

Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm

Theresa Meier, 13.01.2023

How to find conditional independence (CI) relationships in a DAG?

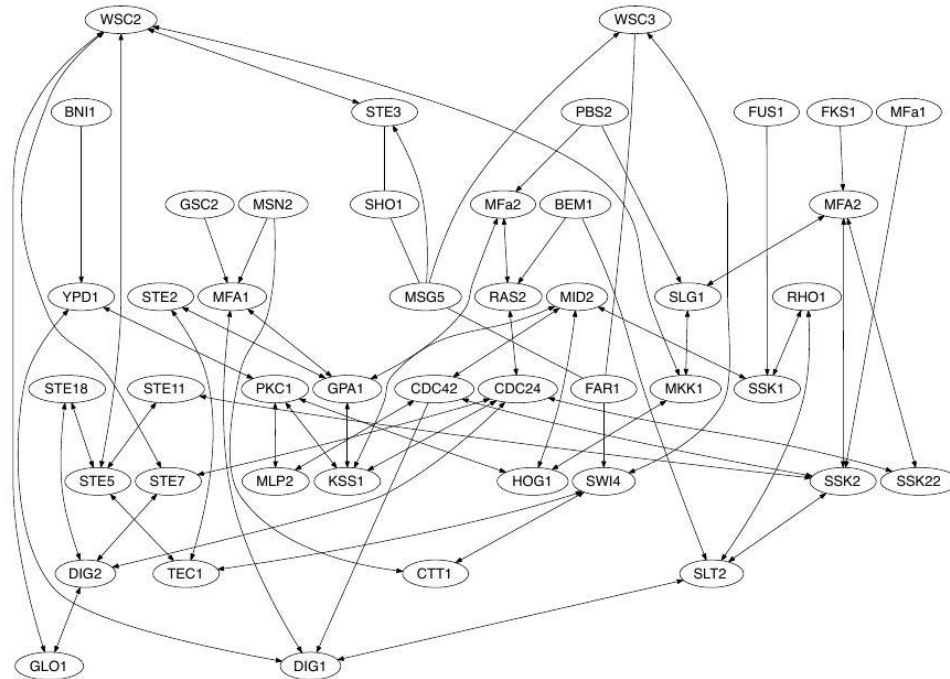


Figure 1: Output of the rank PC-algorithm for yeast gene expression data (Harris and Drten, 2013).

Agenda

- 1) Basic Definitions
- 2) Introduction to the PC-Algorithm
- 3) Limitations of the PC-Algorithm
- 4) The stable PC-Algorithm
- 5) The parallel PC-Algorithm
- 6) Simulation Study in R
- 7) Final Notes and Discussion

Agenda

- 1) **Basic Definitions**
- 2) Introduction to the PC-Algorithm
- 3) Limitations of the PC-Algorithm
- 4) The stable PC-Algorithm
- 5) The parallel PC-Algorithm
- 6) Simulation Study in R
- 7) Final Notes and Discussion

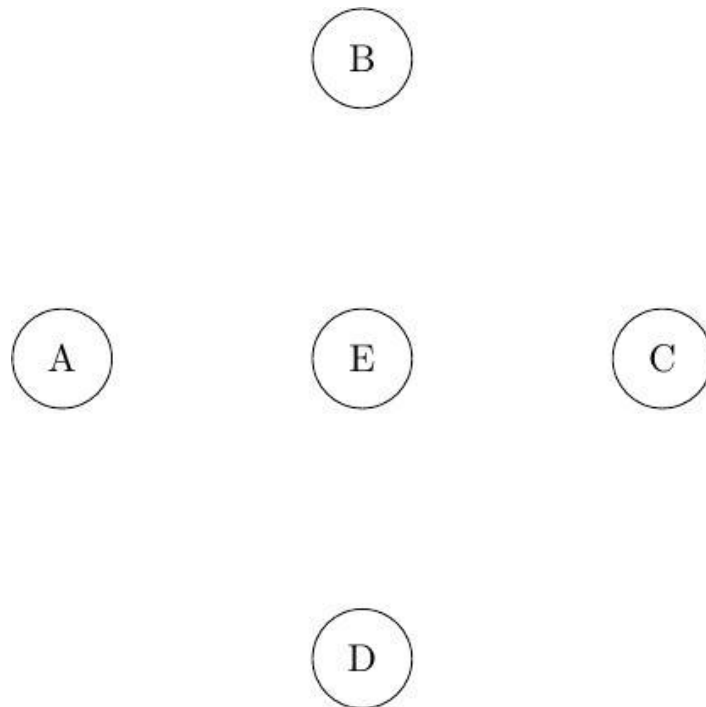
Directed Acyclic Graphs (DAGs)

All following definitions are taken from Harris and Drton (2013), Le et al. (2014) and Koller and Friedman (2009).

Directed Acyclic Graphs (DAGs)

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a graph consisting of:

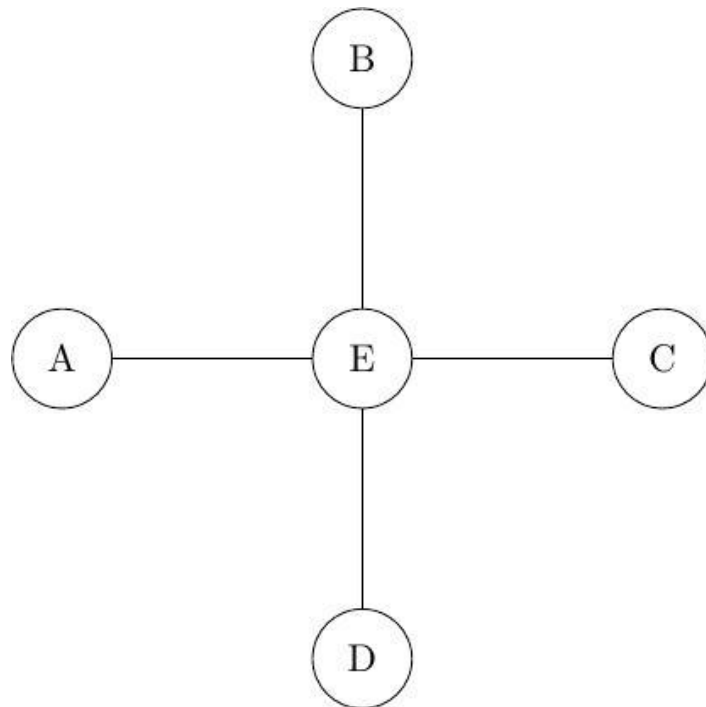
- set of nodes \mathcal{N}



Directed Acyclic Graphs (DAGs)

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a graph consisting of:

- set of nodes \mathcal{N}
- set of edges $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$

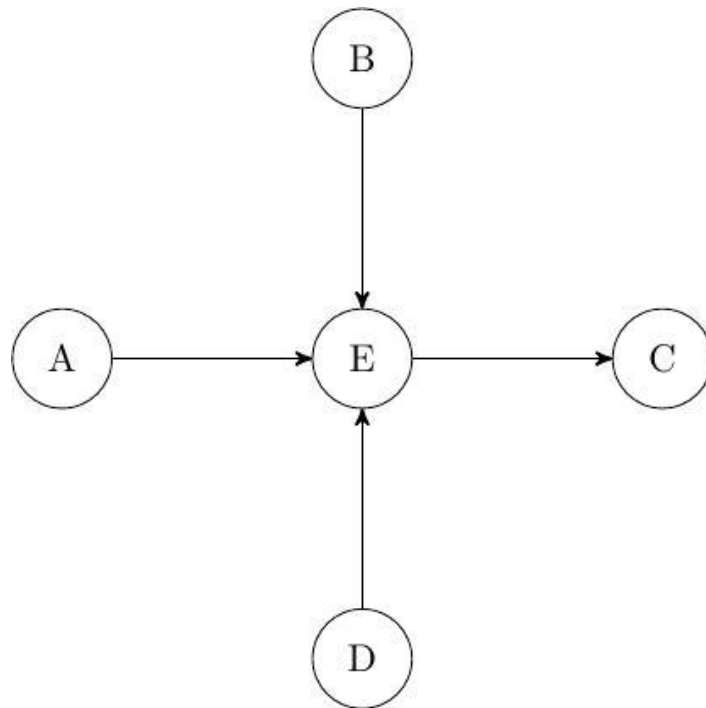


Directed Acyclic Graphs (DAGs)

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a graph consisting of:

- set of nodes \mathcal{N}
- set of edges $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$

\mathcal{G} is a **directed acyclic graph** (DAG) if \mathcal{G} contains only directed edges and has no directed cycles.



Directed Acyclic Graphs (DAGs)

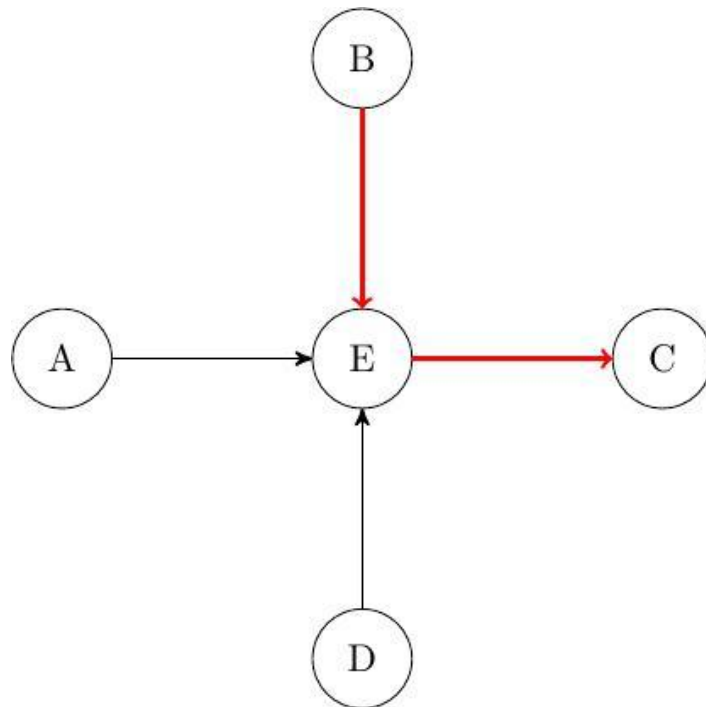
Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a graph consisting of:

- set of nodes \mathcal{N}
- set of edges $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$

\mathcal{G} is a **directed acyclic graph** (DAG) if \mathcal{G} contains only directed edges and has no directed cycles.

A **path** from node A_0 to node A_n in DAG \mathcal{G} is a sequence of distinct nodes (A_0, \dots, A_n) such that for all $1 \leq k \leq n$ either

$$A_{k-1} \rightarrow A_k \in \mathcal{E} \text{ or } A_k \rightarrow A_{k-1} \in \mathcal{E}.$$



Directed Acyclic Graphs (DAGs)

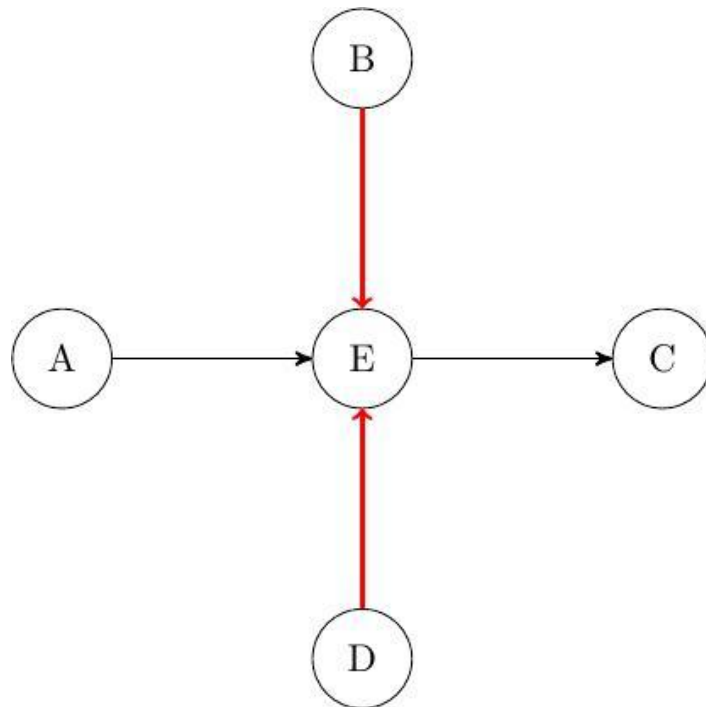
Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a graph consisting of:

- set of nodes \mathcal{N}
- set of edges $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$

\mathcal{G} is a **directed acyclic graph** (DAG) if \mathcal{G} contains only directed edges and has no directed cycles.

A **path** from node A_0 to node A_n in DAG \mathcal{G} is a sequence of distinct nodes (A_0, \dots, A_n) such that for all $1 \leq k \leq n$ either

$$A_{k-1} \rightarrow A_k \in \mathcal{E} \text{ or } A_k \rightarrow A_{k-1} \in \mathcal{E}.$$

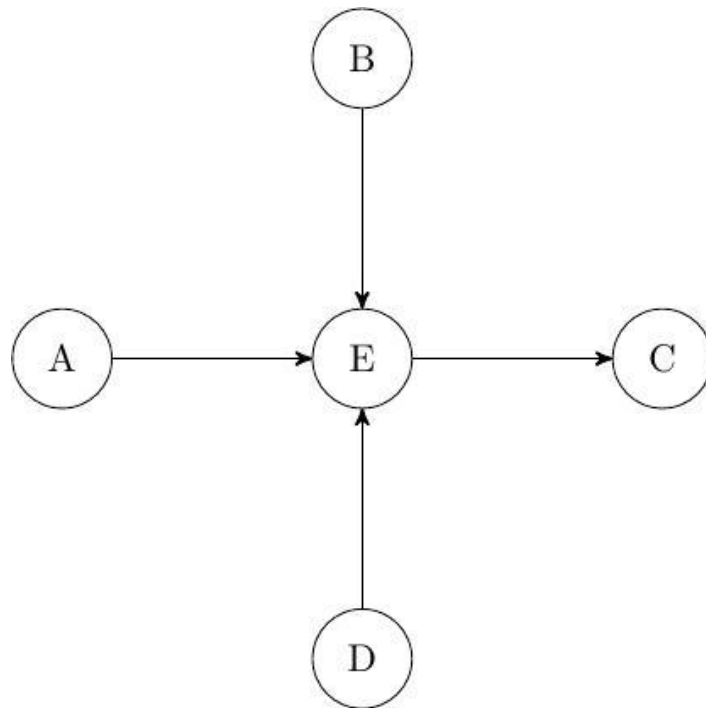


Directed Acyclic Graphs (DAGs)

A **collider** or **v-structure** is a triple of nodes $(A, B, C) \in \mathcal{N}^3$ that induces the subgraph

$$A \rightarrow B \leftarrow C$$

where A and C are not adjacent (i.e. there is no edge between A and C).

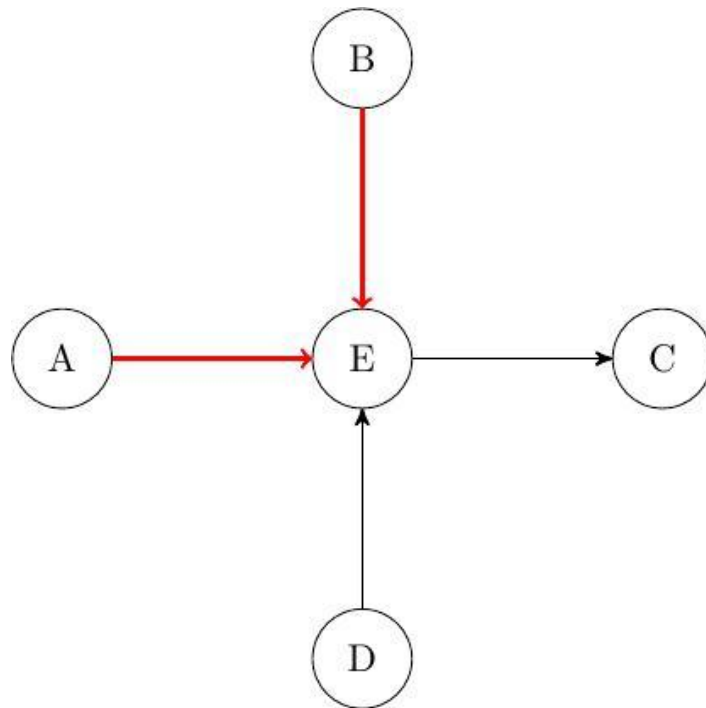


Directed Acyclic Graphs (DAGs)

A **collider** or **v-structure** is a triple of nodes $(A, B, C) \in \mathcal{N}^3$ that induces the subgraph

$$A \rightarrow B \leftarrow C$$

where A and C are not adjacent (i.e. there is no edge between A and C).



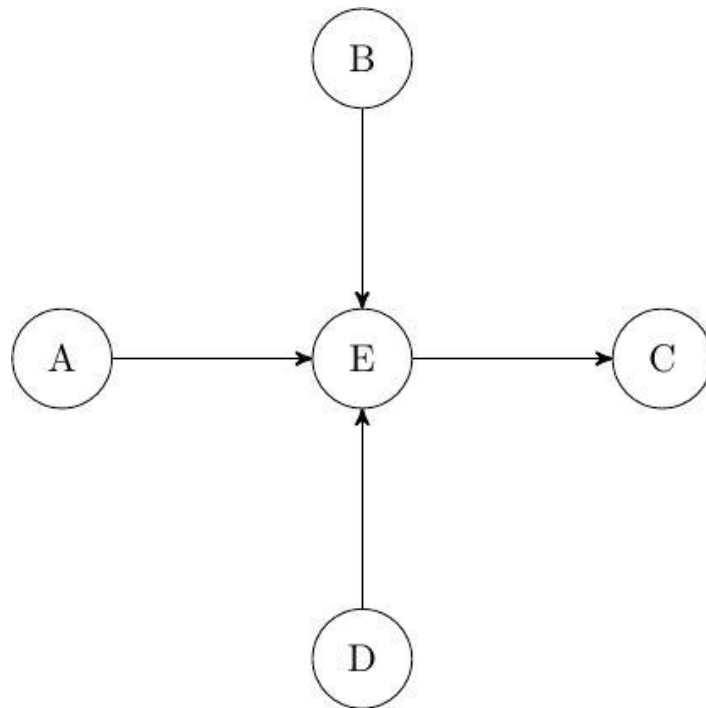
Directed Acyclic Graphs (DAGs)

A **collider** or **v-structure** is a triple of nodes $(A, B, C) \in \mathcal{N}^3$ that induces the subgraph

$$A \rightarrow B \leftarrow C$$

where A and C are not adjacent (i.e. there is no edge between A and C).

The **skeleton** of a DAG is the undirected graph obtained by converting each directed edge into an undirected edge.



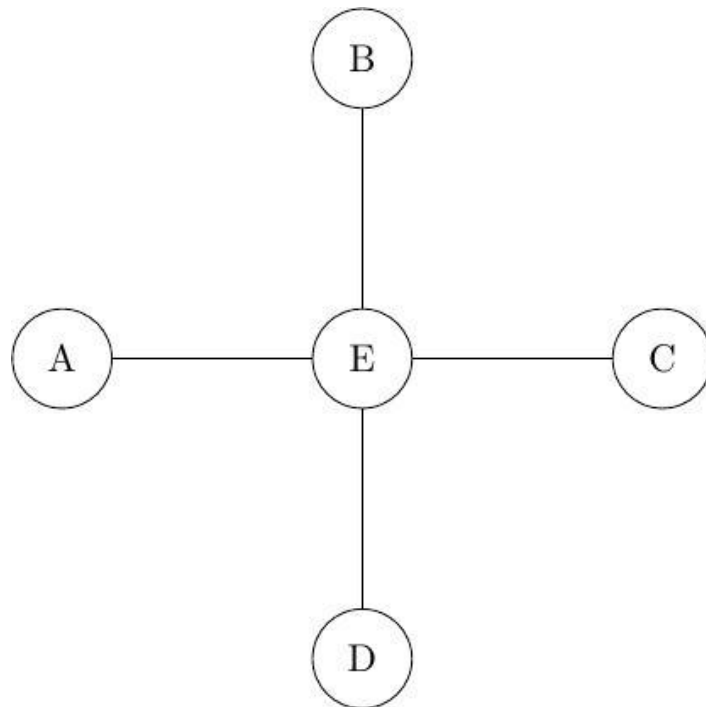
Directed Acyclic Graphs (DAGs)

A **collider** or **v-structure** is a triple of nodes $(A, B, C) \in \mathcal{N}^3$ that induces the subgraph

$$A \rightarrow B \leftarrow C$$

where A and C are not adjacent (i.e. there is no edge between A and C).

The **skeleton** of a DAG is the undirected graph obtained by converting each directed edge into an undirected edge.



d-Separation

Let $A, B \in \mathcal{N}$, $A \neq B$, and $S \subseteq \mathcal{N}$ a set of nodes not containing A and B .

A and B are **d-separated** given S if and only if there exists no undirected path p between A and B such that

- every collider on p has a descendant in S or is in S itself, and
- no other node on p is in S .

Otherwise A and B are called **d-connected**.

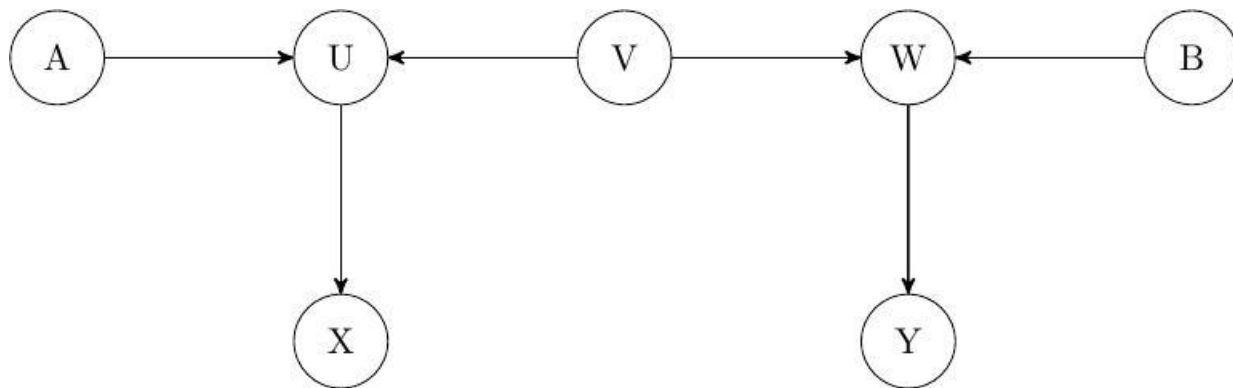
d-Separation

Let $A, B \in \mathcal{N}$, $A \neq B$, and $S \subseteq \mathcal{N}$ a set of nodes not containing A and B .

A and B are **d-separated** given S if and only if there exists no undirected path p between A and B such that

- every collider on p has a descendant in S or is in S itself, and
- no other node on p is in S .

Otherwise A and B are called **d-connected**.



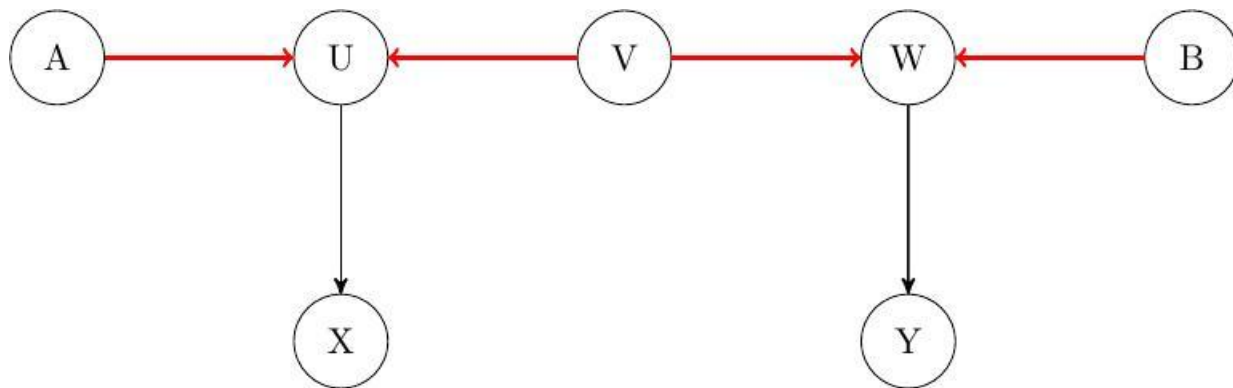
d-Separation

Let $A, B \in \mathcal{N}$, $A \neq B$, and $S \subseteq \mathcal{N}$ a set of nodes not containing A and B .

A and B are **d-separated** given S if and only if there exists no undirected path p between A and B such that

- every collider on p has a descendant in S or is in S itself, and
- no other node on p is in S .

Otherwise A and B are called **d-connected**.



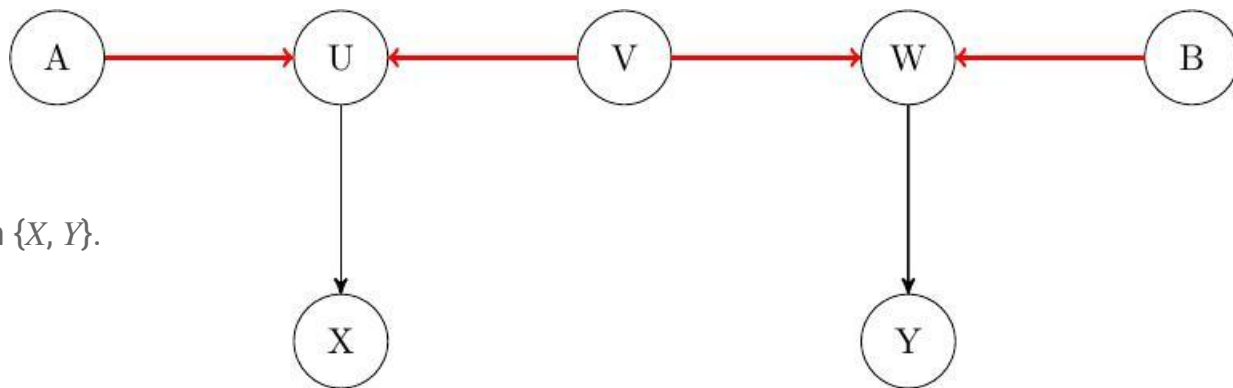
d-Separation

Let $A, B \in \mathcal{N}$, $A \neq B$, and $S \subseteq \mathcal{N}$ a set of nodes not containing A and B .

A and B are **d-separated** given S if and only if there exists no undirected path p between A and B such that

- every collider on p has a descendant in S or is in S itself, and
- no other node on p is in S .

Otherwise A and B are called **d-connected**.



A and B are **d-connected** given $\{X, Y\}$.

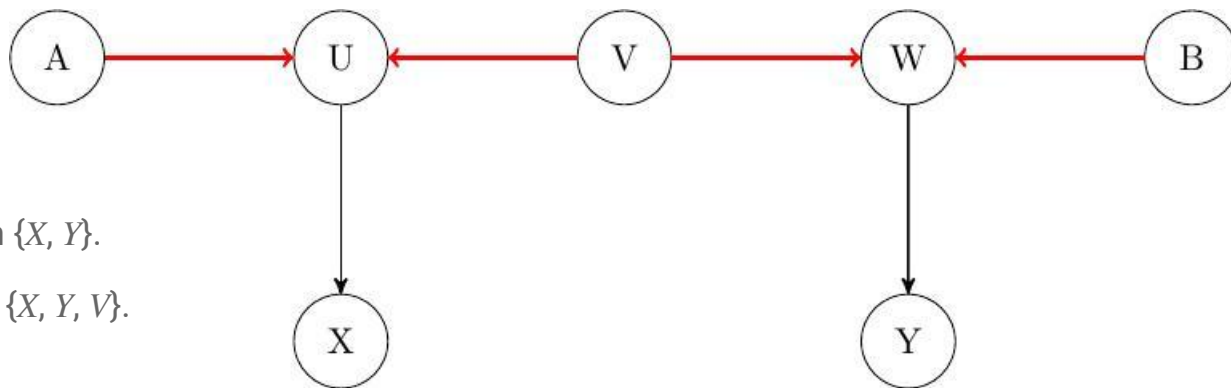
d-Separation

Let $A, B \in \mathcal{N}$, $A \neq B$, and $S \subseteq \mathcal{N}$ a set of nodes not containing A and B .

A and B are **d-separated** given S if and only if there exists no undirected path p between A and B such that

- every collider on p has a descendant in S or is in S itself, and
- no other node on p is in S .

Otherwise A and B are called **d-connected**.



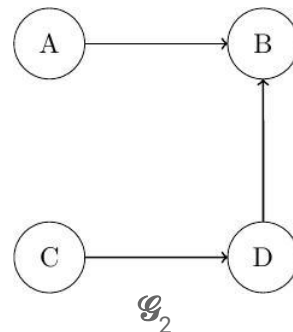
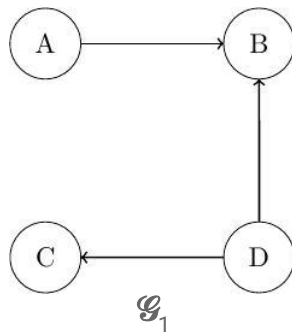
A and B are **d-connected** given $\{X, Y\}$.

A and B are **d-separated** given $\{X, Y, V\}$.

Markov equivalence class

Two DAGs with the same node set \mathcal{N} are called **Markov equivalent** if and only if both share the same skeleton and the same colliders.

→ partitions space of DAGs into **equivalence classes** where all members of an equivalence class encode the same statistical model, i. e. contain the same conditional independence information and the same d-separation relations



Completed partially directed acyclic graph (CPDAG)

Let $[\mathcal{G}]$ be the Markov equivalence class of DAG $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and define the edge set

$$[\mathcal{E}] = \bigcup_{\mathcal{H} \in [\mathcal{G}]} \mathcal{E}(\mathcal{H}),$$

where $\mathcal{E}(\mathcal{H})$ denotes the set of edges of a DAG \mathcal{H} . That is, $(A, B) \in [\mathcal{E}]$ if there exists a DAG $\mathcal{H} \in [\mathcal{G}]$ with the edge $A \rightarrow B$ in its edge set.

The graph $\mathcal{C}(\mathcal{G}) = (\mathcal{N}, [\mathcal{E}])$ is called the **completed partially directed acyclic graph (CPDAG)** and consists of both directed and undirected edges.

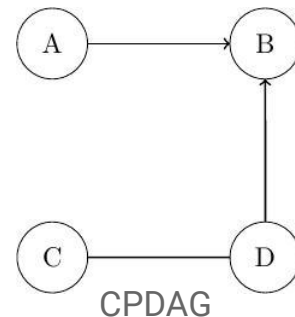
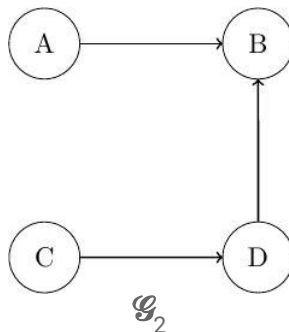
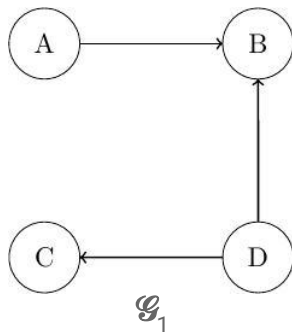
Completed partially directed acyclic graph (CPDAG)

Let $[\mathcal{G}]$ be the Markov equivalence class of DAG $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and define the edge set

$$[\mathcal{E}] = \bigcup_{\mathcal{H} \in [\mathcal{G}]} \mathcal{E}(\mathcal{H}),$$

where $\mathcal{E}(\mathcal{H})$ denotes the set of edges of a DAG \mathcal{H} . That is, $(A, B) \in [\mathcal{E}]$ if there exists a DAG $\mathcal{H} \in [\mathcal{G}]$ with the edge $A \rightarrow B$ in its edge set.

The graph $\mathcal{C}(\mathcal{G}) = (\mathcal{N}, [\mathcal{E}])$ is called the **completed partially directed acyclic graph (CPDAG)** and consists of both directed and undirected edges.



Agenda

- 1) Basic Definitions
- 2) Introduction to the PC-Algorithm**
- 3) Limitations of the PC-Algorithm
- 4) The stable PC-Algorithm
- 5) The parallel PC-Algorithm
- 6) Simulation Study in R
- 7) Final Notes and Discussion

History

- Invented by **Peter Spirtes** and **Clark Glymour** in 1991 as a more efficient alternative to the SGS algorithm
- Many improvements available (stable PC, parallel PC etc.)
- Implementations in R:
 - package *pcalg* by Markus Kalisch
 - package *ParallelPC* by Thuc Duy Le, Tao Hoang, Shu Hu, and Liang Zhang



Image sources:

<https://www.cmu.edu/dietrich/philosophy/people/faculty/spirtes.html> (left),

<https://www.cmu.edu/dietrich/philosophy/people/emeritus/glymour.html> (right)

Theoretical foundation

Theorem 1 (Sprites et al., 2000)

If nodes A and B are not adjacent in a DAG \mathcal{G} , then there is a set of nodes S which is either a subset of

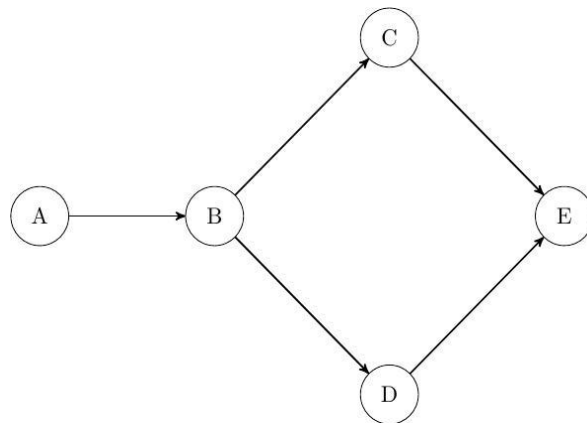
$adj(A, \mathcal{G})$

or of

$adj(B, \mathcal{G})$

such that S *d-separates* A and B in \mathcal{G} .

→ S disconnects A and B



Proof: Sprites et al., 2000, Theorem 6.2 (page 417)

Theoretical foundation

Theorem 2 (Sprites et al., 2000)

Let the nodes \mathcal{N} represent random variables $X = (X_N)_{N \in \mathcal{N}}$ which are multivariate Gaussian distributed \mathcal{P} and let \mathcal{P} be *faithful* to a DAG $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Assume that the perfect conditional independence information about all pairs of variables (A, B) in \mathcal{N} given subsets S is given.

Then, the output of the PC-algorithm is the *CPDAG* that represents \mathcal{G} .

→ Even given infinite amount of data, the PC-algorithm is not able to identify the true DAG, only its equivalence class

Proof: Sprites et al., 2000, Theorem 5.1 (page 410)

Two-Step Computation

Step 1: Learning the Skeleton

Input: Complete undirected graph

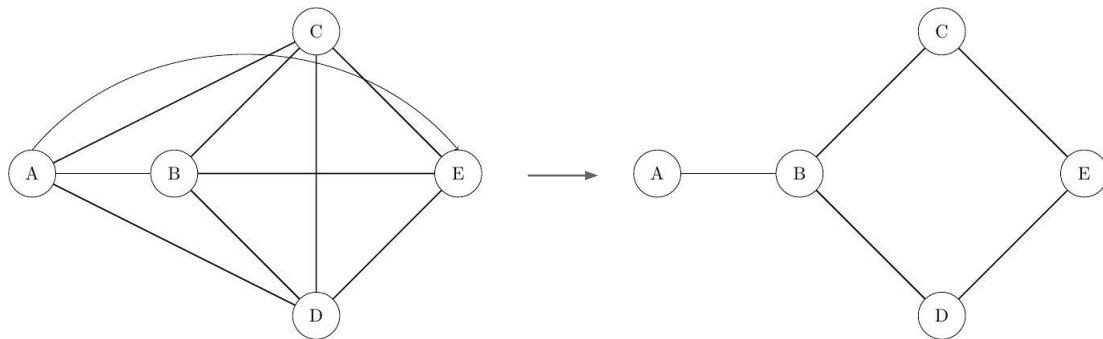
Output: Skeleton of the underlying DAG

Two-Step Computation

Step 1: Learning the Skeleton

Input: Complete undirected graph

Output: Skeleton of the underlying DAG

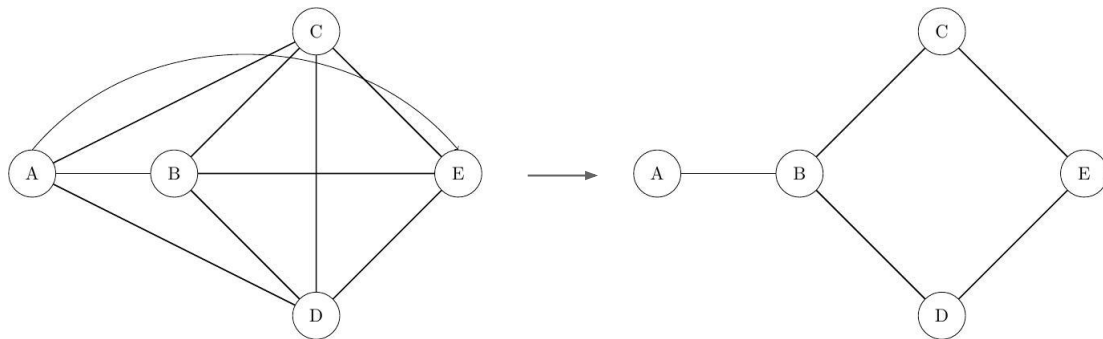


Two-Step Computation

Step 1: Learning the Skeleton

Input: Complete undirected graph

Output: Skeleton of the underlying DAG



Step 2: Learning the directions of the edges

Input: Skeleton of underlying DAG

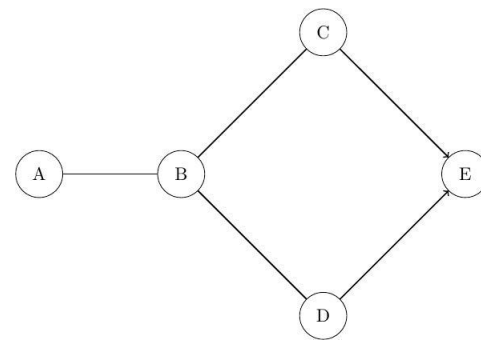
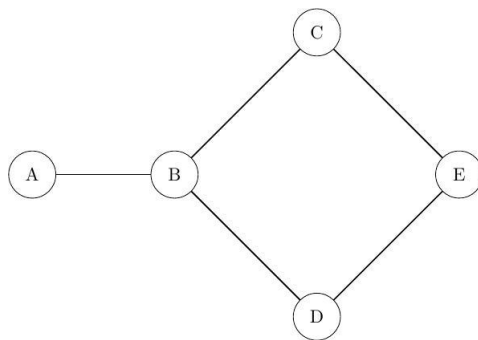
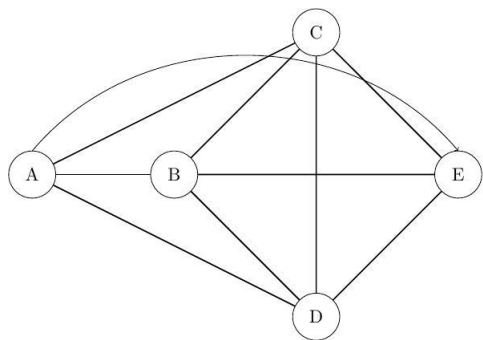
Output: Approximated CPDAG

Two-Step Computation

Step 1: Learning the Skeleton

Input: Complete undirected graph

Output: Skeleton of the underlying DAG



Step 2: Learning the directions of the edges

Input: Skeleton of underlying DAG

Output: Approximated CPDAG

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

end

end

end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;

Kalisch and Bühlmann, 2007

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

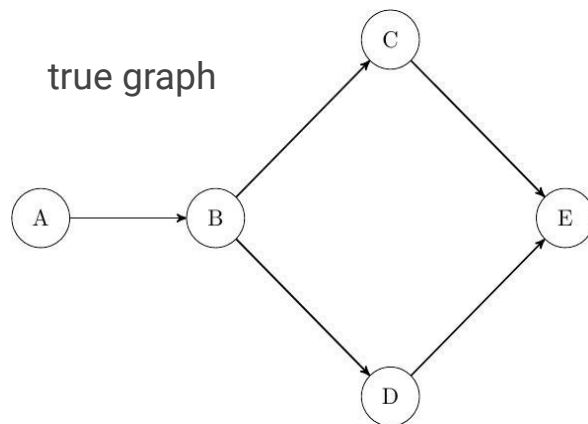
end

end

end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

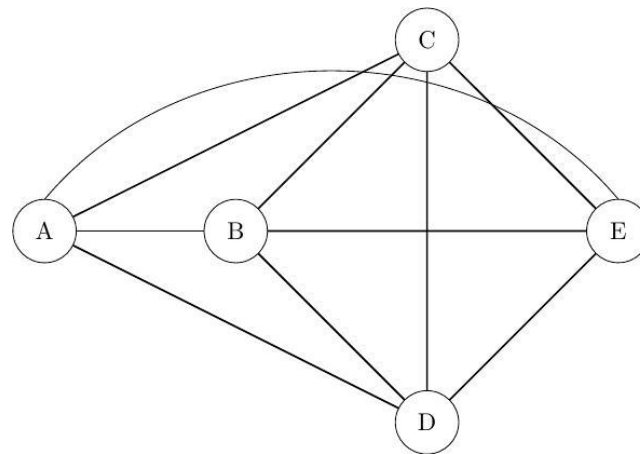
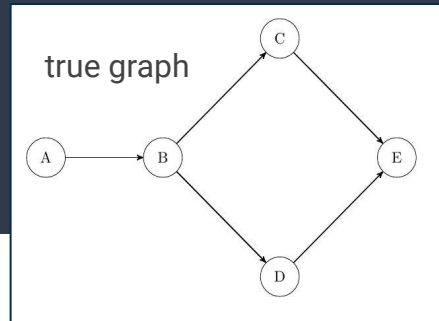
end

end

end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



complete undirected graph

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

 for each ordered pair of adjacent nodes X and Y do

 if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ then

 for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ do

 Test $CI(X, Y|S)$ on significance level α

 if $CI(X, Y|S)$ then

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

 break

 end

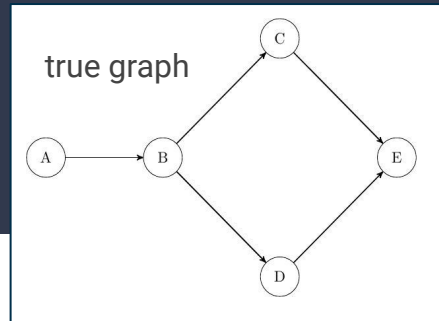
 end

 end

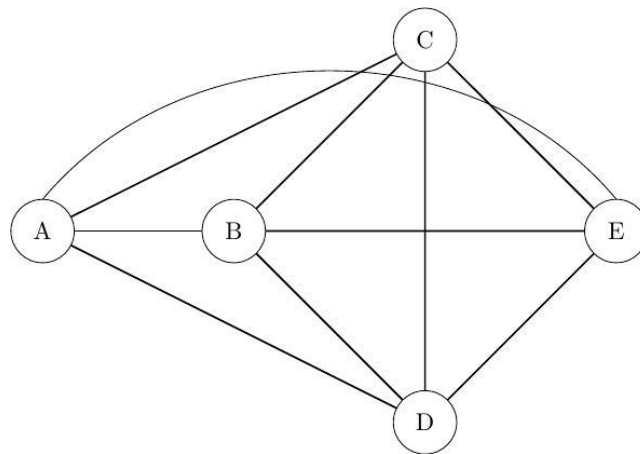
 end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 0$



Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

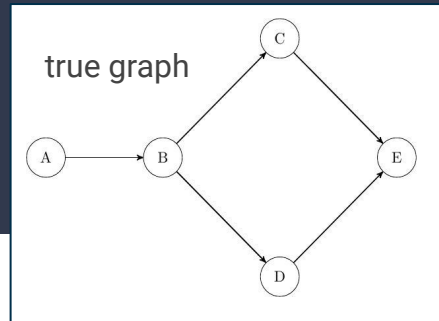
end

end

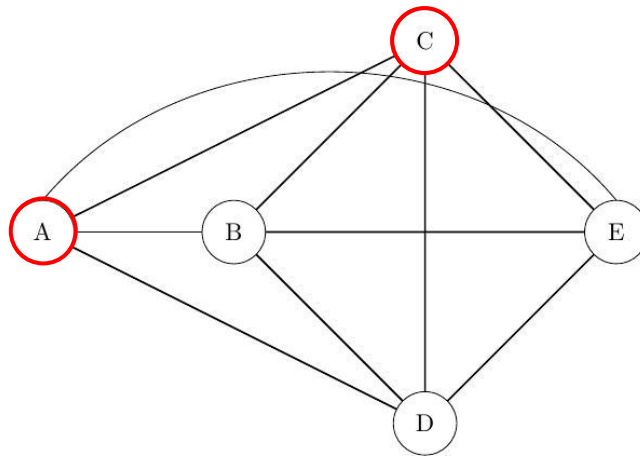
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 0$



Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

 for each ordered pair of adjacent nodes X and Y do

 if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ then

 for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ do

 Test $CI(X, Y|S)$ on significance level α

 if $CI(X, Y|S)$ then

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

 break

 end

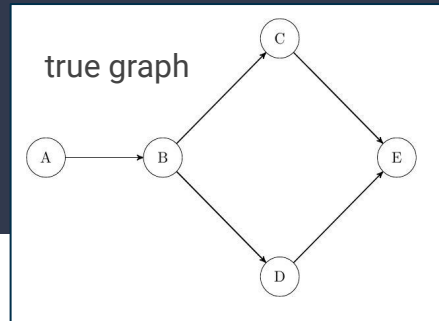
 end

 end

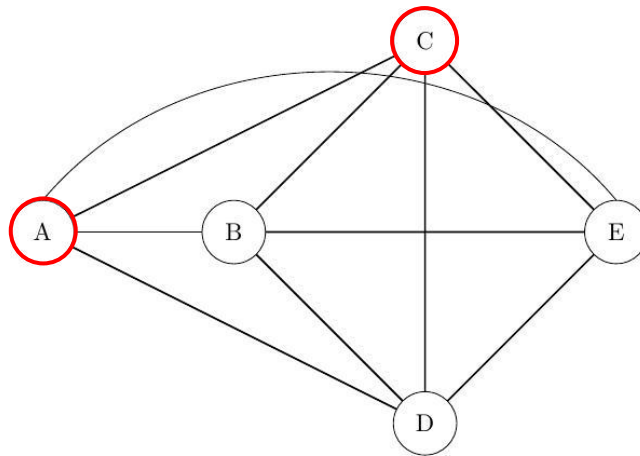
 end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 0$



$adj(A, \mathcal{G}) \setminus \{C\} = \{B, D, E\}$

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

 for each ordered pair of adjacent nodes X and Y do

 if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ then

 for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ do

 Test $CI(X, Y|S)$ on significance level α

 if $CI(X, Y|S)$ then

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

 break

 end

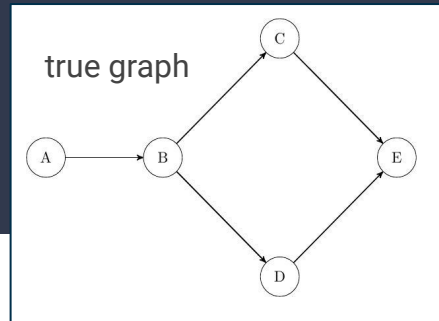
 end

 end

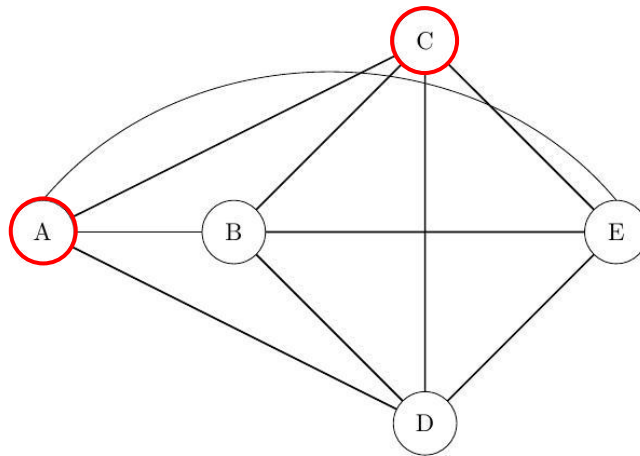
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 0$



$adj(A, \mathcal{G}) \setminus \{C\} = \{B, D, E\}$

$|S| = 0$

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

 for each ordered pair of adjacent nodes X and Y do

 if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ then

 for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ do

 Test $CI(X, Y|S)$ on significance level α

 if $CI(X, Y|S)$ then

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

 break

 end

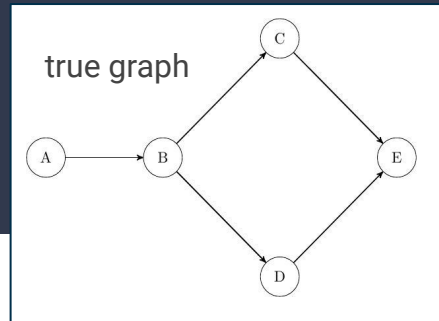
 end

 end

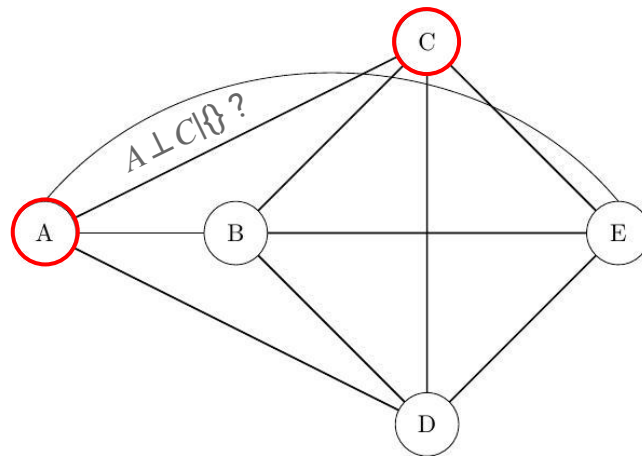
 end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 0$



$adj(A, \mathcal{G}) \setminus \{C\} = \{B, D, E\}$

$|S| = 0$

The CI test in detail

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

end

end

end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;

In practice: perfect CI information is unknown

→ How to estimate conditional independence from the data?

The CI test in detail

Recall: Property of the multivariate normal distribution:

For a multivariate normal distributed random vector $\mathbf{X} \in \mathbb{R}^p$ denote its partial correlation between components $\mathbf{X}^{(i)}$ and $\mathbf{X}^{(j)}$ ($i \neq j$) given set $\{\mathbf{X}^{(r)}, r \in \mathbf{R} \subseteq \{1, \dots, p\} \setminus \{i, j\}\}$ by $\rho_{i,j|\mathbf{R}}$.

Then it holds:

$\rho_{i,j|\mathbf{R}} = 0$ if and only if $\mathbf{X}^{(i)}$ and $\mathbf{X}^{(j)}$ are *conditionally independent* given $\{\mathbf{X}^{(r)}, r \in \mathbf{R}\}$.

→ **Idea:** Estimate partial correlation $\rho_{i,j|\mathbf{R}}$ and test whether it is close to zero

The CI test in detail

Testing of $\rho_{i,j|r}$ via Fisher's z-transform (Kalisch and Bühlmann, 2007):

- **Test statistic:** $Z(i, j | R) = 0.5 \log \left(\frac{1 + \rho_{i,j|R \setminus t}}{1 - \rho_{i,j|R \setminus t}} \right) \stackrel{H_0}{\sim} \mathcal{N}(0,1)$
- **Test:** Reject $H_0: \rho_{i,j|R} = 0$ against $H_1: \rho_{i,j|R} \neq 0$ if

$$(n - |R| - 3)^{0.5} |Z(i, j | R)| > \Phi^{-1}(1 - \alpha/2) ,$$

where Φ denotes the cumulative distribution function of a standard normal distribution

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

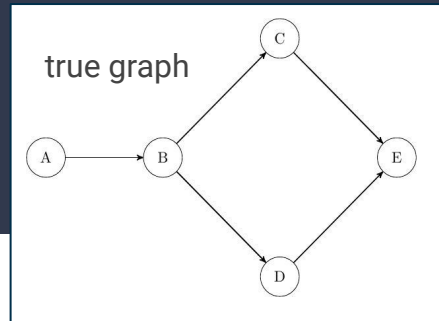
end

end

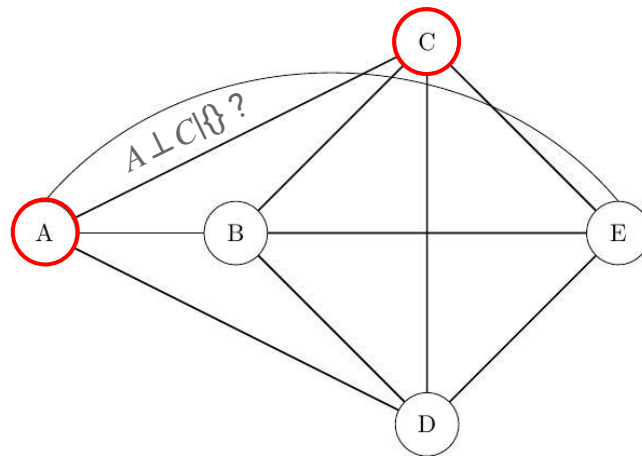
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 0$



$adj(A, \mathcal{G}) \setminus \{C\} = \{B, D, E\}$

$|S| = 0$

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

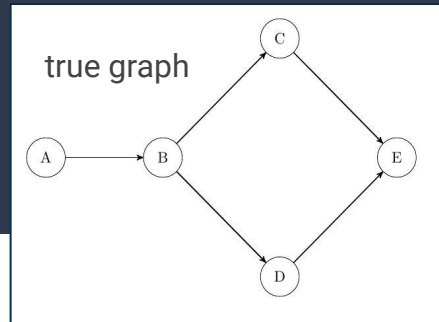
end

end

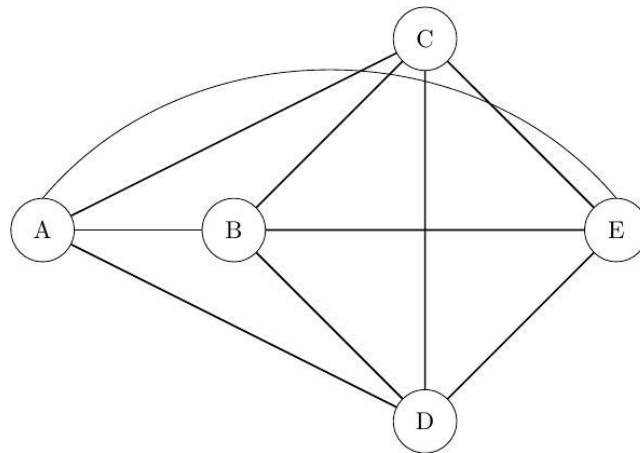
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 1$



Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

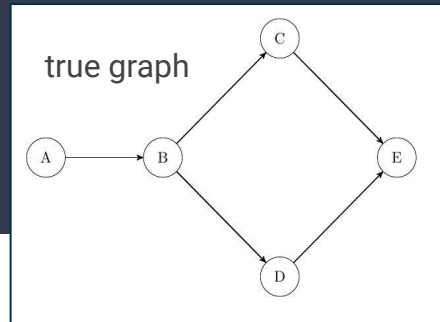
end

end

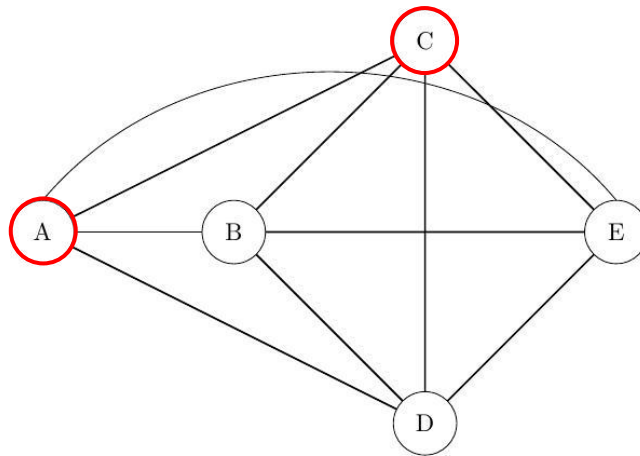
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 1$



Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

 for each ordered pair of adjacent nodes X and Y do

 if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ then

 for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ do

 Test $CI(X, Y|S)$ on significance level α

 if $CI(X, Y|S)$ then

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

 break

 end

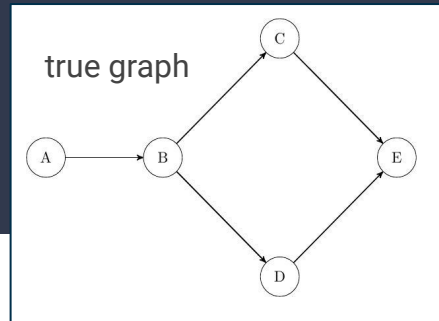
 end

 end

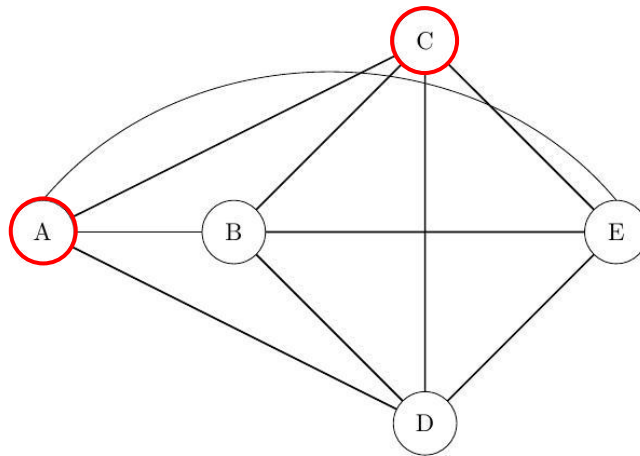
 end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 1$



$adj(A, \mathcal{G}) \setminus \{C\} = \{B, D, E\}$

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

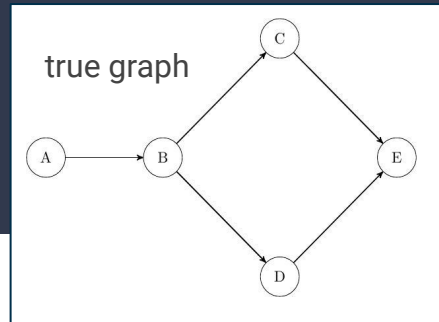
end

end

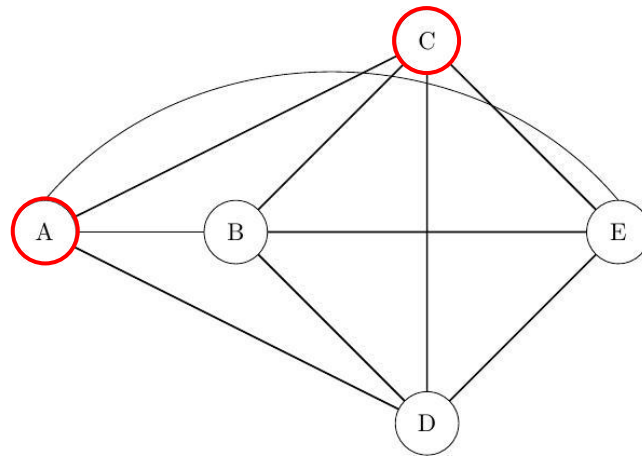
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 1$



$adj(A, \mathcal{G}) \setminus \{C\} = \{B, D, E\}$

$|S| = 1$

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

 for each ordered pair of adjacent nodes X and Y do

 if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ then

 for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ do

 Test $CI(X, Y|S)$ on significance level α

 if $CI(X, Y|S)$ then

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

 break

 end

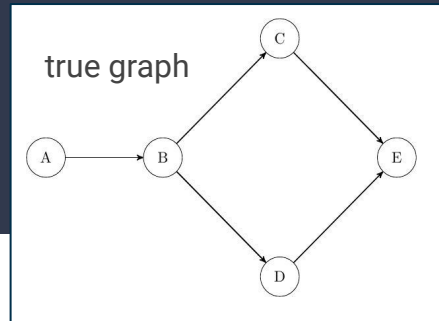
 end

 end

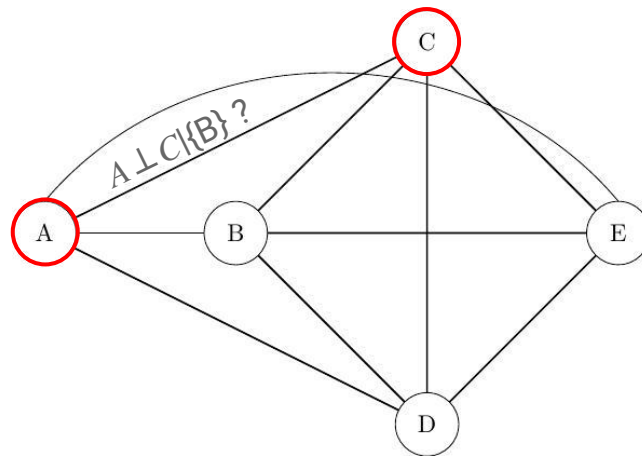
 end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 1$



$adj(A, \mathcal{G}) \setminus \{C\} = \{B, D, E\}$

$|S| = 1$

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

 break

end

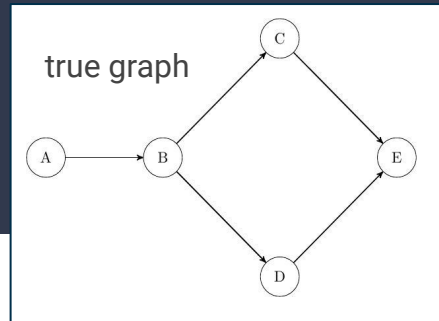
end

end

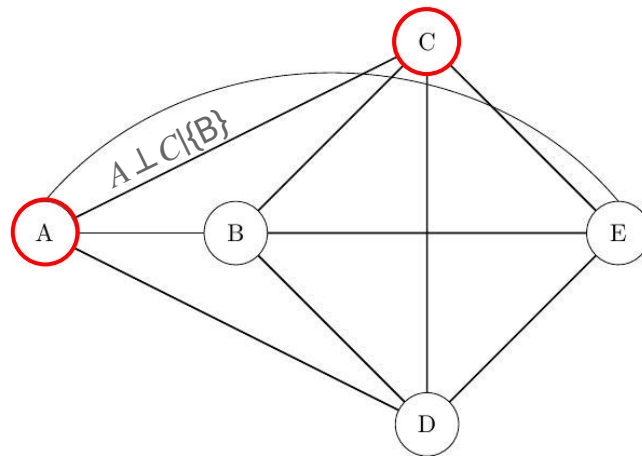
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 1$



$$adj(A, \mathcal{G}) \setminus \{C\} = \{B, D, E\}$$

$$|S| = 1 \rightarrow S = \{B\}$$

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

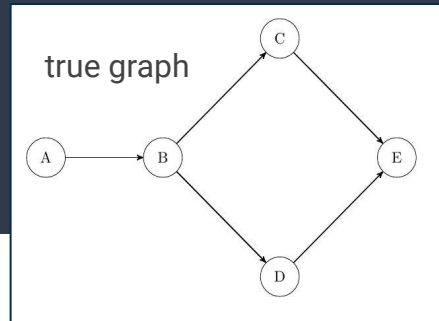
end

end

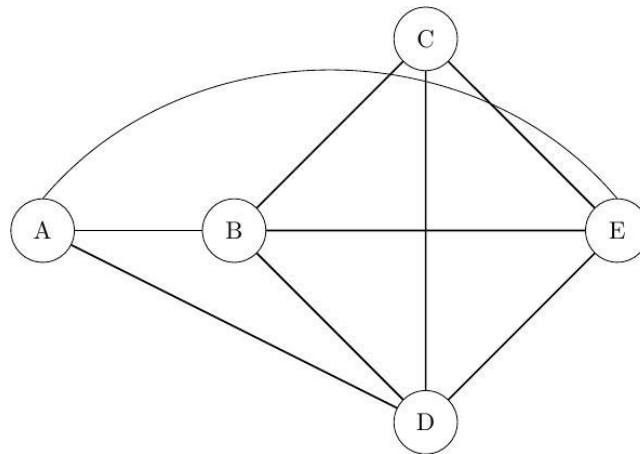
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 1$



$$adj(A, \mathcal{G}) \setminus \{C\} = \{B, D, E\}$$

$$|S| = 1 \rightarrow S = \{B\}$$

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

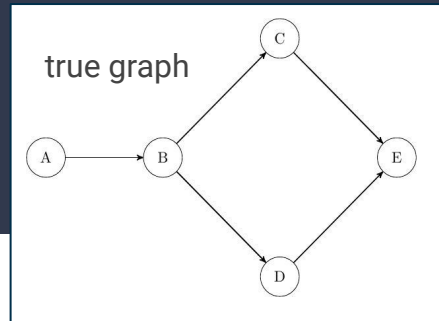
end

end

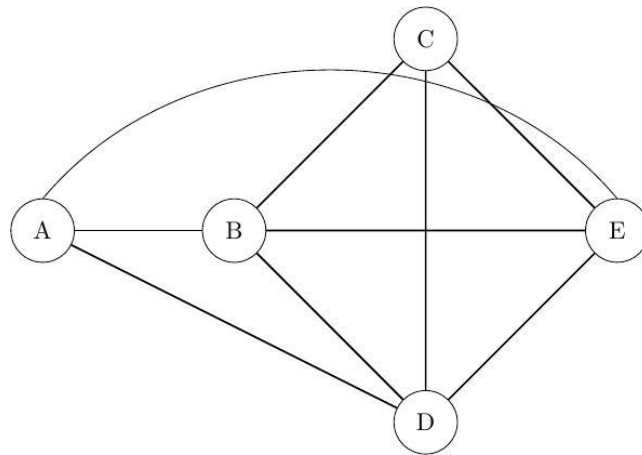
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 1$



$$adj(A, \mathcal{G}) \setminus \{C\} = \{B, D, E\}$$

$$|S| = 1 \rightarrow S = \{B\}$$

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

 break

end

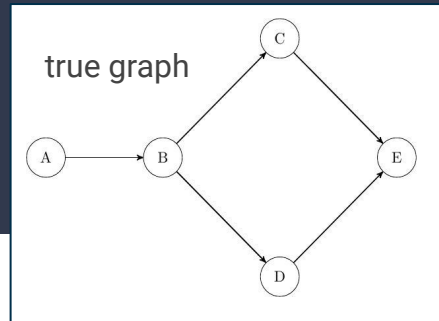
end

end

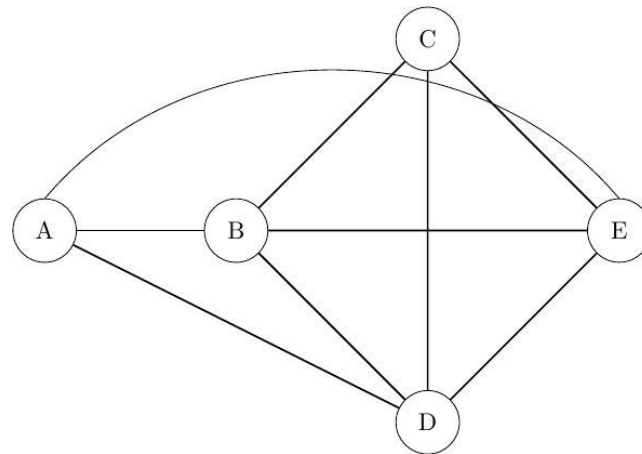
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 1$



$$\mathcal{E}_{new} = \mathcal{E} \setminus \{(A, C)\}$$

$$\mathcal{G}_{new} = (\mathcal{N}, \mathcal{E}_{new})$$

Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

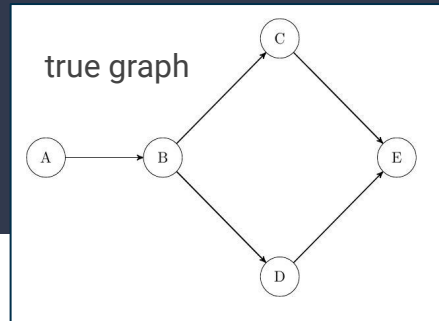
end

end

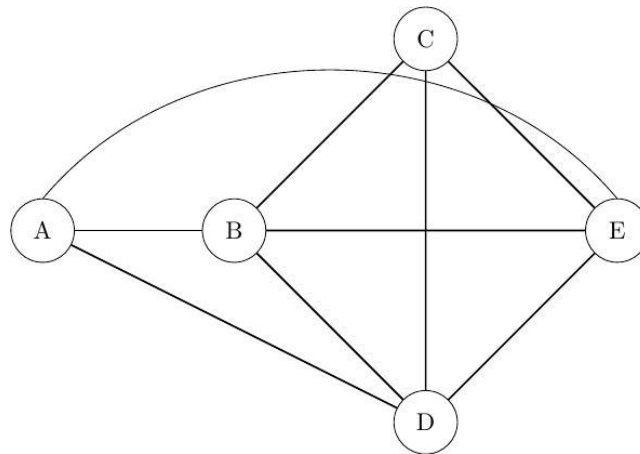
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 1$



Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

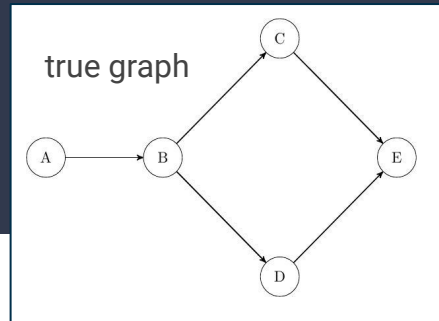
end

end

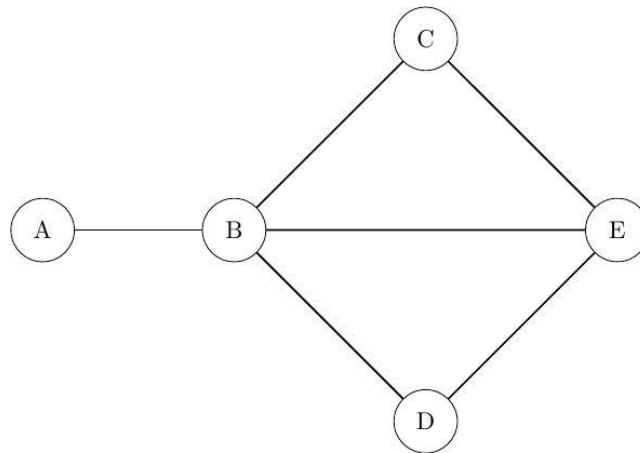
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 2$



Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

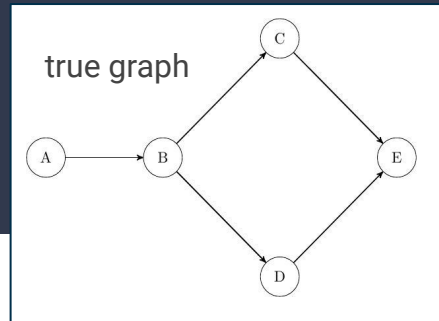
end

end

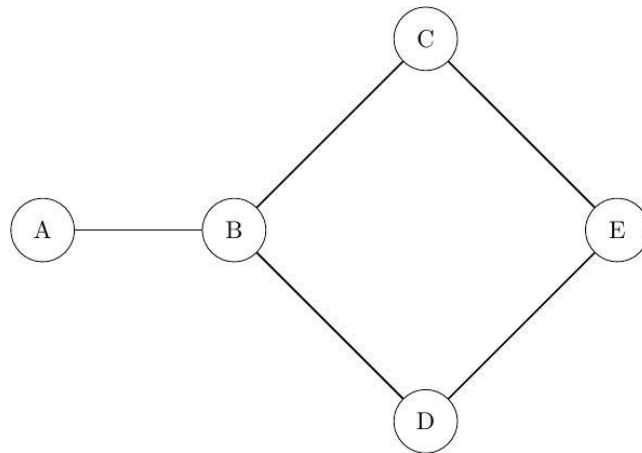
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 3$



Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

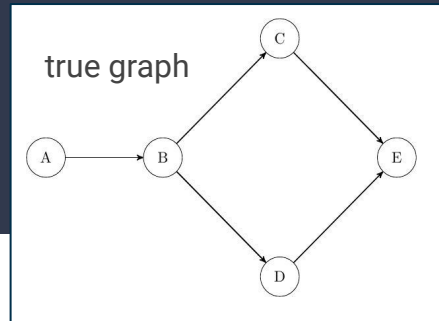
end

end

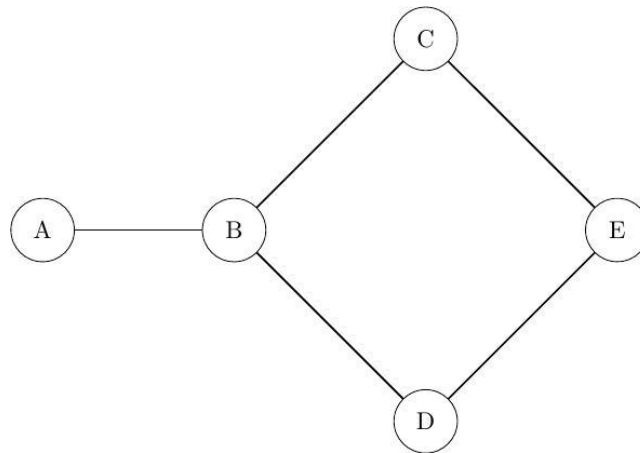
end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;



$d = 3$



Pseudocode

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

break

end

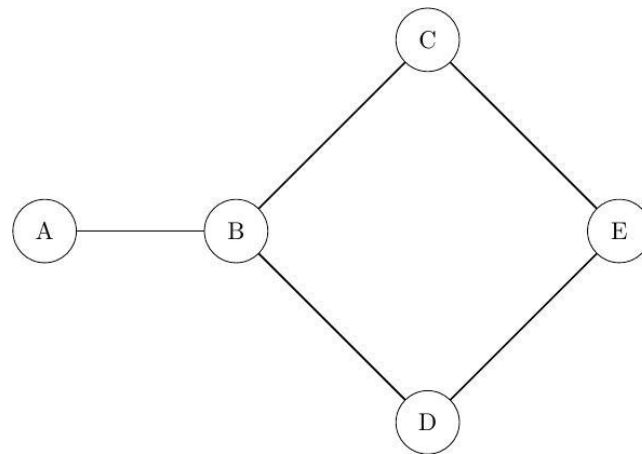
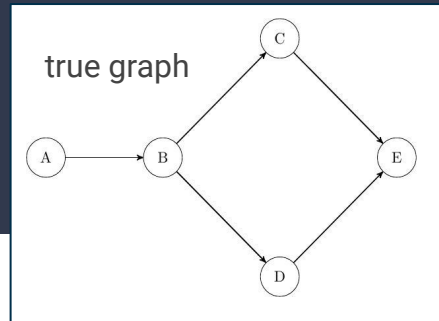
end

end

end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;

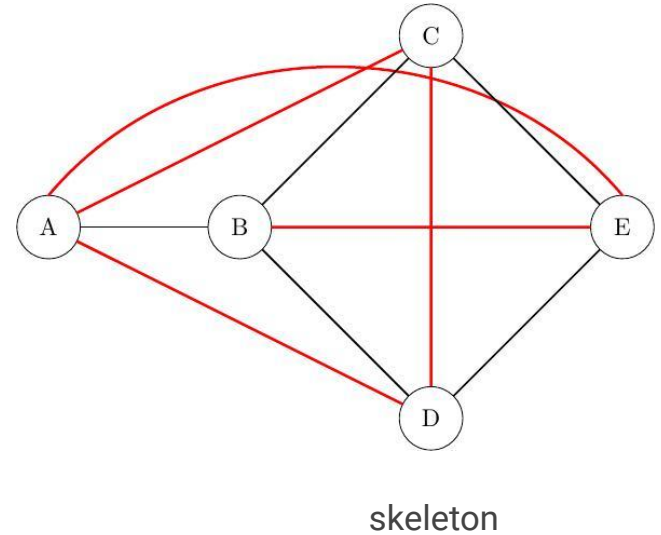
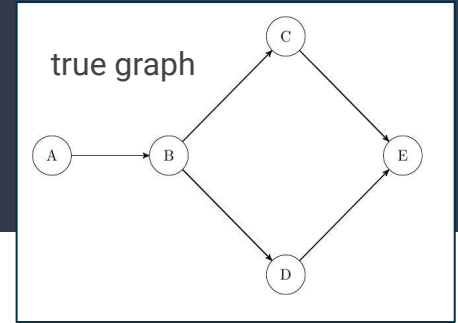


skeleton

Pseudocode

Interim results of step 1:

	Seperating set	Removed edge
$A \perp C \mid \{B\}$	$\{B\}$	$A - C$
$A \perp D \mid \{B\}$	$\{B\}$	$A - D$
$A \perp E \mid \{B\}$	$\{B\}$	$A - E$
$C \perp D \mid \{B\}$	$\{B\}$	$C - D$
$B \perp E \mid \{C, D\}$	$\{C, D\}$	$B - E$



Pseudocode

Algorithm 2: Step 2: Learning the CPDAG

Data: Skeleton \mathcal{G}_{Skel} , separating sets S

Result: CPDAG

for all pairs of non-adjacent nodes X and Y with common neighbor U **do**

if $U \notin S(X, Y)$ **then**

 | Orient $X - U - Y$ as $X \rightarrow U \leftarrow Y$

end

end

Orient as many undirected edges as possible by applying the following rules:

repeat

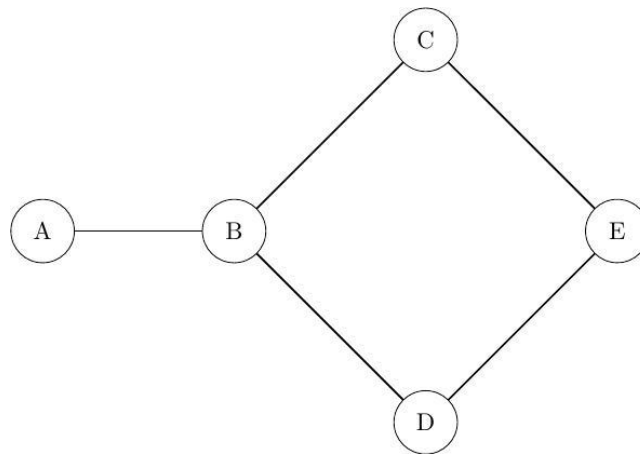
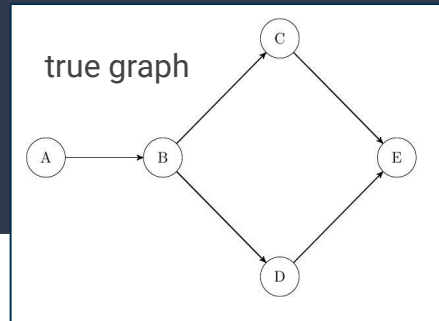
Rule 1 Orient $Y - U$ into $Y \rightarrow U$ when there is edge $X \rightarrow Y$ s.t. X and U are non-adjacent.

Rule 2 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$.

Rule 3 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$ and $X \rightarrow V \rightarrow Y$ s.t. U and V are non-adjacent.

Rule 4 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow V$ and $U \rightarrow V \rightarrow Y$ s.t. U and Y are non-adjacent.

until no more edges can be oriented;



Pseudocode

Algorithm 2: Step 2: Learning the CPDAG

Data: Skeleton \mathcal{G}_{Skel} , separating sets S

Result: CPDAG

for all pairs of non-adjacent nodes X and Y with common neighbor U **do**

if $U \notin S(X, Y)$ **then**

 | Orient $X - U - Y$ as $X \rightarrow U \leftarrow Y$

end

end

Orient as many undirected edges as possible by applying the following rules:

repeat

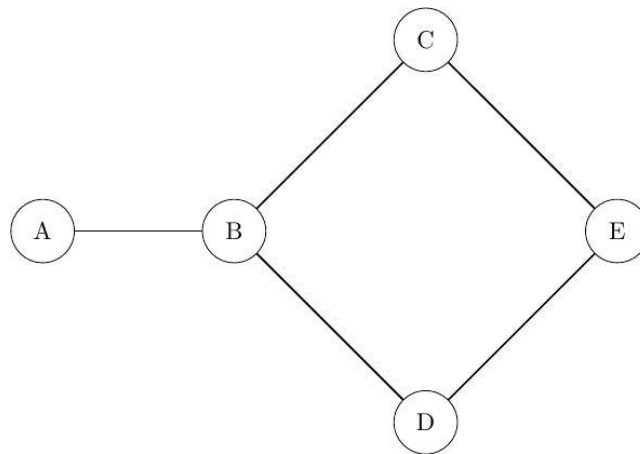
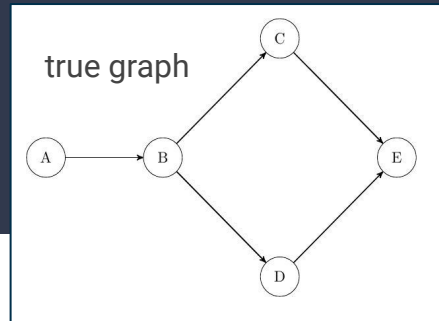
Rule 1 Orient $Y - U$ into $Y \rightarrow U$ when there is edge $X \rightarrow Y$ s.t. X and U are non-adjacent.

Rule 2 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$.

Rule 3 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$ and $X \rightarrow V \rightarrow Y$ s.t. U and V are non-adjacent.

Rule 4 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow V$ and $U \rightarrow V \rightarrow Y$ s.t. U and Y are non-adjacent.

until no more edges can be oriented;



Pseudocode

Algorithm 2: Step 2: Learning the CPDAG

Data: Skeleton \mathcal{G}_{Skel} , separating sets S

Result: CPDAG

for all pairs of non-adjacent nodes X and Y with common neighbor U **do**

if $U \notin S(X, Y)$ **then**

 | Orient $X - U - Y$ as $X \rightarrow U \leftarrow Y$

end

end

Orient as many undirected edges as possible by applying the following rules:

repeat

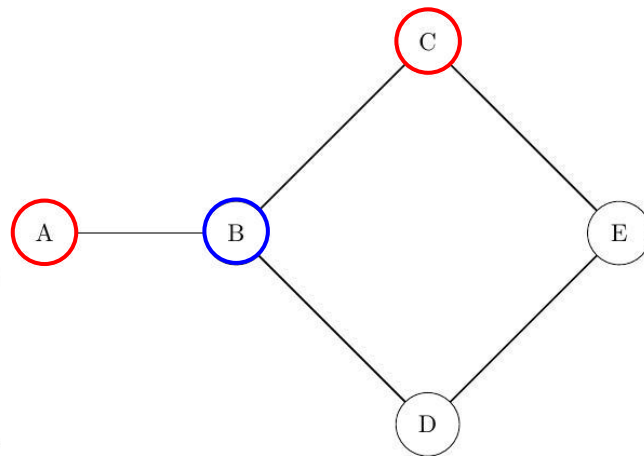
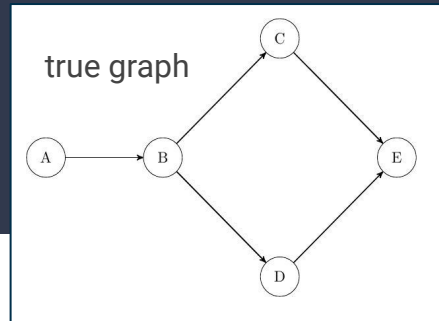
Rule 1 Orient $Y - U$ into $Y \rightarrow U$ when there is edge $X \rightarrow Y$ s.t. X and U are non-adjacent.

Rule 2 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$.

Rule 3 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$ and $X \rightarrow V \rightarrow Y$ s.t. U and V are non-adjacent.

Rule 4 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow V$ and $U \rightarrow V \rightarrow Y$ s.t. U and Y are non-adjacent.

until no more edges can be oriented;



Pseudocode

Algorithm 2: Step 2: Learning the CPDAG

Data: Skeleton \mathcal{G}_{Skel} , separating sets S

Result: CPDAG

for all pairs of non-adjacent nodes X and Y with common neighbor U **do**

if $U \notin S(X, Y)$ **then**

 | Orient $X - U - Y$ as $X \rightarrow U \leftarrow Y$

end

end

Orient as many undirected edges as possible by applying the following rules:

repeat

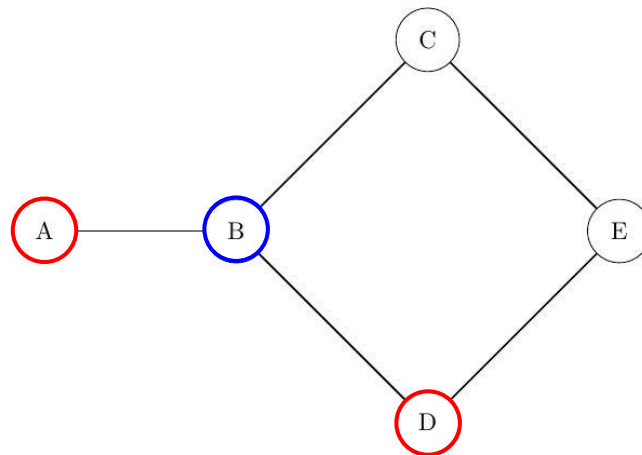
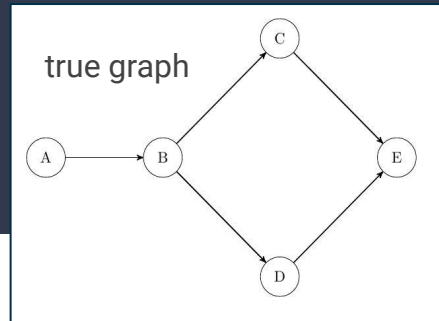
Rule 1 Orient $Y - U$ into $Y \rightarrow U$ when there is edge $X \rightarrow Y$ s.t. X and U are non-adjacent.

Rule 2 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$.

Rule 3 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$ and $X \rightarrow V \rightarrow Y$ s.t. U and V are non-adjacent.

Rule 4 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow V$ and $U \rightarrow V \rightarrow Y$ s.t. U and Y are non-adjacent.

until no more edges can be oriented;



Pseudocode

Algorithm 2: Step 2: Learning the CPDAG

Data: Skeleton \mathcal{G}_{Skel} , separating sets S

Result: CPDAG

for all pairs of non-adjacent nodes X and Y with common neighbor U **do**

if $U \notin S(X, Y)$ **then**

 | Orient $X - U - Y$ as $X \rightarrow U \leftarrow Y$

end

end

Orient as many undirected edges as possible by applying the following rules:

repeat

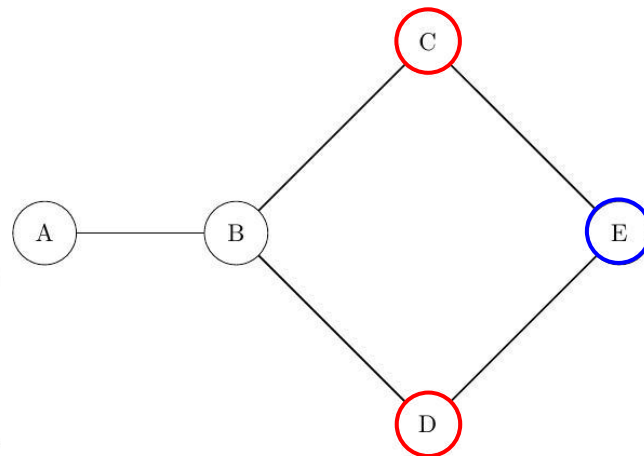
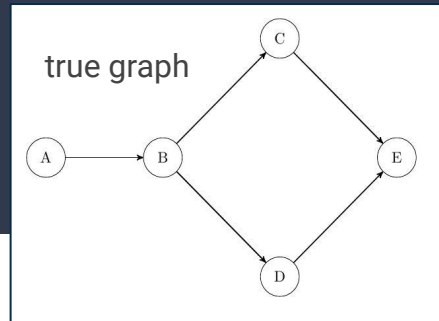
Rule 1 Orient $Y - U$ into $Y \rightarrow U$ when there is edge $X \rightarrow Y$ s.t. X and U are non-adjacent.

Rule 2 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$.

Rule 3 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$ and $X \rightarrow V \rightarrow Y$ s.t. U and V are non-adjacent.

Rule 4 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow V$ and $U \rightarrow V \rightarrow Y$ s.t. U and Y are non-adjacent.

until no more edges can be oriented;



Pseudocode

Algorithm 2: Step 2: Learning the CPDAG

Data: Skeleton \mathcal{G}_{Skel} , separating sets S

Result: CPDAG

for all pairs of non-adjacent nodes X and Y with common neighbor U **do**

if $U \notin S(X, Y)$ **then**

 | Orient $X - U - Y$ as $X \rightarrow U \leftarrow Y$

end

end

Orient as many undirected edges as possible by applying the following rules:

repeat

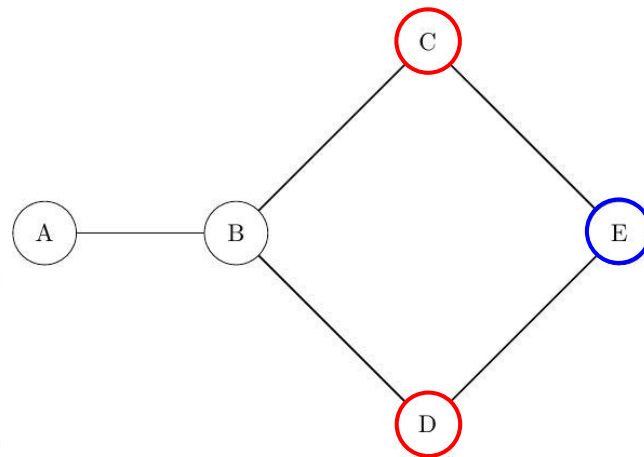
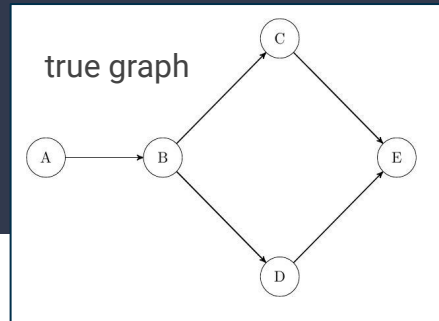
Rule 1 Orient $Y - U$ into $Y \rightarrow U$ when there is edge $X \rightarrow Y$ s.t. X and U are non-adjacent.

Rule 2 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$.

Rule 3 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$ and $X \rightarrow V \rightarrow Y$ s.t. U and V are non-adjacent.

Rule 4 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow V$ and $U \rightarrow V \rightarrow Y$ s.t. U and Y are non-adjacent.

until no more edges can be oriented;



$$S(C, D) = \{B\} \not\ni E$$

Pseudocode

Algorithm 2: Step 2: Learning the CPDAG

Data: Skeleton \mathcal{G}_{Skel} , separating sets S

Result: CPDAG

for all pairs of non-adjacent nodes X and Y with common neighbor U **do**

if $U \notin S(X, Y)$ **then**

 | Orient $X - U - Y$ as $X \rightarrow U \leftarrow Y$

end

end

Orient as many undirected edges as possible by applying the following rules:

repeat

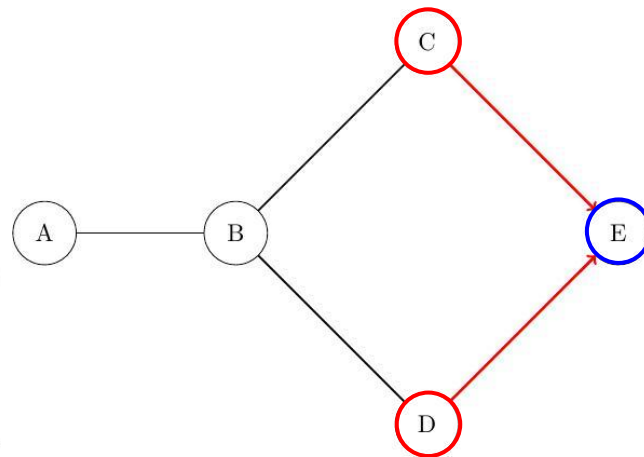
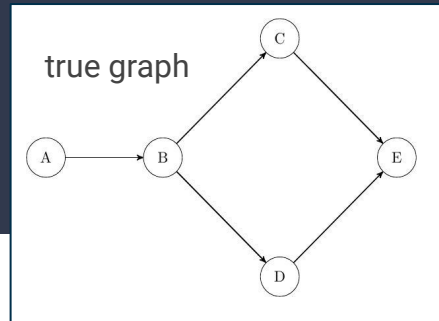
Rule 1 Orient $Y - U$ into $Y \rightarrow U$ when there is edge $X \rightarrow Y$ s.t. X and U are non-adjacent.

Rule 2 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$.

Rule 3 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$ and $X \rightarrow V \rightarrow Y$ s.t. U and V are non-adjacent.

Rule 4 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow V$ and $U \rightarrow V \rightarrow Y$ s.t. U and Y are non-adjacent.

until no more edges can be oriented;



$$S(C, D) = \{B\} \not\ni E$$

Pseudocode

Algorithm 2: Step 2: Learning the CPDAG

Data: Skeleton \mathcal{G}_{Skel} , separating sets S

Result: CPDAG

for all pairs of non-adjacent nodes X and Y with common neighbor U **do**

if $U \notin S(X, Y)$ **then**

 | Orient $X - U - Y$ as $X \rightarrow U \leftarrow Y$

end

end

Orient as many undirected edges as possible by applying the following rules:

repeat

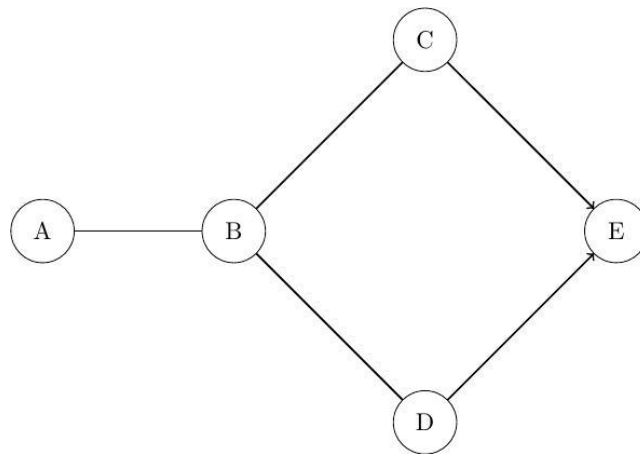
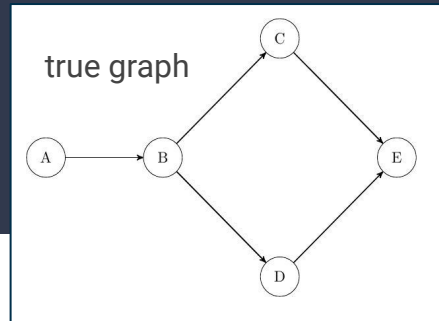
Rule 1 Orient $Y - U$ into $Y \rightarrow U$ when there is edge $X \rightarrow Y$ s.t. X and U are non-adjacent.

Rule 2 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$.

Rule 3 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$ and $X \rightarrow V \rightarrow Y$ s.t. U and V are non-adjacent.

Rule 4 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow V$ and $U \rightarrow V \rightarrow Y$ s.t. U and Y are non-adjacent.

until no more edges can be oriented;



Pseudocode

Algorithm 2: Step 2: Learning the CPDAG

Data: Skeleton \mathcal{G}_{Skel} , separating sets S

Result: CPDAG

for all pairs of non-adjacent nodes X and Y with common neighbor U **do**

if $U \notin S(X, Y)$ **then**

 | Orient $X - U - Y$ as $X \rightarrow U \leftarrow Y$

end

end

Orient as many undirected edges as possible by applying the following rules:

repeat

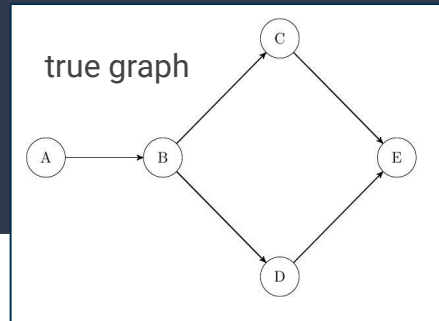
Rule 1 Orient $Y - U$ into $Y \rightarrow U$ when there is edge $X \rightarrow Y$ s.t. X and U are non-adjacent.

Rule 2 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$.

Rule 3 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow Y$ and $X \rightarrow V \rightarrow Y$ s.t. U and V are non-adjacent.

Rule 4 Orient $X - Y$ into $X \rightarrow Y$ when there is $X \rightarrow U \rightarrow V$ and $U \rightarrow V \rightarrow Y$ s.t. U and Y are non-adjacent.

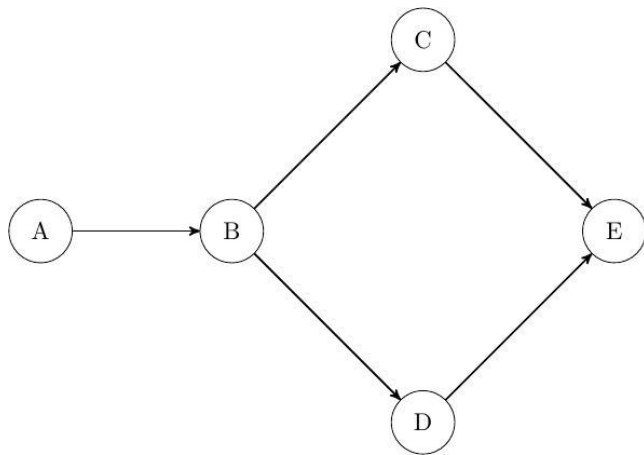
until no more edges can be oriented;



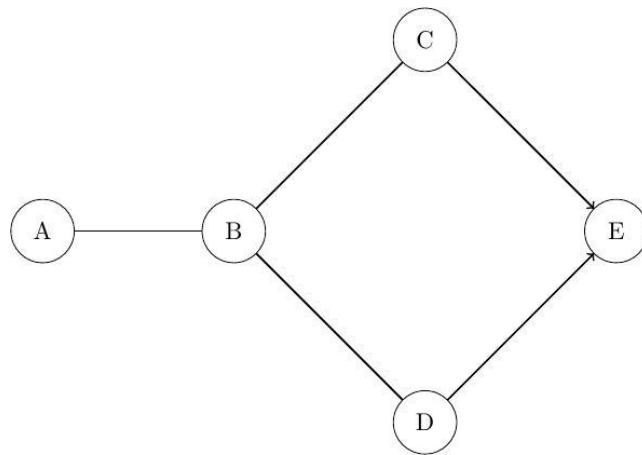
} oriented based on invalid edges
inducing new colliders or directed
edges

Pseudocode

Final result of PC:



true graph



approximated CPDAG

Agenda

- 1) Basic Definitions
- 2) Introduction to the PC-Algorithm
- 3) Limitations of the PC-Algorithm**
- 4) The stable PC-Algorithm
- 5) The parallel PC-Algorithm
- 6) Simulation Study in R
- 7) Final Notes and Discussion

Limitations of the original PC-Algorithm

Ordering

Problem:

Output graph is dependent on the order in which CI tests are performed

Solution: Stable PC-algorithm
(Colombo and Maathuis, 2012)

Runtime

Problem:

Many CI tests necessary
→ high runtime

Solution: Parallel PC-algorithm
(Le et al., 2014)

Gaussian Distribution

Problem:

Consistency only holds for Gaussian distribution
→ what if it is non-Gaussian?

Solution: Rank PC-algorithm
(Harris and Drton, 2013)

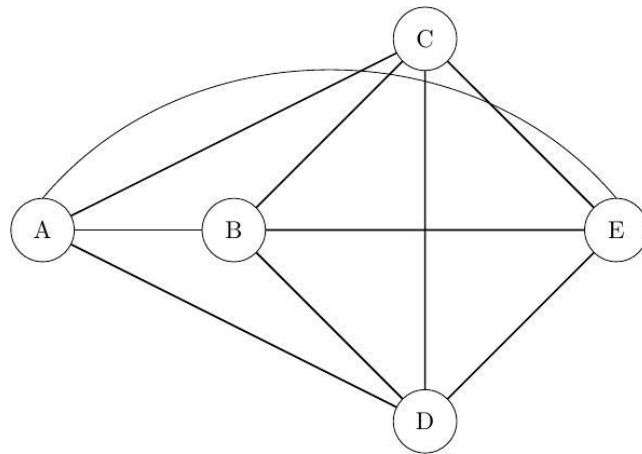
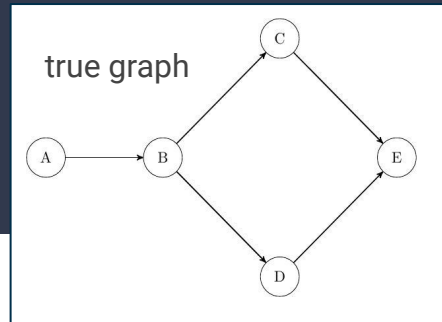
Agenda

- 1) Basic Definitions
- 2) Introduction to the PC-Algorithm
- 3) Limitations of the PC-Algorithm
- 4) The stable PC-Algorithm**
- 5) The parallel PC-Algorithm
- 6) Simulation Study in R
- 7) Final Notes and Discussion

The stable PC-Algorithm

Problem:

Incorrectly removing/retaining edge results in changes of set S

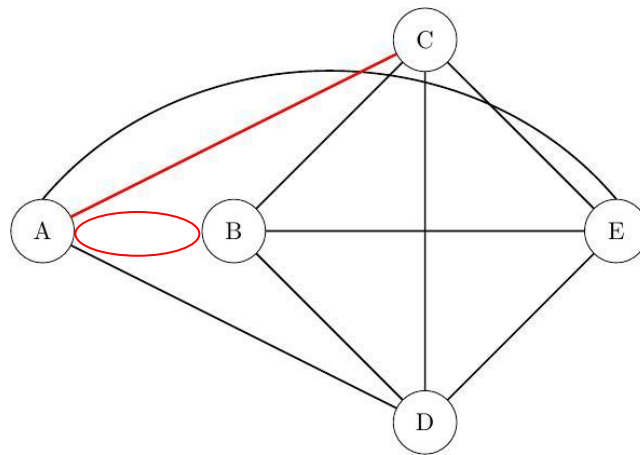
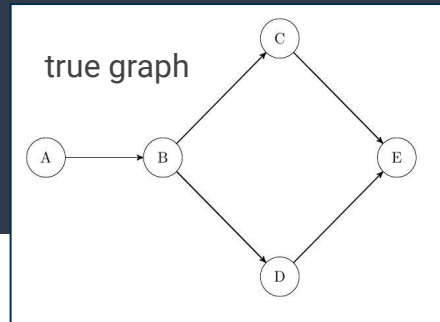


The stable PC-Algorithm

Problem:

Incorrectly removing/retaining edge results in changes of set S

→ output graph depends on order in which CI tests are performed



$$\text{adj}(A, \mathcal{G}) \setminus \{C\} = \{D, E\}$$

$S \ni B$ for all possible sets S

The stable PC-Algorithm

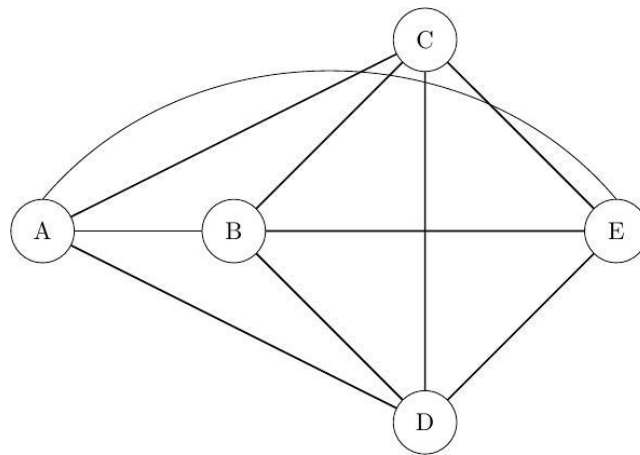
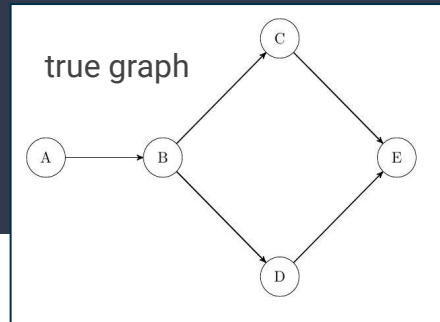
Problem:

Incorrectly removing/retaining edge results in changes of set S

→ output graph depends on order in which CI tests are performed

Idea:

Keep adjacency sets of nodes unchanged in each level d



Idea

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

 break

end

end

end

end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;

Idea

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

 break

end

end

end

end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;

Idea

Algorithm 1: Step 1: Learning the skeleton

Data: Data set D with nodes \mathcal{N} , significance level α

Result: Skeleton \mathcal{G}_{skel} , separating sets S

$d \leftarrow 0$

repeat

for each ordered pair of adjacent nodes X and Y **do**

if $|adj(X, \mathcal{G}) \setminus \{Y\}| \geq d$ **then**

for each subset $S \subseteq adj(X, \mathcal{G}) \setminus \{Y\}$ and $|S| = d$ **do**

 Test $CI(X, Y|S)$ on significance level α

if $CI(X, Y|S)$ **then**

 Remove edge between X and Y

 Save S as separating set of (X, Y)

 Update \mathcal{G} and \mathcal{E}

 break

end

end

end

end

$d \leftarrow d + 1$

until $|adj(X, \mathcal{G}) \setminus \{Y\}| < d$ for every pair of adjacent nodes;

} now: update \mathcal{G} and \mathcal{E} at the end of level d and store the adjacency sets of all nodes for the search of the right S

Limitation

Disadvantages:

- Adjacency sets and set S are possibly larger than in the original PC
→ more CI tests necessary
- Even more increased runtime than for the original PC

Possible Solution: Parallel PC-algorithm

Agenda

- 1) Basic Definitions
- 2) Introduction to the PC-Algorithm
- 3) Limitations of the PC-Algorithm
- 4) The stable PC-Algorithm
- 5) The parallel PC-Algorithm**
- 6) Simulation Study in R
- 7) Final Notes and Discussion

Idea

Parallelism:

- Break down big task into several different smaller subtasks
- Distribute them over different cores of the computer's CPU

Idea

Parallelism:

- Break down big task into several different smaller subtasks
- Distribute them over different cores of the computer's CPU

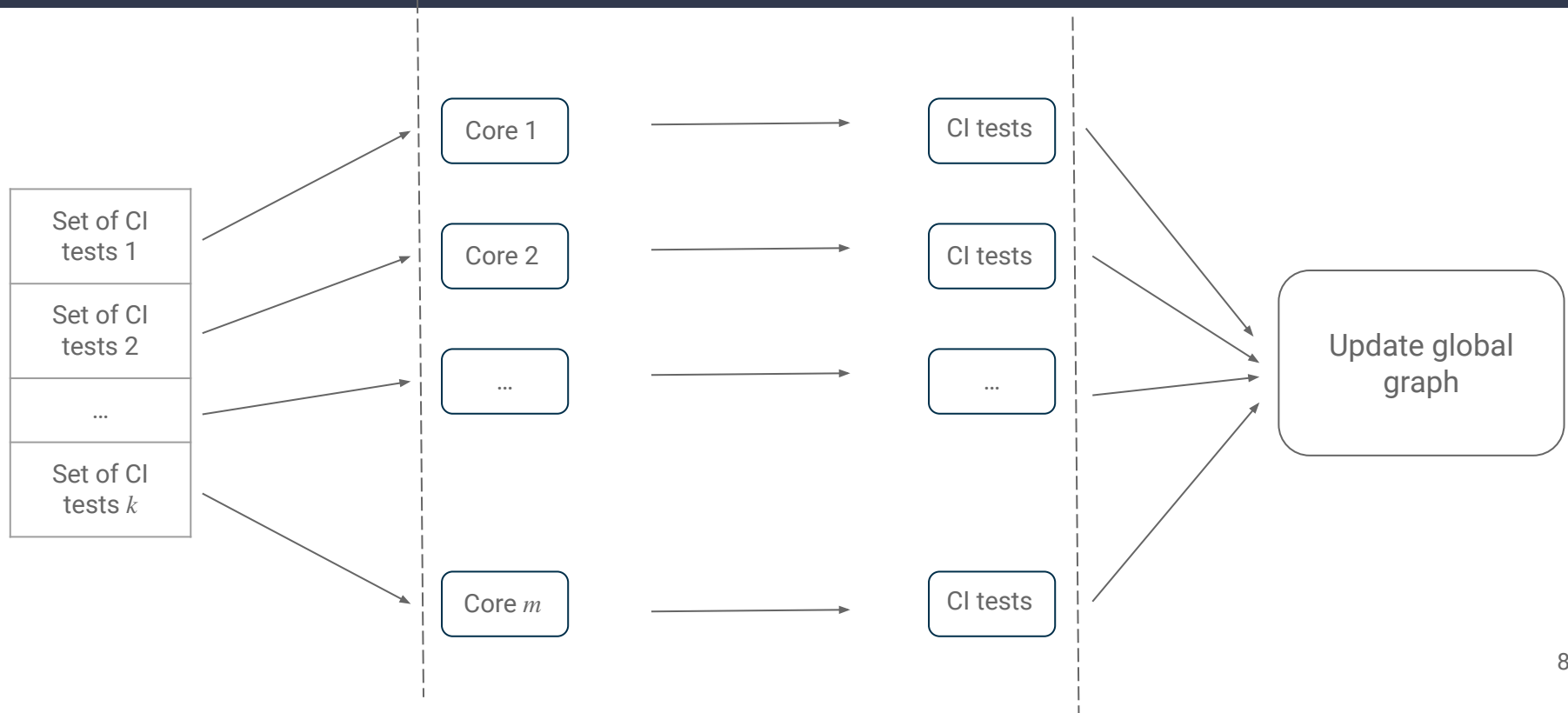
Prerequisite: subtasks need to be independent

- But: CI test results of particular level d influence CI test results of following level $d + 1$
- Idea: parallelize CI tests within each level

Step 1

Step 2

Step 3



Agenda

- 1) Basic Definitions
- 2) Introduction to the PC-Algorithm
- 3) Limitations of the PC-Algorithm
- 4) The stable PC-Algorithm
- 5) The parallel PC-Algorithm
- 6) Simulation Study in R**
- 7) Final Notes and Discussion

Comparison of all previously discussed algorithms

Data simulation

Repeat the following steps:

1. Simulate random DAG \mathcal{G}
 - dimension p
 - sparsity s
2. Draw random samples from \mathcal{G}
 - sample size n
 - distribution

CPDAG construction

Run original/stable/parallel PC-algorithm with different settings for $\alpha = 0.01$ and 10 simulations:

- dimension p
- sample size n
- ordering

Evaluation

Comparison of:

- Run time in s
- Structural Hamming Distance
- True Positive Rate
- False Positive Rate
- True Discovery Rate

Evaluation measures

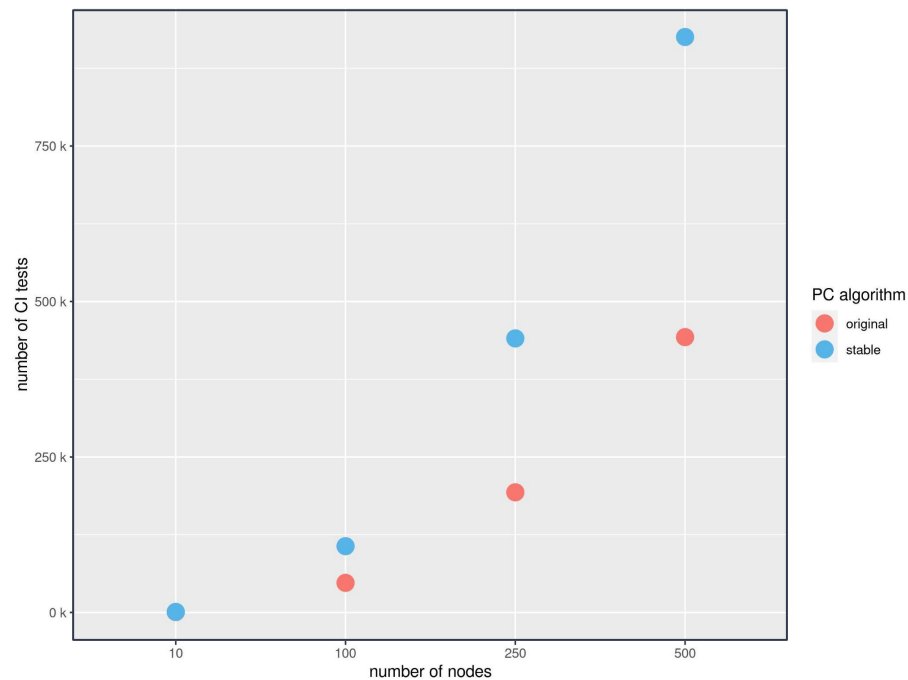
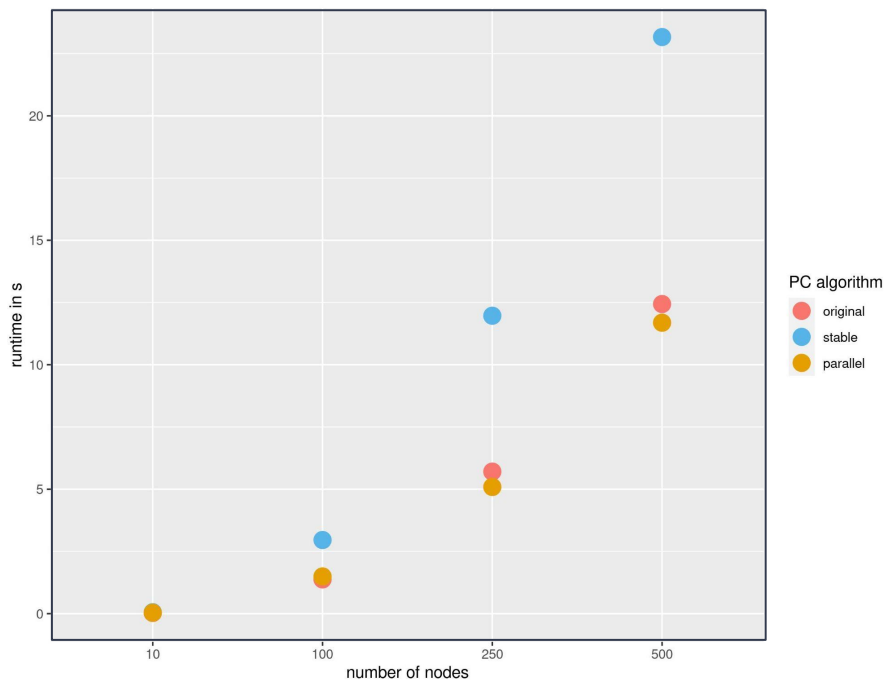
Structural Hamming Distance (SHD)	Number of edge insertions, deletions or flips in order to transform estimated CPDAG to true CPDAG (or vice versa)
True Positive Rate (TPR)	Number of correctly found edges in estimated CPDAG divided by number of true edges in true CPDAG
False Positive Rate (FPR)	Number of incorrectly found edges in estimated CPDAG divided by number of true gaps in true CPDAG
True Discovery Rate (TDR)	Number of correctly found edges divided by number of found edges both in estimated CPDAG

Setting 1: different dimensions

$n = 1000$

$p \in \{10, 100, 250, 500\}$

expected neighborhood size: 5

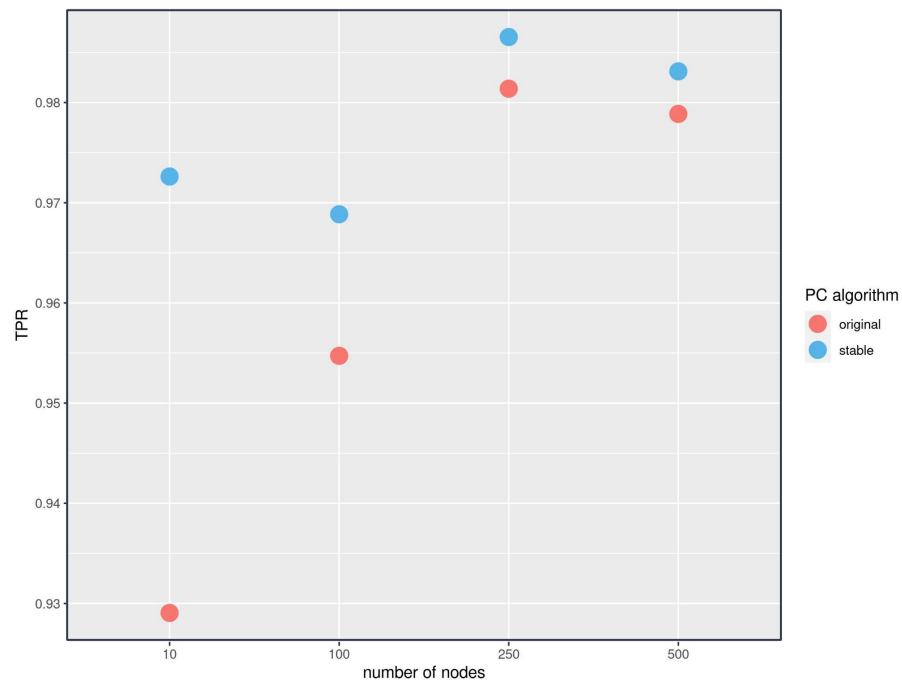
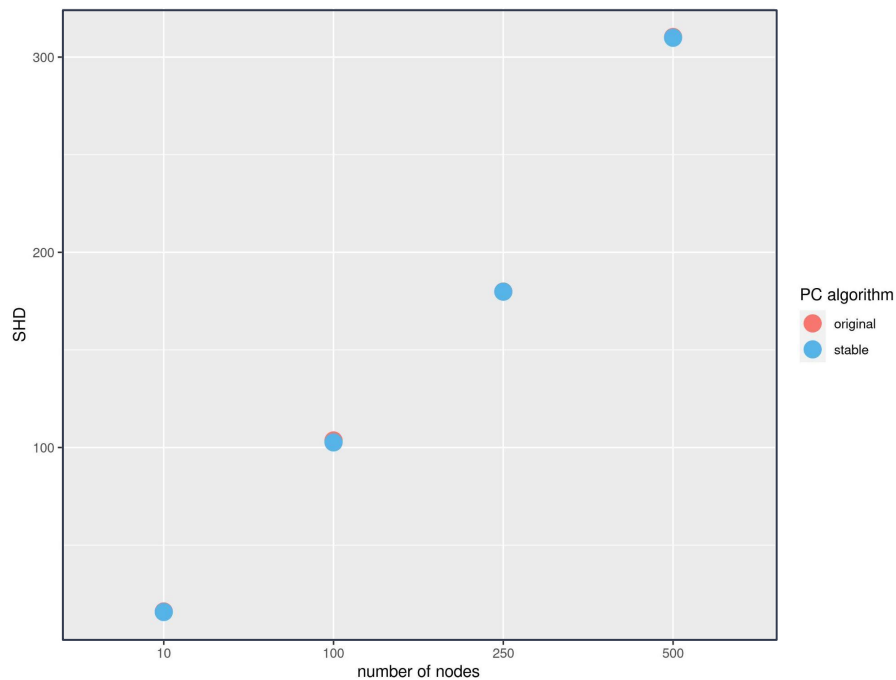


Setting 1: different dimensions

$n = 1000$

$p \in \{10, 100, 250, 500\}$

expected neighborhood size: 5

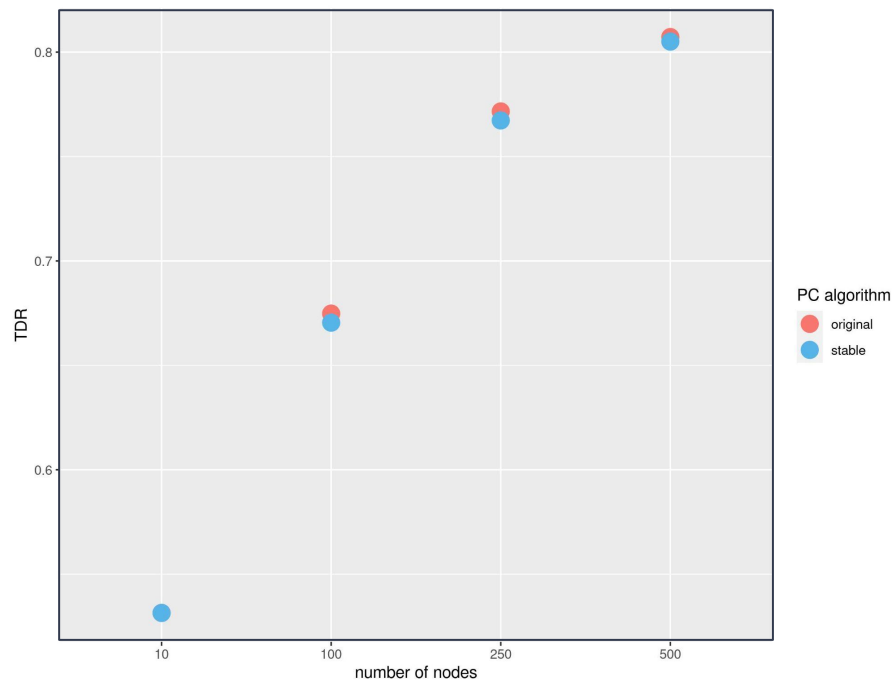
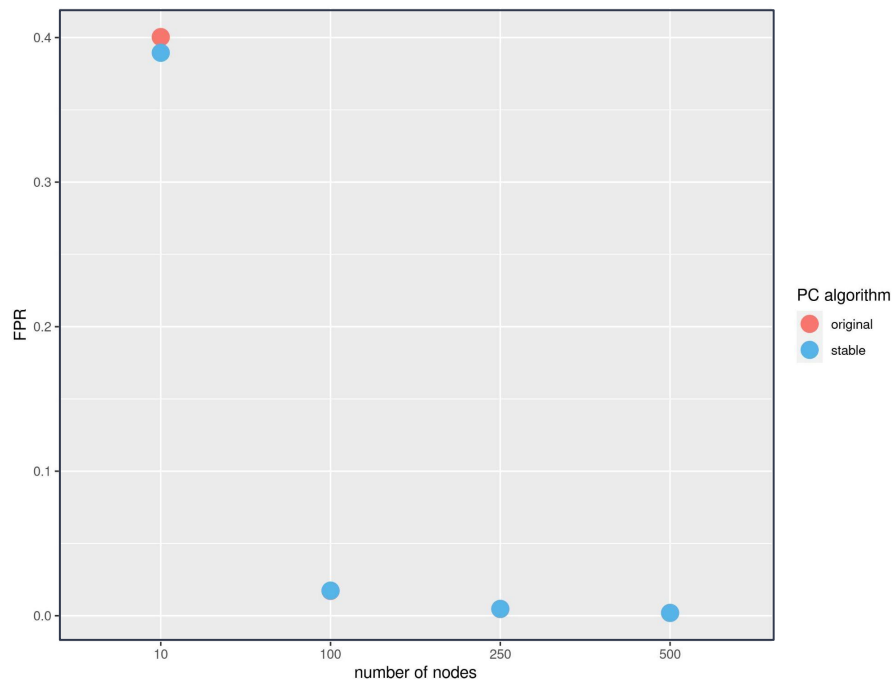


Setting 1: different dimensions

$n = 1000$

$p \in \{10, 100, 250, 500\}$

expected neighborhood size: 5

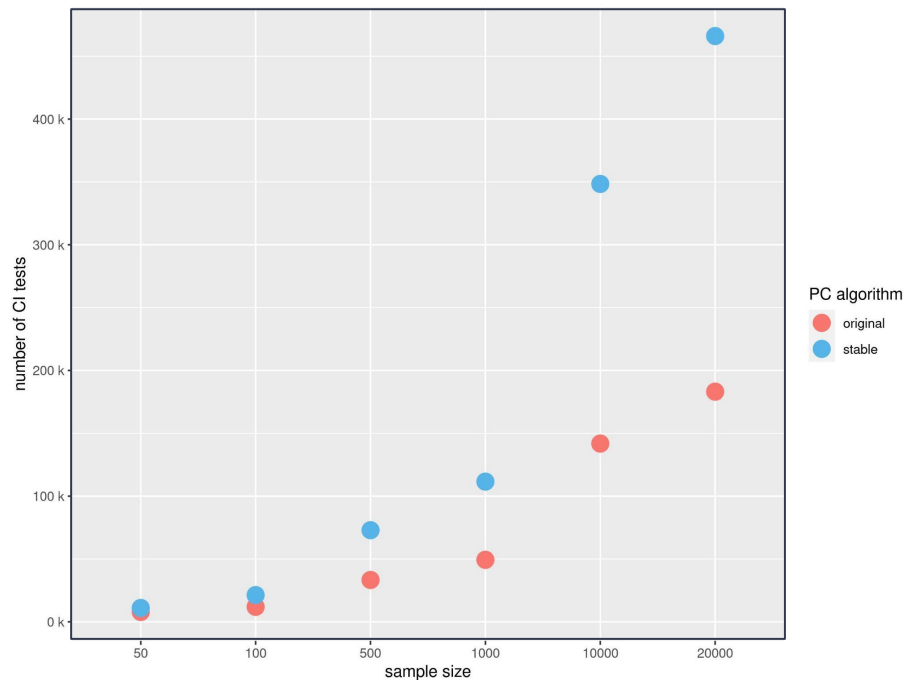
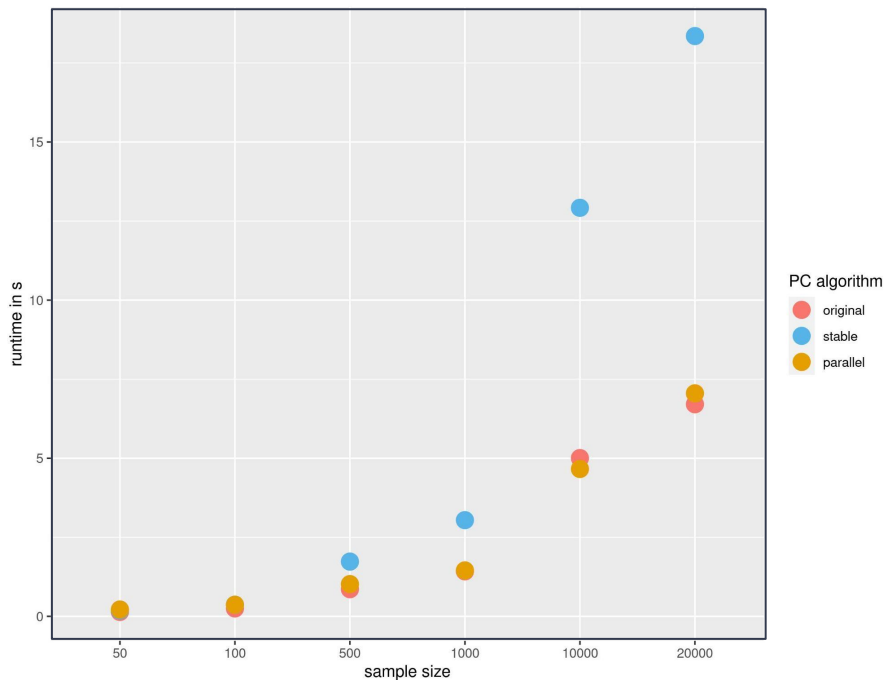


Setting 2: different sample sizes

$n \in \{50, 100, 500, 1k, 10k, 20k\}$

$p = 100$

expected neighborhood size: 5

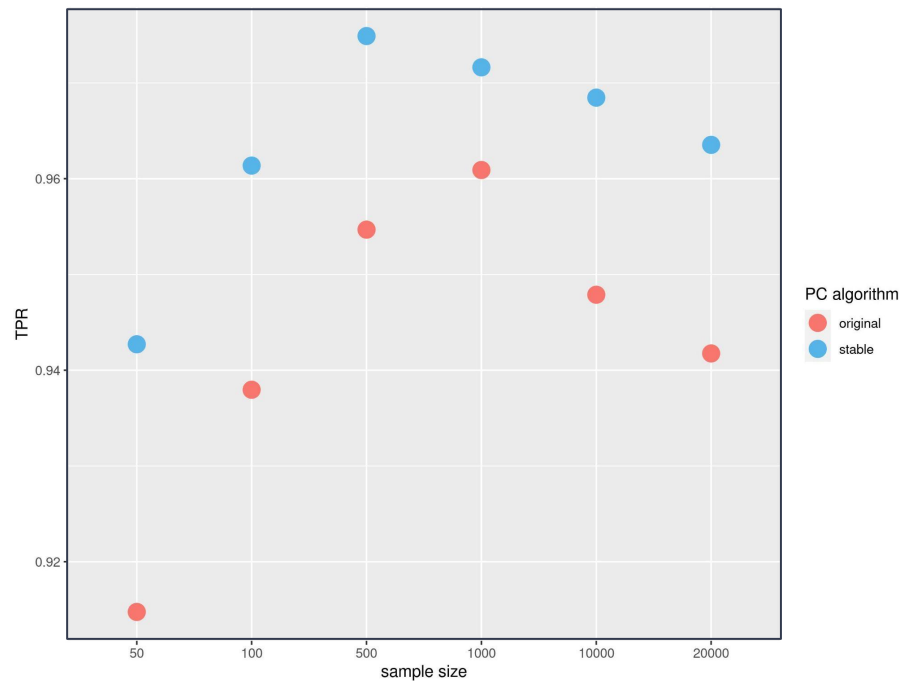
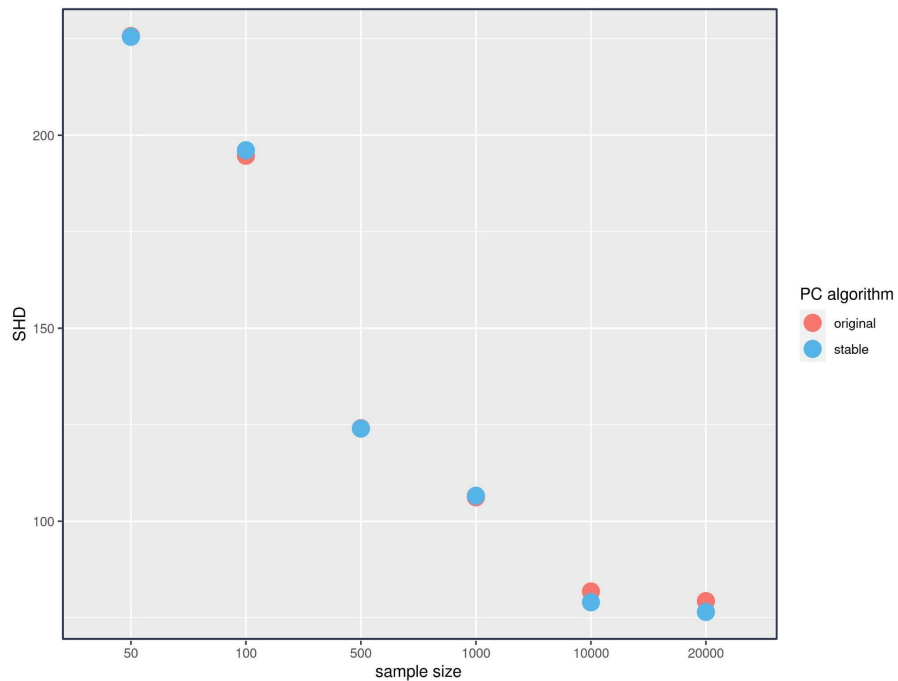


Setting 2: different sample sizes

$n \in \{50, 100, 500, 1k, 10k, 20k\}$

$p = 100$

expected neighborhood size: 5

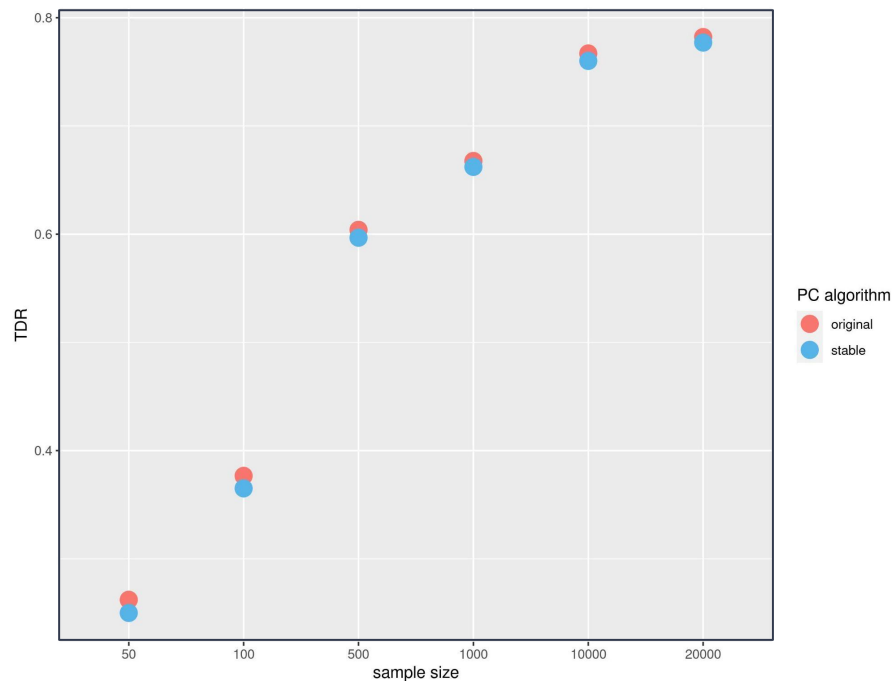
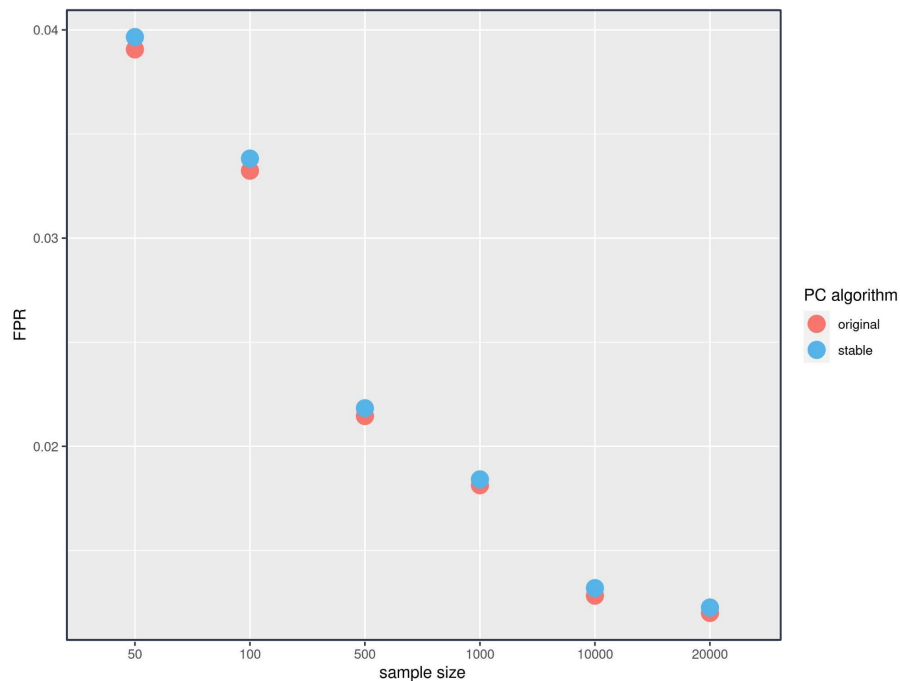


Setting 2: different sample sizes

$n \in \{50, 100, 500, 1k, 10k, 20k\}$

$p = 100$

expected neighborhood size: 5

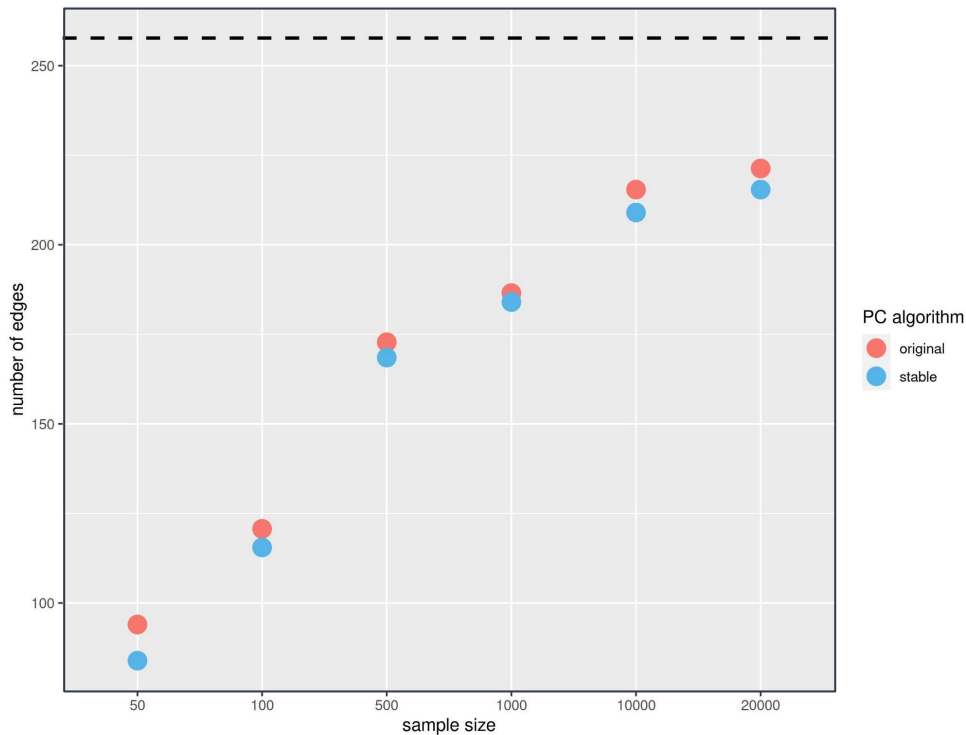


Setting 2: different sample sizes

$n \in \{50, 100, 500, 1k, 10k, 20k\}$

$p = 100$

number of simulations: 10

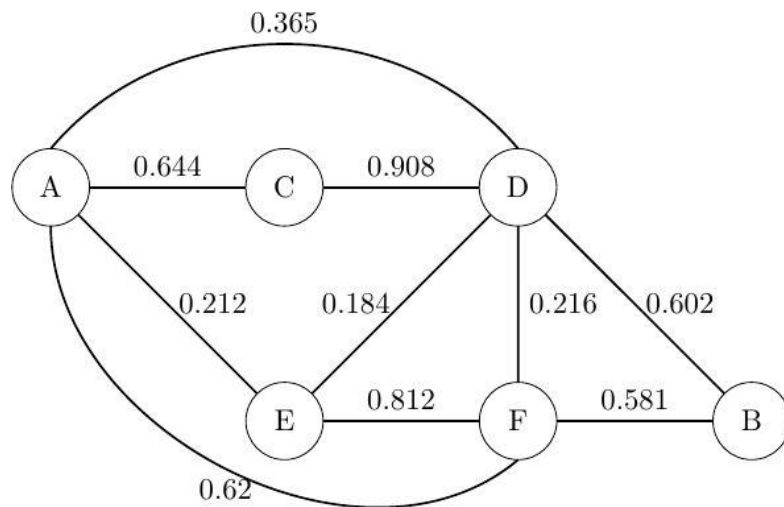


Setting 3: different ordering

Parameter settings:

- sample size: $n = 20$
- number of nodes: $p = 6$
- expected neighborhood size: 3
- significance level: $\alpha = 0.05$
- 4 different orderings of variables

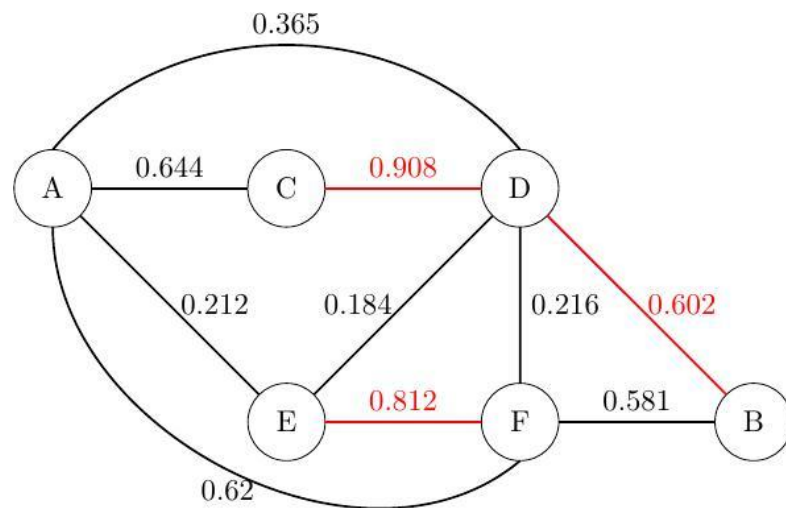
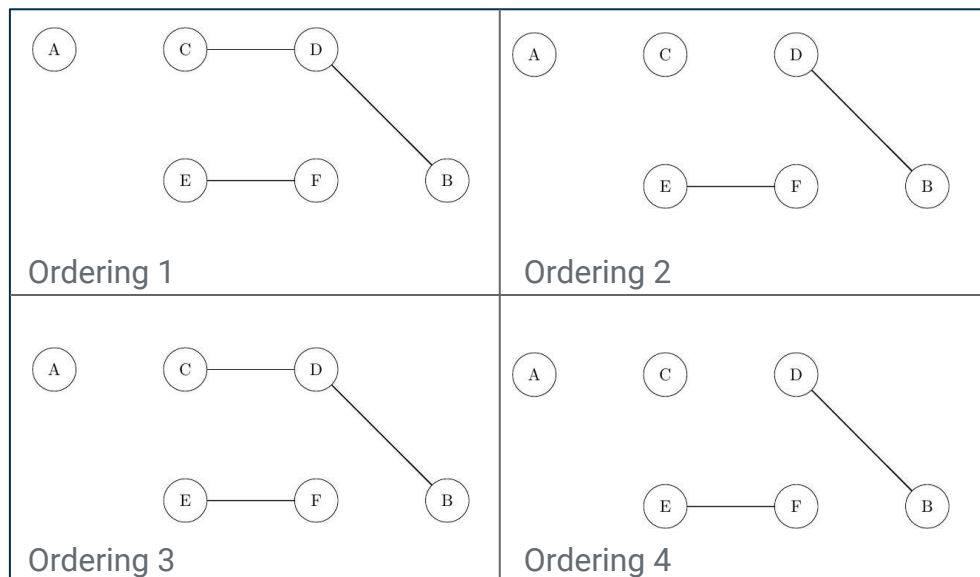
→ Compare different outcomes of original and stable PC-algorithm



true weighted skeleton

Setting 3: different ordering

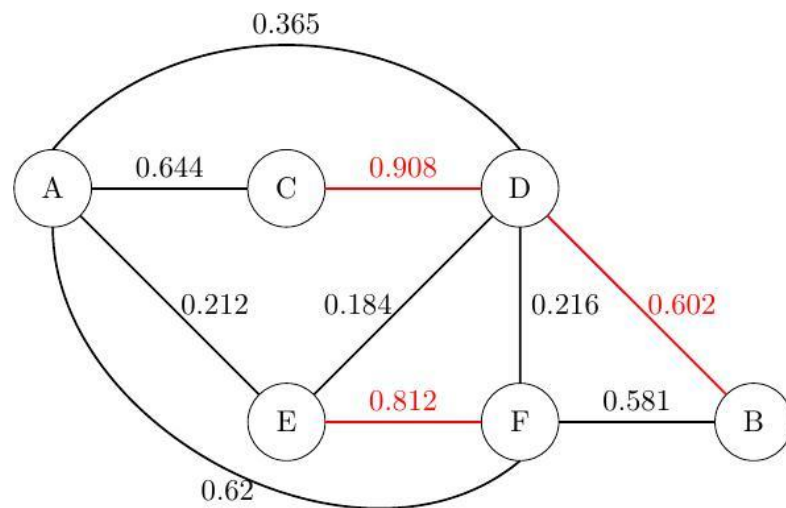
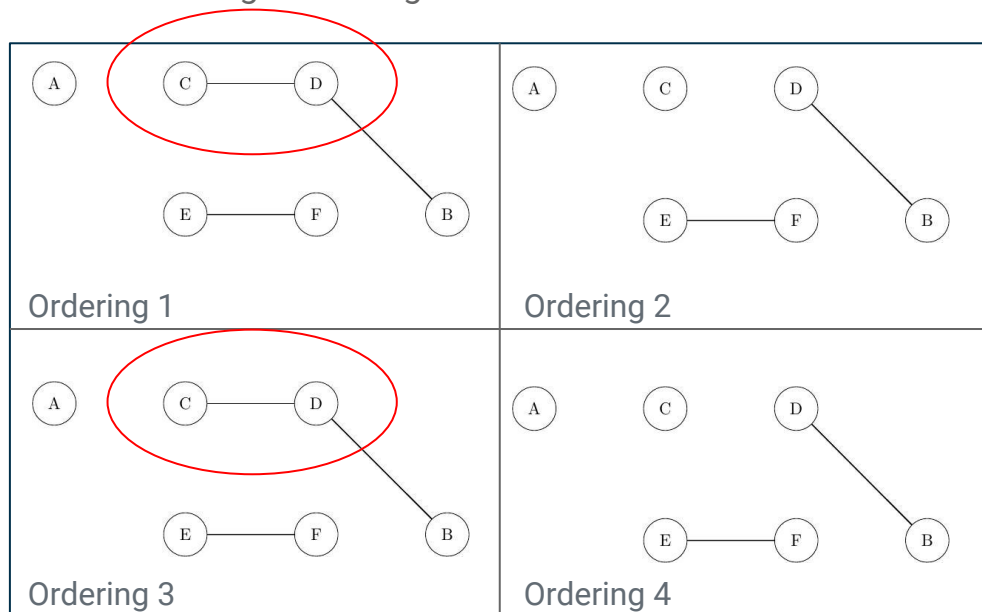
Results of original PC-algorithm



true weighted skeleton

Setting 3: different ordering

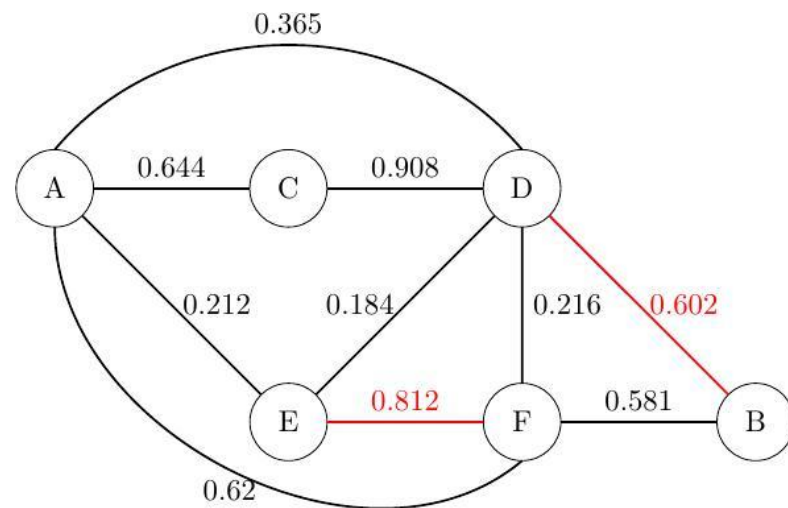
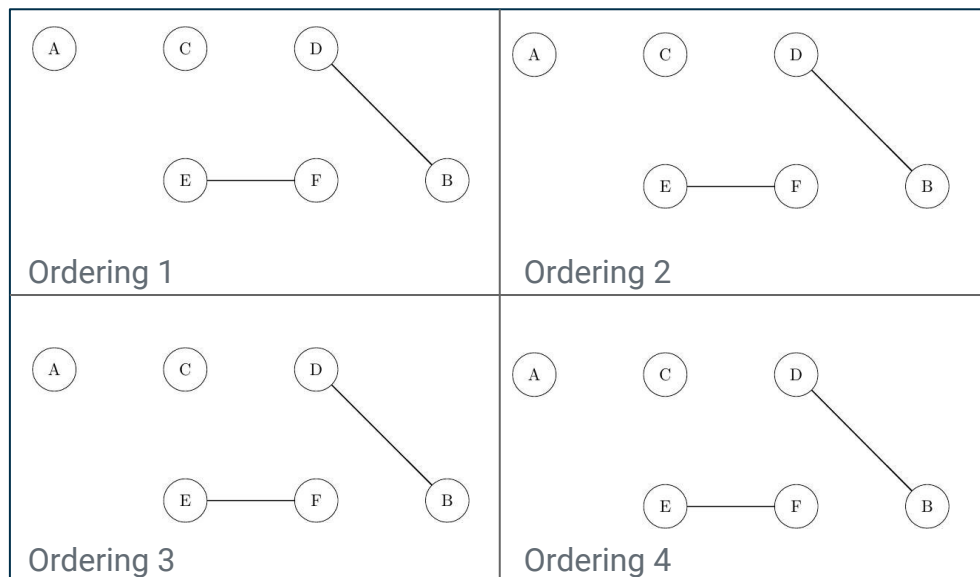
Results of original PC-algorithm



true weighted skeleton

Setting 3: different ordering

Results of stable PC-algorithm



true weighted skeleton

Setting 3: different ordering

Results for higher dimensions:

- sample size: $n = 100$
- number of nodes: $p = 50$
- significance level: $\alpha = 0.05$
- sparsity: $s = 0.06$
- expected neighborhood size: 3
- 10 different orderings

ordering	1	2	3	4	5	6	7	8	9	10
original	106	108	110	110	110	108	110	110	110	106
stable	102	102	102	102	102	102	102	102	102	102

Table 1: Number of edges in the estimated skeleton for 10 different orderings estimated by original and stable PC-algorithm.

Results of the simulation study

- **Runtime:**
 - increasing the dimensionality and increasing the sample size increases the runtime
 - parallel PC is a lot faster than stable PC (especially with high number of cores)
 - original PC: similar to parallel because of less CI tests
- **Dimension:** increasing the dimensionality improves TPR, FPR and TDR, but SHD increases, too
- **Sample size:** increasing the sample size increases the goodness of fit except for TPR

Agenda

- 1) Basic Definitions
- 2) Introduction to the PC-Algorithm
- 3) Limitations of the PC-Algorithm
- 4) The stable PC-Algorithm
- 5) The parallel-PC-Algorithm
- 6) Simulation Study in R
- 7) **Final Notes and Discussion**

Conclusion

Goal of PC-algorithm: detecting conditional independence relationships between nodes in a DAG whose underlying data-generating distribution is faithful and multivariate normal and outputs the corresponding CPDAG

Ordering

Problem:

Output graph is dependent on the order in which CI tests are performed

Solution: Stable PC-algorithm

Runtime

Problem:

Many CI tests necessary
→ high runtime

Solution: Parallel PC-algorithm

Gaussian Distribution

Problem:

Consistency only holds for Gaussian distribution
→ what if it is non-Gaussian?

Solution: Rank PC-algorithm

References

- Harris N., Drton M. (2013). *PC Algorithm for Nonparanormal Graphical Models*, Journal of Machine Learning Research, 14, p. 3365-3383.
- Le T.D., Hoang T., Li J., Liu L., Liu H., Hu S. (2014). *A fast PC algorithm for high dimensional causal discovery with multi-core PCs*, Journal of LaTeX class files, Vol. 13, No. 9.
- Koller D., Friedman N. (2009). *Probabilistic Graphical Models: Principles and Techniques*, MIT Press, Cambridge.
- Sprites P., Glymour C., Scheines R. (2000). *Causation, Prediction, and Search*, 2nd edition, MIT Press, Cambridge.
- Kalisch M., Bühlmann P. (2007). *Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm*, Journal of Machine Learning Research, 8, p. 613-636.
- Colombo D., Maathuis, M.H. (2012). *A modification of the PC algorithm yielding order-independant skeletons*, ArXiv e-prints.
- Kalisch M., Mächler M., Colombo D., Maathuis, M.H., Bühlmann P. (2012). *Causal Inference Using Graphical Models with the R package pcalg*, Journal of Statistical Software, Vol. 47, Issue 11.