Bayesian Networks

I. Bayesian Networks / 3. Constrained-based Structure Learning

Prof. Dr. Lars Schmidt-Thieme, L. B. Marinho, K. Buza

Information Systems and Machine Learning Lab (ISMLL)
Institute of Economics and Information Systems
& Institute of Computer Science
University of Hildesheim
http://www.ismll.uni-hildesheim.de

1. **Markov Equivalence and DAG patterns**

2. **PC Algorithm**

Markov-equivalence

Definition 1. Let G, H be two graphs on a set V (undirected or DAGs).
G and H are called **markov-equivalent**, if they have the same independency model, i.e.

$$I_G(X, Y|Z) \Leftrightarrow I_H(X, Y|Z), \quad \forall X, Y, Z \subseteq V$$

The notion of markov-equivalence for undirected graphs is uninteresting, as every undirected graph is markov-equivalent only to itself (corollary of uniqueness of minimal representation!).

Markov-equivalence

Why is markov-equivalence important?

1. in structure learning, the set of all graphs over V is our search space.
   $\rightsquigarrow$ if we can restrict searching to equivalence classes, the search space becomes smaller.

2. if we interpret the edges of our graph as causal relationships between variables, it is of interest,
   - which edges are necessary
     (i.e., occur in all instances of the equivalence class), and
   - which edges are only possible
     (i.e., occur in some instances of the equivalence class, but not in some others; i.e., there are alternative explanations).

Markov-equivalence

Definition 2. Let G be a directed graph. We call a chain

$$p_1 - p_2 - p_3$$

uncoupled if there is no edge between p_1 and p_3.

Lemma 1 (markov-equivalence criterion, [PGV90])**.** *Let G and H be two DAGs on the vertices V.*

G and H are markov-equivalent if and only if

(i) G and H have the same links ($u(G) = u(H)$) and

(ii) G and H have the same uncoupled head-to-head meetings.

The set of uncoupled head-to-head meetings is also denoted as **V-structure** of G.

Markov-equivalence / examples

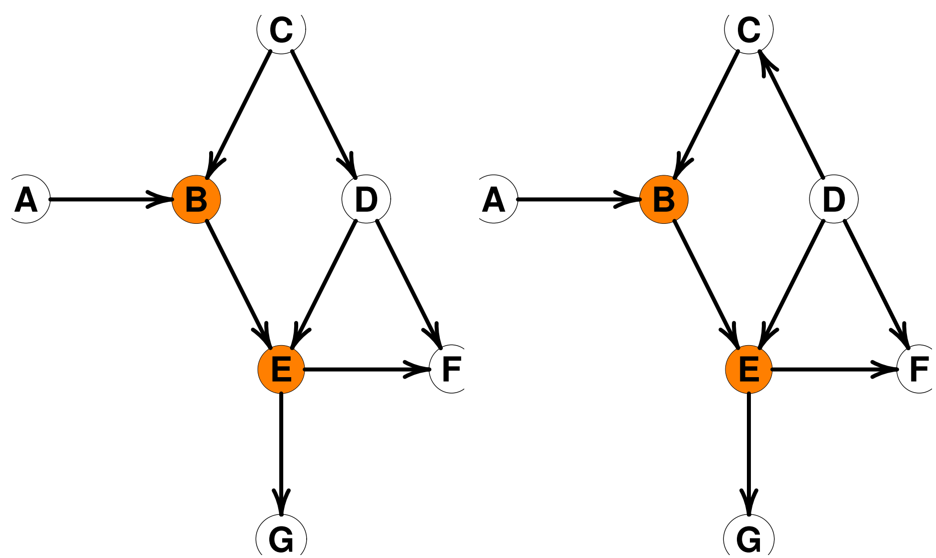


Figure 1: Example for markov-equivalent DAGs.

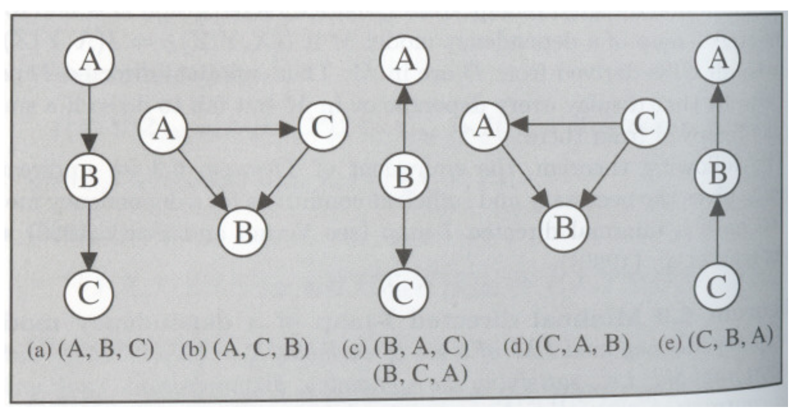


Figure 2: Which minimal DAG-representations of I are equivalent? [CGH97, p. 240]

Directed graph patterns

Definition 3. Let V be a set and $E \subseteq V^2 \cup \mathcal{P}^2(V)$ a set of ordered and unordered pairs of elements of V with $(v,w),(w,v) \notin E$ for $v,w \in V$ with $\{v,w\} \in E$.
Then $G := (V,E)$ is called a **directed graph pattern**. The elements of V are called vertices, the elemtents of E **edges**: unordered pairs are called **undirected edges**, ordered pairs **directed edges**.

We say, a directed graph pattern H is **a pattern of the directed graph** G, if there is an orientation of the unoriented edges of H that yields G, i.e.

$$(v,w) \in E_G \Rightarrow \begin{cases} (v,w) \in E_H \text{ or} \\ \{v,w\} \in E_H \end{cases}$$

$$(v,w) \in E_G \Leftarrow (v,w) \in E_H$$

$$\left.\begin{array}{l} (v,w) \in E_G \text{ or} \\ (w,v) \in E_G \end{array}\right\} \Leftarrow \{v,w\} \in E_H$$

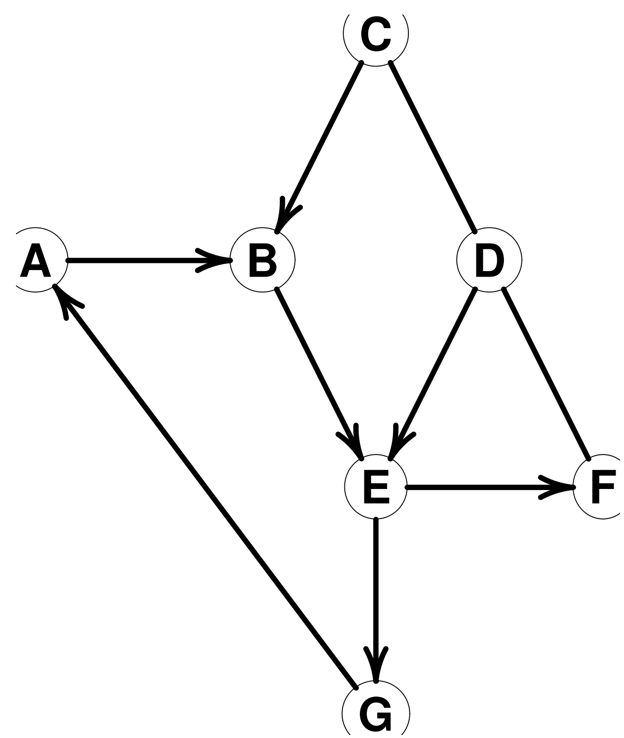


Figure 3: Directed graph pattern.

DAG patterns

Definition 4. A directed graph pattern H is called an **acyclic directed graph pattern** (DAG pattern), if

- it is the directed graph pattern of a DAG G

  or equivalently

- H does not contain a completely directed cycle, i.e. there is no sequence $v_1, \ldots, v_n \in V$ with $(v_i, v_{i+1}) \in E$ for $i = 1, \ldots, n-1$ (i.e. the directed graph got by dropping undirected edges is a DAG).

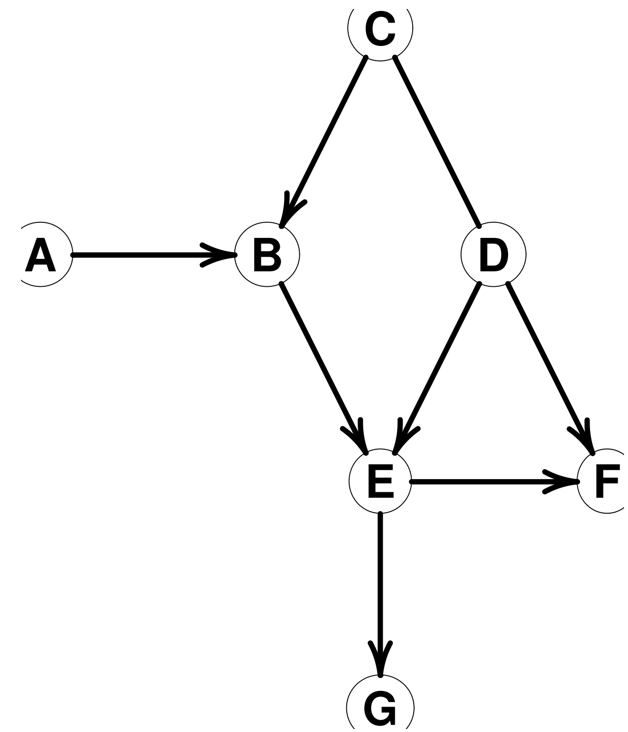


Figure 4: DAG pattern.

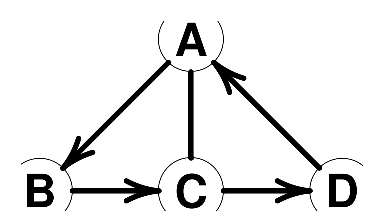


Figure 5: Directed graph pattern that is not a DAG pattern.

DAG patterns represent markov equivalence classes

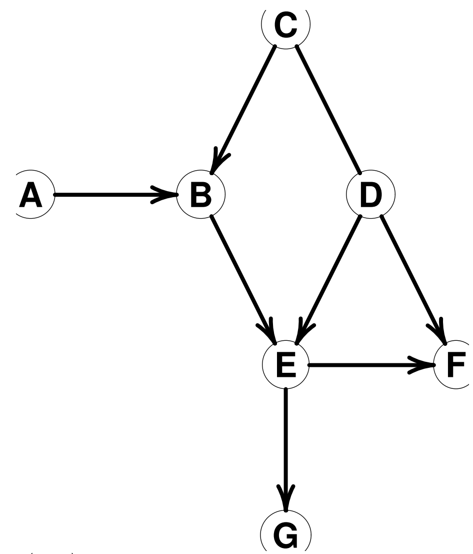
Lemma 2. *Each markov equivalence class corresponds uniquely to a DAG pattern G:*

(i) The markov equivalence class consists of all DAGs that G is a pattern of, i.e., that give G by dropping the directions of some edges that are not part of an uncoupled head-to-head meeting,

(ii) The DAG pattern contains a directed edge (v, w), if all representatives of the markov equivalence class contain this directed edge, otherwise (i.e. if some representatives have (v, w), some others (w, v)) the DAG pattern contains the undirected edge $\{v, w\}$.

The directed edges of the DAG pattern are also called **irreversible** or **compelled**, the undirected edges are also called **reversible**.

DAG patterns represent markov equivalence classes / example



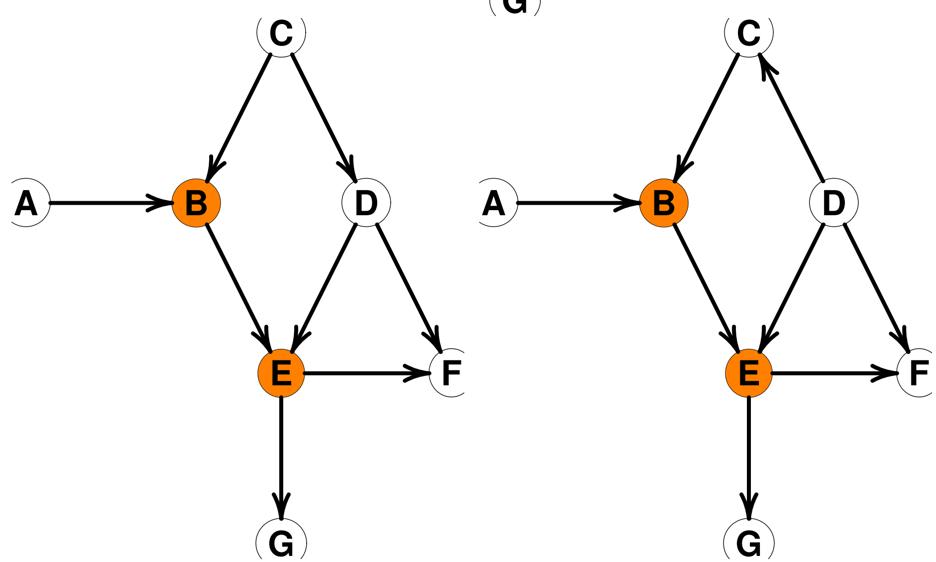
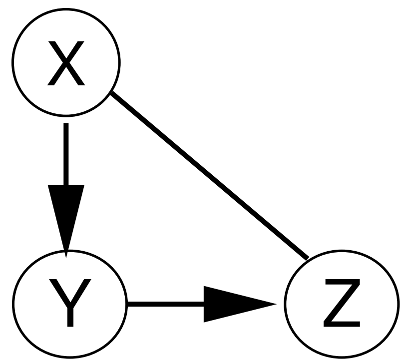


Figure 6: DAG pattern and its markov equivalence class representatives.

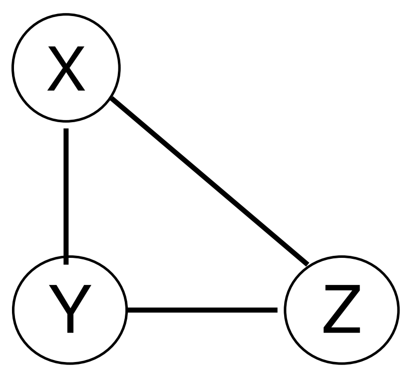
DAG patterns represent markov equivalence classes

But beware, not every DAG pattern represents a Markov-equivalence class !

Example:



is not a DAG pattern of a Markov-equivalence class, but

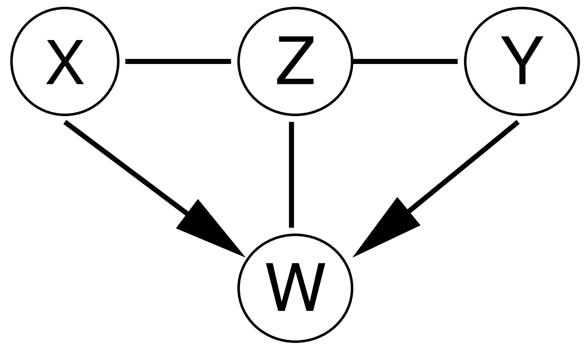


is.

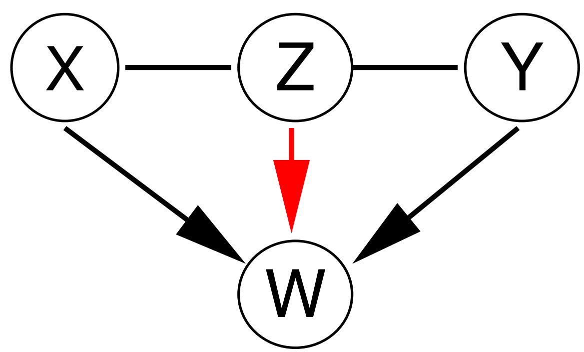
DAG patterns represent markov equivalence classes

But just skeleton plus uncoupled head-to-head meetings do not make a DAG pattern that represents a markov-equivalence class either.

Example:



is not a DAG pattern that represents a Markov-equivalence class, as any of its represenatives also has $Z \rightarrow W$. But



is.

Computing DAG patterns

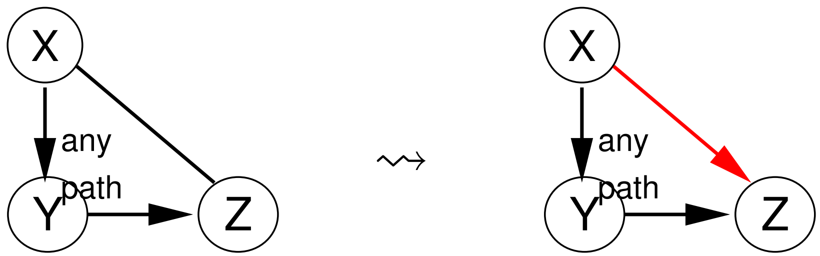
So, to compute the DAG pattern that represents the equivalence class of a given DAG,

1. start with the skeleton plus all head-to-head-meetings,

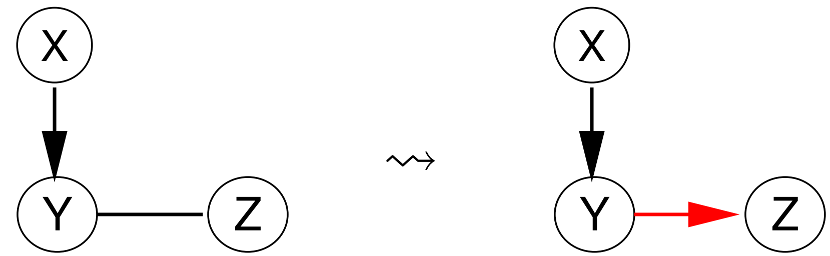
2. add entailed edges successively (saturating).

Saturating DAG patterns

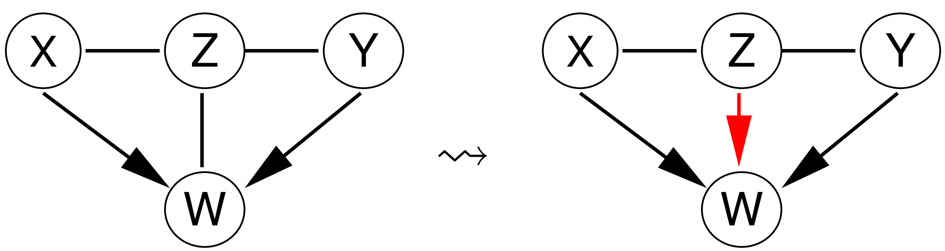
rule 1:



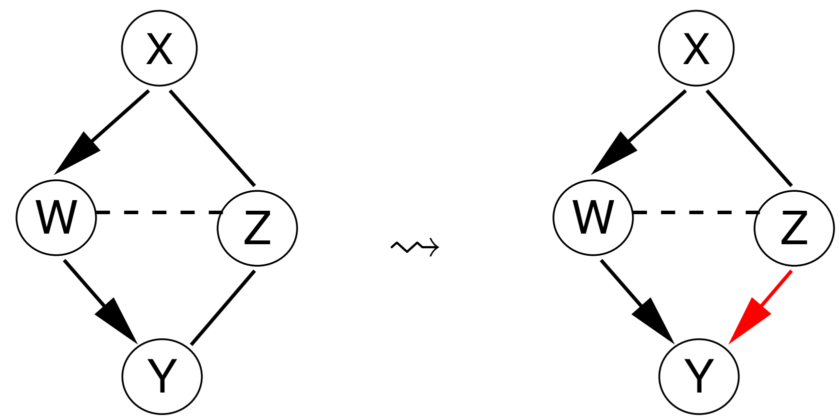
rule 2:



rule 3:



rule 4:



Dashed link can be $W \to Z$, $W \leftarrow Z$, or $W - Z$
(so rule 4 is actually a compact notation for 3 rules).

Computing DAG patterns

saturate(graph pattern $G = (V, E)$) :
apply rules 1–4 to G until no more rule matches
return G

dag-pattern(graph $G = (V, E)$) :
$H := (V, F)$ with $F := \{\{x, y\} \mid (x, y) \in E\}$
for $X \rightarrow Z \leftarrow Y$ uncoupled head-to-head-meeting in G **do**
    orient $X \rightarrow Z \leftarrow Y$ in H
od
saturate(H)
return H

Figure 7: Algorithm for computing the DAG pattern of the Markov-equivalence class of a given DAG.

Lemma 3. *For a given graph G, algorithm 7 computes correctly the DAG pattern that represents its Markov-equivalence class.*

Furthermore, here, even the rule set 1–3 will do and is non-redundant.

See [Mee95] for a proof.

Summary

- Some DAGs encode the same independency relation (**Markov equivalence**).
- A Markov equivalence class can be represented by a **DAG pattern**.
  (but not all DAG patterns represent a Markov equivalence class!)
- For a given DAG, its DAG pattern can be computed by
  1. start from the undirected skeleton,
  2. add all directions of **uncoupled head-to-head meetings**,
  3. **saturate infered directions** (using 3 rules).

1. **Markov Equivalence and DAG patterns**

2. **PC Algorithm**

Types of Methods for Structure Learning

There are three types of structure learning algorithms for Bayesian networks:

1. **constrained-based learning** (e.g., PC),
2. **searching with a target function** (e.g., K2),
3. **hybrid methods** (e.g., sparse candidate).

Computing the Skeleton

Lemma 4 (Edge Criterion)**.** *Let* $G := (V, E)$ *be a DAG and* $X, Y \in V$. *Then it is equivalent:*

(i) X and Y cannot be separated by any $\mathcal{Z}$, i.e.,

$$\neg I_G(X, Y \mid \mathcal{Z}) \quad \forall \mathcal{Z} \subseteq V \setminus \{X, Y\}$$

(ii) There is an edge between X and Y, i.e.,

$$(X, Y) \in E \text{ or } (Y, X) \in E$$

Definition 5. Any $\mathcal{Z} \subseteq V \setminus \{X, Y\}$ with $I_G(X, Y \mid \mathcal{Z})$ is called a **separator of** X **and** Y.

$$\mathrm{Sep}(X, Y) := \{\mathcal{Z} \subseteq V \setminus \{X, Y\} \mid I_G(X, Y \mid \mathcal{Z})\}$$

Computing the Skeleton / Separators

```
separators-basic(set of variables V, independency relation I) :
Allocate S : P^2(V) → P(V) ∪ {none}
for {X, Y} ⊆ V do
    S({X, Y}) := none
    for T ⊆ V \ {X, Y} do
        if I(X, Y | T)
            S({X, Y}) := T
            break
        fi
    od
od
return S
```

Figure 8: Compute a separator for each pair of variables.

Example / 1/3 – Computing the Skeleton

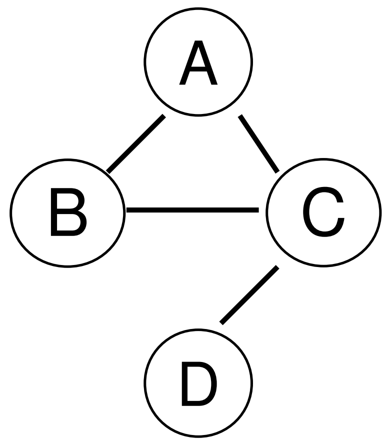
Let I be the following independency structure:

$$I(A, D \,|\, C), \quad I(A, D \,|\, \{C, B\}), \quad I(B, D)$$

Then we can compute the following separators:

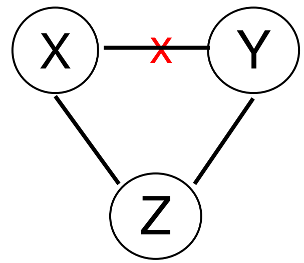
$$S(A, B) := \text{none}$$
$$S(A, C) := \text{none}$$
$$S(A, D) := \{C\}$$
$$S(B, C) := \text{none}$$
$$S(B, D) := \emptyset$$
$$S(C, D) := \text{none}$$

Thus, the skeleton of the Bayesian Network representing I looks like



Computing the V-structure

Lemma 5 (Uncoupled Head-to-head Meeting Criterion). *Let* $G := (V, E)$ *be a DAG,* $X, Y, Z \in V$ *with*



Then it is equivalent:

(i) $X \to Z \leftarrow Y$ is an uncoupled head-to-head meeting, i.e.,

$$(X, Z), (Y, Z) \in E, (X, Y), (Y, X) \notin E$$

(ii) Z is not contained in any separator of X and Y, i.e.,

$$Z \notin S \quad \forall S \in \mathrm{Sep}(X, Y)$$

(iii) Z is not contained in at least one separator of X and Y, i.e.,

$$Z \notin S \quad \exists S \in \mathrm{Sep}(X, Y)$$

Computing Skeleton and V-structure

vstructure(set of variables V, independency relation I) :
$S :=$ *separators*(V, I)
$G := (V, E)$ with $E := \{\{X, Y\} \mid S(\{X, Y\}) = \text{none}\}$
for $X, Y, Z \in V$ with $X - Z - Y, X \not- Y$ **do**
    if $Z \notin S(X, Y)$
        orient $X - Z - Y$ as $X \rightarrow Z \leftarrow Y$
    fi
od
return G

Figure 9: Compute skeleton and v-structure.

learn-structure-pc(set of variables V, independency relation I) :
$G :=$ *vstructure*(V, I)
saturate(G)
return G

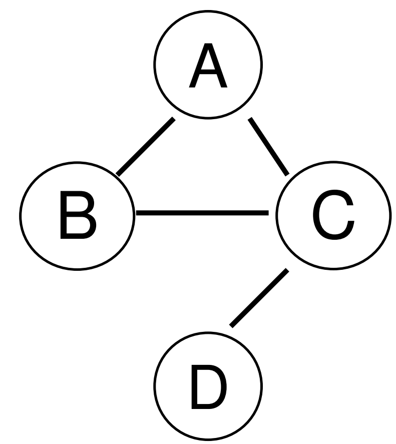
Figure 10: Learn structure of a Bayesian Network (SGS/PC algorithm, [**?**]).

Example / 2/3 – Computing the V-Structure

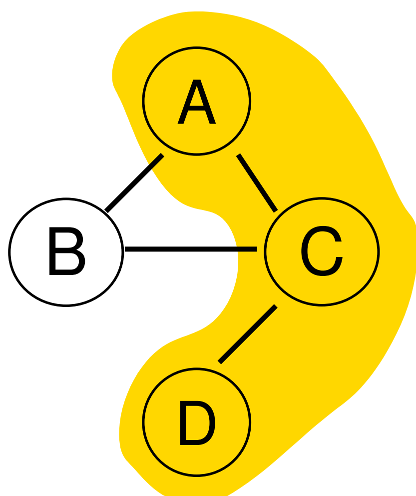
Separators:

$$S(A,B) := \text{none}$$
$$S(A,C) := \text{none}$$
$$S(A,D) := \{C\}$$
$$S(B,C) := \text{none}$$
$$S(B,D) := \emptyset$$
$$S(C,D) := \text{none}$$

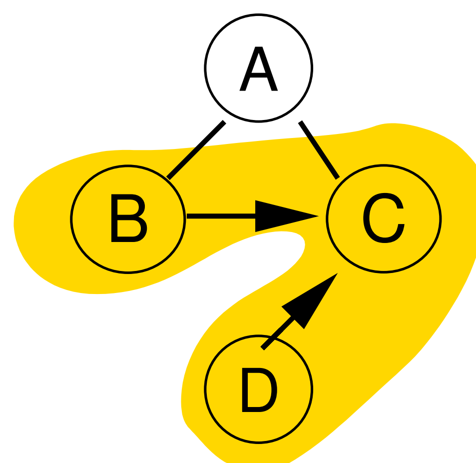
Skeleton:



Checking A–C–D:

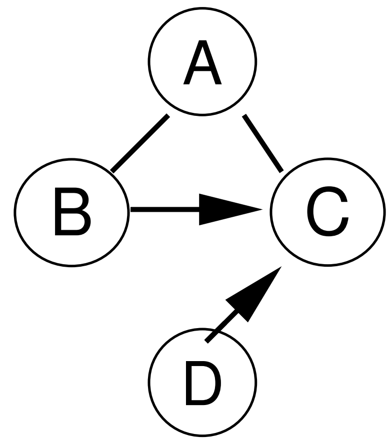


Checking B–C–D:

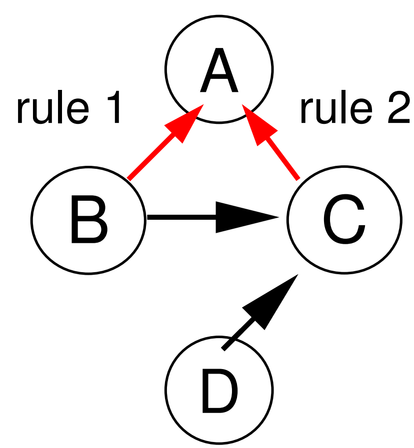


Example / 3/3 – Saturating

Skeleton and v-structure:



Saturating:



Number of Independency Tests

Let there be n variables.

For each of the $\binom{n}{2}$ pairs of variables, there are 2^{n-2} candidates for possible separators.

$$\text{number of } I\text{-tests} = \binom{n}{2} 2^{n-2}$$

Example: $n = 4$:

$$\binom{n}{2} 2^{n-2} = \binom{4}{2} 2^2 = 6 \cdot 4 = 24$$

If we start with small separators and stop once a separator has been found, we still have to check

$$4 \cdot (1 + 2 + 1) + 1 \cdot (1 + 2) + 1 \cdot 1 = 20$$

Number of Independency Tests

Can we reduce the number of tests for a given pair of variable by reusing results for other pairs of variables?

Lemma 6. *Let* $G := (V, E)$ *be a DAG and* $X, Y \in V$ *separated. Then*

$$I(X, Y \mid \mathrm{pa}(X)) \quad \textit{or} \quad I(X, Y \mid \mathrm{pa}(Y))$$

As we do not know directions of edges at the skeleton recovery step, we use the weaker result:

$$I(X, Y \mid \mathrm{fan}(X)) \quad \text{or} \quad I(X, Y \mid \mathrm{fan}(Y))$$

Computing the Skeleton / Separators

separators-remove-edges(separator map S, skeleton graph G, independency relation I)
$i := 0$
while $\exists X \in V : |\operatorname{fan}_G(X)| > i$ **do**
  for $\{X, Y\} \in E$ with $\operatorname{fan}_G(X)| > i$ or $\operatorname{fan}_G(Y)| > i$ **do**
    for $T \in \mathcal{P}^i(\operatorname{fan}_G(X) \setminus \{Y\}) \cup \mathcal{P}^i(\operatorname{fan}_G(Y) \setminus \{X\})$ **do**
      if $I(X, Y \mid T)$
        $S(\{X, Y\}) := T$
        $E := E \setminus \{\{X, Y\}\}$
        break
      fi
    od
  od
  $i := i + 1$
od
return S

separators-interlaced(set of variables V, independency relation I) :
Allocate $S : \mathcal{P}^2(V) \rightarrow \mathcal{P}(V) \cup \{\text{none}\}$
$S(\{X, Y\}) := \text{none} \quad \forall \{X, Y\} \subseteq V$
$G := (V, E)$ with $E := \mathcal{P}^2(V)$
separators-remove-edges(S, G, I)
return S

Figure 11: Compute a separator for each pair of variables.

Example / Computing the Separators (1/3)

$$I(A, D \mid C), \quad I(A, D \mid \{C, B\}), \quad I(B, D)$$

$i = 0$:

A, B, $T = \emptyset$: —

C, $T = \emptyset$: —

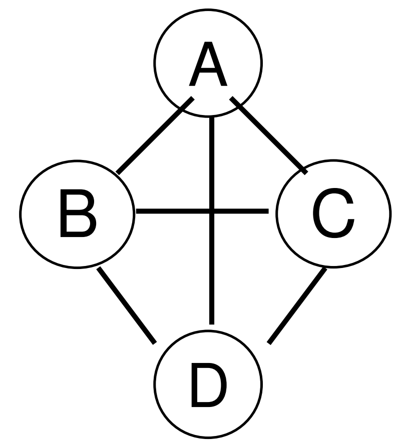
D, $T = \emptyset$: —

B, C, $T = \emptyset$: —

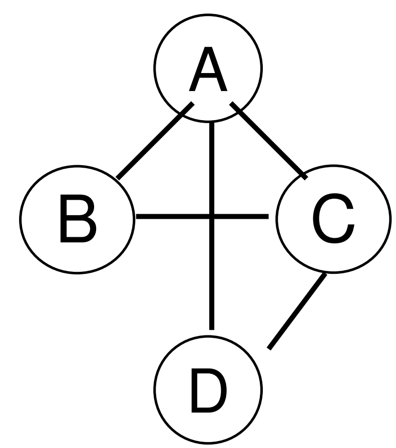
D, $T = \emptyset$: $D(\{B, D\}) = \emptyset$

C, D, $T = \emptyset$: —

initial graph:



after update for B, D:



Example / Computing the Separators (2/3)

$$I(A, D \mid C), \quad I(A, D \mid \{C, B\}), \quad I(B, D)$$

$i = 1$:

A, B, $T = \{C\}, \{D\}$: —

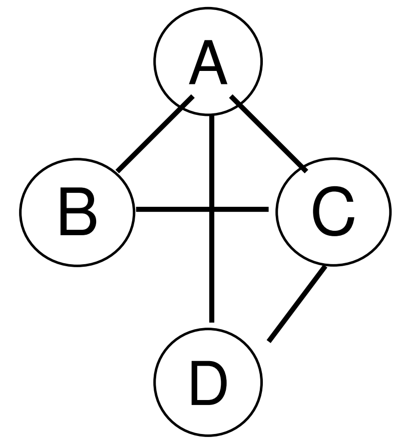
C, $T = \{B\}, \{D\}$: —

D, $T = \{B\}, \{C\}$: $S(\{A, D\}) = \{C\}$

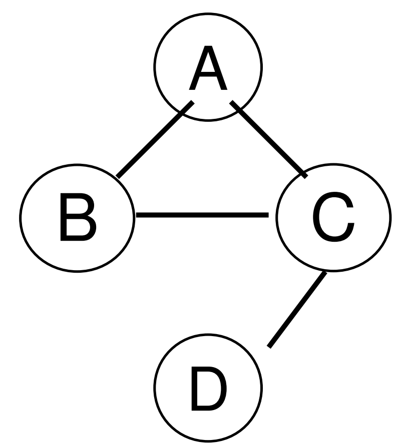
B, C, $T = \{A\}, \{D\}$: —

C, D, $T = \{A\}, \{B\}$: —

after update for B, D:



after update for A, D:



Example / Computing the Separators (3/3)

$$I(A, D \mid C), \quad I(A, D \mid \{C, B\}), \quad I(B, D)$$

$i = 2:$

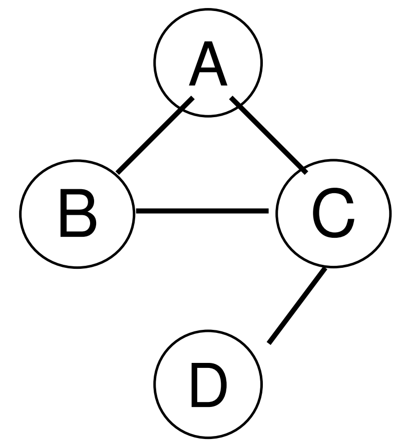
A, C, $T = \{B, D\}$: —

B, C, $T = \{A, D\}$: —

C, D, $T = \{A, B\}$: —

total: 19 I-tests.

after update for A, D:



Algorithms – SGS vs. PC

SGS/PC with `separators-basic` is called **SGS algorithm** ([**?**], 1990).

SGS/PC with `separators-interlaced` is called **PC algorithm** ([**?**], 1991).

Implementations are available:

- in Tetrad
  `http://www.phil.cmu.edu/projects/tetrad/`
  (class files & javadocs, no sources)

- in Hugin (commercial).

References

[CGH97] Enrique Castillo, José Manuel Gutiérrez, and Ali S. Hadi. *Expert Systems and Probabilistic Network Models*. Springer, New York, 1997.

[Mee95] C. Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence,* Montreal, QU, pages 403–418. Morgan Kaufmann, August 1995.

[PGV90] J. Pearl, D. Geiger, and T. S. Verma. The logic of influence diagrams. In R. M. Oliver and J. Q. Smith, editors, *Influence Diagrams, Belief Networks and Decision Analysis*. Wiley, Sussex, 1990. (a shorter version originally appeared in Kybernetica, Vol. 25, No. 2, 1989).