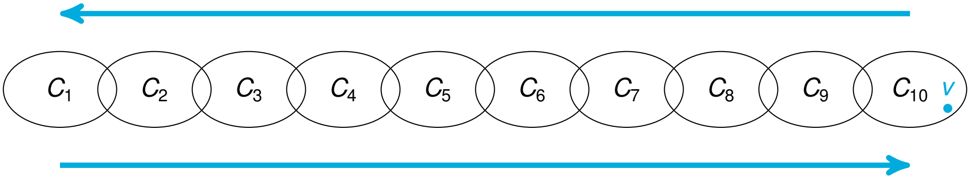


## THE ALGORITHM

if $r_\Phi(x) < r_\Phi(y)$, then $y \prec_\sigma x$

rightmost vertex ordering

(for semi-proper clique ordering)




lexicographic vertex ordering

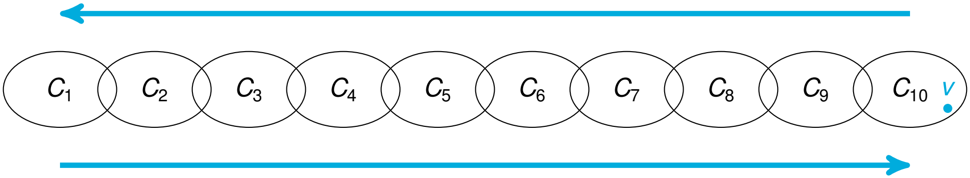
(for any consecutive clique ordering)

# THE ALGORITHM

if $r_\Phi(x) < r_\Phi(y)$, then $y \prec_\sigma x$

rightmost vertex ordering

(for semi-proper clique ordering)




lexicographic vertex ordering

(for any consecutive clique ordering)

if $(\ell_\Phi(x), r_\Phi(x)) < (\ell_\Phi(y), r_\Phi(y))$, then $x \prec_\sigma y$

Lemma

*A **lexicographic vertex ordering** of a semi-proper clique ordering is a **connected PEO**.*

## THE ALGORITHM

### Lemma

*A **lexicographic vertex ordering** of a semi-proper clique ordering is a **connected PEO**.*

### Theorem

*A **2-sweep** of any strictly monotone graph search*

## THE ALGORITHM

### Lemma

*A **lexicographic vertex ordering** of a semi-proper clique ordering is a **connected PEO**.*

### Theorem

*A **2-sweep** of any strictly monotone graph search*

*starting in **end vertex** $v$ of a connected PEO*

# THE ALGORITHM

## Lemma

*A **lexicographic vertex ordering** of a semi-proper clique ordering is a **connected PEO**.*

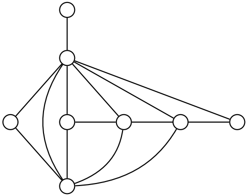
## Theorem

*A **2-sweep** of any strictly monotone graph search*

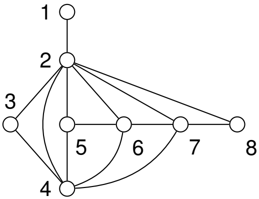
*starting in **end vertex** $v$ of a connected PEO*

*results in a **connected PEO** ending with $v$.*

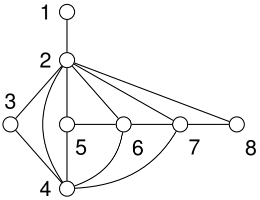
# HOW TO FIND THE END VERTEX?

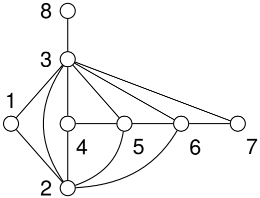


# HOW TO FIND THE END VERTEX?

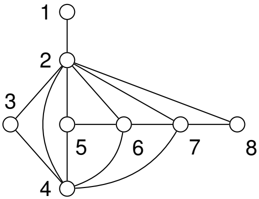


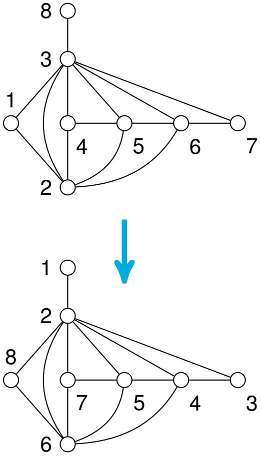
## HOW TO FIND THE END VERTEX?



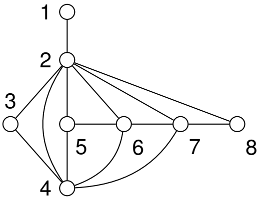



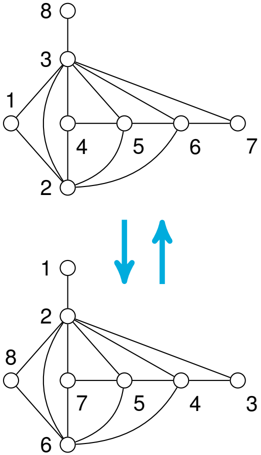
# HOW TO FIND THE END VERTEX?



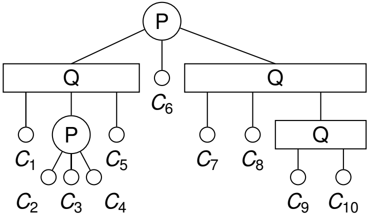
# HOW TO FIND THE END VERTEX?



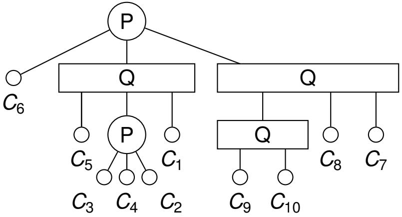



# PQ-TREES

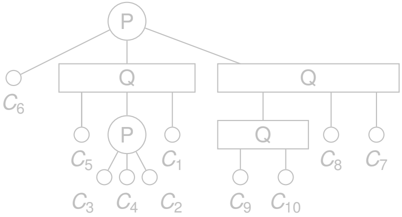
## PQ-TREES



## PQ-TREES

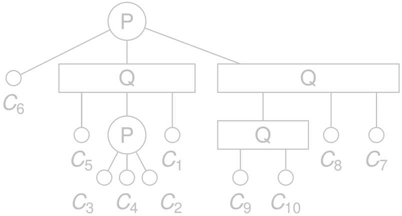


# PQ-TREES

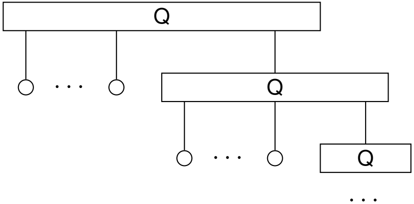



**Semi-proper clique ordering**

# PQ-TREES




## Semi-proper clique ordering



# Corollary

*Every semi-proper interval graph has **at most two** semi-proper clique orderings.*

## Corollary

*Every semi-proper interval graph has* ***at most two*** *semi-proper clique orderings.*

## Theorem

*There is a* ***linear-time recognition algorithm*** *for semi-proper interval graphs.*

## Corollary

*Every semi-proper interval graph has **at most two** semi-proper clique orderings.*

## Theorem

*There is a **linear-time recognition algorithm** for semi-proper interval graphs.*

1. compute **PQ-tree**

## Corollary

*Every semi-proper interval graph has **at most two** semi-proper clique orderings.*

## Theorem

*There is a **linear-time recognition algorithm** for semi-proper interval graphs.*

1. compute **PQ-tree**
2. **check** whether the tree can have the **right structure**

## Corollary

*Every semi-proper interval graph has **at most two** semi-proper clique orderings.*

## Theorem

*There is a **linear-time recognition algorithm** for semi-proper interval graphs.*

1. compute **PQ-tree**
2. **check** whether the tree can have the **right structure**
3. **rearrange** the tree such that it has the **right structure**

## Corollary

*Every semi-proper interval graph has* ***at most two*** *semi-proper clique orderings.*

## Theorem

*There is a* ***linear-time recognition algorithm*** *for semi-proper interval graphs.*

1. compute **PQ-tree**
2. **check** whether the tree can have the **right structure**
3. **rearrange** the tree such that it has the **right structure**
4. find the two **semi-proper clique orderings** and two respective **end vertices**

## Corollary

*Every semi-proper interval graph has* ***at most two*** *semi-proper clique orderings.*

## Theorem

*There is a* ***linear-time recognition algorithm*** *for semi-proper interval graphs.*

1. compute **PQ-tree**
2. **check** whether the tree can have the **right structure**
3. **rearrange** the tree such that it has the **right structure**
4. find the two **semi-proper clique orderings** and two respective **end vertices**
5. make a **2-sweep** of LBFS starting with **both end vertices**

## Corollary

*Every semi-proper interval graph has **at most two** semi-proper clique orderings.*

## Theorem

*There is a **linear-time recognition algorithm** for semi-proper interval graphs.*

1. compute **PQ-tree**
2. **check** whether the tree can have the **right structure**
3. **rearrange** the tree such that it has the **right structure**
4. find the two **semi-proper clique orderings** and two respective **end vertices**
5. make a **2-sweep** of LBFS starting with **both end vertices**
6. **check** whether one of these vertex orderings is a **connected PEO**

## Corollary

***Semi-proper interval graphs*** *with* $k$ *maximal cliques have* $\leq \mathbf{2^{k-1}}$ *consecutive clique orderings.*

*This bound is* ***tight****.*

## Corollary

***Semi-proper interval graphs*** *with* $k$ *maximal cliques have* $\leq 2^{k-1}$ *consecutive clique orderings. This bound is* ***tight****.*

## Observation

*Connected* ***interval graphs*** *with* $k$ *maximal cliques can have* $k!$ *consecutive clique orderings.*

## Corollary

***Semi-proper interval graphs*** *with* $k$ *maximal cliques have* $\leq 2^{k-1}$ *consecutive clique orderings. This bound is* ***tight****.*

## Observation

*Connected* ***interval graphs*** *with* $k$ *maximal cliques can have* $k!$ *consecutive clique orderings.*

## Observation

*Connected* ***proper interval graphs*** *have* ***at most two*** *consecutive clique orderings.*

# HAMILTONIAN PATHS AND CYCLES

### Theorem (Bertossi, 1983; Chen et al, 1997; Broersma et al, 2015)

*Let $G$ be a proper interval graph.*

1. *$G$ has a Hamiltonian path if and only if $G$ is connected.*
2. *$G$ has a Hamiltonian cycle if and only if $G$ is 2-connected.*
3. *$G$ is $k$-Hamilton-connected if and only if $G$ is $(k+3)$-connected.*

## Theorem (S, 2022)

*Let $G$ be a **semi-proper** interval graph.*

1. *$G$ has a Hamiltonian path if and only if $G$ is connected.*
2. *$G$ has a Hamiltonian cycle if and only if $G$ is 2-connected.*
3. *$G$ is $k$-Hamilton-connected if and only if $G$ is $(k+3)$-connected.*

## HAMILTONIAN PATHS AND CYCLES

### Theorem (S, 2022)

*Let $G$ be a* ***semi-proper*** *interval graph.*

1. *$G$ has a Hamiltonian path if and only if $G$ is connected.*
2. *$G$ has a Hamiltonian cycle if and only if $G$ is 2-connected.*
3. *$G$ is $k$-Hamilton-connected if and only if $G$ is $(k + 3)$-connected.*

### Corollary

*The* ***longest path*** *and the* ***longest cycle*** *of an semi-proper interval graph can be found in* ***linear time****.*

## FURTHER RESEARCH

- complexity of problems that are hard or open on interval graphs

## FURTHER RESEARCH

- complexity of problems that are hard or open on interval graphs

  **max-cut**

  (hard on interval, open on proper interval)

## FURTHER RESEARCH

- complexity of problems that are hard or open on interval graphs

  **max-cut**

  (hard on interval, open on proper interval)

  **geodetic number** and **optimal linear arrangement**

  (hard on interval, polynomial on proper interval)

## FURTHER RESEARCH

- complexity of problems that are hard or open on interval graphs

  **max-cut**

  (hard on interval, open on proper interval)

  **geodetic number** and **optimal linear arrangement**

  (hard on interval, polynomial on proper interval)

  **x-y-Hamiltonian path problem**

  (open on interval, polynomial on proper interval)

# FURTHER RESEARCH

- complexity of problems that are hard or open on interval graphs

  **max-cut**

  (hard on interval, open on proper interval)

  **geodetic number** and **optimal linear arrangement**

  (hard on interval, polynomial on proper interval)

  **x-y-Hamiltonian path problem**

  (open on interval, polynomial on proper interval)

**Thanks. . . !**