Learning causal DAGs using adaptive interventions

Davin Choo

This talk is based on joint work with
Arnab Bhattacharyya, Themis Gouleakis, Kirankumar Shiragur



Suppose we are given some data and we want to discover causal relationships between them

	X_1	X_2	X_3	X_4	X_5	X_6
Sample 1	0.22	0.04	0.84	0.48	0.98	0.82
Sample 2	0.87	0.17	0.61	0.67	0.67	0.23
Sample 3	0.55	0.54	0.67	0.86	0.93	0.23
…	…	…	…	…	…	…
Sample M	0.12	0.95	0.79	0.47	0.05	0.92

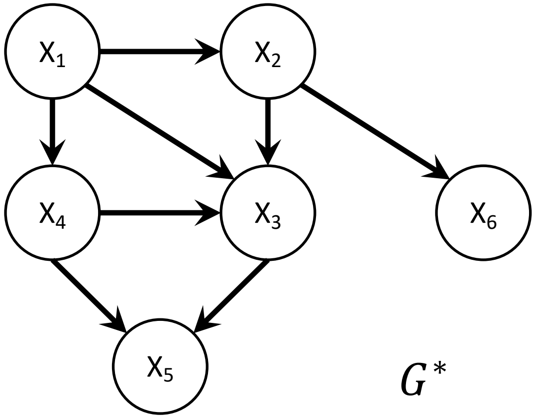
Suppose we are given some data and we want to discover causal relationships between them

	X_1	X_2	X_3	X_4	X_5	X_6
Sample 1	0.22	0.04	0.84	0.48	0.98	0.82
Sample 2	0.87	0.17	0.61	0.67	0.67	0.23
Sample 3	0.55	0.54	0.67	0.86	0.93	0.23
…	…	…	…	…	…	…
Sample M	0.12	0.95	0.79	0.47	0.05	0.92

Genetics	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6
Finance	AAPL	GOOGL	MSFT	AMZN	META	TSLA
…	…	…	…	…	…	…
Health care	Diet	Exercise	Weight	Blood pressure	Blood glucose	Cholesterol levels

One possible way: use graphical modelling

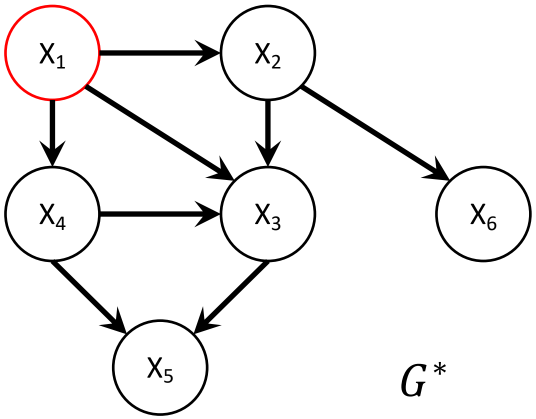
	X_1	X_2	X_3	X_4	X_5	X_6
Sample 1	0.22	0.04	0.84	0.48	0.98	0.82
Sample 2	0.87	0.17	0.61	0.67	0.67	0.23
Sample 3	0.55	0.54	0.67	0.86	0.93	0.23
…	…	…	…	…	…	…
Sample M	0.12	0.95	0.79	0.47	0.05	0.92



G^*

A directed acyclic graphs (DAG) representation

	X_1	X_2	X_3	X_4	X_5	X_6
Sample 1	0.22	0.04	0.84	0.48	0.98	0.82
Sample 2	0.87	0.17	0.61	0.67	0.67	0.23
Sample 3	0.55	0.54	0.67	0.86	0.93	0.23
...	...	...	...	...	...	...
Sample M	0.12	0.95	0.79	0.47	0.05	0.92



G^*

$$X_1 = f_1(\epsilon_1)$$

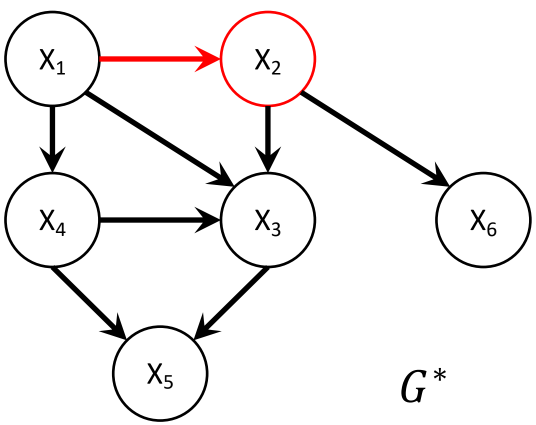
Structural equation model (SEM)

ϵ_1

noise

A directed acyclic graphs (DAG) representation

	X_1	X_2	X_3	X_4	X_5	X_6
Sample 1	0.22	0.04	0.84	0.48	0.98	0.82
Sample 2	0.87	0.17	0.61	0.67	0.67	0.23
Sample 3	0.55	0.54	0.67	0.86	0.93	0.23
…	…	…	…	…	…	…
Sample M	0.12	0.95	0.79	0.47	0.05	0.92



G^*

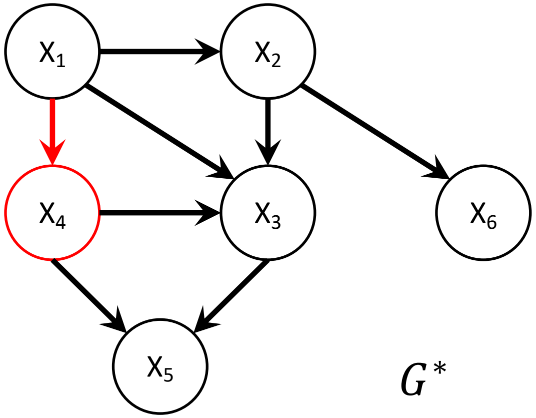
$$X_1 = f_1(\epsilon_1)$$
$$X_2 = f_2(X_1, \epsilon_2)$$

Structural equation model (SEM)

$\epsilon_1, \epsilon_2,$ independent noise

A directed acyclic graphs (DAG) representation

	X_1	X_2	X_3	X_4	X_5	X_6
Sample 1	0.22	0.04	0.84	0.48	0.98	0.82
Sample 2	0.87	0.17	0.61	0.67	0.67	0.23
Sample 3	0.55	0.54	0.67	0.86	0.93	0.23
…	…	…	…	…	…	…
Sample M	0.12	0.95	0.79	0.47	0.05	0.92



G^*

$$X_1 = f_1(\epsilon_1)$$
$$X_2 = f_2(X_1, \epsilon_2)$$

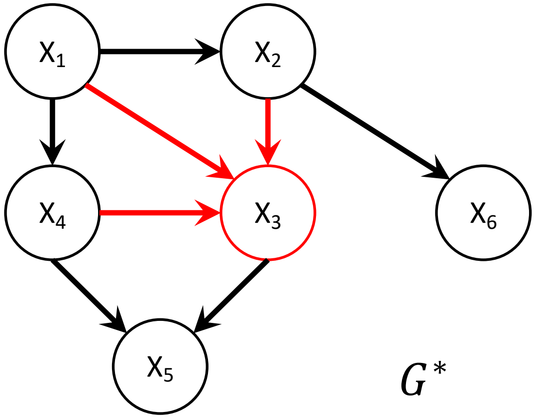
$$X_4 = f_4(X_1, \epsilon_4)$$

Structural equation model (SEM)

$\epsilon_1, \epsilon_2, \quad \epsilon_4$ independent noise

A directed acyclic graphs (DAG) representation

	X_1	X_2	X_3	X_4	X_5	X_6
Sample 1	0.22	0.04	0.84	0.48	0.98	0.82
Sample 2	0.87	0.17	0.61	0.67	0.67	0.23
Sample 3	0.55	0.54	0.67	0.86	0.93	0.23
…	…	…	…	…	…	…
Sample M	0.12	0.95	0.79	0.47	0.05	0.92



G^*

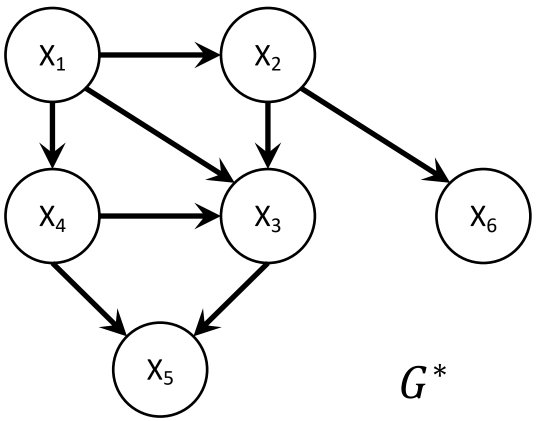
$$X_1 = f_1(\epsilon_1)$$
$$X_2 = f_2(X_1, \epsilon_2)$$
$$X_3 = f_3(X_1, X_2, X_4, \epsilon_3)$$
$$X_4 = f_4(X_1, \epsilon_4)$$

Structural equation model (SEM)

$\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4$ independent noise

A directed acyclic graphs (DAG) representation

	X_1	X_2	X_3	X_4	X_5	X_6
Sample 1	0.22	0.04	0.84	0.48	0.98	0.82
Sample 2	0.87	0.17	0.61	0.67	0.67	0.23
Sample 3	0.55	0.54	0.67	0.86	0.93	0.23
…	…	…	…	…	…	…
Sample M	0.12	0.95	0.79	0.47	0.05	0.92



G^*

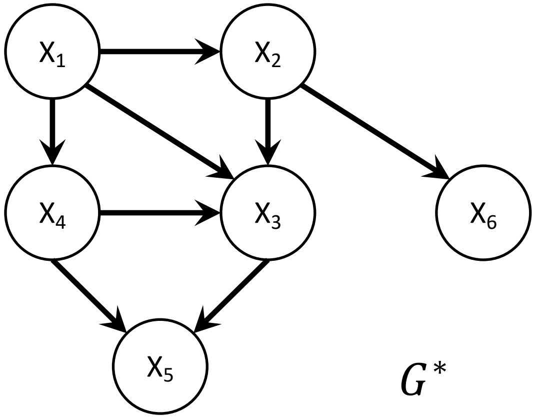
$$
\begin{aligned}
X_1 &= f_1(\epsilon_1) \\
X_2 &= f_2(X_1, \epsilon_2) \\
X_3 &= f_3(X_1, X_2, X_4, \epsilon_3) \\
X_4 &= f_4(X_1, \epsilon_4) \\
X_5 &= f_5(X_3, X_4, \epsilon_5) \\
X_6 &= f_6(X_2, \epsilon_6)
\end{aligned}
$$

$\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5, \epsilon_6$ independent noise

Structural equation model (SEM)

A directed acyclic graphs (DAG) representation

	X_1	X_2	X_3	X_4	X_5	X_6
Sample 1	0.22	0.04	0.84	0.48	0.98	0.82
Sample 2	0.87	0.17	0.61	0.67	0.67	0.23
Sample 3	0.55	0.54	0.67	0.86	0.93	0.23
…	…	…	…	…	…	…
Sample M	0.12	0.95	0.79	0.47	0.05	0.92



G^*

$$X_1 = f_1(\epsilon_1)$$
$$X_2 = f_2(X_1, \epsilon_2)$$
$$X_3 = f_3(X_1, X_2, X_4, \epsilon_3)$$
$$X_4 = f_4(X_1, \epsilon_4)$$
$$X_5 = f_5(X_3, X_4, \epsilon_5)$$
$$X_6 = f_6(X_2, \epsilon_6)$$

$\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5, \epsilon_6$ independent noise

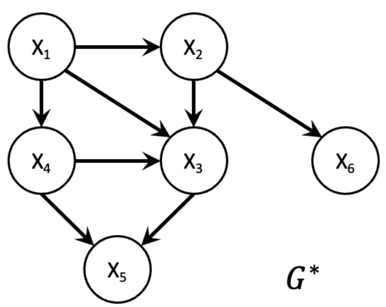
Structural equation model (SEM)

Using the Bayesian network, one can decompose the joint distribution as follows:

$$\Pr[X_1] \cdot \Pr[X_2 | X_1] \cdot \Pr[X_4 | X_1] \cdot \Pr[X_3 | X_1, X_2, X_4] \cdot \Pr[X_5 | X_3, X_4] \cdot \Pr[X_6 | X_2]$$

Conditional independence (CI) tests

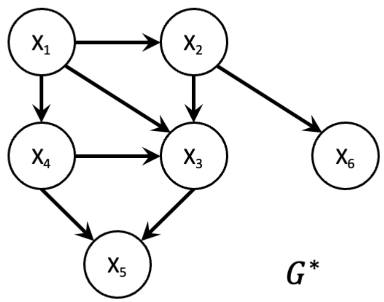
- A standard way (under some causal assumptions*) to recover graph structure from data is to perform CI tests
    - e.g. PC (Peter-Clark) algorithm* [Spirtes, Glymour, Scheines, Heckerman 2000]



*See backup slides if time permits

Conditional independence (CI) tests

- A standard way (under some causal assumptions*) to recover graph structure from data is to perform CI tests
  - e.g. PC (Peter-Clark) algorithm* [Spirtes, Glymour, Scheines, Heckerman 2000]



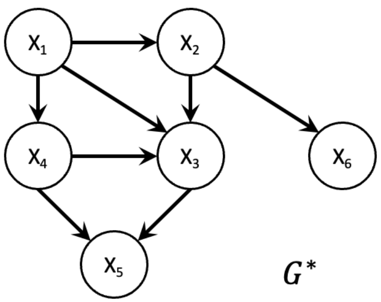
Get samples

	X_1	X_2	X_3	X_4	X_5	X_6
Sample 1	0.22	0.04	0.84	0.48	0.98	0.82
Sample 2	0.87	0.17	0.61	0.67	0.67	0.23
Sample 3	0.55	0.54	0.67	0.86	0.93	0.23
...	...	...	...	...	...	...
Sample M	0.12	0.95	0.79	0.47	0.05	0.92

*See backup slides if time permits

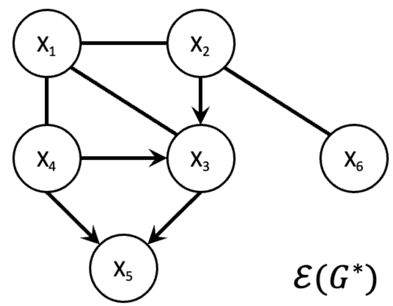
Conditional independence (CI) tests

- A standard way (under some causal assumptions*) to recover graph structure from data is to perform CI tests
  - e.g. PC (Peter-Clark) algorithm* [Spirtes, Glymour, Scheines, Heckerman 2000]



G^*

Essential graph $\mathcal{E}(G^*)$
Partially oriented G^* that represents the equivalence class $[G^*]$



$\mathcal{E}(G^*)$

(Recover up to an equivalence class)

Get samples

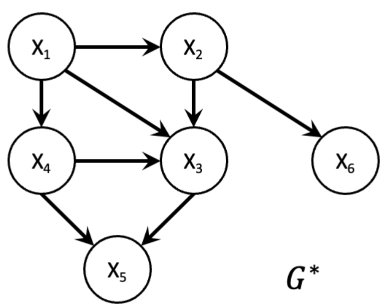
	X_1	X_2	X_3	X_4	X_5	X_6
Sample 1	0.22	0.04	0.84	0.48	0.98	0.82
Sample 2	0.87	0.17	0.61	0.67	0.67	0.23
Sample 3	0.55	0.54	0.67	0.86	0.93	0.23
...	...	...	...	...	...	...
Sample M	0.12	0.95	0.79	0.47	0.05	0.92

Do CI tests
- Recover skeleton
- Orient *some* edges

*See backup slides if time permits

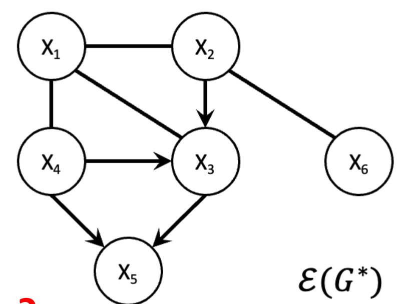
Conditional independence (CI) tests

- A standard way (under some causal assumptions*) to recover graph structure from data is to perform CI tests
  - e.g. PC (Peter-Clark) algorithm* [Spirtes, Glymour, Scheines, Heckerman 2000]



G^*

Essential graph $\mathcal{E}(G^*)$
Partially oriented G^* that represents the equivalence class $[G^*]$



$\mathcal{E}(G^*)$

What are these kinds of edges?
What makes them special?

(Recover up to an equivalence class)

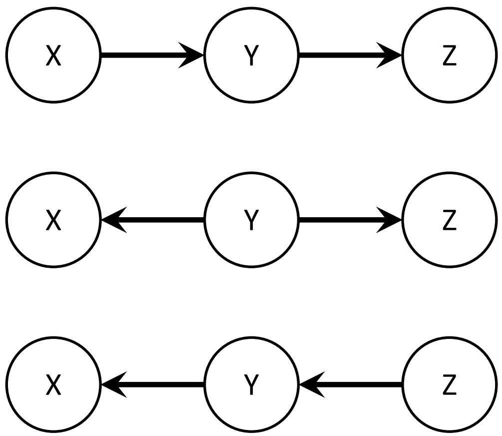
Get samples

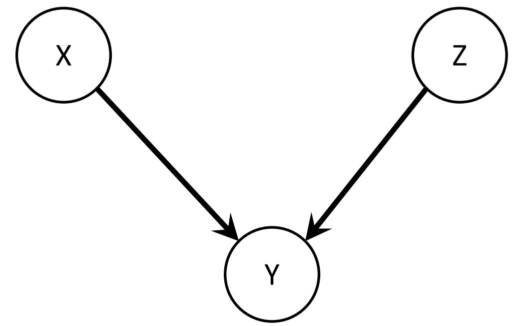
	X_1	X_2	X_3	X_4	X_5	X_6
Sample 1	0.22	0.04	0.84	0.48	0.98	0.82
Sample 2	0.87	0.17	0.61	0.67	0.67	0.23
Sample 3	0.55	0.54	0.67	0.86	0.93	0.23
…	…	…	…	…	…	…
Sample M	0.12	0.95	0.79	0.47	0.05	0.92

Do CI tests
- Recover skeleton
- Orient *some* edges

*See backup slides if time permits

Unshielded colliders / v-structures



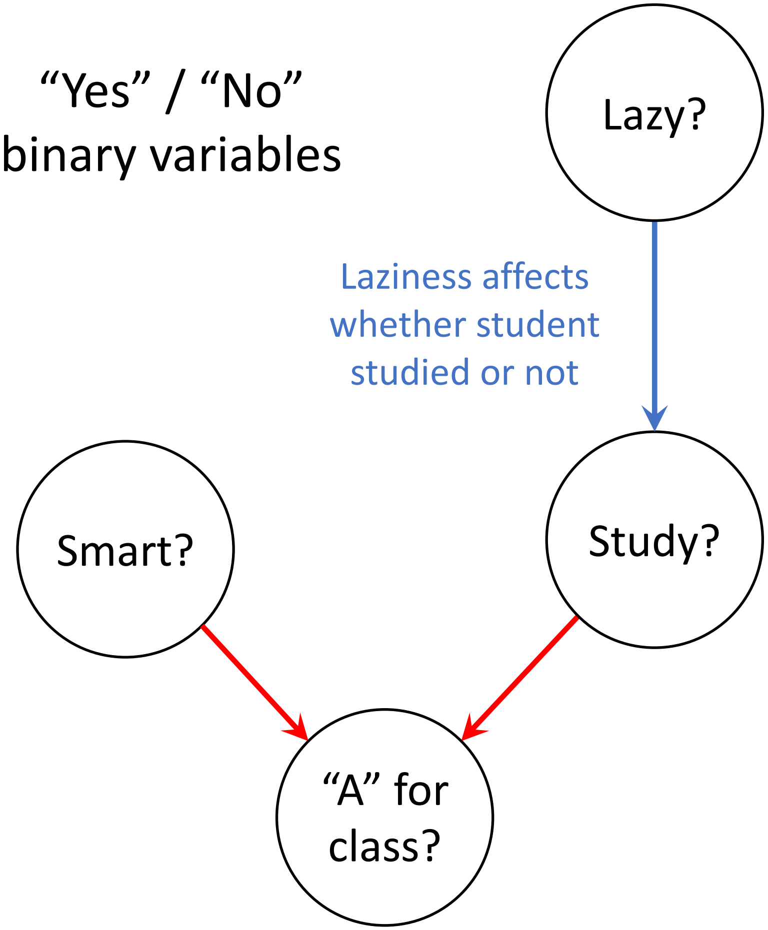


$$X \not\perp\!\!\!\perp Y$$
$$X \not\perp\!\!\!\perp Z$$
$$Y \not\perp\!\!\!\perp Z$$
$$X \not\perp\!\!\!\perp Y \mid Z$$
$$X \perp\!\!\!\perp Z \mid Y$$
$$Y \not\perp\!\!\!\perp Z \mid X$$

$$X \not\perp\!\!\!\perp Y$$
$$X \perp\!\!\!\perp Z$$
$$Y \not\perp\!\!\!\perp Z$$
$$X \not\perp\!\!\!\perp Y \mid Z$$
$$X \not\perp\!\!\!\perp Z \mid Y$$
$$Y \not\perp\!\!\!\perp Z \mid X$$

Toy example

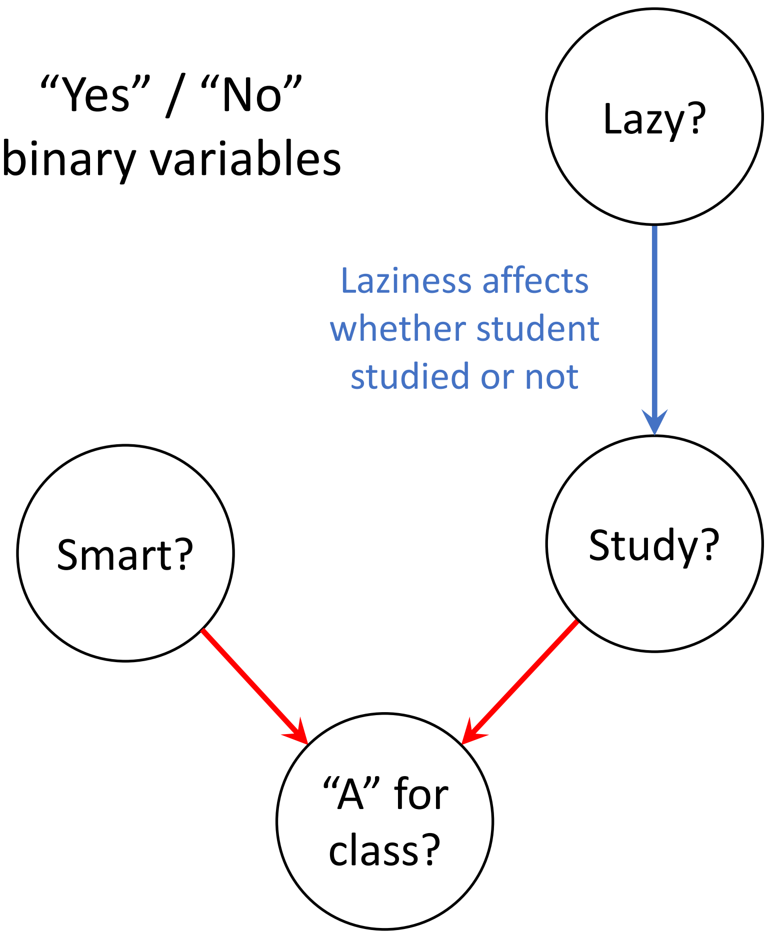
“Yes” / “No”
binary variables



Chance of “A” depends on whether student studied and whether student is smart

Toy example

"Yes" / "No"
binary variables



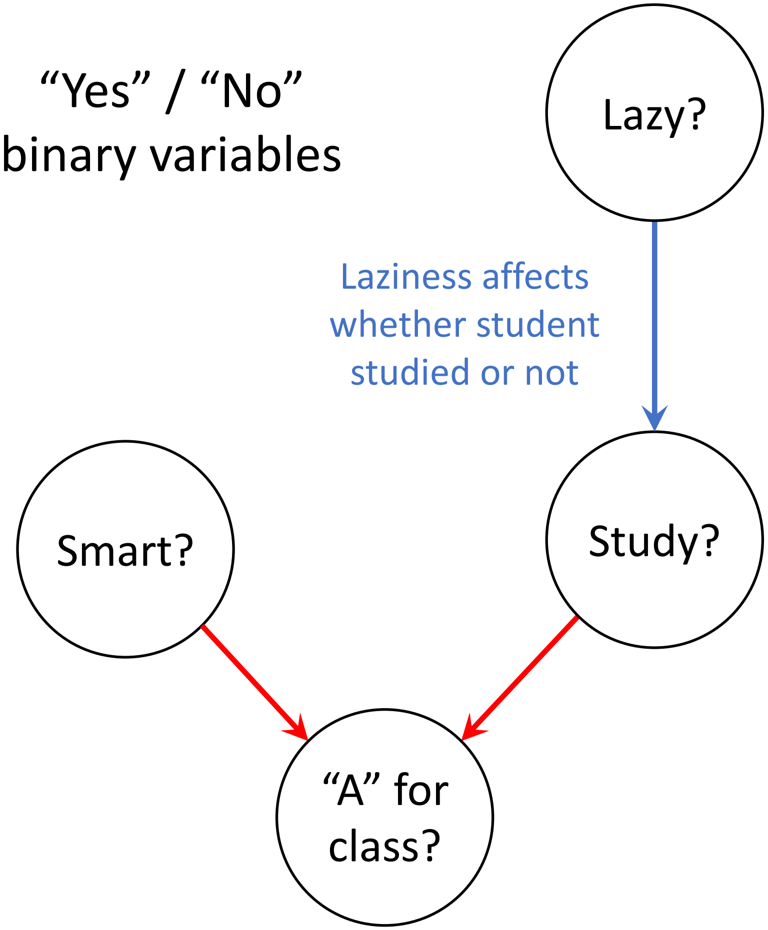
Chance of "A" depends on whether student studied and whether student is smart

Lazy $\not\perp\!\!\!\perp$ "A"

Lazy students tend to NOT get "A"
(because they usually don't study)

Toy example

"Yes" / "No"
binary variables



Chance of "A" depends on whether student studied and whether student is smart

Lazy $\not\perp\!\!\!\perp$ "A"

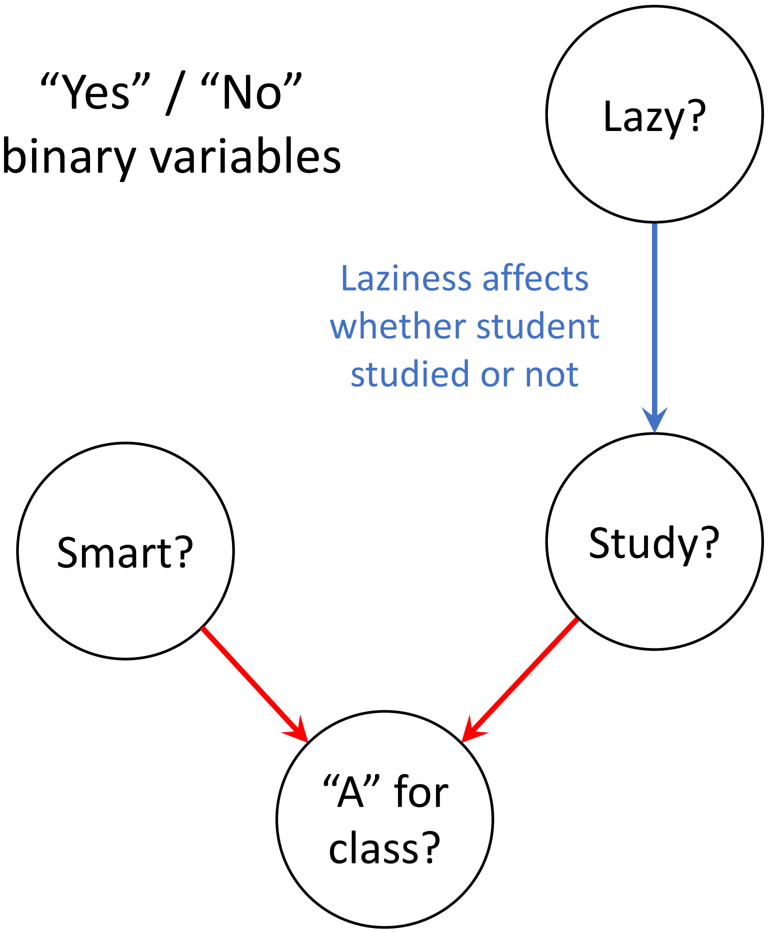
Lazy students tend to NOT get "A"
(because they usually don't study)

Lazy $\perp\!\!\!\perp$ "A" | Study

If we knew whether student studied, the laziness of the student is irrelevant to the grade

Toy example

"Yes" / "No"
binary variables



Chance of "A" depends on whether student studied and whether student is smart

Lazy $\not\!\perp\!\!\!\perp$ "A"

Lazy students tend to NOT get "A"
(because they usually don't study)

Lazy $\perp\!\!\!\perp$ "A" | Study

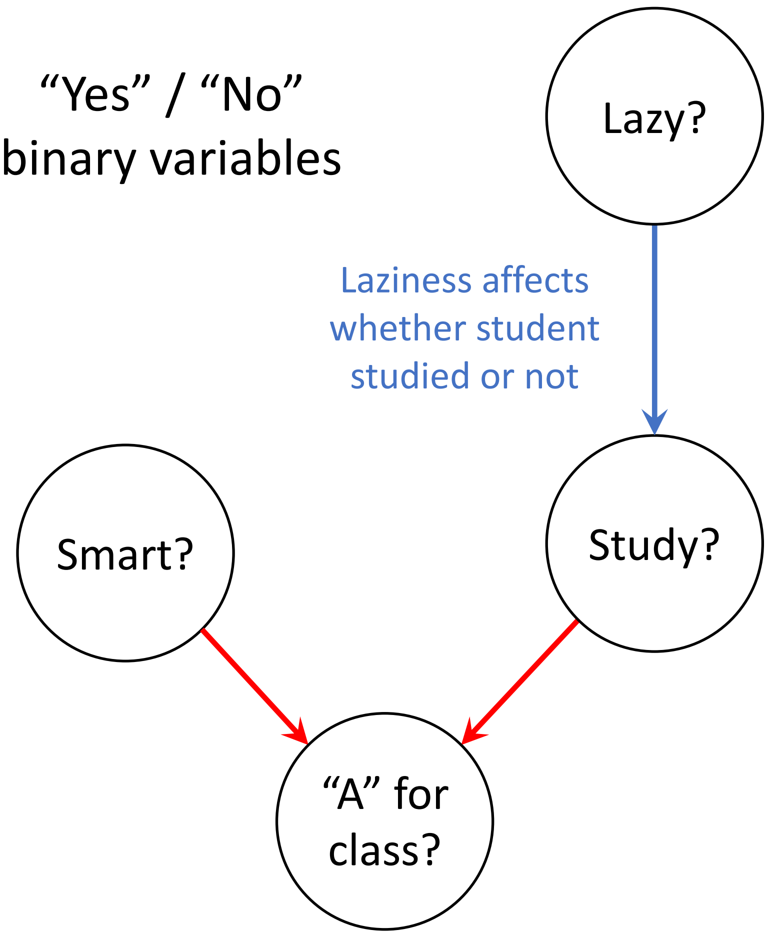
If we knew whether student studied, the laziness of the student is irrelevant to the grade

Lazy $\perp\!\!\!\perp$ Smart

Modelling assumption: Smart students are equally likely to be lazy or hard working

Toy example

“Yes” / “No”
binary variables



Chance of “A” depends on whether student studied and whether student is smart

$\text{Lazy} \not\perp\!\!\!\perp \text{“A”}$

Lazy students tend to NOT get “A”
(because they usually don’t study)

$\text{Lazy} \perp\!\!\!\perp \text{“A”} \mid \text{Study}$

If we knew whether student studied, the laziness of the student is irrelevant to the grade

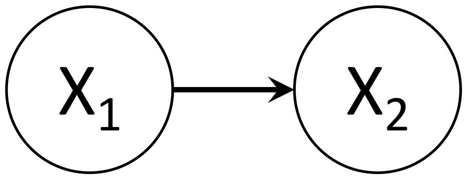
$\text{Lazy} \perp\!\!\!\perp \text{Smart}$

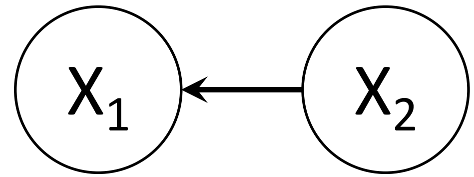
Modelling assumption: Smart students are equally likely to be lazy or hard working

$\text{Lazy} \not\perp\!\!\!\perp \text{Smart} \mid \text{“A”}$

Roughly speaking, “A” if student smart OR studied.
e.g. if NOT smart, then LIKELY to have studied, which implies student was UNLIKELY to be lazy

Two equivalent causal models





- $X_1 = \epsilon_1$
- $X_2 = a \cdot X_1 + \epsilon_2$
- $\epsilon_1 \sim N(0, 1)$
- $\epsilon_2 \sim N(0, 1)$

- $X_1 = \frac{a}{a^2+1} \cdot X_2 + \epsilon_1$
- $X_2 = \epsilon_2$
- $\epsilon_1 \sim N\left(0, \frac{1}{a^2+1}\right)$
- $\epsilon_2 \sim N(0, a^2 + 1)$

Data from both are fully characterized by covariance matrix $\begin{bmatrix} 1 & a \\ a & a^2 + 1 \end{bmatrix}$

Example from: https://youtu.be/rE6IMfSkOU0?t=849. See https://github.com/csquires/6.S091-causality for full course

Two equivalent causal models

- $X_1 =$
- $X_2 =$
- $\epsilon_1 \sim$
- $\epsilon_2 \sim$

Data fro

a
$a^2 + 1$

How to get around non-identifiability issues from observational data?

1. Make assumptions about functional form of SEM
   - e.g. Non-Gaussian noise
2. **Perform interventions (more on this later)**
   - e.g. randomized controlled trials

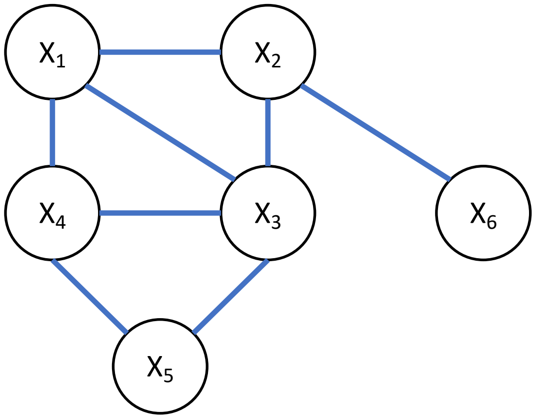
Markov Equivalence Class (MEC)

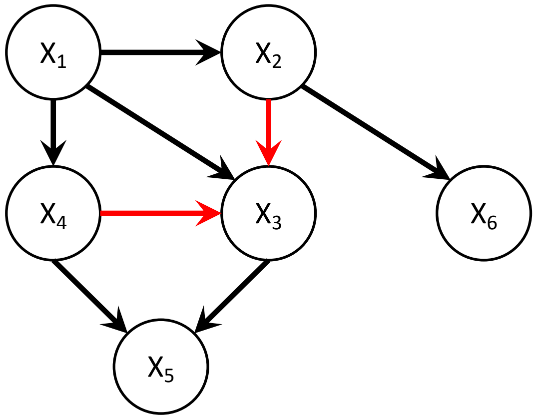
- Two DAGs are Markov equivalent if they encode the same CI relations
- Theorem [Verma, Pearl 1990; Andersson, Madigan, Perlman 1997]

  G and G’ are Markov equivalent **if and only if**

  1) G and G’ have the same skeleton
  2) G and G’ have the same v-structures

- skeleton and v-structures of DAG G^* earlier





- For any DAG G^*, we use $[G^*]$ to denote its MEC

Essential graphs $\mathcal{E}(G^*)$

- Used to graphically represent a MEC $[\mathrm{G}^*]$
- DAGs in same MEC have the same essential graph

Essential graphs $\mathcal{E}(G^*)$

- Used to graphically represent a MEC $[\mathrm{G}^*]$
- DAGs in same MEC have the same essential graph
- Partially oriented DAG
  - $X \sim Y$ is oriented as $X \rightarrow Y$ if **all** DAGs in the MEC agree
  - $X \sim Y$ is unoriented arc if there **exists** disagreement
    - $\exists G_1, G_2 \in [G^*]$ in MEC such that $X \rightarrow Y$ in G_1 and $X \leftarrow Y$ in G_2.

Essential graphs $\mathcal{E}(G^*)$

- Used to graphically represent a MEC $[\mathrm{G}^*]$
- DAGs in same MEC have the same essential graph
- Partially oriented DAG
  - $X \sim Y$ is oriented as $X \rightarrow Y$ if **all** DAGs in the MEC agree
  - $X \sim Y$ is unoriented arc if there **exists** disagreement
    - $\exists G_1, G_2 \in [G^*]$ in MEC such that $X \rightarrow Y$ in G_1 and $X \leftarrow Y$ in G_2.
- How to compute essential graph $\mathcal{E}(G^*)$ of G^*?
  1. Look at skeleton of G^*
  2. Orient v-structures in G^*
  3. Apply Meek rules [Meek 1995]

Meek rules [Meek 1995]

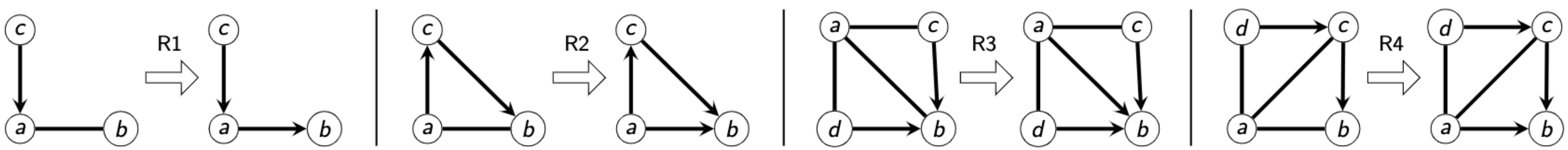
- **Sound** and **complete**
  (with respect to arc orientations with acyclic completion)

We won't miss out on any information

We won't wrongly orient arcs

Meek rules [Meek 1995]

- **Sound** and **complete**
  (with respect to arc orientations with acyclic completion)



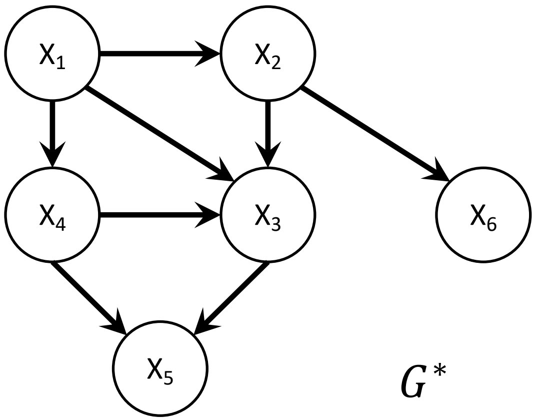
If $b \leftarrow a$,
then v-structure

If $b \leftarrow a$,
then cycle

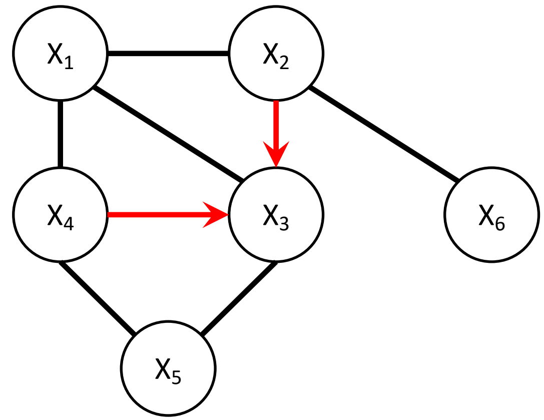
If $b \leftarrow a$, then unoriented arcs would have been oriented **in the same way** in all DAGs within the MEC (via R2)

- Converge in polynomial time [Wienöbst, Bannach, Liśkiewicz 2021]

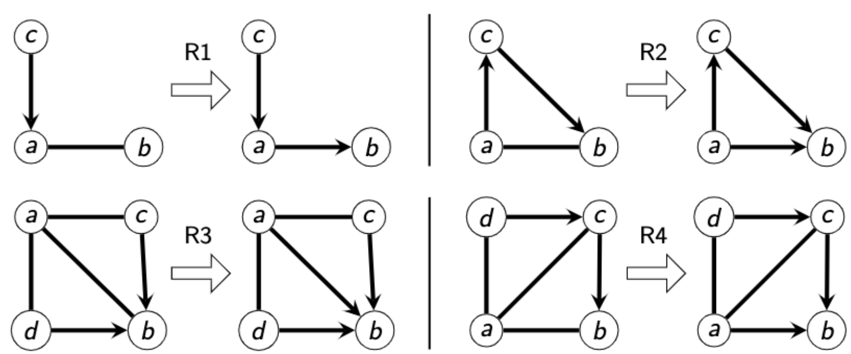
Essential graph example



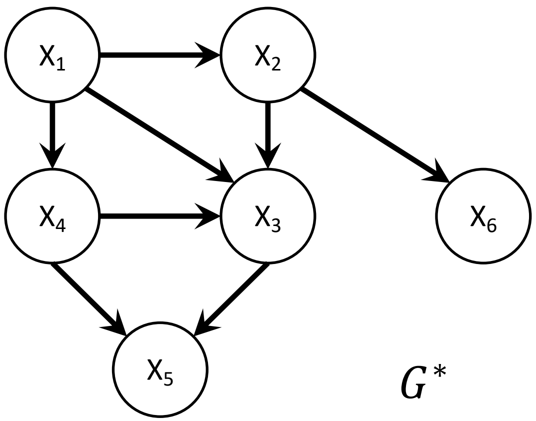
G^*

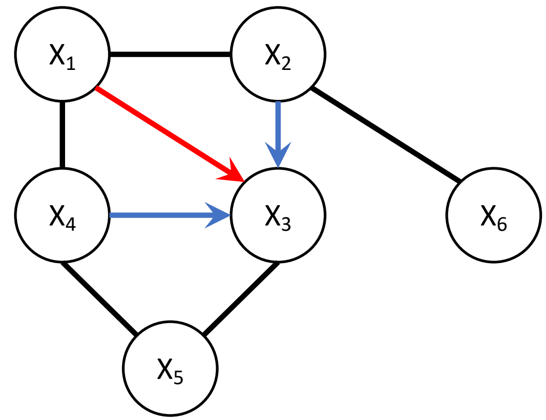


- Use CI tests: recover skeleton and v-structures

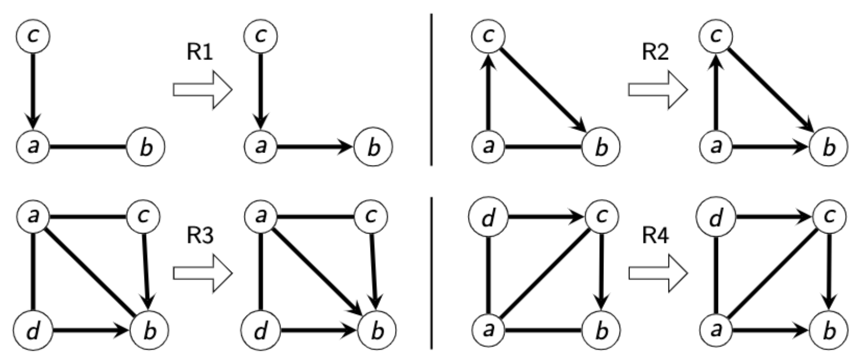


Essential graph example

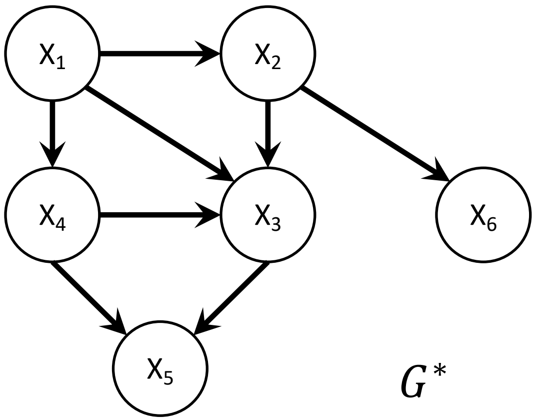


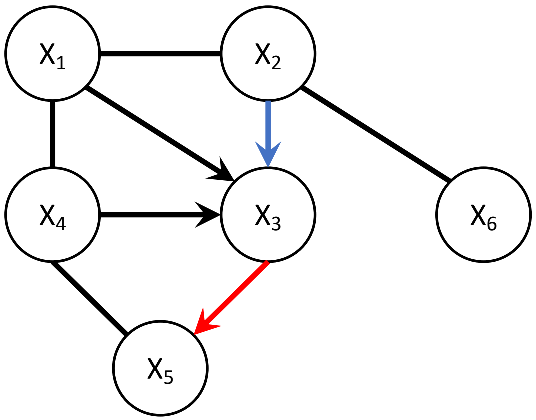


- Use CI tests: recover skeleton and v-structures
- Meek R3

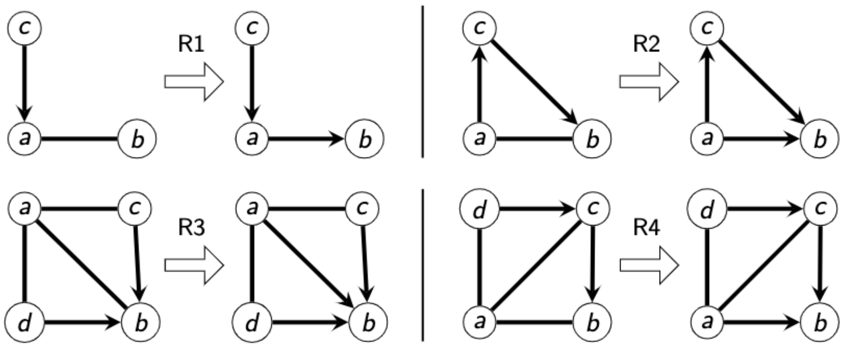


Essential graph example

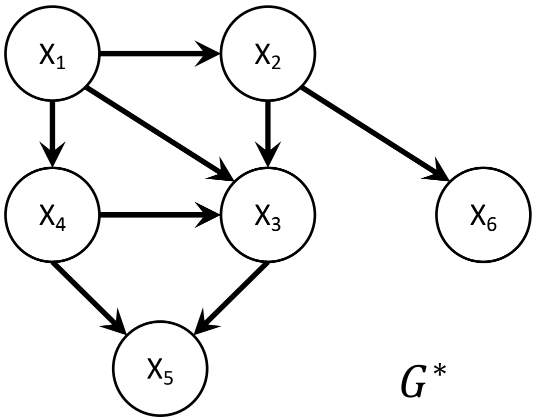


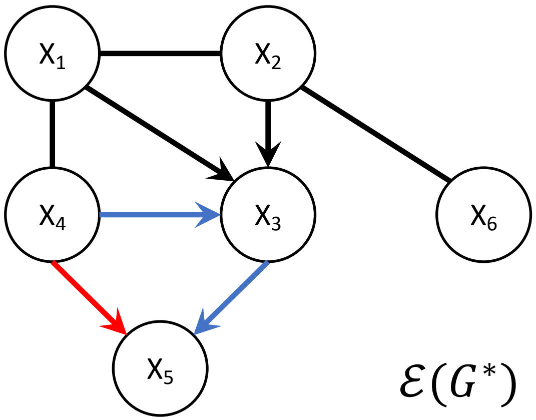


- Use CI tests: recover skeleton and v-structures
- Meek R3
- Meek R1

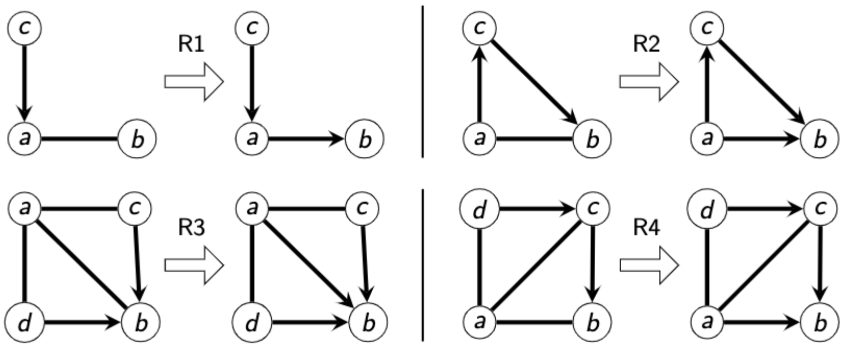


Essential graph example

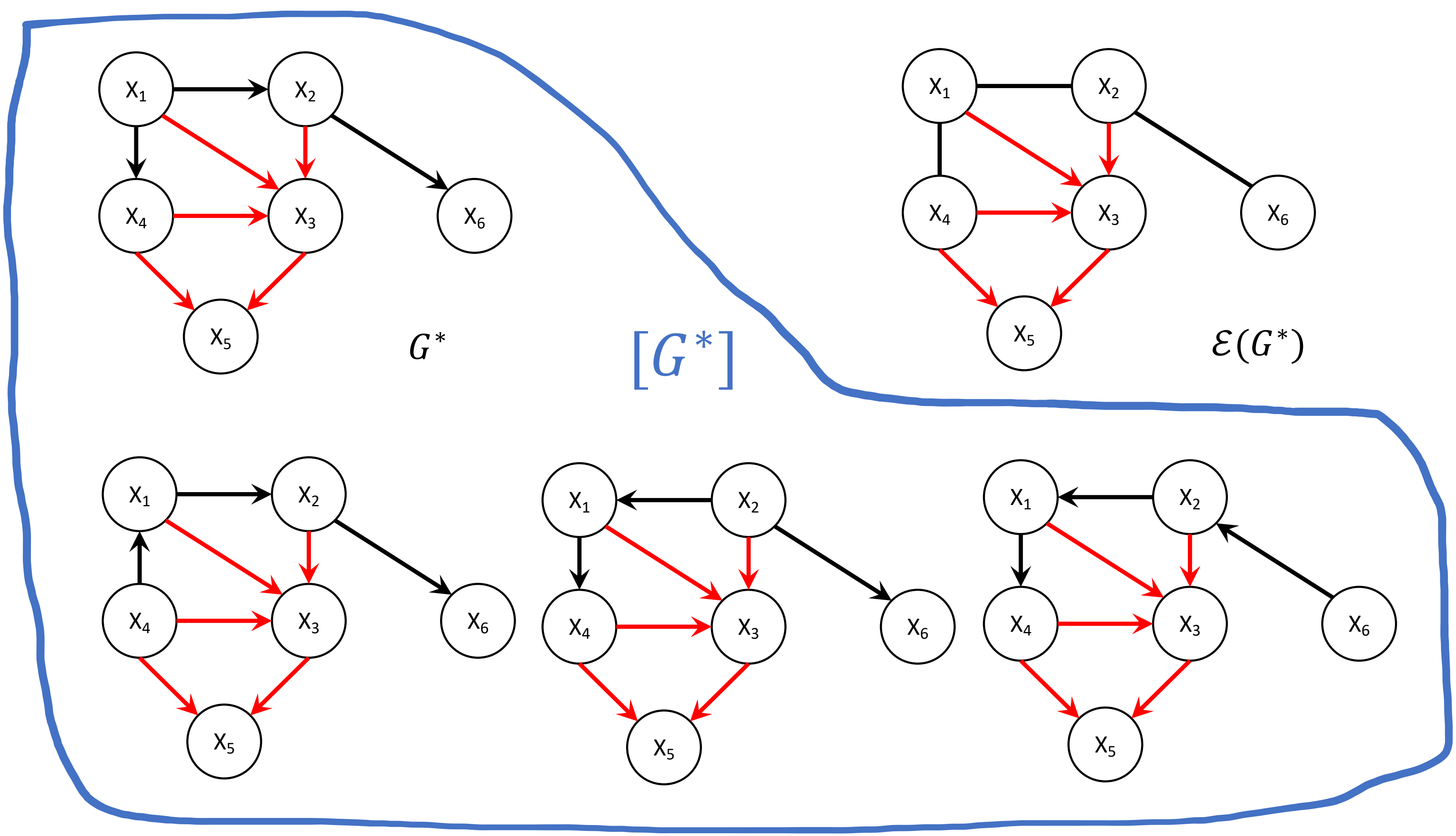




- Use CI tests: recover skeleton and v-structures
- Meek R3
- Meek R1
- Meek R2



Essential graph example



For this talk…

- Some standard causal assumptions
  - Causal sufficiency: no unobserved causal variables
  - Faithfulness: $\perp\!\!\!\perp$ in data $\Rightarrow$ $\perp\!\!\!\perp$ in graph
  - Oracle access to conditional independencies
- Simplifying assumptions for this talk
  - Hard interventions (see next slide)
  - Atomic intervention: One vertex per intervention
  - Each vertex has unit cost
- Objective
  - Minimize total number of vertices intervened

For this talk…

- Som

  -
  -
  -
- Sim

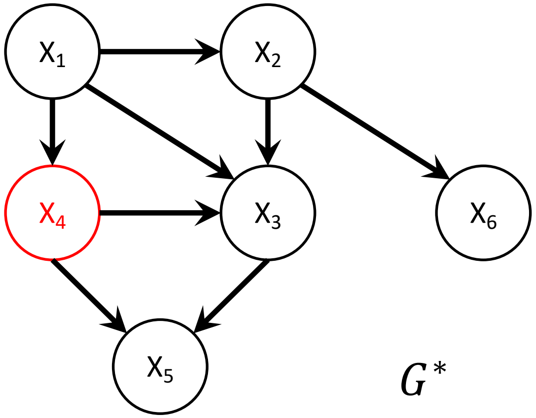
  -
  -
  - Each vertex has unit cost

We can abstract structure learning as a graph problem with specialized causal graph manipulation operations

Goal: Fully recover G^*

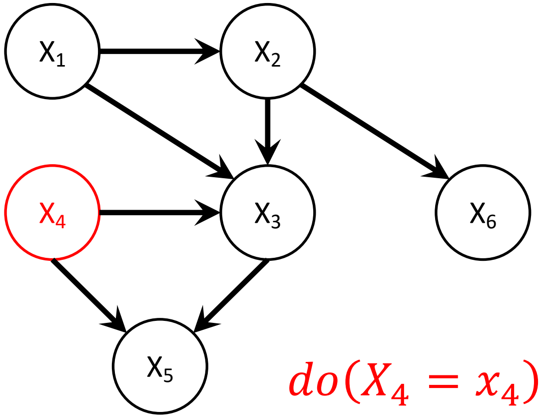
- Objective
  - Minimize total number of vertices intervened

Hard interventions



G^*

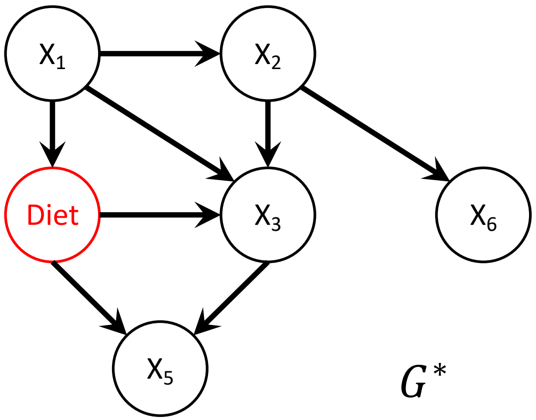
$X_1 = f_1(\epsilon_1)$
$X_2 = f_2(X_1, \epsilon_2)$
$X_3 = f_3(X_1, X_2, X_4, \epsilon_3)$
$X_4 = f_4(X_1, \epsilon_4)$
$X_5 = f_5(X_3, X_4, \epsilon_5)$
$X_6 = f_6(X_2, \epsilon_6)$
$\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5, \epsilon_6$ independent noise



$do(X_4 = x_4)$

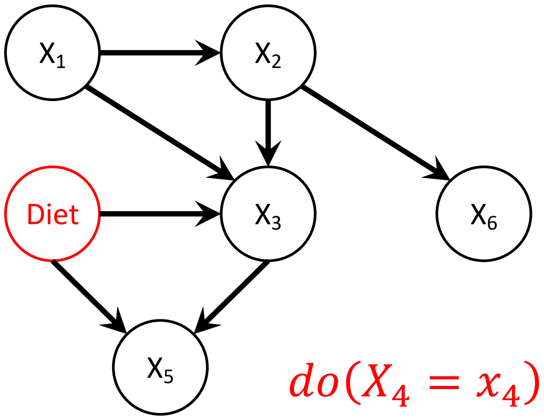
$X_1 = f_1(\epsilon_1)$
$X_2 = f_2(X_1, \epsilon_2)$
$X_3 = f_3(X_1, X_2, X_4, \epsilon_3)$
$X_4 =$ intervened value x_4
$X_5 = f_5(X_3, X_4, \epsilon_5)$
$X_6 = f_6(X_2, \epsilon_6)$
$\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5, \epsilon_6$ independent noise

Hard interventions



$$
\begin{aligned}
X_1 &= f_1(\epsilon_1)\\
X_2 &= f_2(X_1, \epsilon_2)\\
X_3 &= f_3(X_1, X_2, X_4, \epsilon_3)\\
X_4 &= f_4(X_1, \epsilon_4)\\
X_5 &= f_5(X_3, X_4, \epsilon_5)\\
X_6 &= f_6(X_2, \epsilon_6)
\end{aligned}
$$

$\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5, \epsilon_6$ independent noise

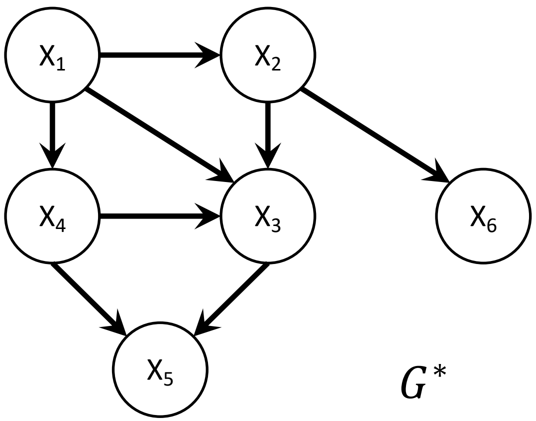


$$
\begin{aligned}
X_1 &= f_1(\epsilon_1)\\
X_2 &= f_2(X_1, \epsilon_2)\\
X_3 &= f_3(X_1, X_2, X_4, \epsilon_3)\\
X_4 &= \text{Eat Z apples a day}\\
X_5 &= f_5(X_3, X_4, \epsilon_5)\\
X_6 &= f_6(X_2, \epsilon_6)
\end{aligned}
$$

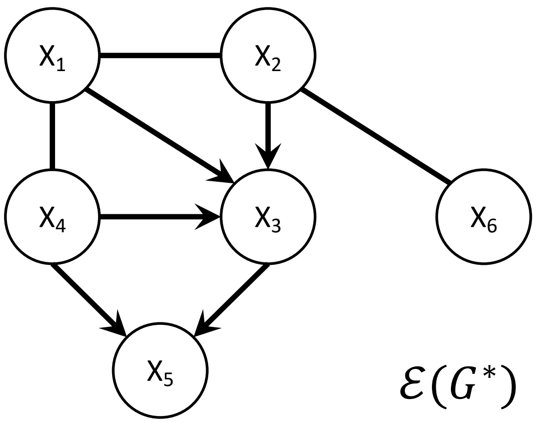
$\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5, \epsilon_6$ independent noise

What can we recover?

(Hidden)

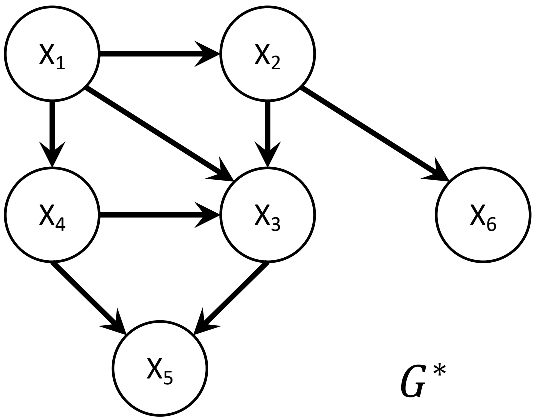


(What we can see)



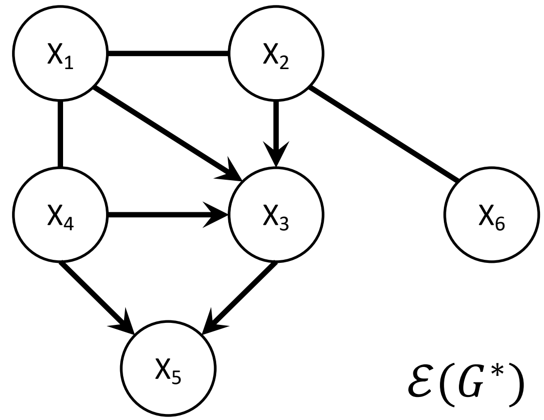
What can we recover?

(Hidden)

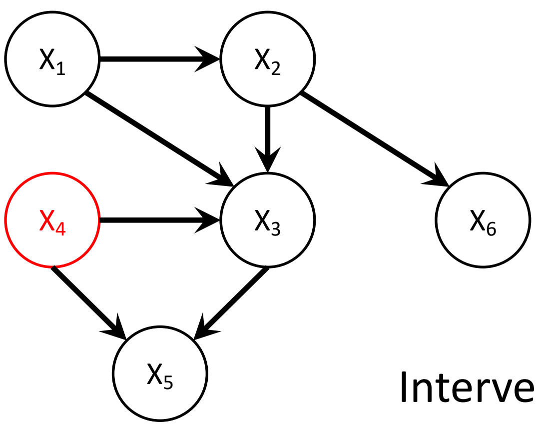


G^*

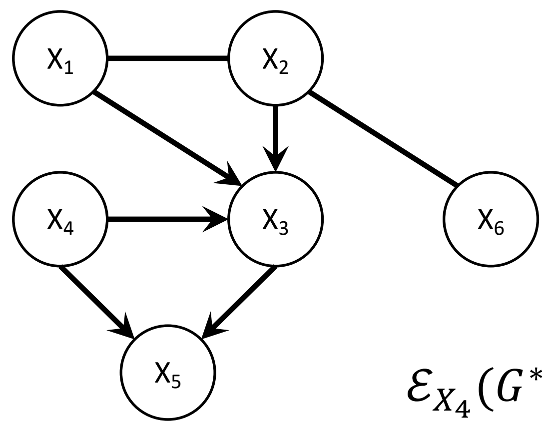
(What we can see)



$\mathcal{E}(G^*)$



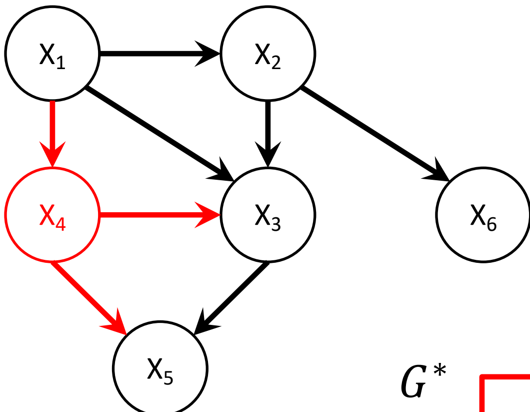
Intervene on X_4



$\mathcal{E}_{X_4}(G^*)$

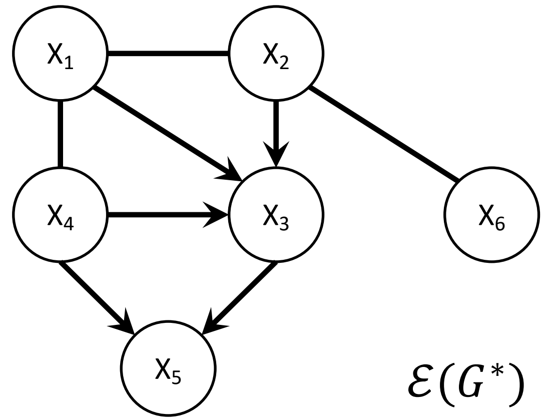
What can we recover?

(Hidden)



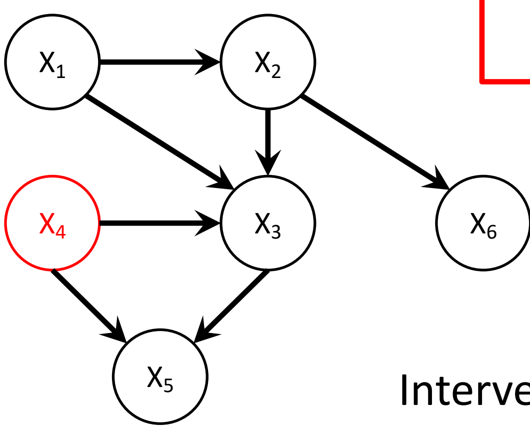
G^*

(What we can see)

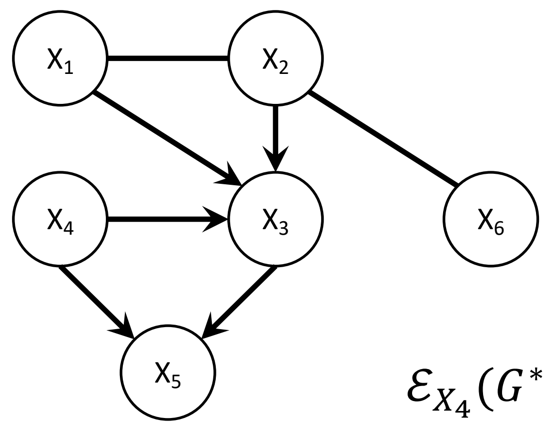


$\mathcal{E}(G^*)$

Intervening on X_4 lets us recover arc directions incident to X_4



Intervene on X_4



$\mathcal{E}_{X_4}(G^*)$

Two classes of interventions

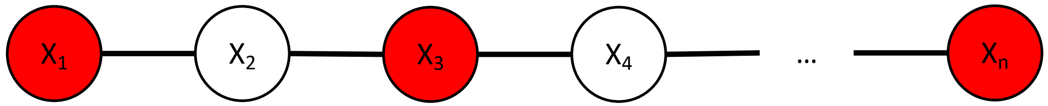
- Non-adaptive
  - Given MEC $[G^*]$, decide on a single fixed set of interventions that recovers *any possible* $G^* \in [G^*]$
  - Need to intervene on a $skel\big(\mathcal{E}(G^*)\big)$*-separating system*
    [Kocaoglu, Dimakis, Vishwanath 2017]
- Adaptive
  - Given MEC $[G^*]$,
    - Decide on first intervention
    - See outcome
    - Decide on second intervention
    - See outcome
    - …

G-separating system [Kocaoglu, Dimakis, Vishwanath 2017]

- Fix an undirected graph $G = (V, E)$
- A subset $\mathcal{J} \subseteq 2^V$ is a called a G-separating system if
  - For every edge $\{u, v\} \in E$, $\exists$ intervention $\mathrm{I} \in \mathcal{J}$ such that either $(u \in I \wedge v \notin I)$ or $(u \notin I \wedge v \in I)$
  - i.e. "every edge must be cut"
- Atomic interventions $\equiv$ vertex cover of G

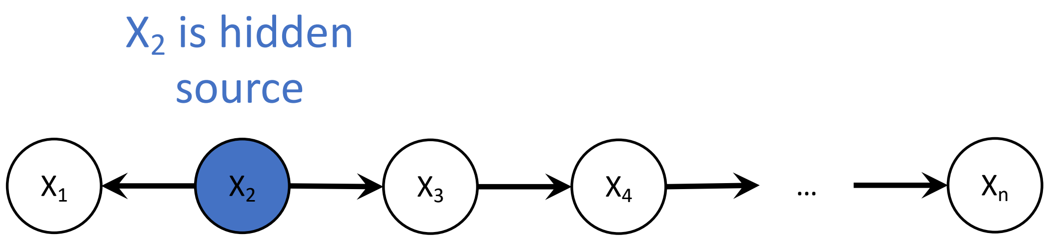
Power of adaptivity

- Path essential graph
  - n possible DAGs (pick a source node and orient away)
  - G-separating system needs $\geq \left\lfloor \frac{n}{2} \right\rfloor \in \Omega(n)$ vertices



Power of adaptivity

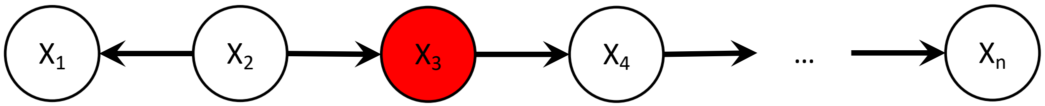
- Path essential graph
  - n possible DAGs (pick a source node and orient away)
  - G-separating system needs $\geq \left\lfloor \frac{n}{2} \right\rfloor \in \Omega(n)$ vertices



- Meanwhile, adaptive search can act like binary search!
  i.e. only $\mathcal{O}(\log n)$ interventions required

Power of adaptivity

- Path essential graph
  - n possible DAGs (pick a source node and orient away)
  - G-separating system needs $\geq \left\lfloor \frac{n}{2} \right\rfloor \in \Omega(n)$ vertices

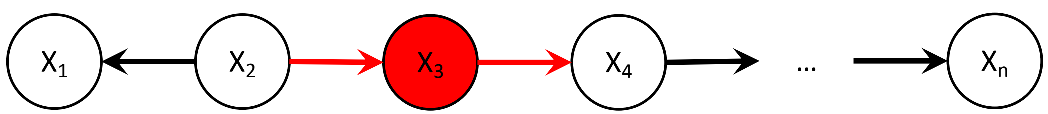


Suppose we intervene on X_3

- Meanwhile, adaptive search can act like binary search!
  i.e. only $\mathcal{O}(\log n)$ interventions required

Power of adaptivity

- Path essential graph
  - n possible DAGs (pick a source node and orient away)
  - G-separating system needs $\geq \left\lfloor \frac{n}{2} \right\rfloor \in \Omega(n)$ vertices

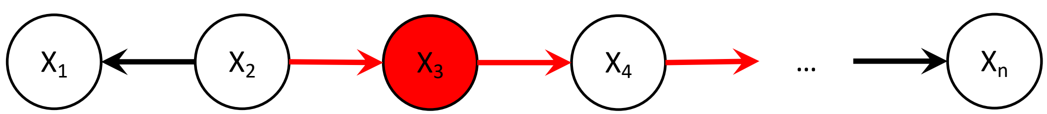


Recover incident edges

  - Meanwhile, adaptive search can act like binary search!
    i.e. only $\mathcal{O}(\log n)$ interventions required

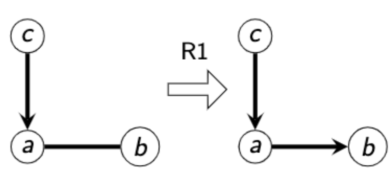
Power of adaptivity

- Path essential graph
  - n possible DAGs (pick a source node and orient away)
  - G-separating system needs $\geq \left\lfloor \frac{n}{2} \right\rfloor \in \Omega(n)$ vertices



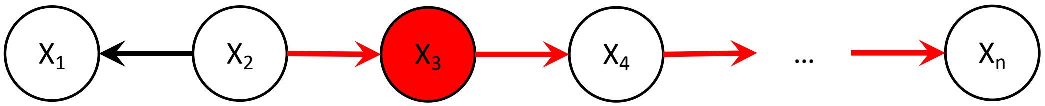
Meek R1

- Meanwhile, adaptive search can act like binary search!
  i.e. only $\mathcal{O}(\log n)$ interventions required



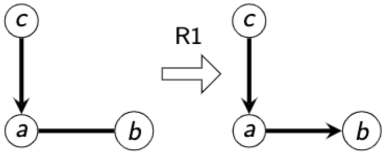
Power of adaptivity

- Path essential graph
  - n possible DAGs (pick a source node and orient away)
  - G-separating system needs $\geq \left\lfloor \frac{n}{2} \right\rfloor \in \Omega(n)$ vertices



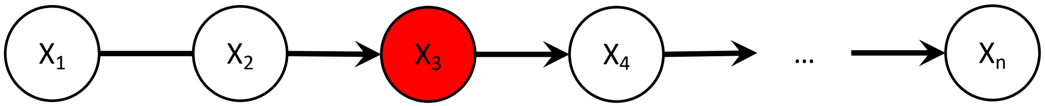
Meek R1

- Meanwhile, adaptive search can act like binary search!
  i.e. only $\mathcal{O}(\log n)$ interventions required



Power of adaptivity

- Path essential graph
  - n possible DAGs (pick a source node and orient away)
  - G-separating system needs $\geq \left\lfloor \frac{n}{2} \right\rfloor \in \Omega(n)$ vertices



Progress after intervening on X_3

Conclusion: The hidden source must be "on the left side" of X_3

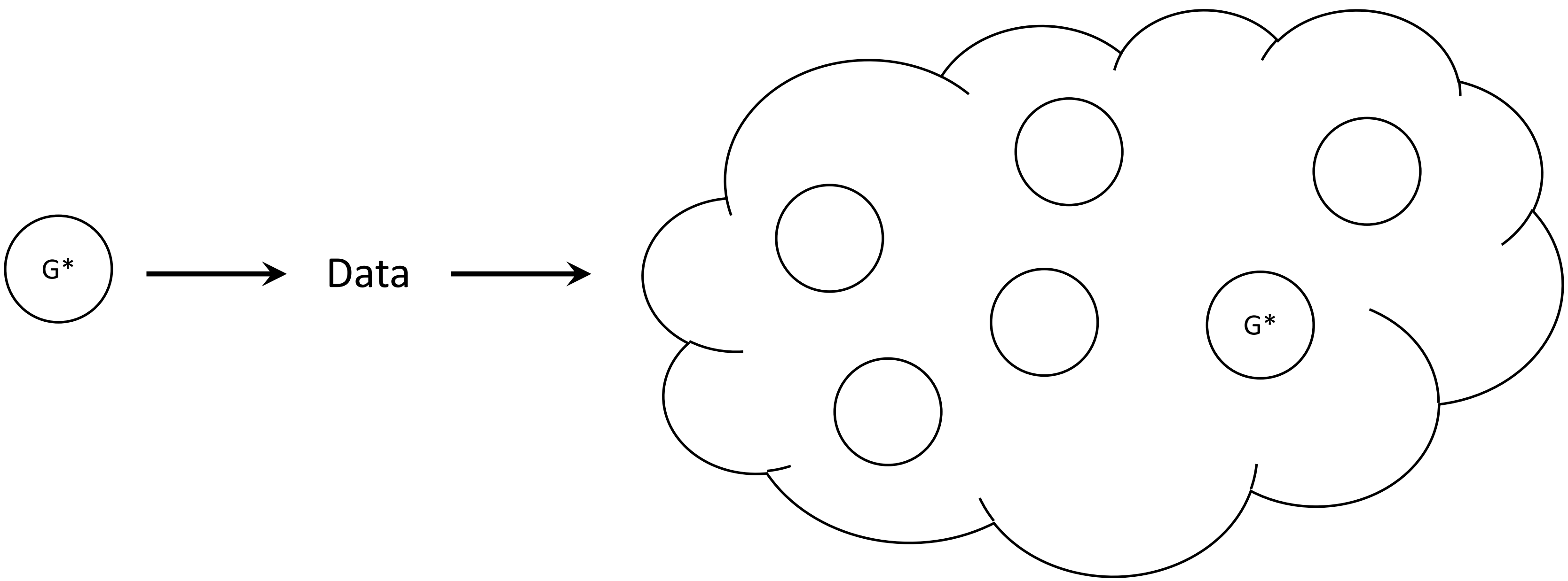
- Meanwhile, adaptive search can act like binary search!

  i.e. only $\mathcal{O}(\log n)$ interventions required

Problem setup

Identify G*

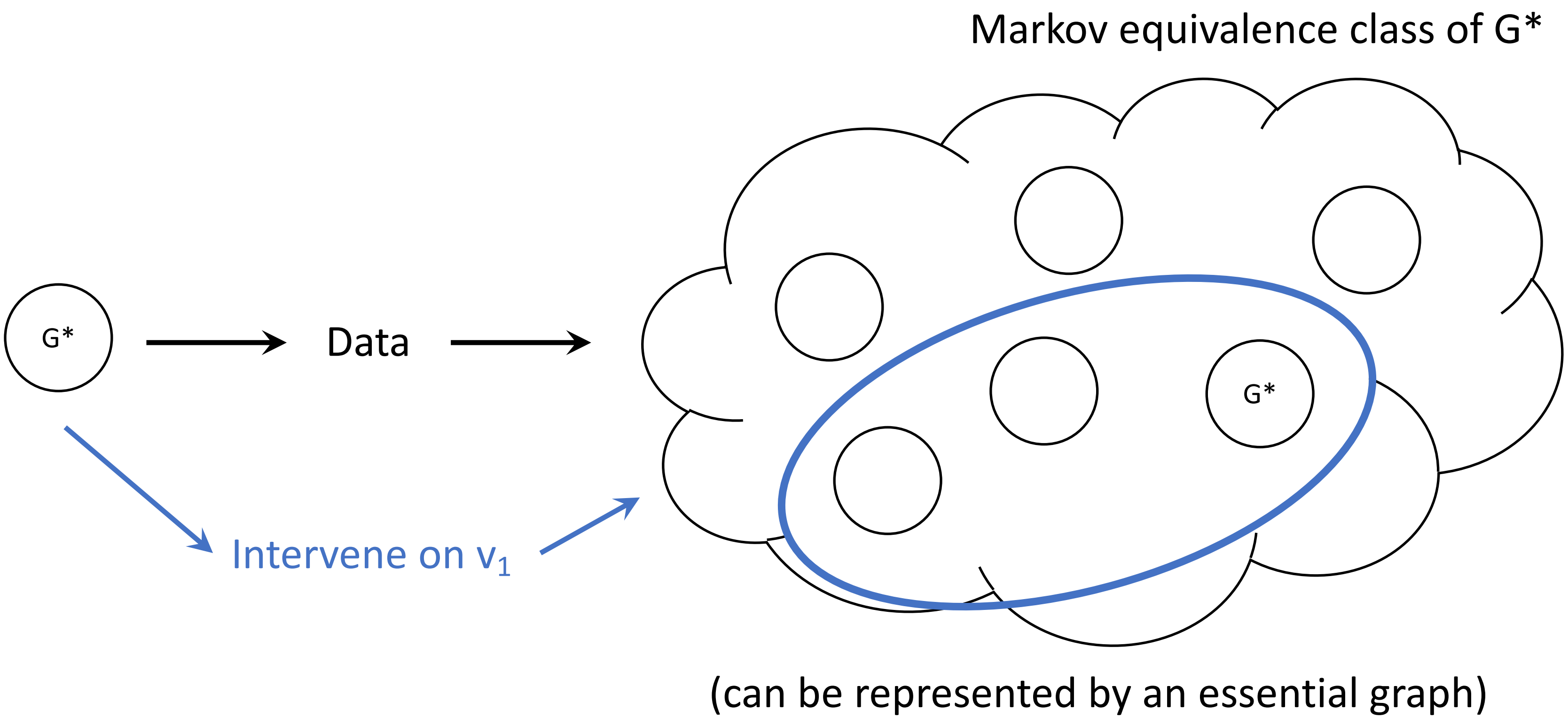
Markov equivalence class of G*



(can be represented by an essential graph)

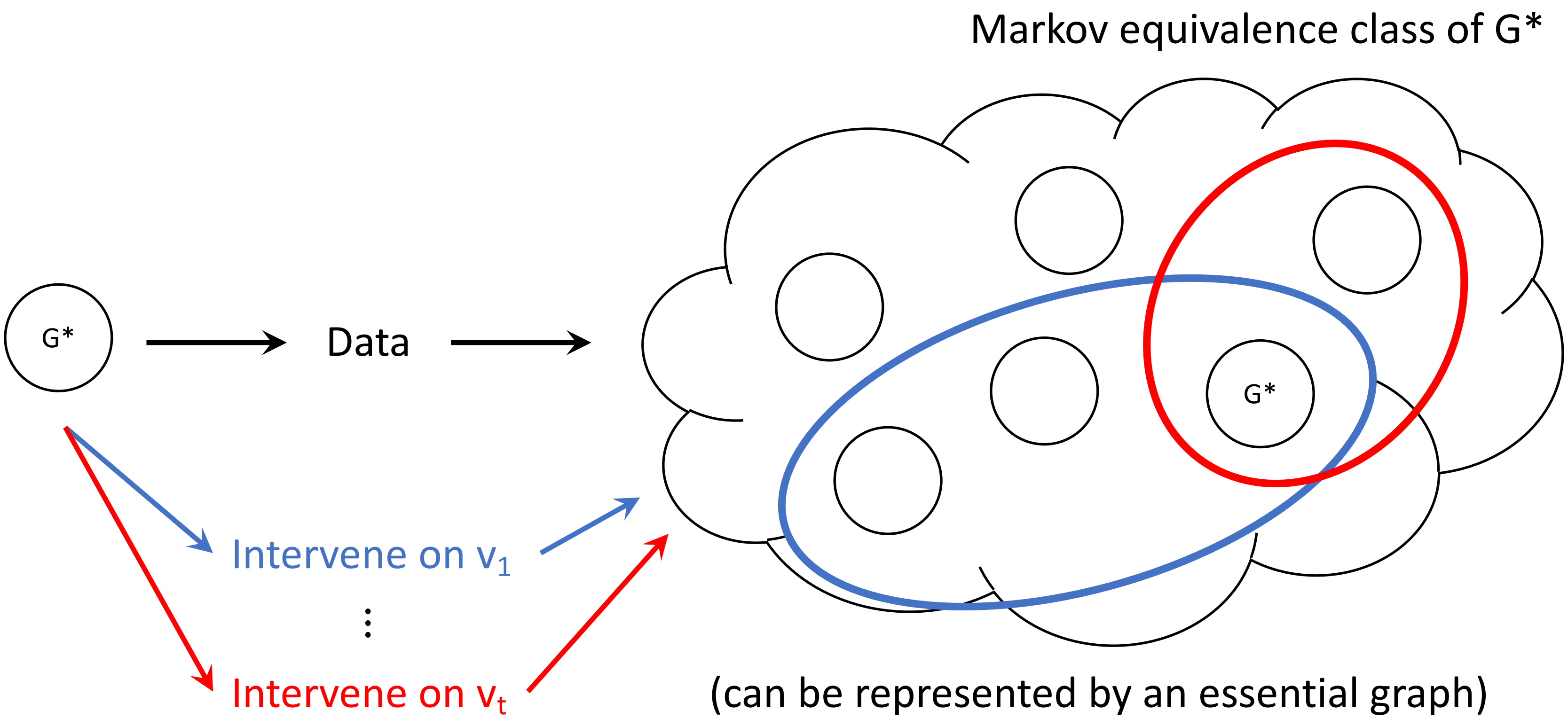
Problem setup

Identify G* using **interventions**



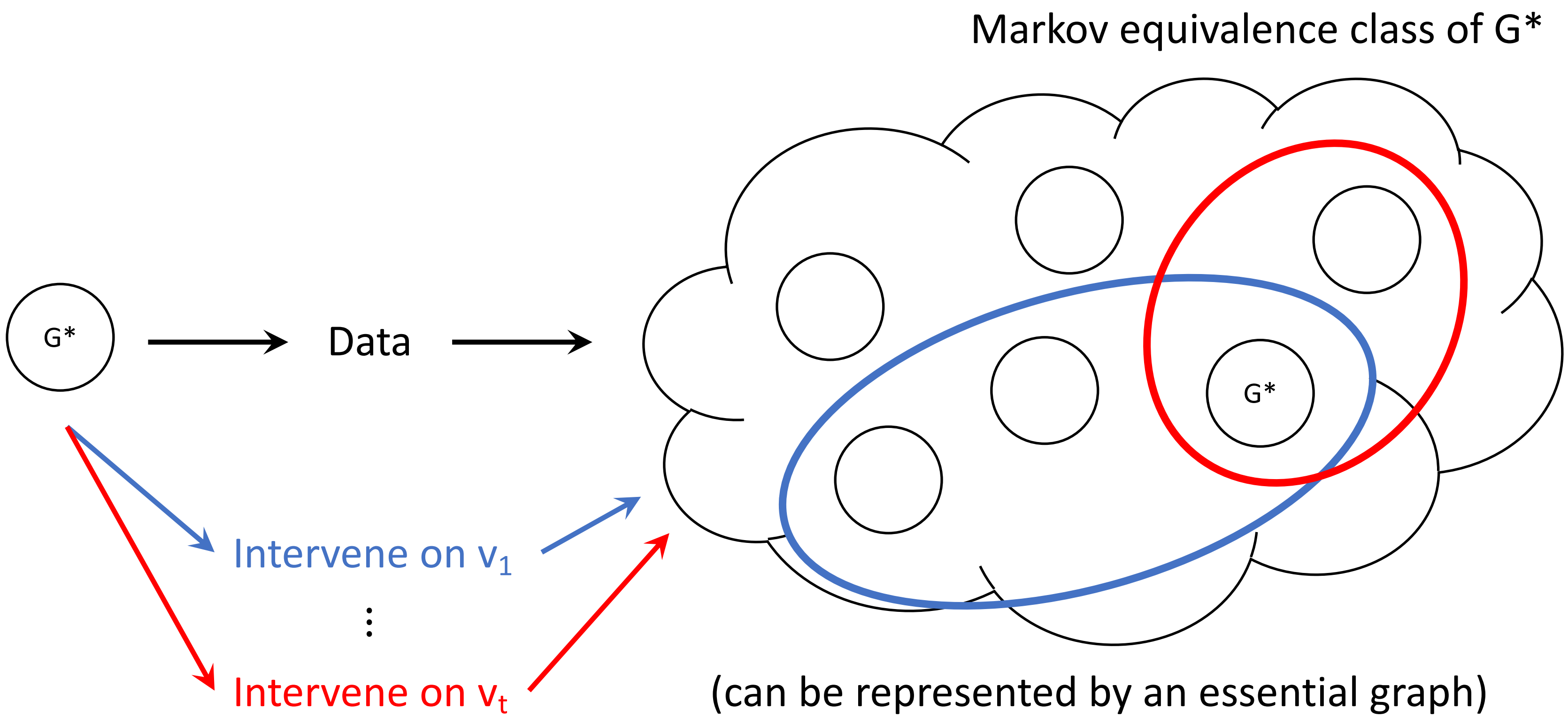
Problem setup

Identify G* using **interventions**



Problem setup

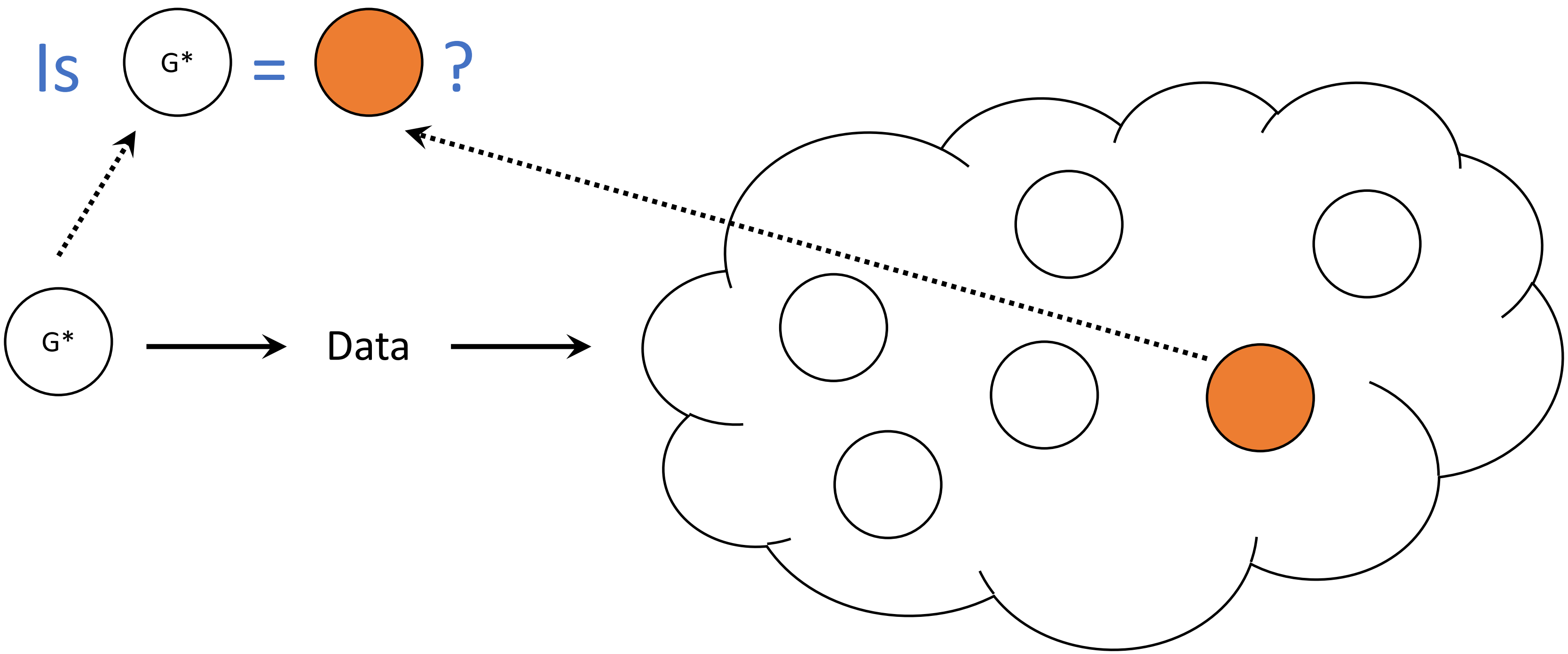
Identify G* using **as few interventions as possible** (minimize t)



(can be represented by an essential graph)

Verification: A simpler problem

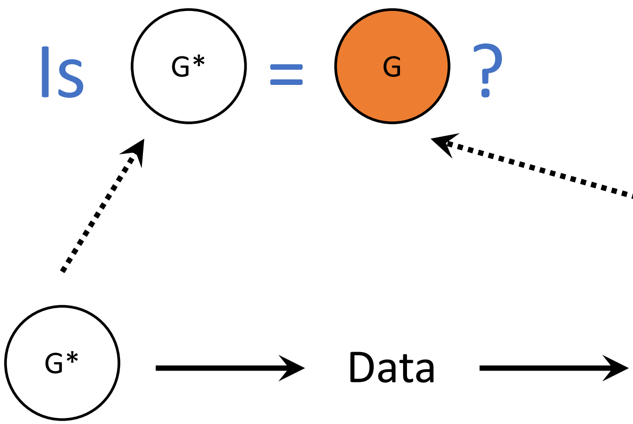
Question:

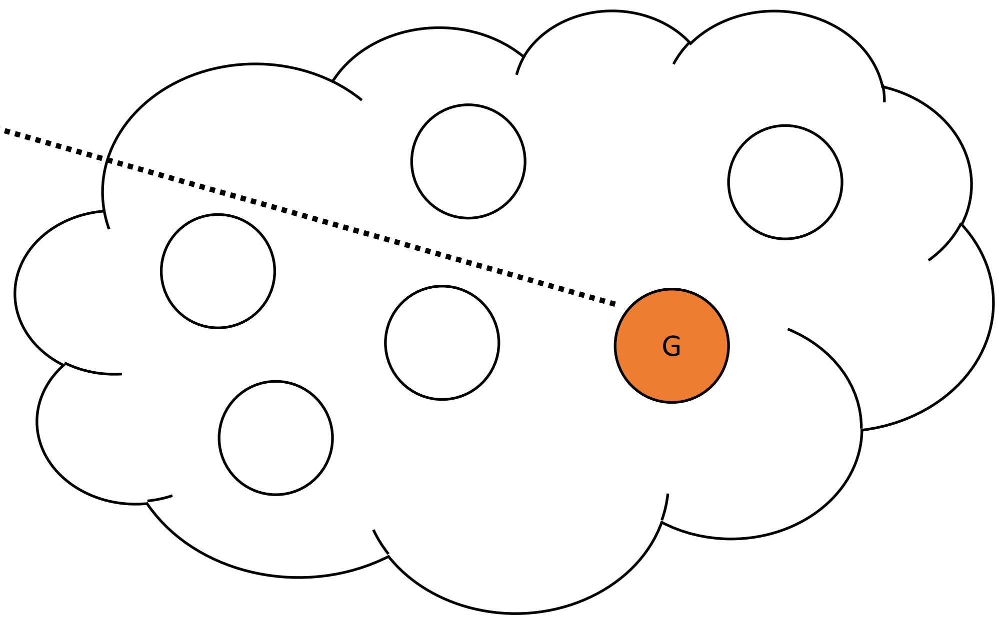


(can be represented by an essential graph)

Verification: A simpler problem

Question:





(can be represented by an essential graph)

Let $\boldsymbol{\nu(G)}$ be the minimum number of interventions needed to answer this question

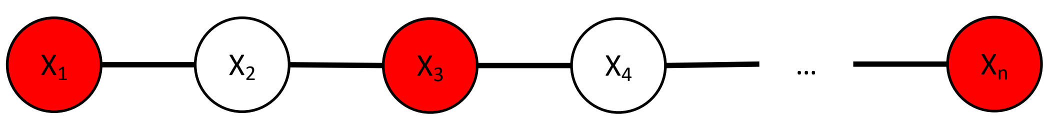
(Note: $\nu(\mathrm{G}^*)$ is a natural lower bound for adaptive search)

The verification problem

- Given MEC $[G^*]$ and some $\mathrm{G} \in [G^*]$,
  check whether $\mathrm{G} = G^*$ using interventions
  - Denote the minimum number required by $\nu(\mathrm{G})$
  - $\nu(G^*)$ is **lower bound** for **searching** for G^* within $[G^*]$

The verification problem

- Given MEC $[G^*]$ and some $\mathrm{G} \in [G^*]$, check whether $\mathrm{G} = G^*$ using interventions
  - Denote the minimum number required by $\nu(\mathrm{G})$
  - $\nu(G^*)$ is **lower bound** for **searching** for G^* within $[G^*]$
- Trivial solution
  - Compute minimum vertex cover on all unoriented arcs of the essential graph $\mathcal{E}(G) = \mathcal{E}(G^*)$
  - Check if revealed orientations agree with G
  - Worst case: $\Omega(n)$ interventions, e.g. on a line



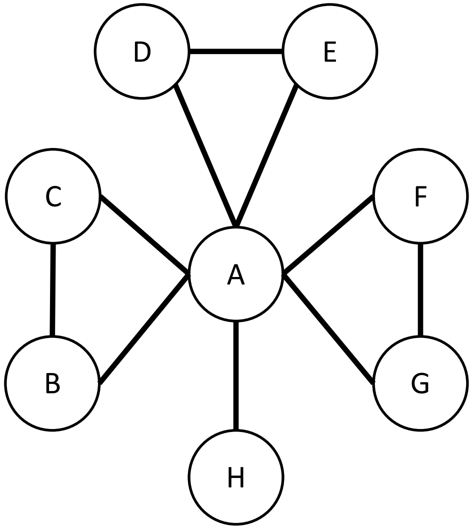
What was known

Maximal clique size

1. $\nu(\mathrm{G}) \geq \left\lfloor \frac{\omega(G)}{2} \right\rfloor$ [Squires, Magliacane, Greenewald, Katz, Kocaoglu, Shanmugam 2020]

Number of maximal cliques

2. $\left\lceil \frac{n-r}{2} \right\rceil \leq \nu(\mathrm{G}) \leq n - r$ [Porwal, Srivastava, Sinha 2022]



MEC $[G^*]$

n = 8, $\omega(G)$ = 3, r = 4

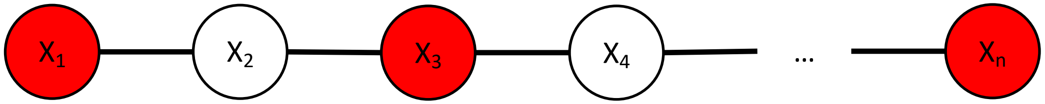
1. $1 \leq \nu(\mathrm{G})$
2. $2 \leq \nu(\mathrm{G}) \leq 4$

Characterization via covered edges

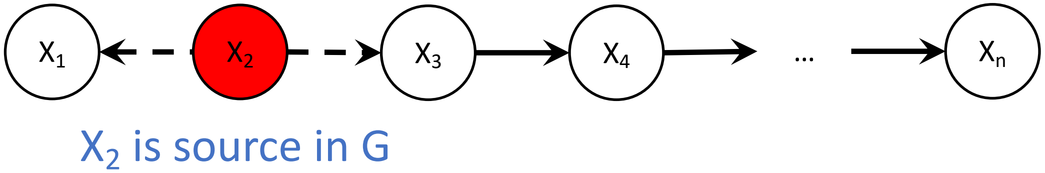
<u>Claim</u>: A set $\mathcal{I} \subseteq V$ is a verifying set for DAG $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ **if and only** if $\mathcal{I}$ is a minimum vertex cover of the *covered edges* [Chickering 1995] of G

- $u \sim v$ is covered edge if they have same parents

Naïve:



Our characterization:



Characterization via covered edges

Claim: A set $\mathcal{J} \subseteq V$ is a verifying set for DAG $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ **if and only** if $\mathcal{J}$ is a minimum vertex cover of the *covered edges* [Chickering 1995] of G

- $u \sim v$ is covered edge if they have same parents

Proof sketch:

- ($\Rightarrow$) Suppose we have a verifying set. Fix any covered edge $u \sim v$ where neither endpoint intervened. Case analysis that all 4 Meek rules will not orient $u \sim v$ will not be oriented.
- ($\Leftarrow$) Suppose we intervened on some minimum vertex cover of the covered edges. Fix a topological ordering π of vertices. Argue via induction that any edges belonging to the prefix of π is will be oriented.

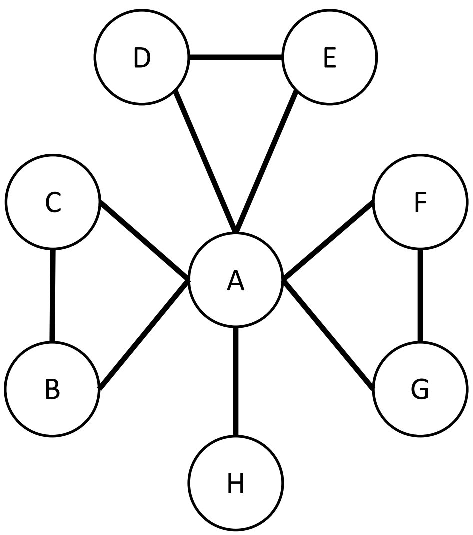
The overall proof is short (≤ 1 page in total) and quite subtle.

Comparison

Maximal clique size

Number of maximal cliques

1. $\nu(\mathrm{G}) \geq \left\lfloor \frac{\omega(G)}{2} \right\rfloor$ [SMG+20]
2. $\left\lceil \frac{n-r}{2} \right\rceil \leq \nu(\mathrm{G}) \leq n - r$ [PSS22]



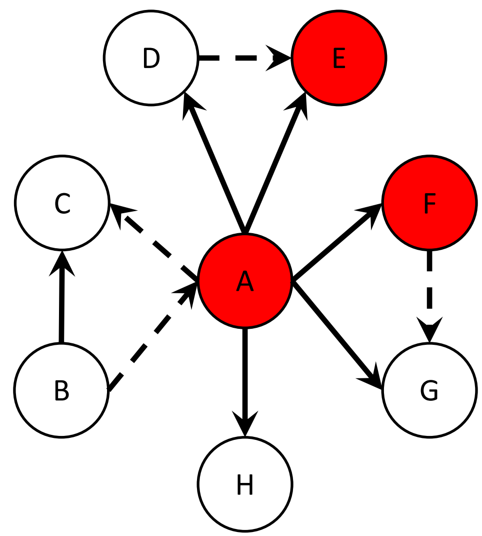
MEC $[G^*]$

n = 8, $\omega(G)$ = 3, r = 4

1. $1 \leq \nu(\mathrm{G})$
2. $2 \leq \nu(\mathrm{G}) \leq 4$

We can compute **exact** $\nu(\mathrm{G})$ for any given $\mathrm{G} \in [G^*]$

In fact, $\nu(\mathrm{G}) \in \{3,4\}$ for any $\mathrm{G} \in [G^*]$



One possible DAG from $[G^*]$

Efficient computation

- Wait… minimum vertex cover is NP-hard in general!

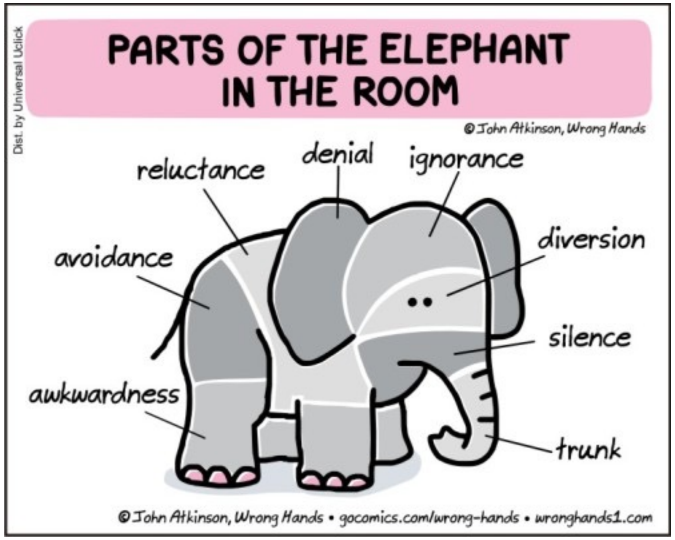
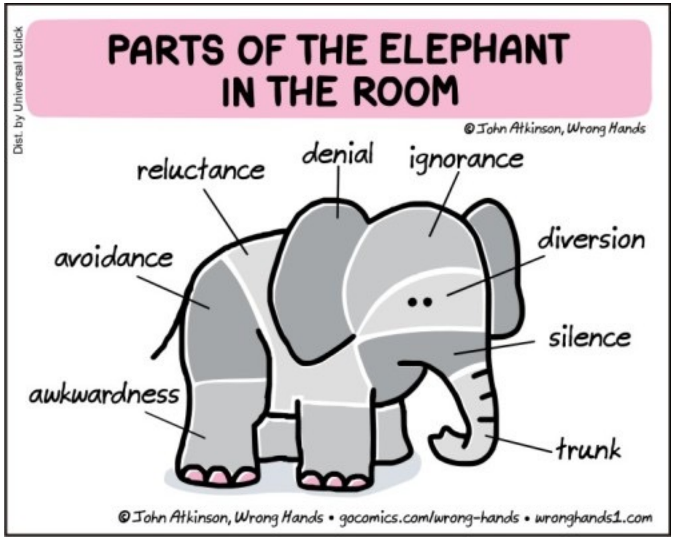


Image credit: https://wronghands1.com/2018/01/12/parts-of-the-elephant-in-the-room/

Efficient computation

- Wait… minimum vertex cover is NP-hard in general!



- Claim: Covered edges induce a forest
- Implication: $\nu(\mathrm{G})$ can be computed **exactly** via DP

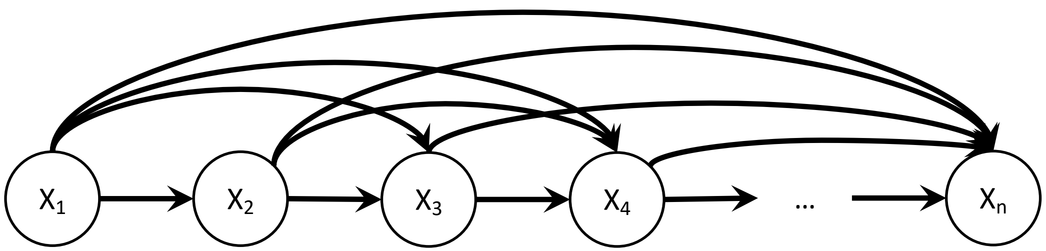
Image credit: https://wronghands1.com/2018/01/12/parts-of-the-elephant-in-the-room/

Through the lens of covered edges

- Covered edges cannot have both endpoints as sink of any maximal clique, so $\nu(\mathrm{G}) \leq n - r$

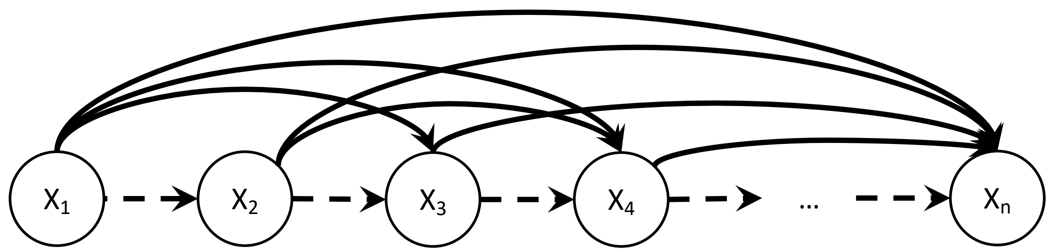
Through the lens of covered edges

- Covered edges cannot have both endpoints as sink of any maximal clique, so $\nu(\mathrm{G}) \leq n - r$
- G is a clique ⇒ Prior work: $\nu(\mathrm{G}) = \left\lfloor \frac{n}{2} \right\rfloor$



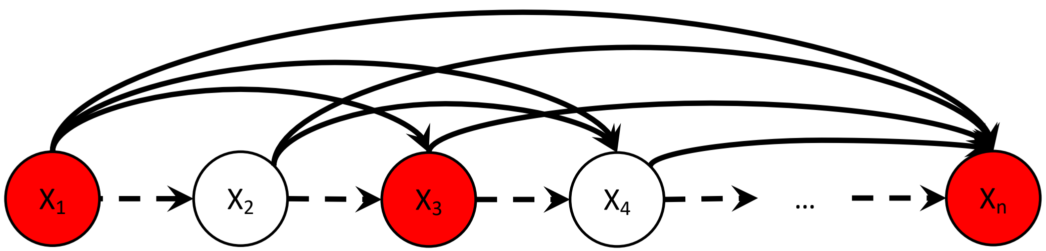
Through the lens of covered edges

- Covered edges cannot have both endpoints as sink of any maximal clique, so $\nu(\mathrm{G}) \leq n - r$
- G is a clique ⇒ Prior work: $\nu(\mathrm{G}) = \left\lfloor \frac{n}{2} \right\rfloor$



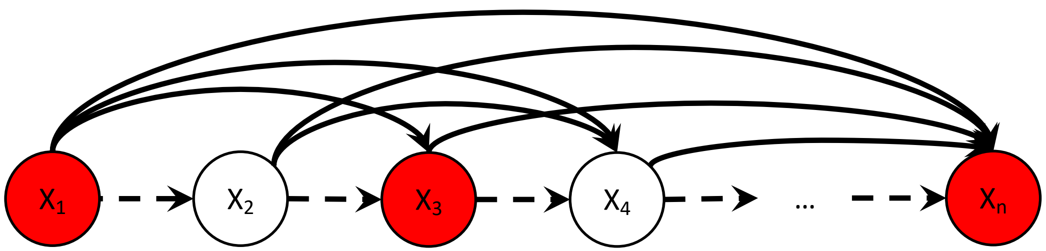
Through the lens of covered edges

- Covered edges cannot have both endpoints as sink of any maximal clique, so $\nu(\mathrm{G}) \leq n - r$
- G is a clique $\Rightarrow$ Prior work: $\nu(\mathrm{G}) = \left\lfloor \frac{n}{2} \right\rfloor$

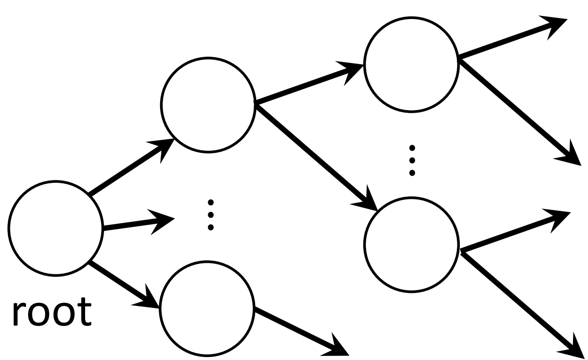


Through the lens of covered edges

- Covered edges cannot have both endpoints as sink of any maximal clique, so $\nu(\mathrm{G}) \leq n - r$
- G is a clique ⇒ Prior work: $\nu(\mathrm{G}) = \left\lfloor \frac{n}{2} \right\rfloor$

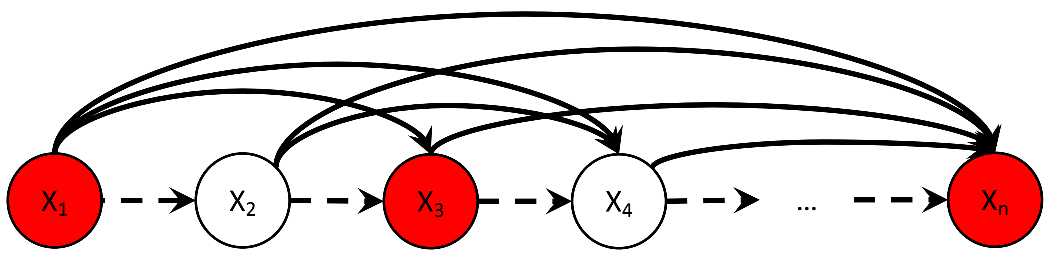


- G is a tree ⇒ Prior work: $\nu(\mathrm{G}) = 1$

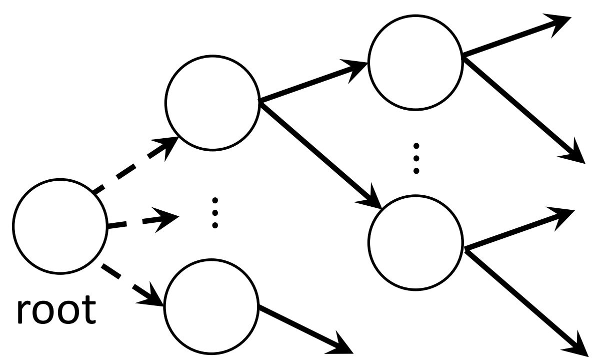


Through the lens of covered edges

- Covered edges cannot have both endpoints as sink of any maximal clique, so $\nu(\mathrm{G}) \leq n - r$
- G is a clique ⇒ Prior work: $\nu(\mathrm{G}) = \left\lfloor \frac{n}{2} \right\rfloor$

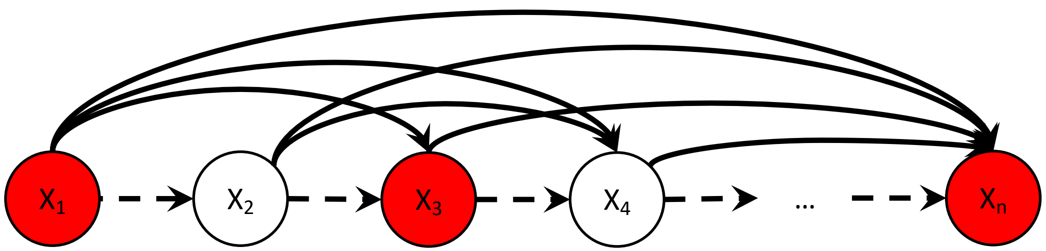


- G is a tree ⇒
  Prior work: $\nu(\mathrm{G}) = 1$

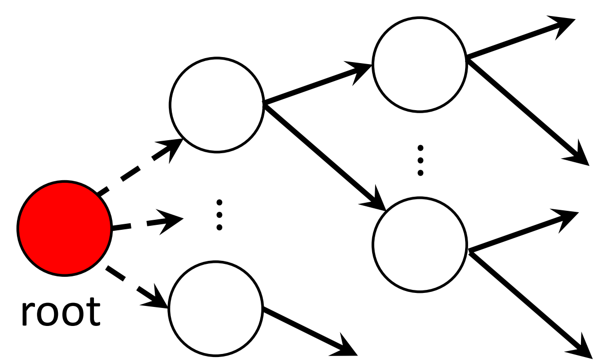


Through the lens of covered edges

- Covered edges cannot have both endpoints as sink of any maximal clique, so $\nu(\mathrm{G}) \leq n - r$
- G is a clique ⇒ Prior work: $\nu(\mathrm{G}) = \left\lfloor \frac{n}{2} \right\rfloor$



- G is a tree ⇒
  Prior work: $\nu(\mathrm{G}) = 1$

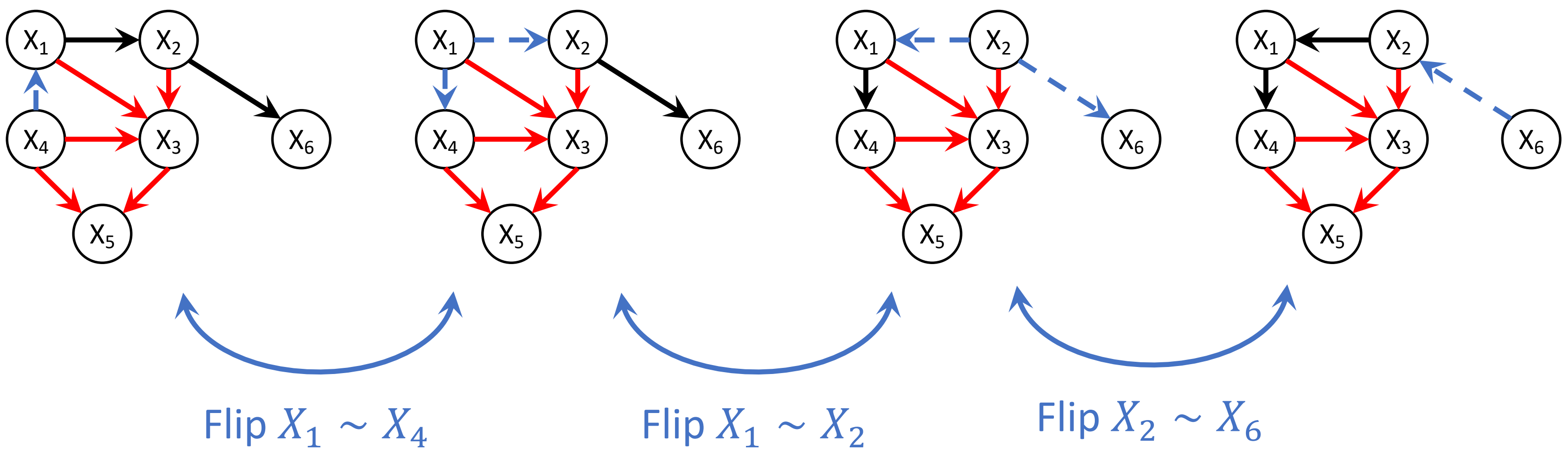


Through the lens of covered edges

- For non-adaptive interventions, we must intervene on a G-separating system

Through the lens of covered edges

- For non-adaptive interventions, we must intervene on a G-separating system
  - Two graphs have the same MEC $[G^*]$ **if and only if** there is a sequence of covered edge reversals that transform between them [Chickering 1995]



Through the lens of covered edges

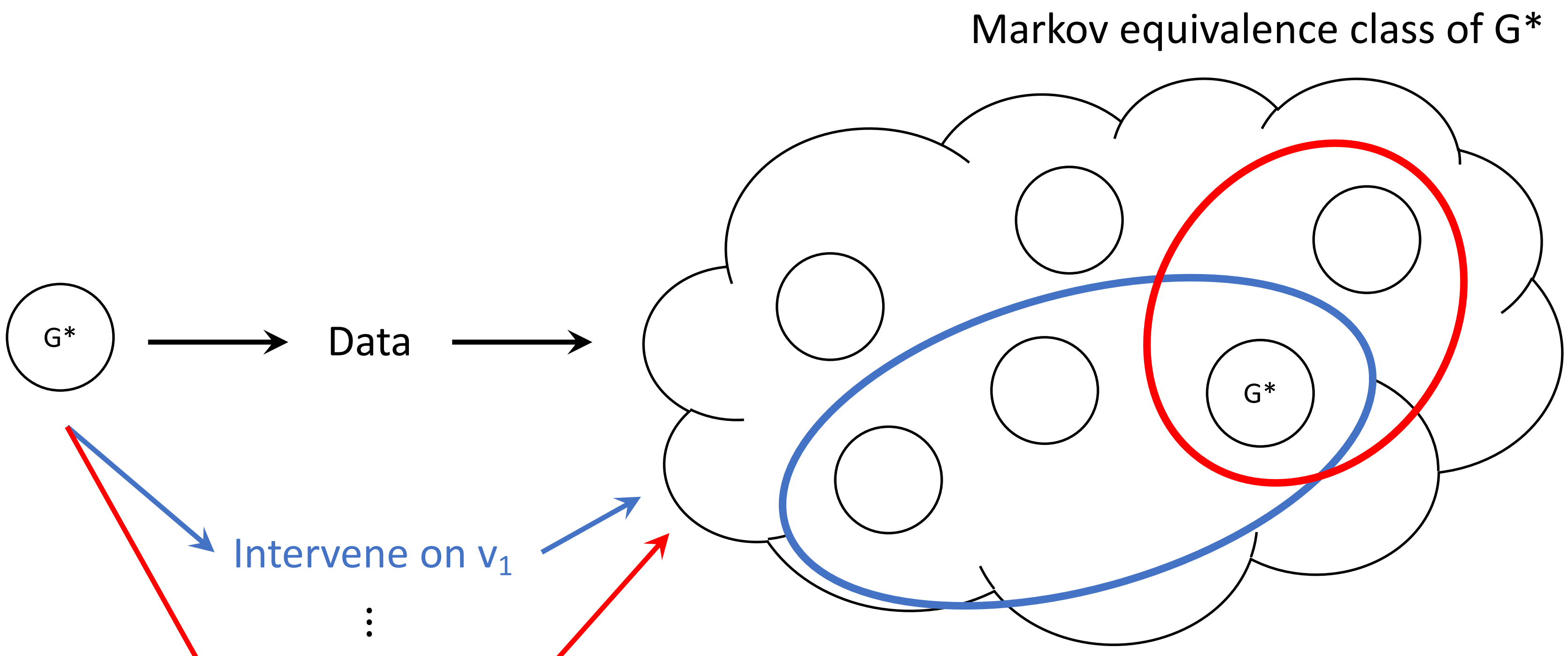
- For non-adaptive interventions, we must intervene on a G-separating system
  - Two graphs have the same MEC $[G^*]$ **if and only if** there is a sequence of covered edge reversals that transform between them [Chickering 1995]
  - Unoriented in $\mathcal{E}(G^*) \Rightarrow$ Covered edge in *some* $\mathrm{G} \in [G^*]$

Through the lens of covered edges

- For non-adaptive interventions, we must intervene on a G-separating system
  - Two graphs have the same MEC $[G^*]$ **if and only if** there is a sequence of covered edge reversals that transform between them [Chickering 1995]
  - Unoriented in $\mathcal{E}(G^*) \Rightarrow$ Covered edge in *some* $\mathrm{G} \in [G^*]$
  - So, "non-adaptive must cut all unoriented in $\mathcal{E}(G^*)$", i.e. a G-separating system

The search problem

Identify G* using **as few interventions as possible** (minimize t)



(can be represented by an essential graph)

The search problem

- Given MEC $[G^*]$ and recover G^* using interventions
  - We know at least $\nu(G^*)$ is necessary
  - Prior works only have guarantees for special classes of graphs: cliques, trees, intersection incomparable, etc.

The search problem

- Given MEC $[G^*]$ and recover G^* using interventions
  - We know at least $\nu(G^*)$ is necessary
  - Prior works only have guarantees for special classes of graphs: cliques, trees, intersection incomparable, etc.
- Punchline: $\mathcal{O}\big(\log n \cdot \nu(G^*)\big)$ interventions suffice
  - "Search is almost as easy as verification"

The search problem

- Given MEC $[G^*]$ and recover G^* using interventions
  - We know at least $\nu(G^*)$ is necessary
  - Prior works only have guarantees for special classes of graphs: cliques, trees, intersection incomparable, etc.
- Punchline: $\mathcal{O}\left(\log n \cdot \nu(G^*)\right)$ interventions suffice
  - "Search is almost as easy as verification"
  - Algorithm does not even know what $\nu(G^*)$ is!

The search problem

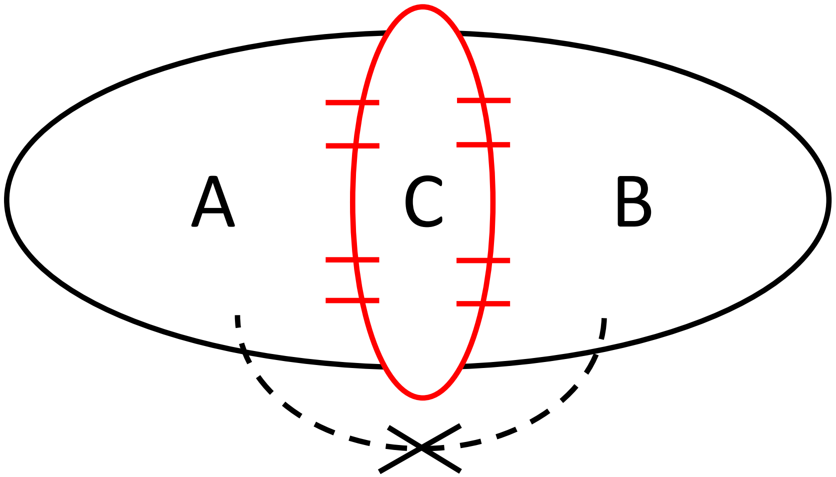
- Given MEC $[G^*]$ and recover G^* using interventions
  - We know at least $\nu(G^*)$ is necessary
  - Prior works only have guarantees for special classes of graphs: cliques, trees, intersection incomparable, etc.
- Punchline: $\mathcal{O}\big(\log n \cdot \nu(G^*)\big)$ interventions suffice
  - "Search is almost as easy as verification"
  - Algorithm does not even know what $\nu(G^*)$ is!
  - $\Omega(\log n)$ is unavoidable when $[G^*]$ is a path on n nodes
    - $\nu(G^*) = 1$
    - "Cannot do better than binary search"

The adaptive search algorithm

- Intervene and **ignore oriented arcs** ⇒ Chordal graph. Handle each connected component [Hauser, Bühlmann 2012, 2014]

The adaptive search algorithm

- Intervene and **ignore oriented arcs** $\Rightarrow$ Chordal graph. Handle each connected component [Hauser, Bühlmann 2012, 2014]
- For any chordal graph G, one can compute in polynomial time a clique separator C [Gilbert, Rose, Edenbrandt 1984]
  - $|A|, |B| \leq \frac{|V(G)|}{2}$; C is a clique, i.e. $|C| \leq \omega(G)$



Graph separator theorem for chordal graph

The adaptive search algorithm

- Intervene and **ignore oriented arcs** $\Rightarrow$ Chordal graph. Handle each connected component [Hauser, Bühlmann 2012, 2014]
- For any chordal graph G, one can compute in polynomial time a clique separator C [Gilbert, Rose, Edenbrandt 1984]
  - $|A|, |B| \leq \frac{|V(G)|}{2}$; C is a clique, i.e. $|C| \leq \omega(G)$
- Algorithm: Find clique separator C_H in each component H; Intervene on all nodes in C_H's; Recurse
- Analysis:
  - $\mathcal{O}(\log n)$ rounds suffices $\leftarrow$ [Gilbert, Rose, Edenbrandt 1984]
  - $\mathcal{O}\big(\nu(G^*)\big)$ per round $\leftarrow$ We prove new lower bound on $\nu(G^*)$

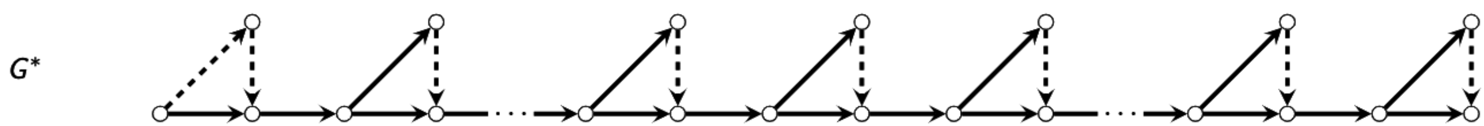
A lower bound

Intuition [HB12,14]: In any interventional essential graph, interventions across different "connected components" *do not* help.

Claim: Fix an essential graph and some DAG G in it. Then,

[SMG+20]

$$\nu(G) \geq \sum_{\substack{\text{connected components} \\ H \in \text{ after removing oriented arcs}}} \left\lfloor \frac{\omega(H)}{2} \right\rfloor$$



Lower bound from claim: $\nu(G^*) \geq \left\lfloor \frac{3}{2} \right\rfloor = 1$

But, from our covered edge characterization, we know that $\nu(G^*) \approx \frac{n}{2}$

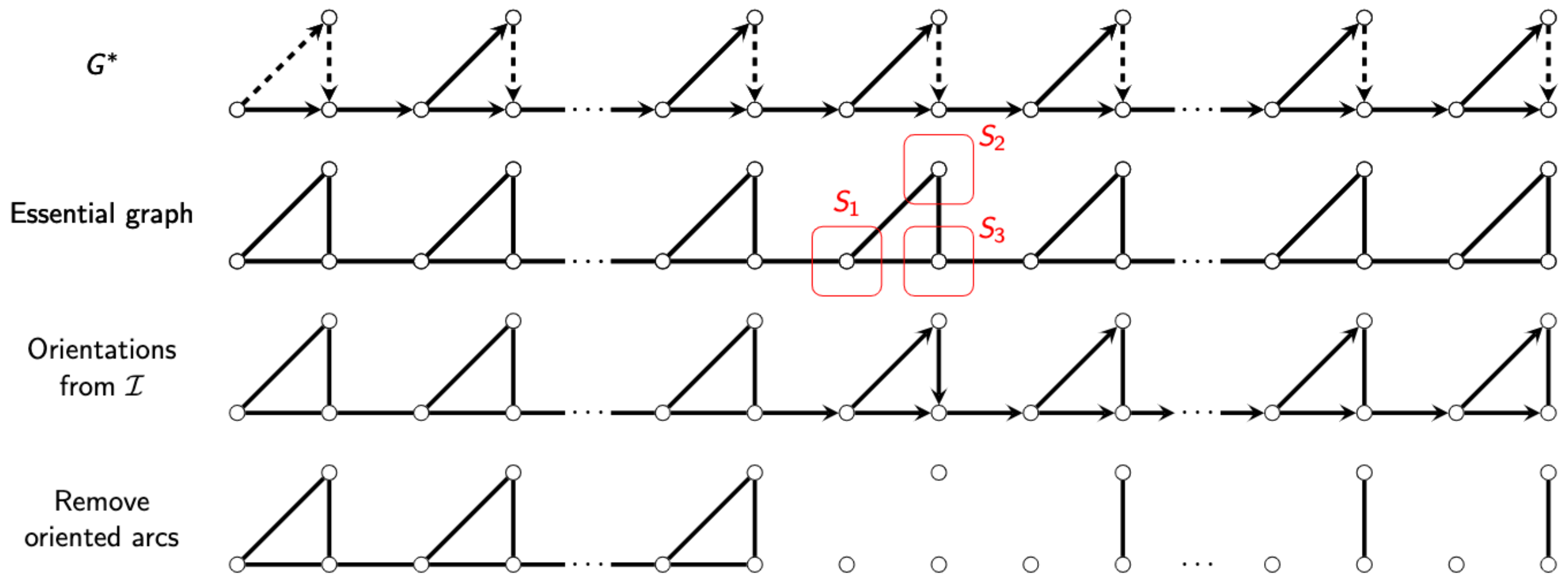
A stronger (but not computable) lower bound

Intuition [HB12,14]: In any interventional essential graph, interventions across different "connected components" *do not* help.

Claim: Fix an essential graph and some DAG G in it. Then,

[CSB22]

$$\nu(G) \geq \max_{\substack{\text{atomic}\\ \text{interventions}\\ S_1,\ldots,S_t}} \sum_{\substack{\text{connected components}\\ H \in \text{ after removing oriented arcs}\\ \text{after interventions } S_1,\ldots,S_t}} \left\lfloor \frac{\omega(H)}{2} \right\rfloor$$



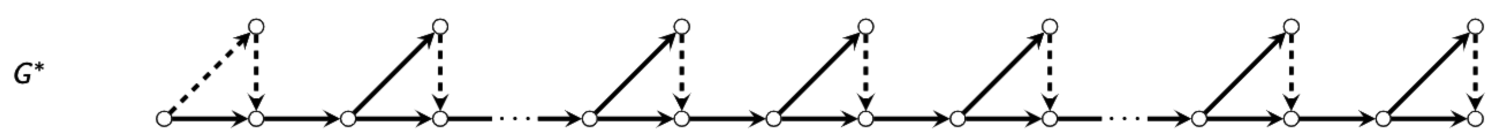
A stronger (but not computable) lower bound

Intuition [HB12,14]: In any interventional essential graph, interventions across different “connected components” *do not* help.

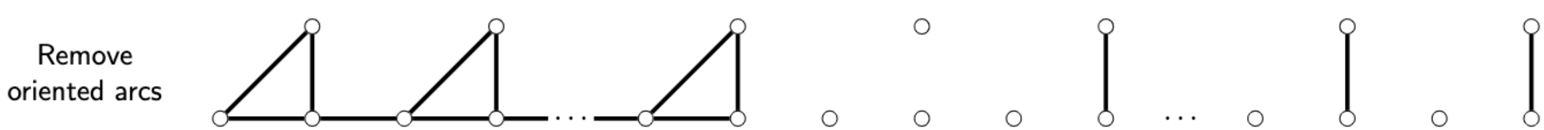
Claim: Fix an essential graph and some DAG G in it. Then,

[CSB22]

$$\nu(G) \geq \max_{\substack{\text{atomic}\\ \text{interventions}\\ S_1,\ldots,S_t}} \sum_{\substack{\text{connected components}\\ H \in \text{ after removing oriented arcs}\\ \text{after interventions } S_1,\ldots,S_t}} \left\lfloor \frac{\omega(H)}{2} \right\rfloor$$

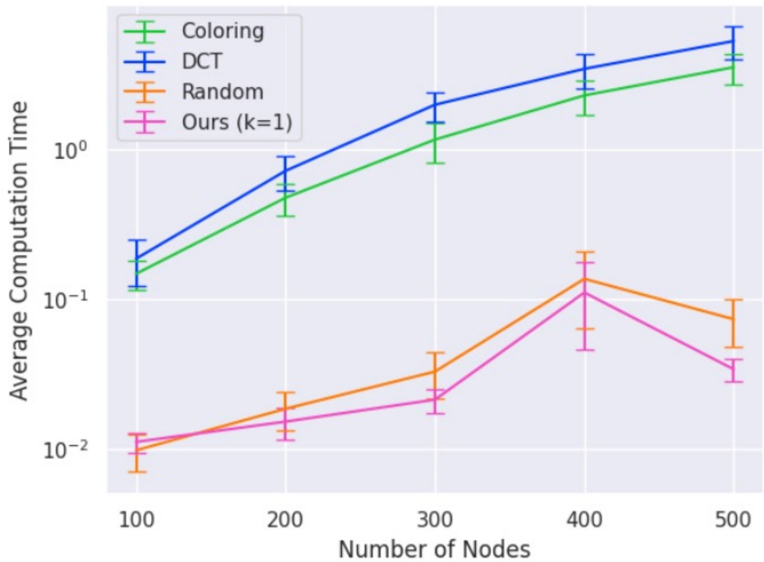


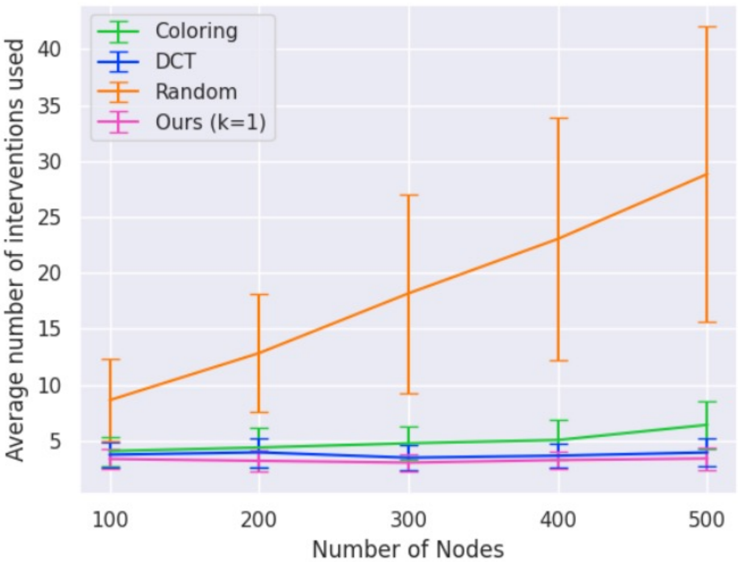
$$\nu(G^*) \geq \left\lfloor \frac{3}{2} \right\rfloor + 1 + \cdots + 1 \in \Omega(n)$$



The adaptive search algorithm

- Qualitatively, our algorithm is competitive with state-of-the-art adaptive search algorithms
  - We run $\sim 10\times$ faster in some experiments



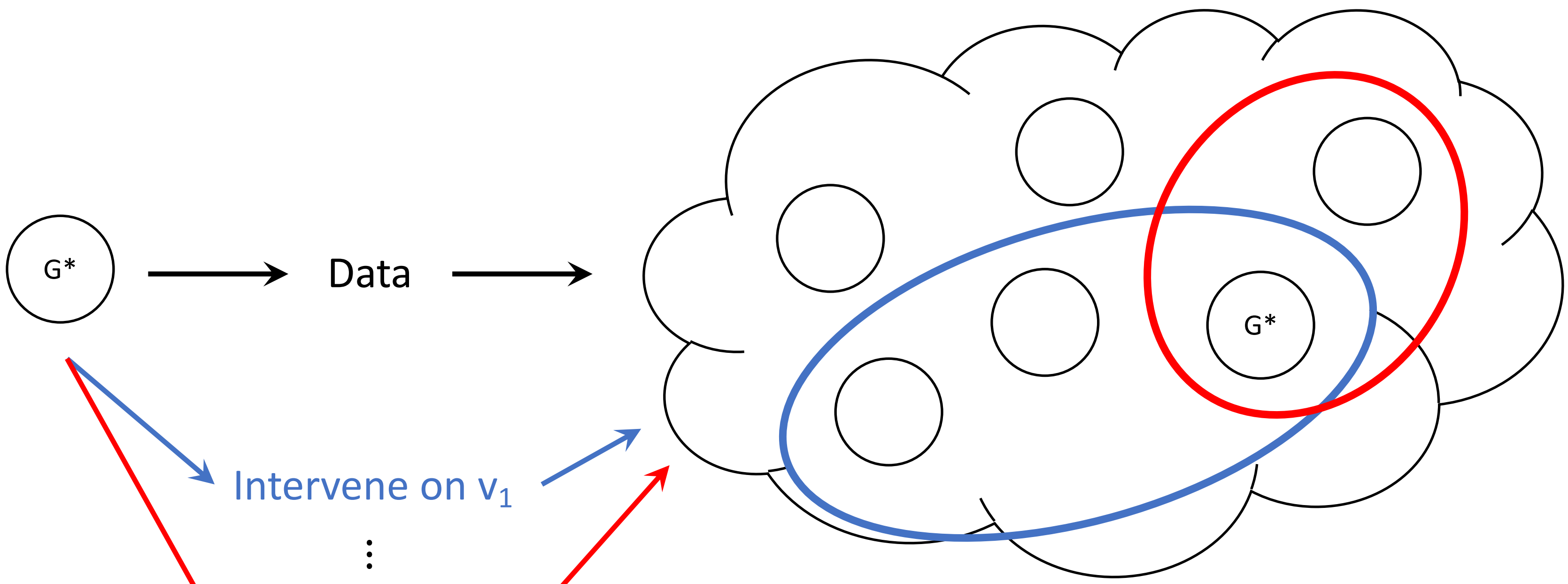


Implementation and more experimental results: https://github.com/cxjdavin/verification-and-search-algorithms-for-causal-DAGs

Problem setup

Identify G* using **as few interventions as possible** (minimize t)

Markov equivalence class of G*



(can be represented by an essential graph)

Verification: $\nu(G^*)$ = size of minimum vertex cover of covered edges [CSB22]
Search: $\mathcal{O}\left(\log n \cdot \nu(G^*)\right)$ interventions suffice [CSB22]

But wait, there’s more!

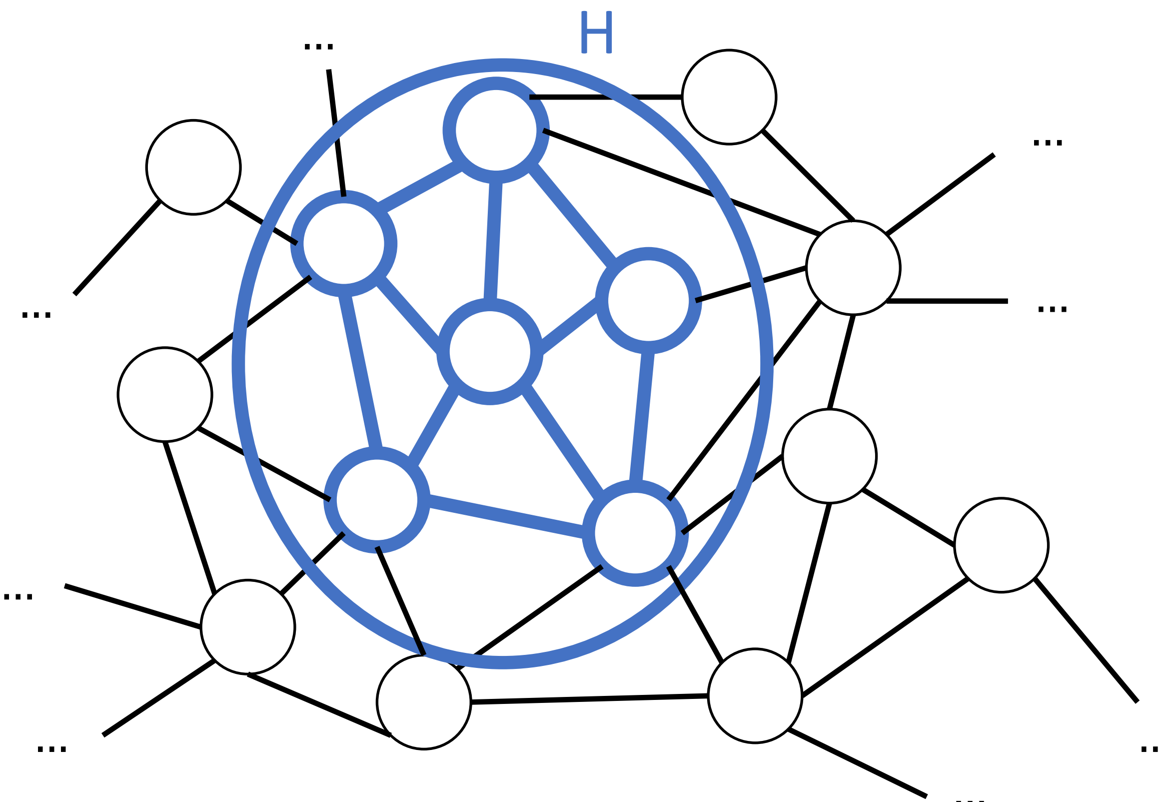
Other extensions / questions

- What if the causal graph is HUGE?
- What if we consult domain experts for advice?
- What if we intervene >1 vertex per intervention?
  - Bounded size interventions
- What if vertices have different interventional costs?
  - Additive cost ⇒ cost of intervention is cost of all vertices in it
- What if we have limited rounds of adaptivity?
  - At most r rounds, for $r < \log n$
- Can we weaken/remove the causal assumptions?
  - What if there are hidden confounders?
  - What if we don't have faithfulness?
  - What if we have finite samples? i.e. May incur error in CI checks
  - Beyond hard interventions? Soft/unknown interventions, etc.

Note: Some are in the papers mentioned earlier in this talk; some are solved and under submission. If you are interested, come talk to me to find out more ☺

Backup slides

What if causal graph is HUGE?



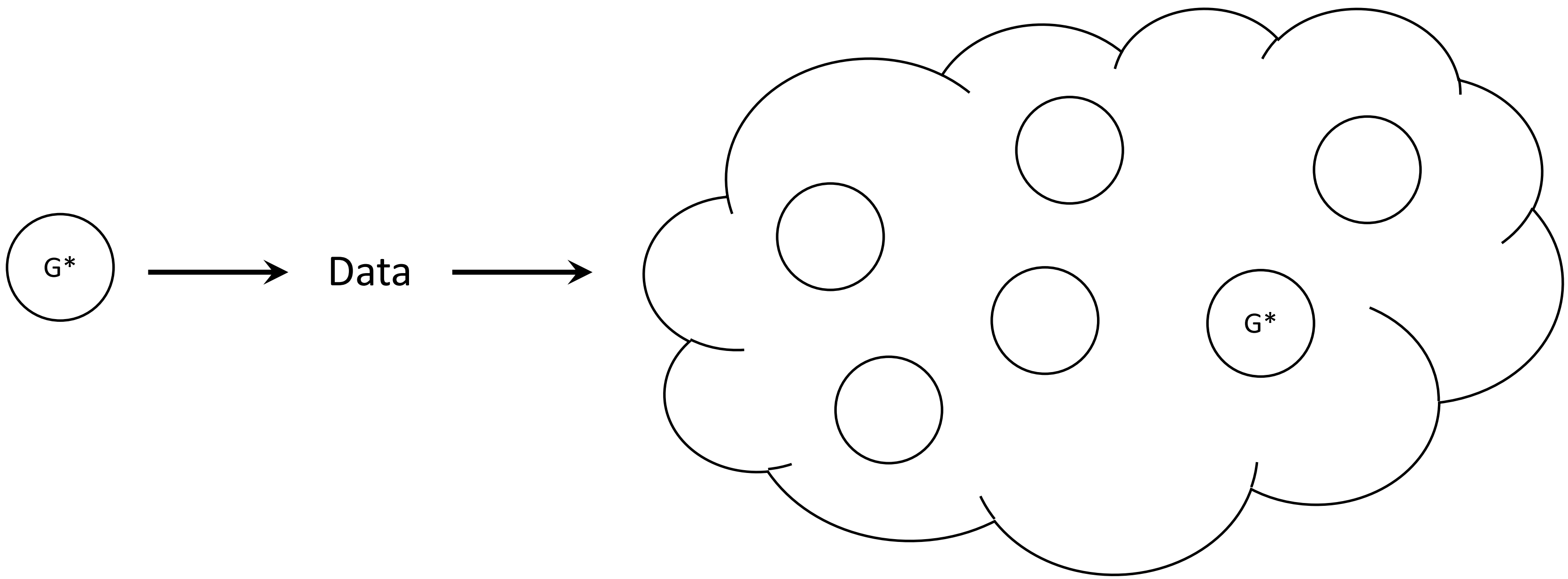
Local causal discovery:

Only care about a small subgraph of the larger graph

(Informal) Verification: Generalization of “DP on covered edge forest” [CS23]

(Informal) Search: $\mathcal{O}\left(\log |H| \cdot \nu(G^*)\right)$ interventions suffices [CS23]

In many problem domains…



There are domain experts!

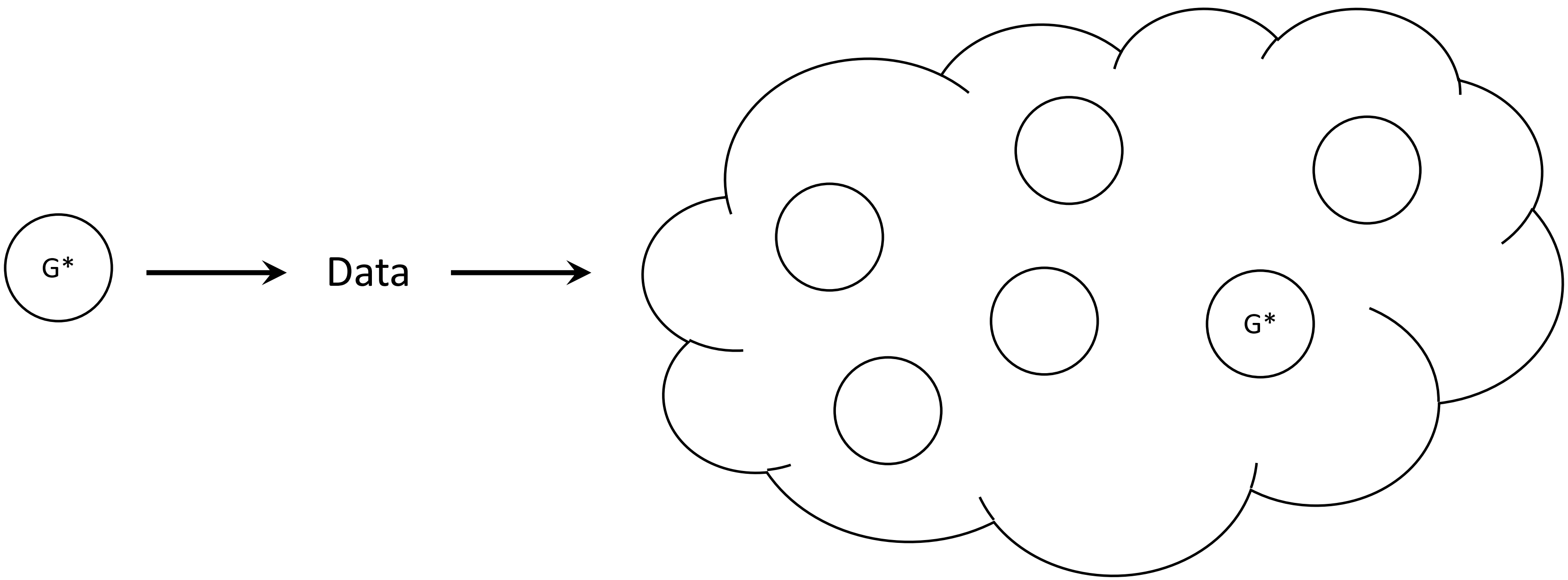


Image credit: https://thenounproject.com/icon/doctor-1285618/

There are domain experts!

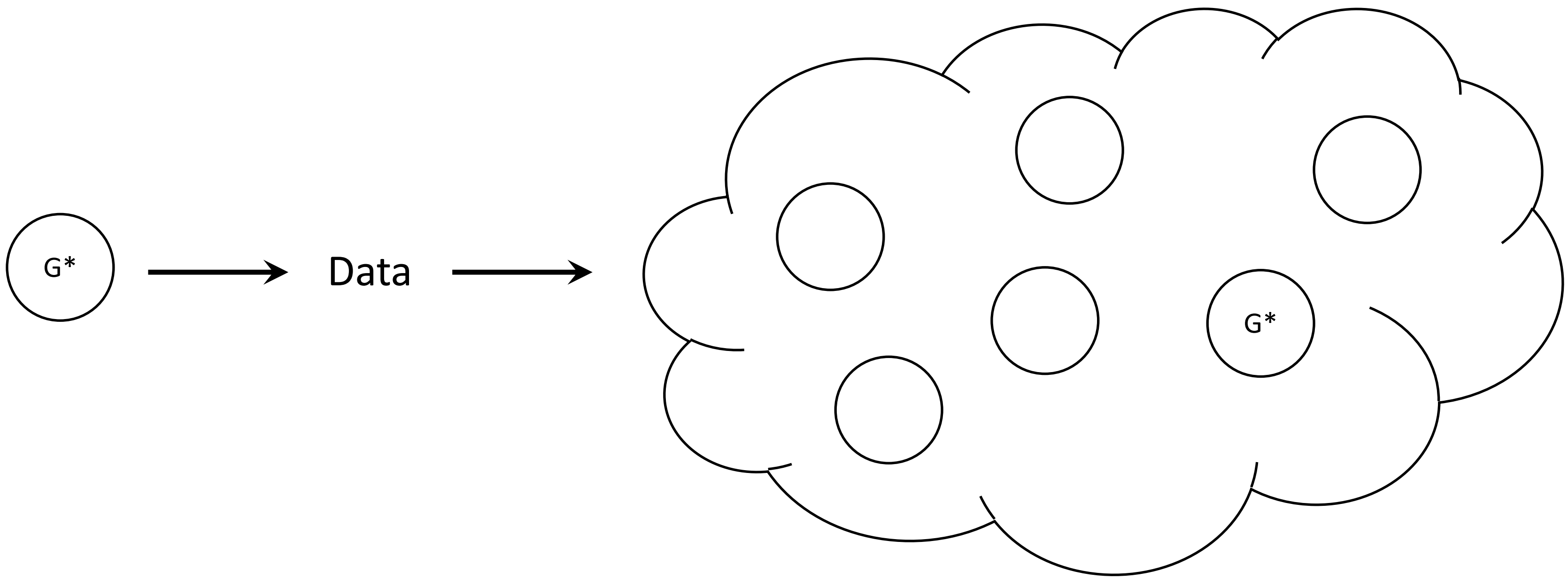
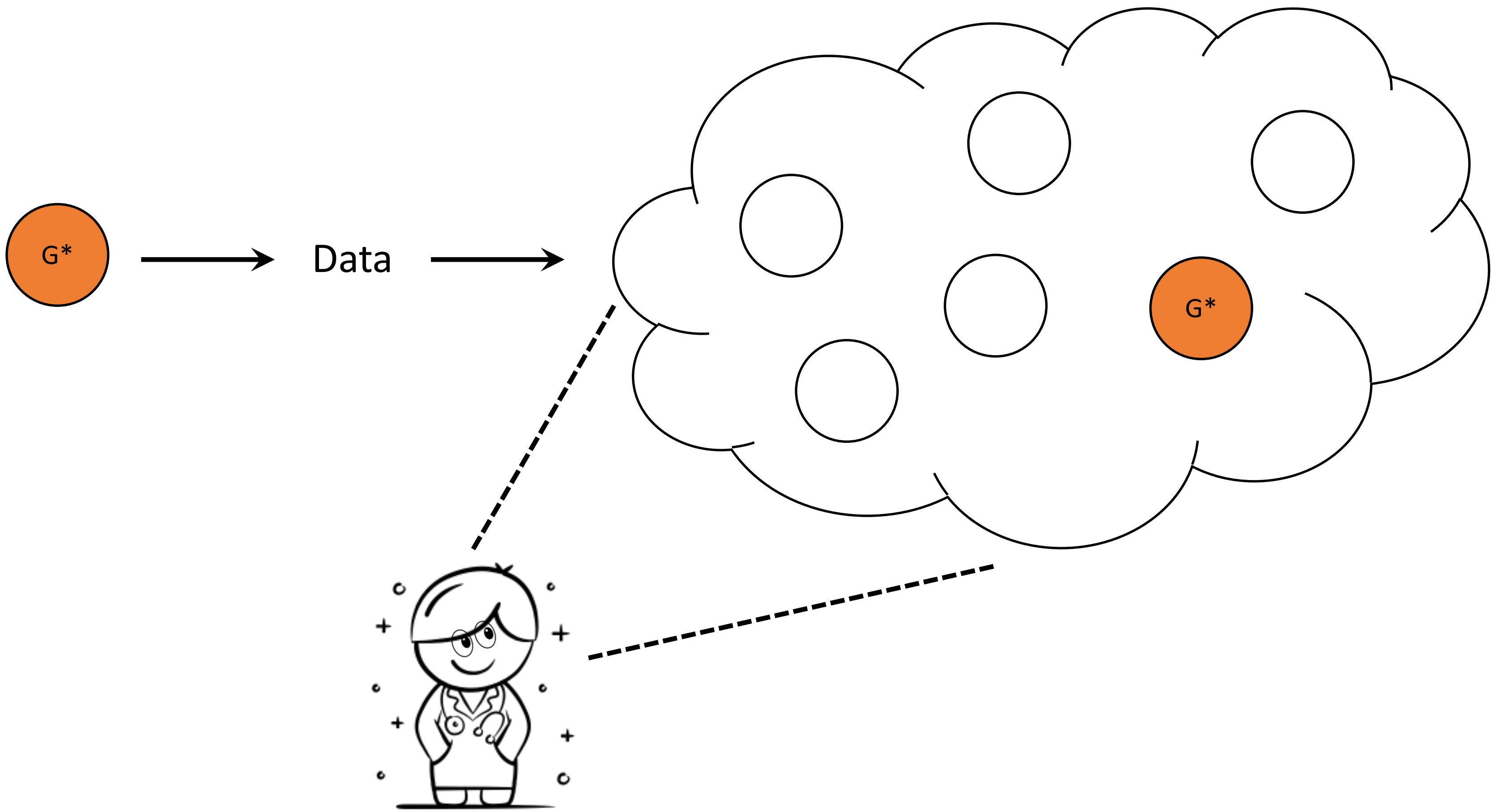
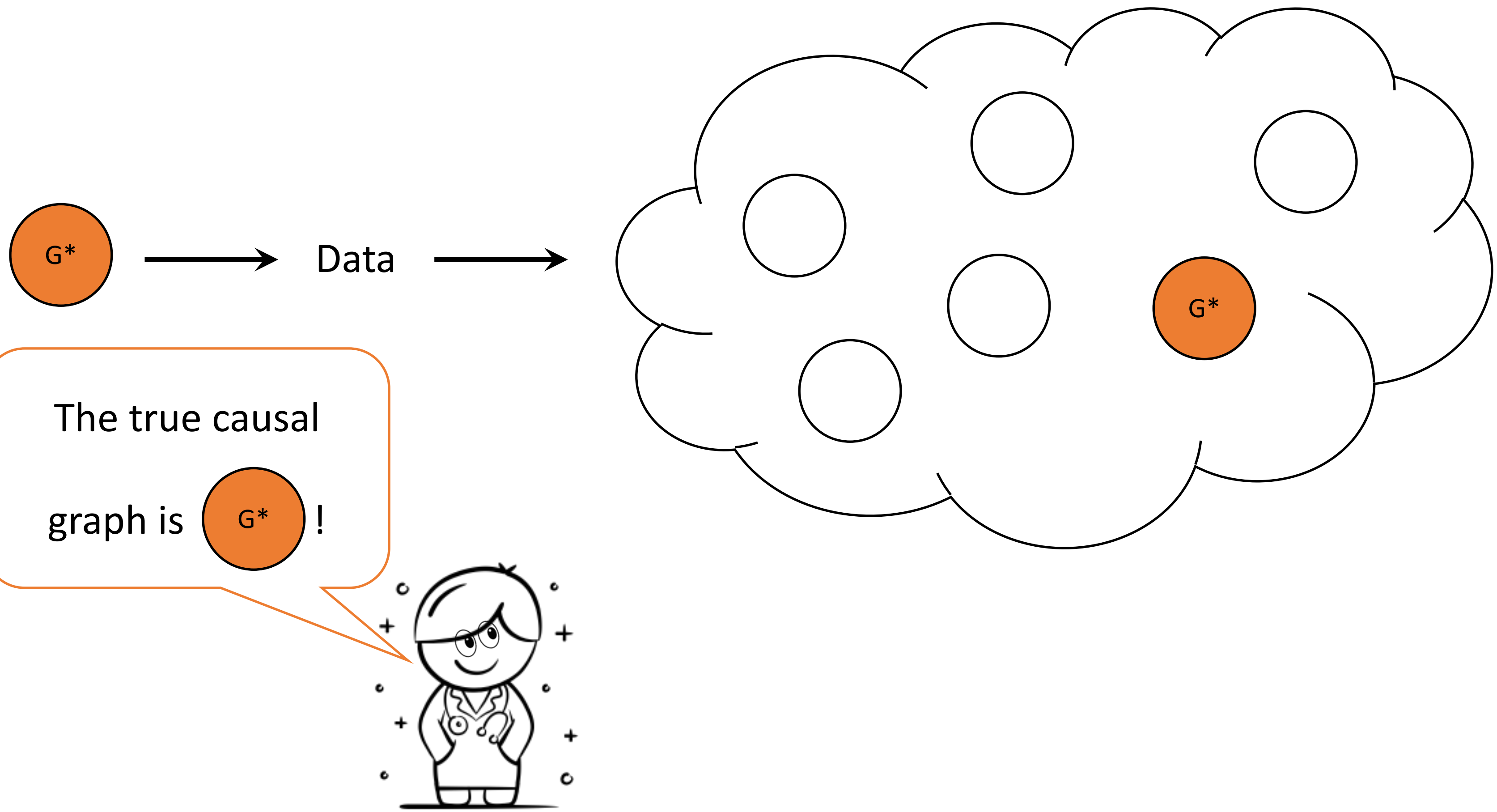


Image credit: https://thenounproject.com/icon/look-4968809/

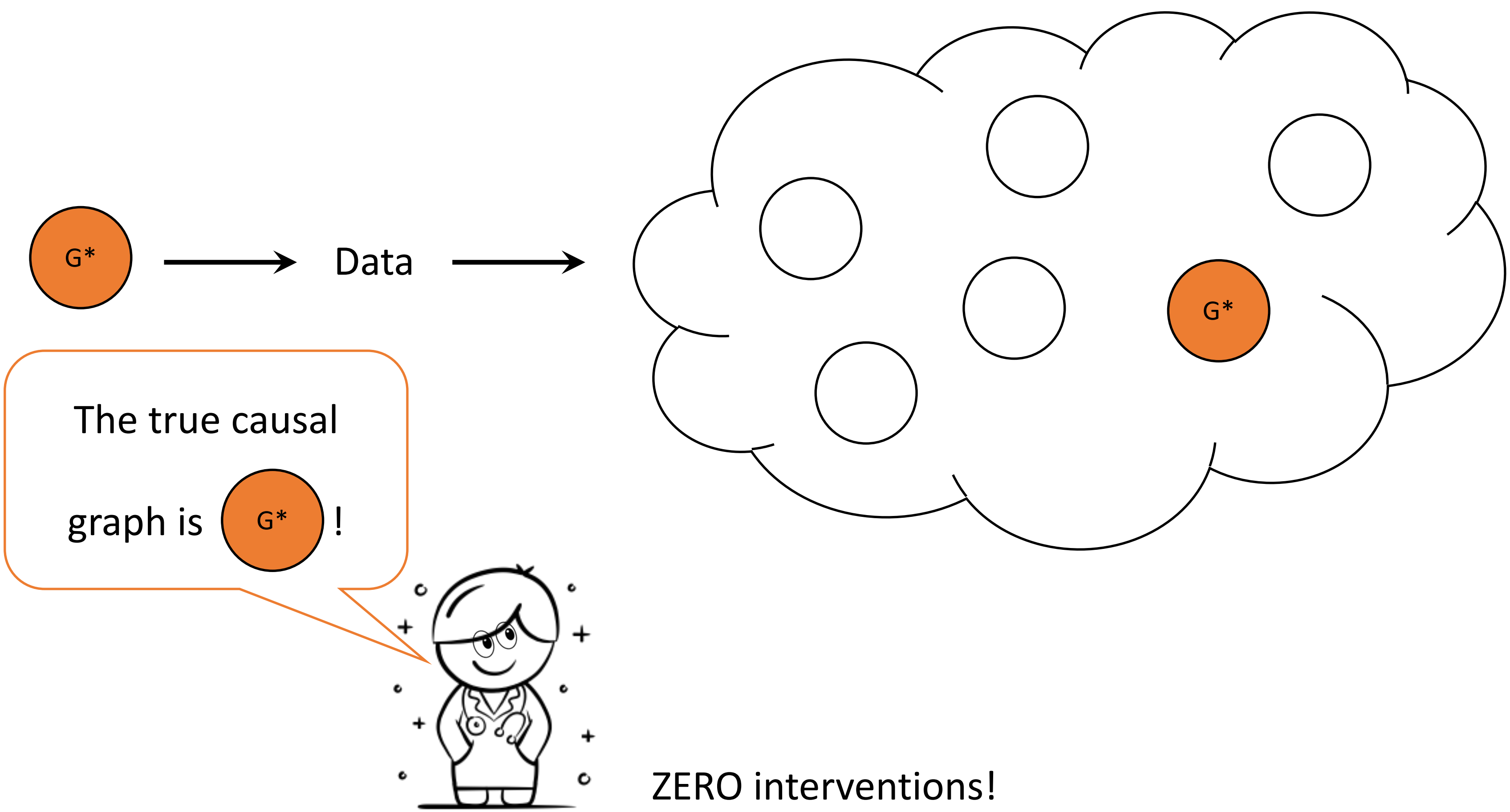
There are domain experts!



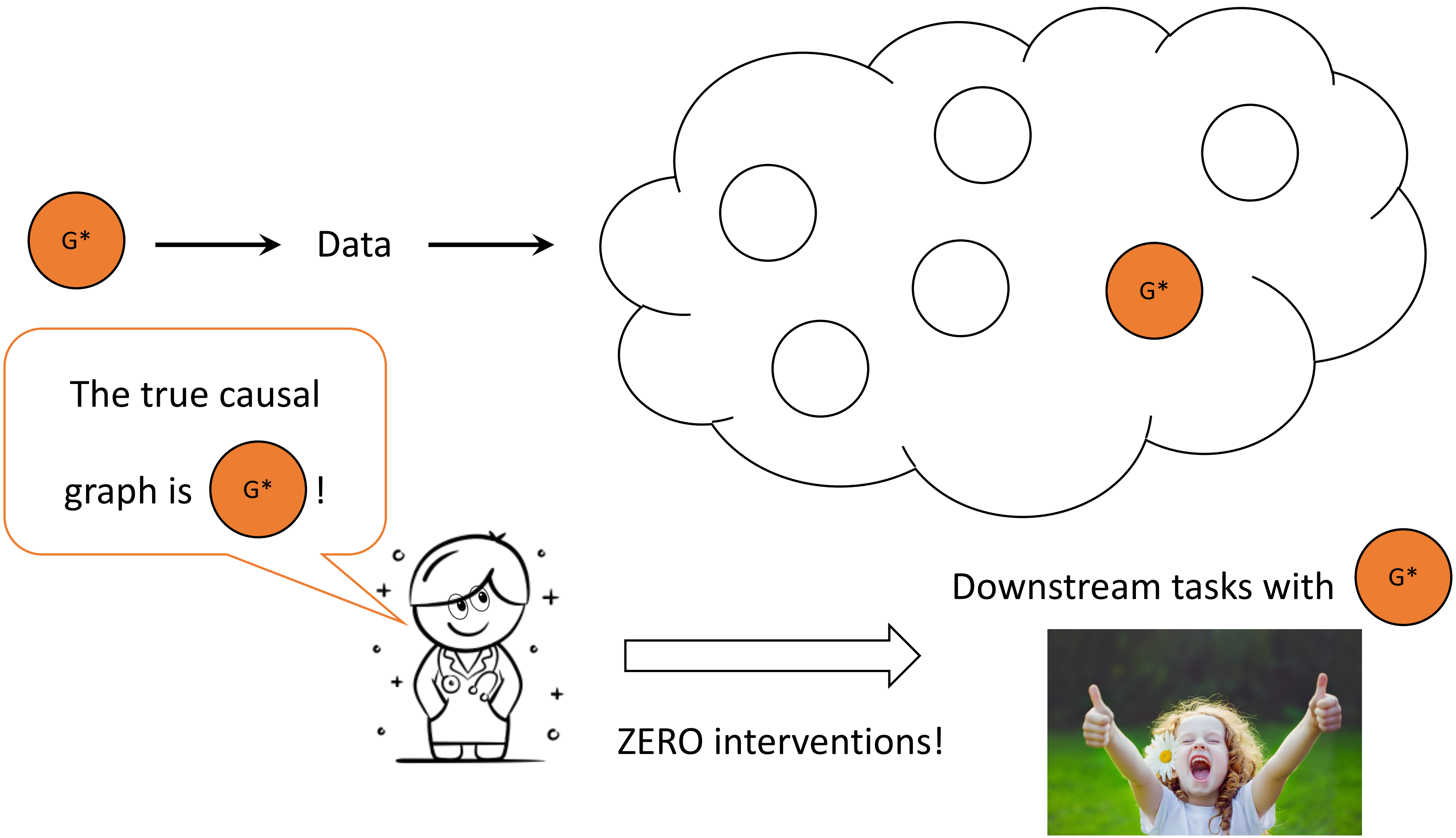
There are domain experts!



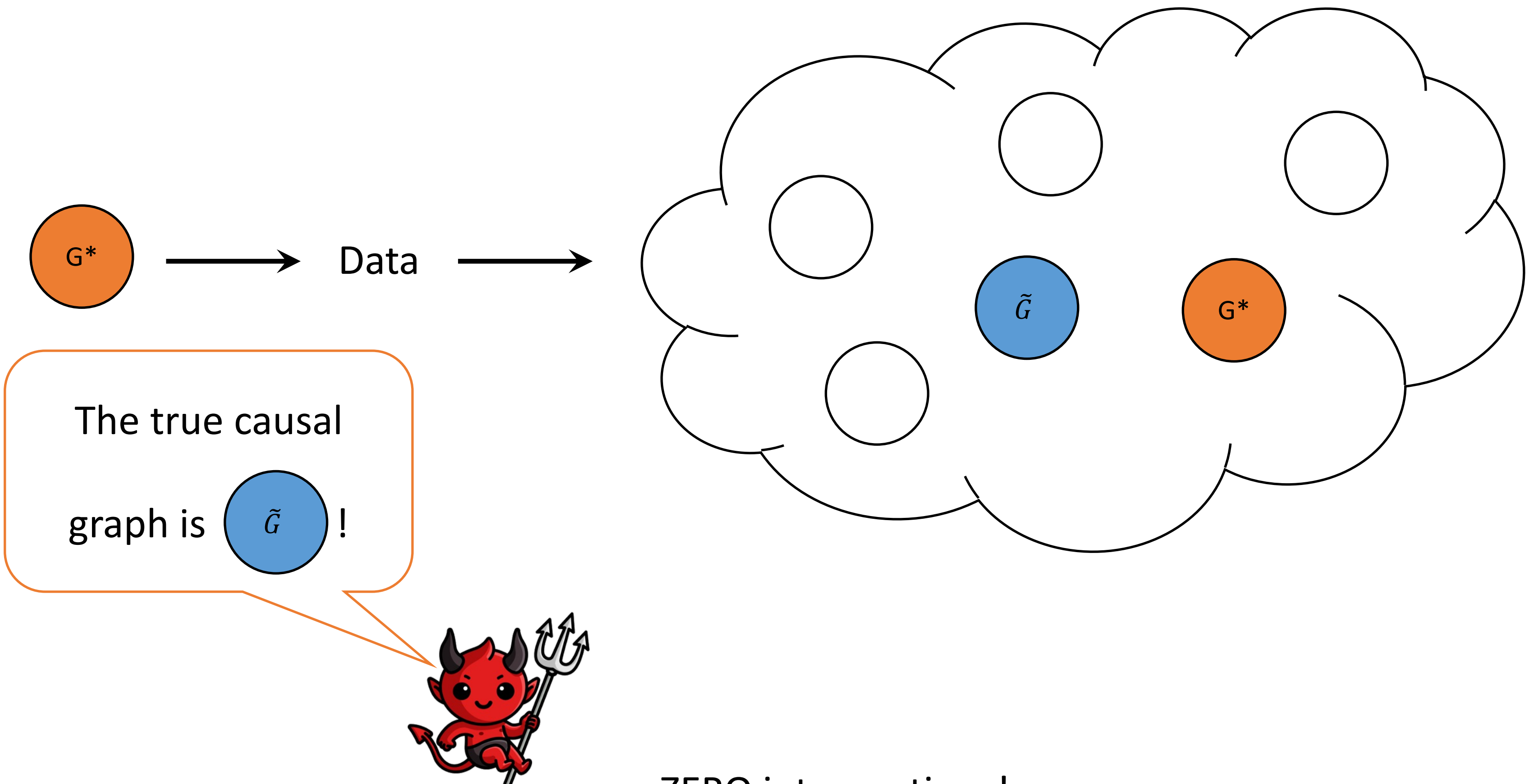
There are domain experts!



There are domain experts!



But... experts can be wrong



ZERO interventions!

Image credit: https://dribbble.com/shots/14489872-Devil

But… experts can be wrong

G^* → Data →

$\tilde{G}$ G^*

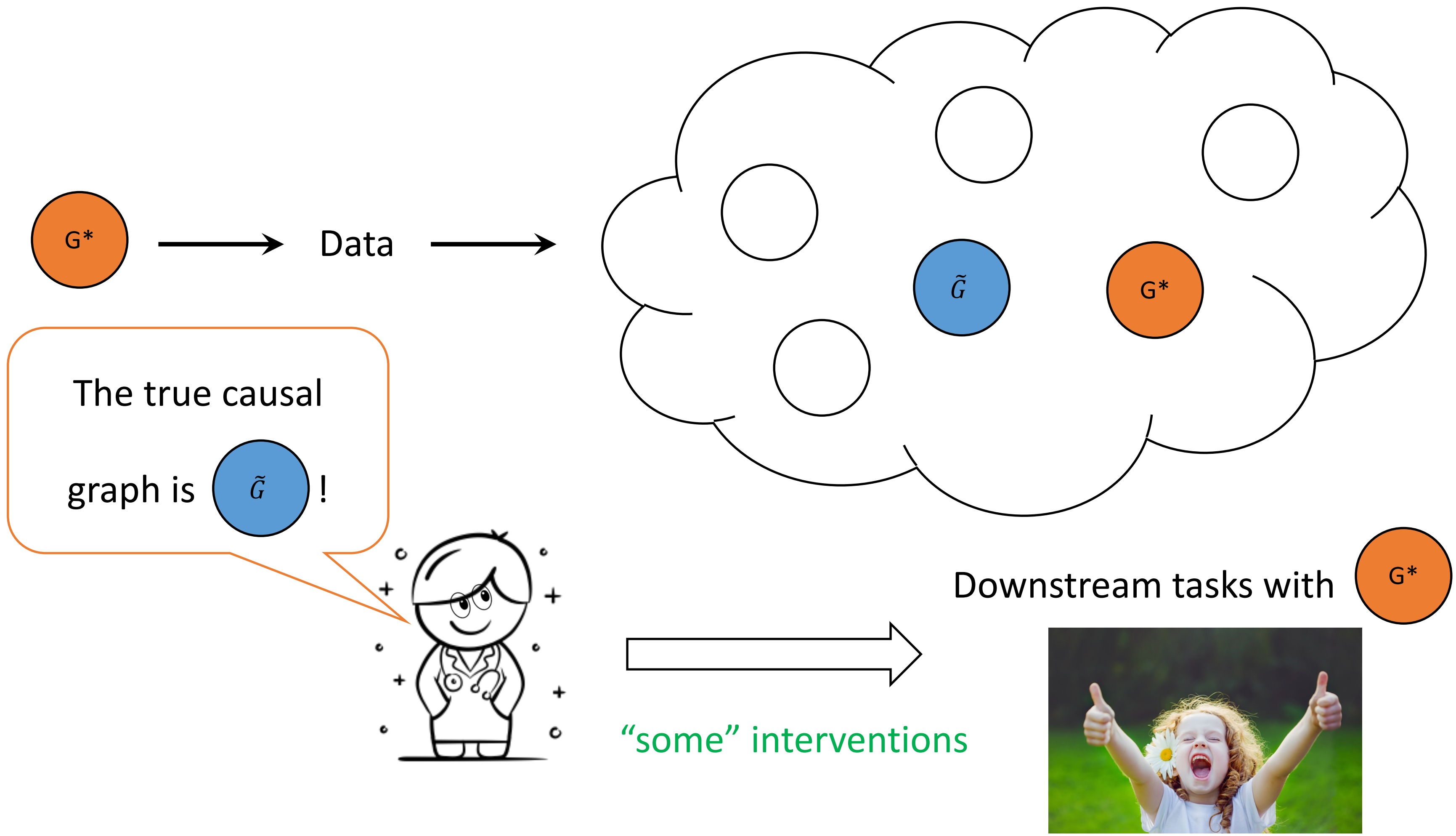
The true causal graph is $\tilde{G}$!

ZERO interventions!

Downstream tasks with $\tilde{G}$

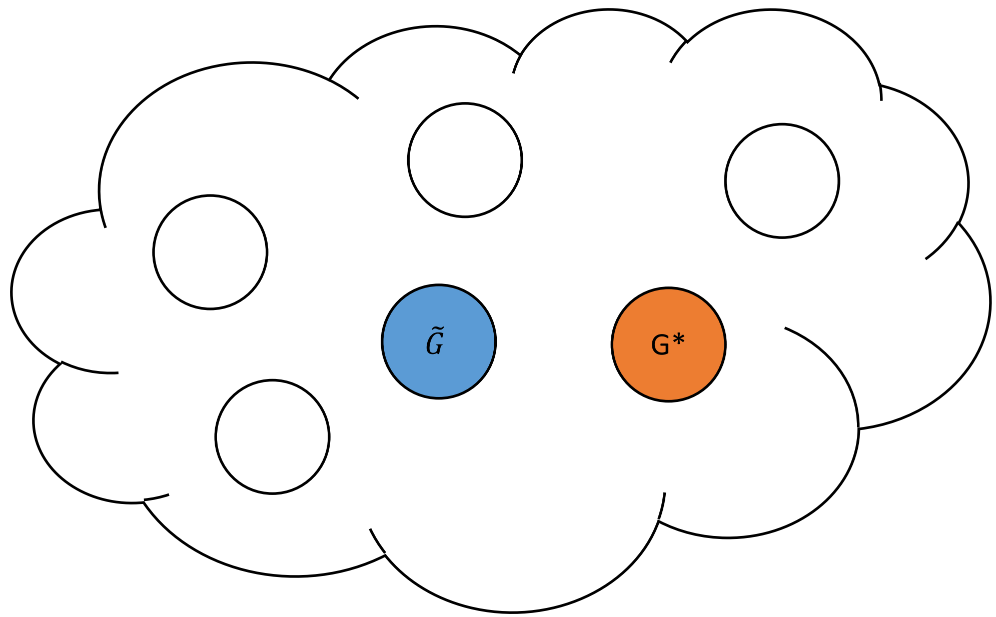
Image credit: https://dribbble.com/shots/3759014-Atomic-Illustrations/attachments/3759014-Atomic-Illustrations?mode=media

Searching with imperfect advice



Searching with imperfect advice

G* ⟶ Data ⟶



The true causal graph is $\tilde{G}$!

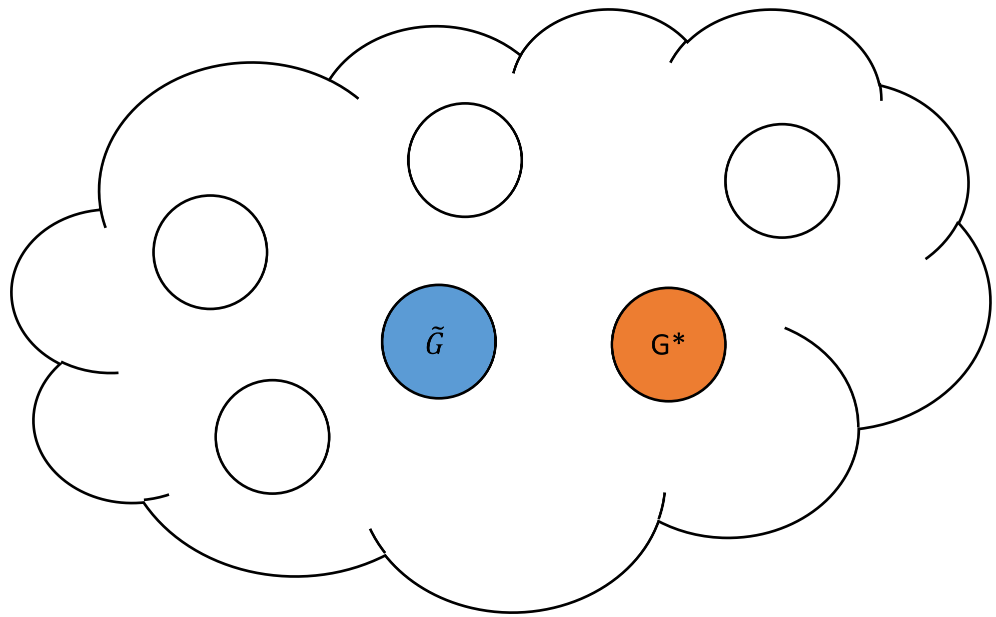
“some” interventions

Downstream tasks with G*

Advice search: $\mathcal{O}\left(\log \psi(G^*, \tilde{G}) \cdot \nu(G^*)\right)$ interventions [CGB23]

Searching with imperfect advice

G* ⟶ Data ⟶



The true causal graph is $\tilde{G}$!

Quality of advice $\tilde{G}$

$0 \leq \psi(G^*, \tilde{G}) \leq n$

(good) (bad)

"some" interventions

Downstream tasks with G*

Advice search: $\mathcal{O}\left(\log \psi(G^*, \tilde{G}) \cdot \nu(G^*)\right)$ interventions [CGB23]

d-separation

- Consider a path $X \sim \cdots \sim Y$ in the DAG
  - $X \sim \cdots \sim Y$ is blocked by a set $\boldsymbol{Z}$ if either holds:
    1. Along the path, we have
       $X \sim \cdots \rightarrow W \rightarrow \cdots \sim Y$ or
       $X \sim \cdots \leftarrow W \leftarrow \cdots \sim Y$ or
       $X \sim \cdots \leftarrow W \rightarrow \cdots \sim Y$,
       where $W \in \boldsymbol{Z}$
    2. Along the path, we have collider $X \sim \cdots \rightarrow W \leftarrow \cdots \sim Y$,
       where W and its descendants are **not** in $\boldsymbol{Z}$
  - $\boldsymbol{Z}$ could be the empty set
- We write as $X \perp\!\!\!\perp_{\mathrm{G}} Y \mid \boldsymbol{Z}$
- Notion generalizes to sets $\boldsymbol{X}$ and $\boldsymbol{Y}$

Common causality assumptions

- Markov assumption

$$X \perp\!\!\!\perp_{\mathrm{G}} Y \mid Z \Rightarrow X \perp\!\!\!\perp_{\mathrm{P}} Y \mid Z$$

"If d-separated in graph, then conditionally independent in data"

- Faithfulness

$$X \perp\!\!\!\perp_{\mathrm{G}} Y \mid Z \Leftarrow X \perp\!\!\!\perp_{\mathrm{P}} Y \mid Z$$

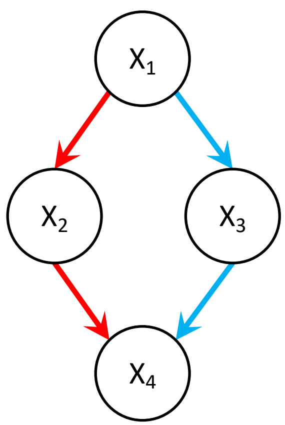
"If conditionally independent in data, then d-separated in graph"

Common causality assumptions

- Faithfulness

$$X \perp\!\!\!\perp_{\mathrm{G}} Y \mid Z \Leftarrow X \perp\!\!\!\perp_{\mathrm{P}} Y \mid Z$$

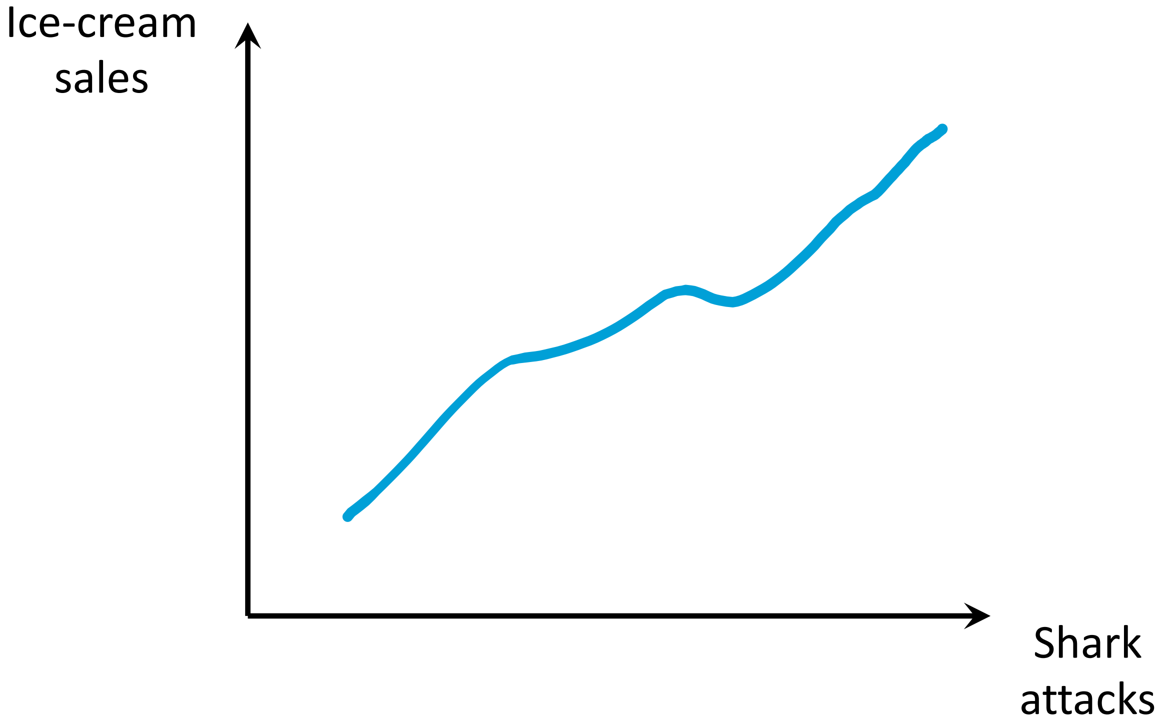
  - No "cancellations" in the distribution
  - Toy example (ignoring noise terms):



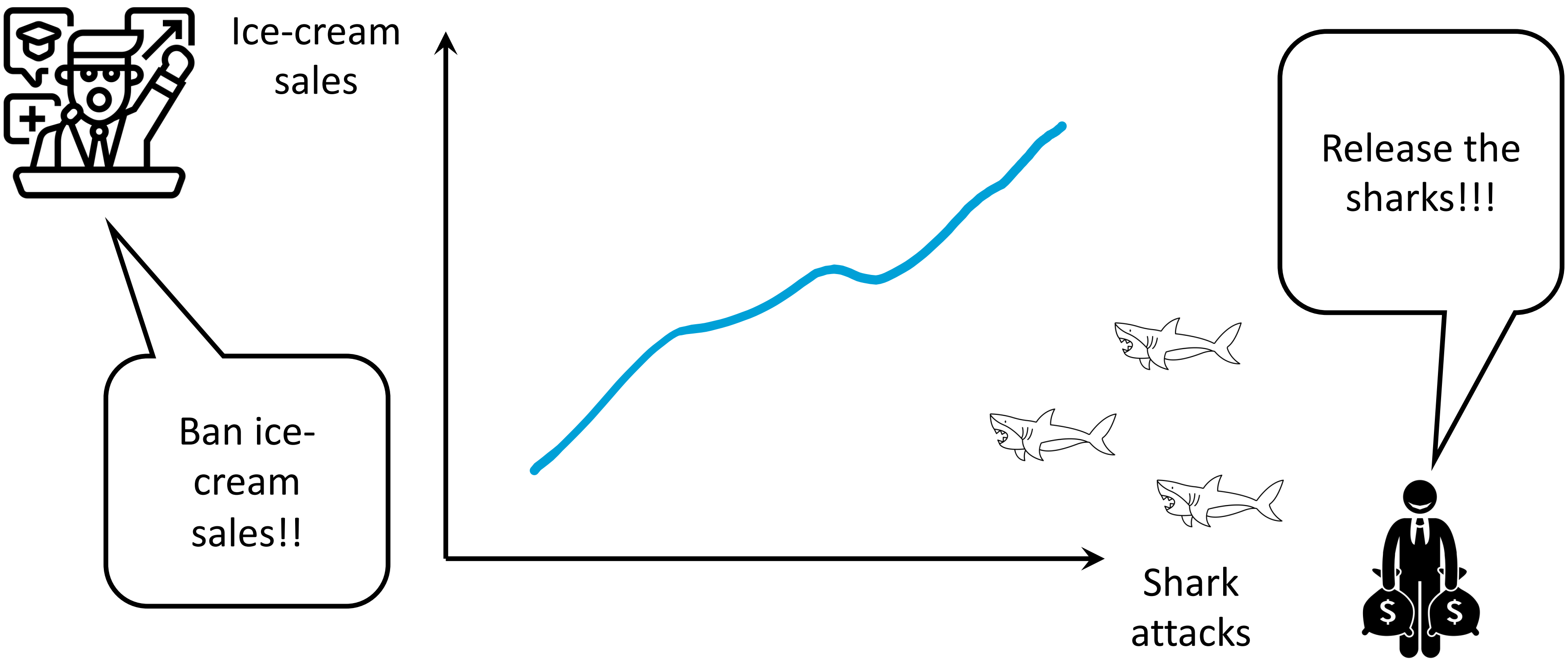
SEM:
$$X_2 = a\, X_1$$
$$X_3 = b\, X_1$$
$$X_4 = c\, X_2 + d\, X_3 = (ac + bd)\, X_1$$

Consider scenario where red and blue paths "cancel out"
If $ac = -bd$, then $X_4 = 0$ always, and we have $\mathrm{X}_1 \perp\!\!\!\perp_{\mathrm{P}} X_4$
If faithfulness holds, then the DAG should show $\mathrm{X}_1 \perp\!\!\!\perp_{\mathrm{G}} X_4$
But X_1 and X_4 **not** d-separated in this DAG
So, faithfulness violated when $ac = -bd$

Common causality assumptions



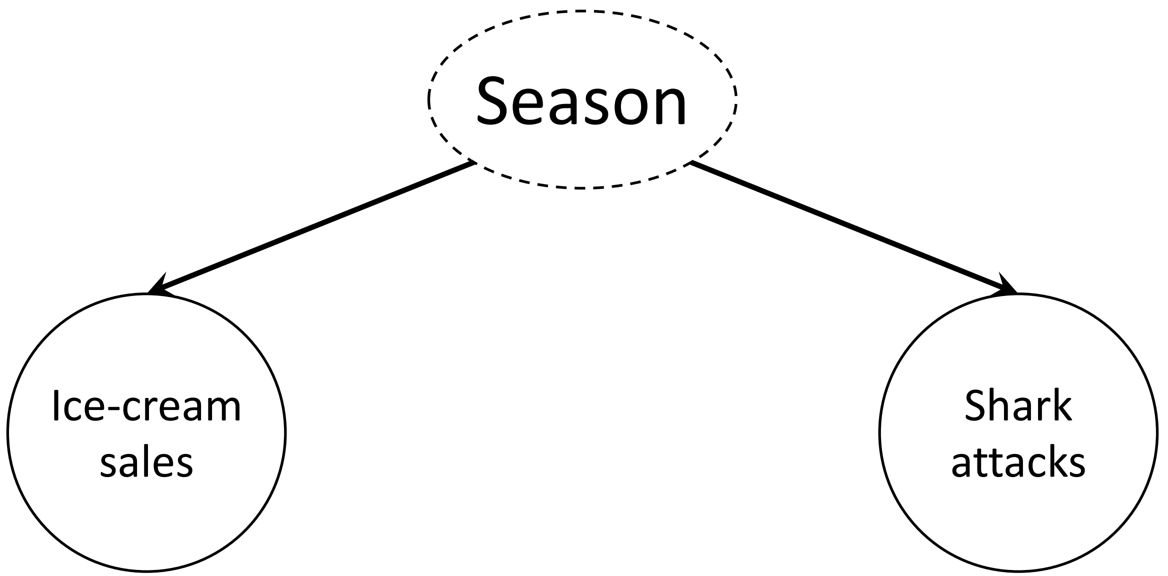
Common causality assumptions



https://thenounproject.com/icon/politician-3189333/
https://thenounproject.com/icon/evil-businessman-with-money-2758402/
https://thenounproject.com/icon/shark-4022671/

Common causality assumptions

- Causal sufficiency
  - No unobserved confounders / common cause



When warm weather, more people buy ice-cream, and more people go to beaches

PC algorithm [Spirtes, Glymour, Scheines, Heckerman 2000]

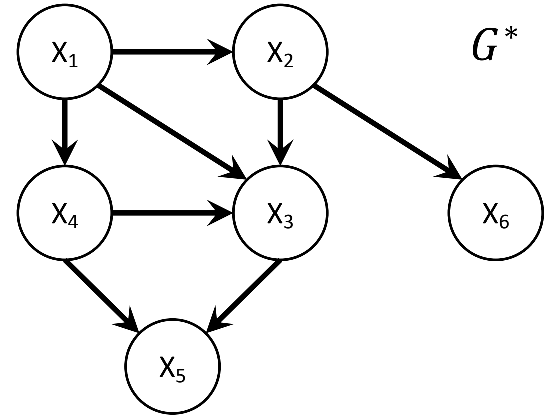
- A classic constraint-based method for causal graph discovery
- Steps
  1. **Identify skeleton** **(See backup slides if time permits)**
     - Start with complete undirected graph
     - Remove edges $X \sim Y$ when $X \perp\!\!\!\perp Y \mid Z$ for conditioning set Z from $\emptyset, \{x_1\}, \dots, \{x_n\}, \{x_1, x_2\}, \dots, \{x_{n-1}, x_n\}, \dots, \{x_1, \dots, x_n\}$
  2. **Identify v-structures**
     - Consider any path $X \sim Y \sim Z$ without $X \sim Z$
     - If Y was **not** used to remove edge $X \sim Y$ in step 1, then it must be the case that $X \rightarrow Y \leftarrow Z$
  3. **Orient more edges using the discovered v-structures**
     - Apply Meek rules
- Fact: If we can always correctly determine conditional independencies, then PC will output G^*

Key takeaway: With enough samples, we can recover essential graph

PC algorithm [Spirtes, Glymour, Scheines, Heckerman 2000]

- A classic constraint-based method for causal graph discovery
- Steps
  1. Identify skeleton
     - Start with complete undirected graph
     - Remove edges $X \sim Y$ when $X \perp\!\!\!\perp Y \mid Z$ for conditioning set Z from $\emptyset, \{x_1\}, \dots, \{x_n\}, \{x_1, x_2\}, \dots, \{x_{n-1}, x_n\}, \dots, \{x_1, \dots, x_n\}$
  2. Identify v-structures
     - Consider any path $X \sim Y \sim Z$ without $X \sim Z$
     - If Y was **not** used to remove edge $X \sim Y$ in step 1, then it must be the case that $X \rightarrow Y \leftarrow Z$
  3. Orient more edges using the discovered v-structures
     - Apply Meek rules
- Fact: If we can always correctly determine conditional independencies, then PC will output G^*

Example: PC algorithm



1. Identify skeleton

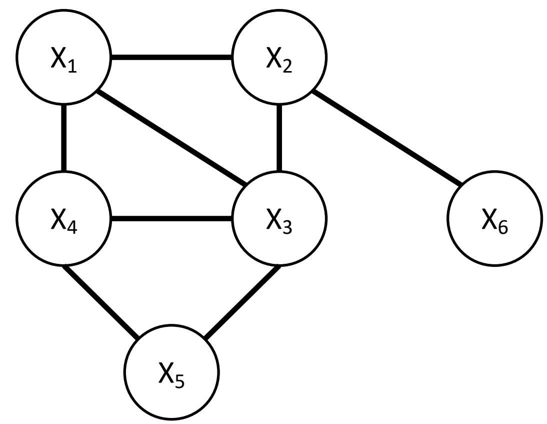
$X_1 \perp\!\!\!\perp X_5 \mid X_3, X_4$
$X_1 \perp\!\!\!\perp X_6 \mid X_2$

$X_2 \perp\!\!\!\perp X_4 \mid X_1$
$X_2 \perp\!\!\!\perp X_5 \mid X_3, X_4$

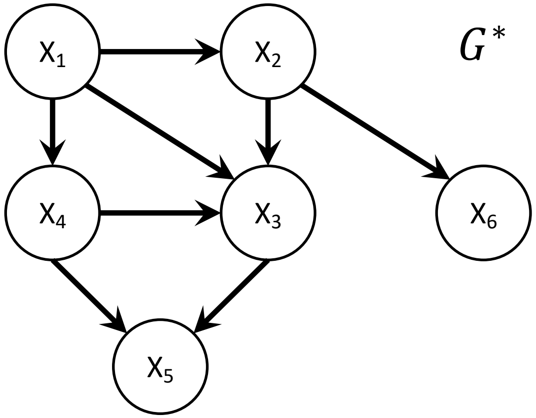
$X_3 \perp\!\!\!\perp X_6 \mid X_2$

$X_4 \perp\!\!\!\perp X_6 \mid X_1$ or $X_4 \perp\!\!\!\perp X_6 \mid X_2$

$X_5 \perp\!\!\!\perp X_6 \mid X_2$



Example: PC algorithm



2. Identify v-structures

$X_1 \perp\!\!\!\perp X_5 \mid X_3, X_4$
$X_1 \perp\!\!\!\perp X_6 \mid X_2$

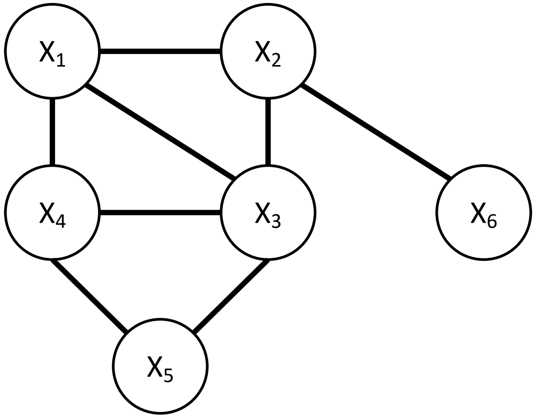
$X_2 \perp\!\!\!\perp X_4 \mid X_1$
$X_2 \perp\!\!\!\perp X_5 \mid X_3, X_4$

$X_3 \perp\!\!\!\perp X_6 \mid X_2$

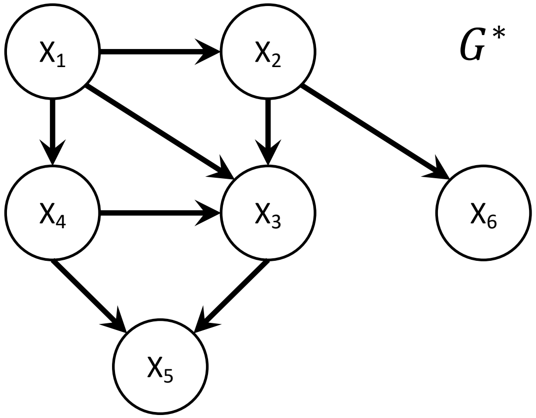
$X_4 \perp\!\!\!\perp X_6 \mid X_1$ or $X_4 \perp\!\!\!\perp X_6 \mid X_2$

$X_5 \perp\!\!\!\perp X_6 \mid X_2$

Look at all triples $A \sim B \sim C$ and $A \nsim C$
If $C \notin sepset(A, B)$, then $A \rightarrow B \leftarrow C$



Example: PC algorithm



2. Identify v-structures

$X_1 \perp\!\!\!\perp X_5 \mid X_3, X_4$
$X_1 \perp\!\!\!\perp X_6 \mid X_2$

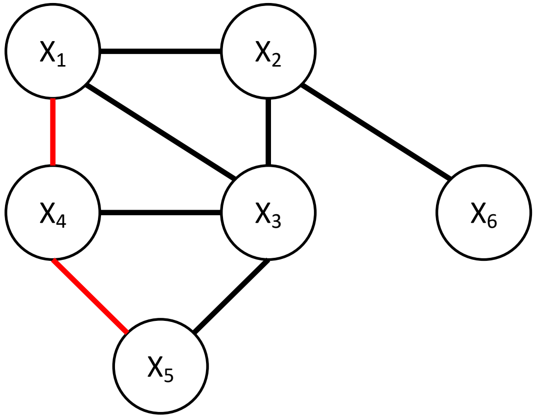
$X_2 \perp\!\!\!\perp X_4 \mid X_1$
$X_2 \perp\!\!\!\perp X_5 \mid X_3, X_4$

$X_3 \perp\!\!\!\perp X_6 \mid X_2$

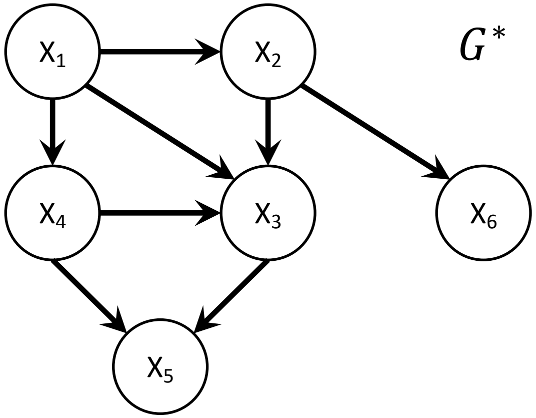
$X_4 \perp\!\!\!\perp X_6 \mid X_1$ or $X_4 \perp\!\!\!\perp X_6 \mid X_2$

$X_5 \perp\!\!\!\perp X_6 \mid X_2$

Look at all triples $A \sim B \sim C$ and $A \nsim C$
If $C \notin sepset(A, B)$, then $A \to B \leftarrow C$



Example: PC algorithm



2. Identify v-structures

$X_1 \perp\!\!\!\perp X_5 \mid X_3, X_4$
$X_1 \perp\!\!\!\perp X_6 \mid X_2$

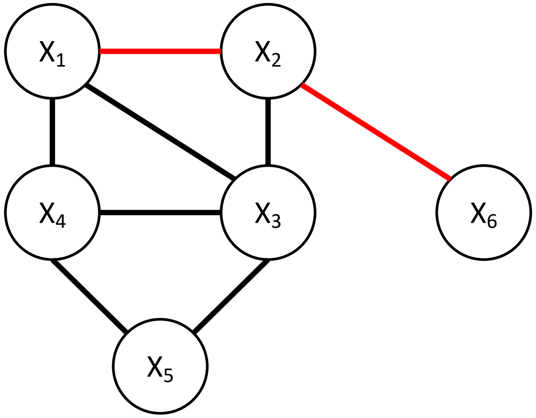
$X_2 \perp\!\!\!\perp X_4 \mid X_1$
$X_2 \perp\!\!\!\perp X_5 \mid X_3, X_4$

$X_3 \perp\!\!\!\perp X_6 \mid X_2$

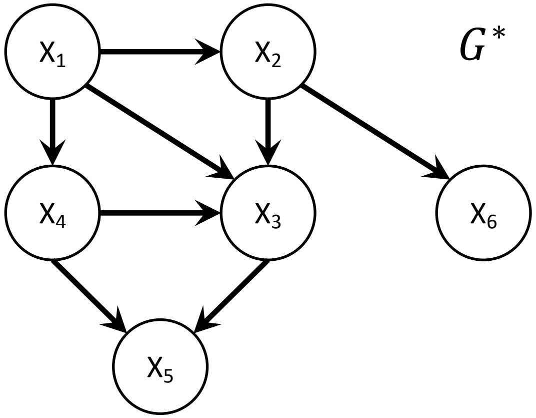
$X_4 \perp\!\!\!\perp X_6 \mid X_1$ or $X_4 \perp\!\!\!\perp X_6 \mid X_2$

$X_5 \perp\!\!\!\perp X_6 \mid X_2$

Look at all triples $A \sim B \sim C$ and $A \nsim C$
If $C \notin sepset(A, B)$, then $A \rightarrow B \leftarrow C$



Example: PC algorithm



G^*

2. Identify v-structures

$X_1 \perp\!\!\!\perp X_5 \mid X_3, X_4$
$X_1 \perp\!\!\!\perp X_6 \mid X_2$

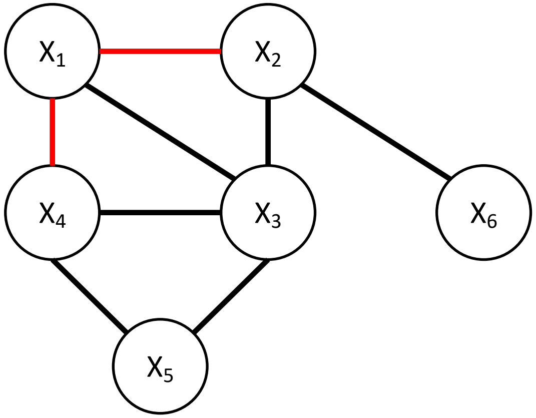
$X_2 \perp\!\!\!\perp X_4 \mid X_1$
$X_2 \perp\!\!\!\perp X_5 \mid X_3, X_4$

$X_3 \perp\!\!\!\perp X_6 \mid X_2$

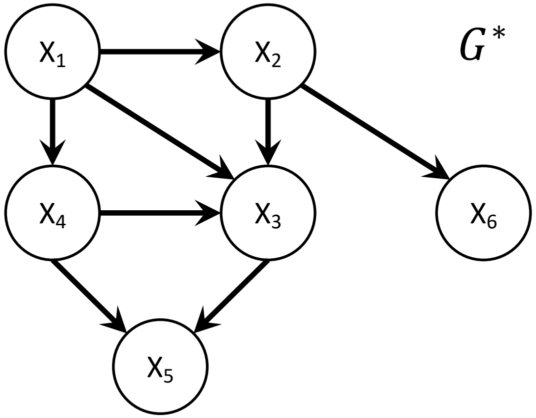
$X_4 \perp\!\!\!\perp X_6 \mid X_1$ or $X_4 \perp\!\!\!\perp X_6 \mid X_2$

$X_5 \perp\!\!\!\perp X_6 \mid X_2$

Look at all triples $A \sim B \sim C$ and $A \nsim C$
If $C \notin sepset(A, B)$, then $A \rightarrow B \leftarrow C$



Example: PC algorithm



2. Identify v-structures

$X_1 \perp\!\!\!\perp X_5 \mid X_3, X_4$
$X_1 \perp\!\!\!\perp X_6 \mid X_2$

$X_2 \perp\!\!\!\perp X_4 \mid X_1$
$X_2 \perp\!\!\!\perp X_5 \mid X_3, X_4$

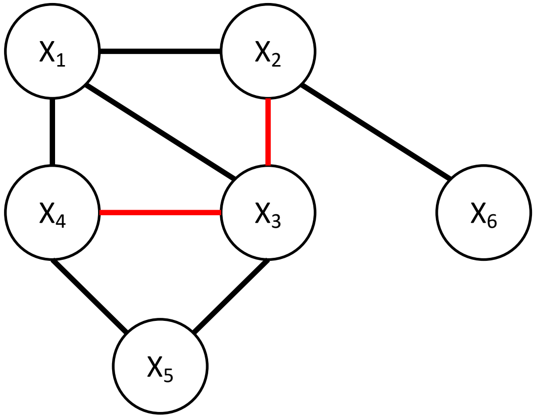
$X_3 \perp\!\!\!\perp X_6 \mid X_2$

$X_4 \perp\!\!\!\perp X_6 \mid X_1$ or $X_4 \perp\!\!\!\perp X_6 \mid X_2$

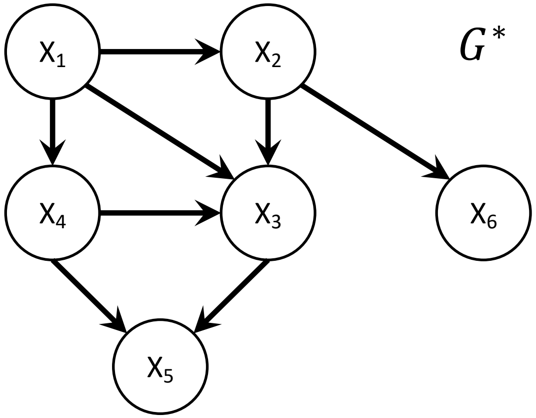
$X_5 \perp\!\!\!\perp X_6 \mid X_2$

Look at all triples $A \sim B \sim C$ and $A \nsim C$

If $C \notin sepset(A, B)$, then $A \rightarrow B \leftarrow C$



Example: PC algorithm



2. Identify v-structures

$X_1 \perp\!\!\!\perp X_5 \mid X_3, X_4$
$X_1 \perp\!\!\!\perp X_6 \mid X_2$

$X_2 \perp\!\!\!\perp X_4 \mid X_1$
$X_2 \perp\!\!\!\perp X_5 \mid X_3, X_4$

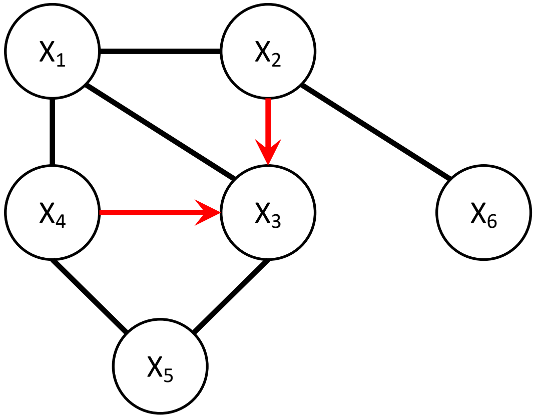
$X_3 \perp\!\!\!\perp X_6 \mid X_2$

$X_4 \perp\!\!\!\perp X_6 \mid X_1$ or $X_4 \perp\!\!\!\perp X_6 \mid X_2$

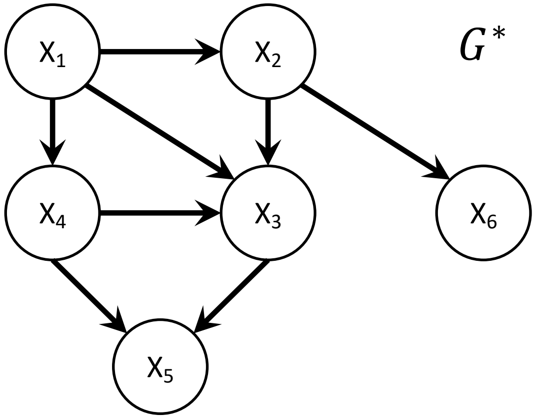
$X_5 \perp\!\!\!\perp X_6 \mid X_2$

Look at all triples $A \sim B \sim C$ and $A \nsim C$

If $C \notin sepset(A, B)$, then $A \rightarrow B \leftarrow C$



Example: PC algorithm



2. Identify v-structures

$X_1 \perp\!\!\!\perp X_5 \mid X_3, X_4$
$X_1 \perp\!\!\!\perp X_6 \mid X_2$

$X_2 \perp\!\!\!\perp X_4 \mid X_1$
$X_2 \perp\!\!\!\perp X_5 \mid X_3, X_4$

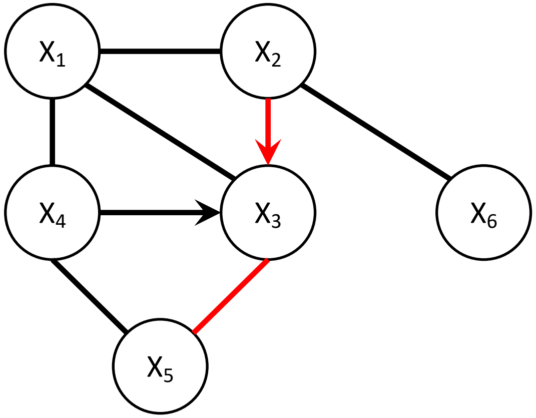
$X_3 \perp\!\!\!\perp X_6 \mid X_2$

$X_4 \perp\!\!\!\perp X_6 \mid X_1$ or $X_4 \perp\!\!\!\perp X_6 \mid X_2$

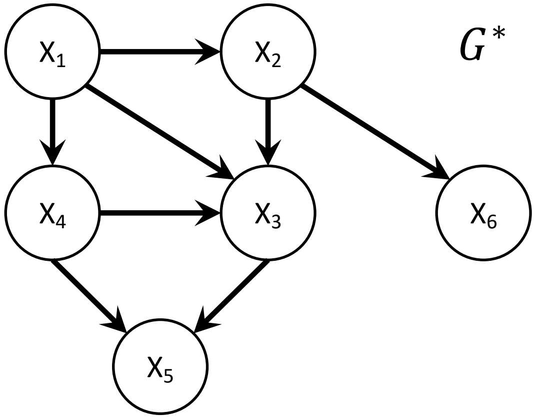
$X_5 \perp\!\!\!\perp X_6 \mid X_2$

Look at all triples $A \sim B \sim C$ and $A \nsim C$

If $C \notin sepset(A, B)$, then $A \rightarrow B \leftarrow C$



Example: PC algorithm



2. Identify v-structures

$X_1 \perp\!\!\!\perp X_5 \mid X_3, X_4$
$X_1 \perp\!\!\!\perp X_6 \mid X_2$

$X_2 \perp\!\!\!\perp X_4 \mid X_1$
$X_2 \perp\!\!\!\perp X_5 \mid X_3, X_4$

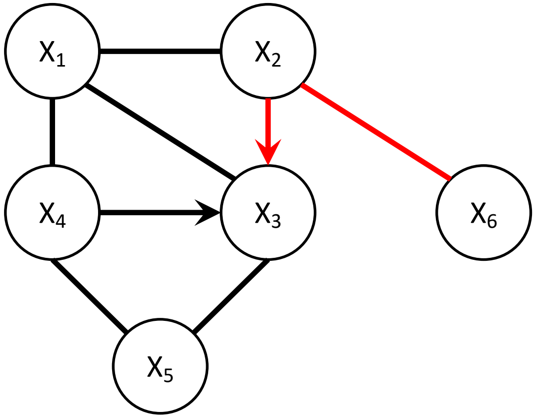
$X_3 \perp\!\!\!\perp X_6 \mid X_2$

$X_4 \perp\!\!\!\perp X_6 \mid X_1$ or $X_4 \perp\!\!\!\perp X_6 \mid X_2$

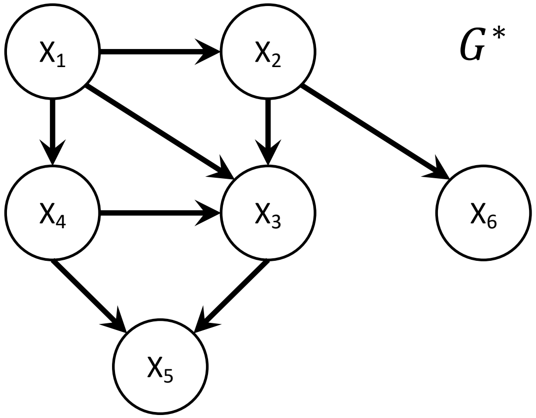
$X_5 \perp\!\!\!\perp X_6 \mid X_2$

Look at all triples $A \sim B \sim C$ and $A \nsim C$

If $C \notin sepset(A, B)$, then $A \rightarrow B \leftarrow C$



Example: PC algorithm



2. Identify v-structures

$X_1 \perp\!\!\!\perp X_5 \mid X_3, X_4$
$X_1 \perp\!\!\!\perp X_6 \mid X_2$

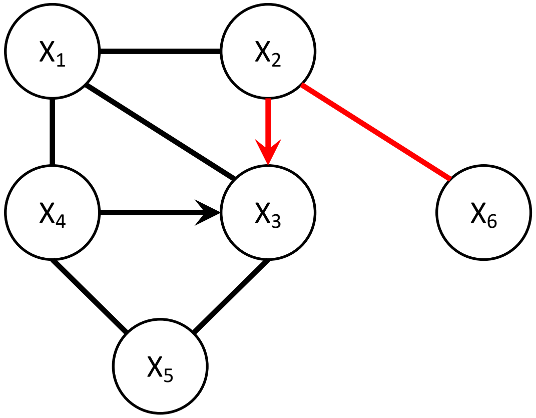
$X_2 \perp\!\!\!\perp X_4 \mid X_1$
$X_2 \perp\!\!\!\perp X_5 \mid X_3, X_4$

$X_3 \perp\!\!\!\perp X_6 \mid X_2$

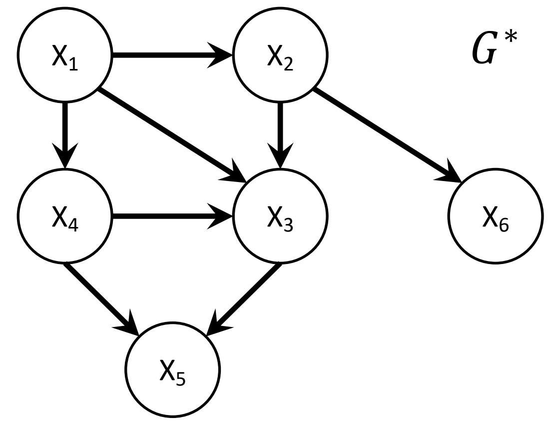
$X_4 \perp\!\!\!\perp X_6 \mid X_1$ or $X_4 \perp\!\!\!\perp X_6 \mid X_2$

$X_5 \perp\!\!\!\perp X_6 \mid X_2$

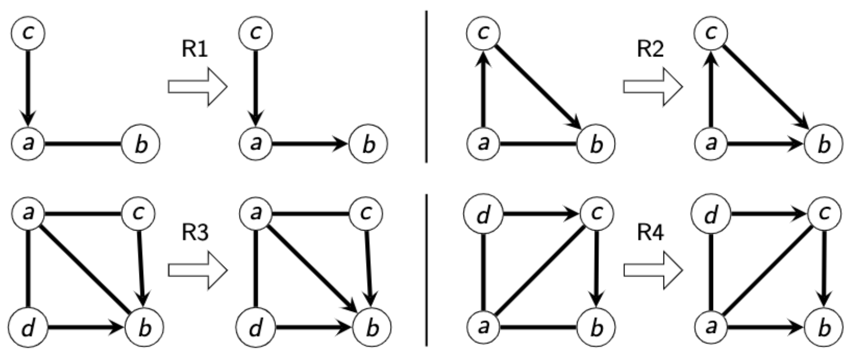
Look at all triples $A \sim B \sim C$ and $A \nsim C$
If $C \notin sepset(A, B)$, then $A \rightarrow B \leftarrow C$

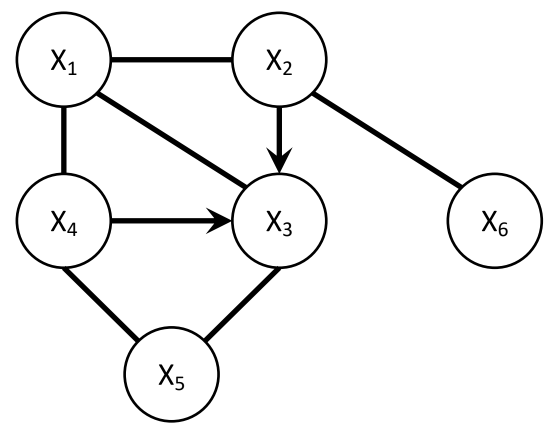


Example: PC algorithm

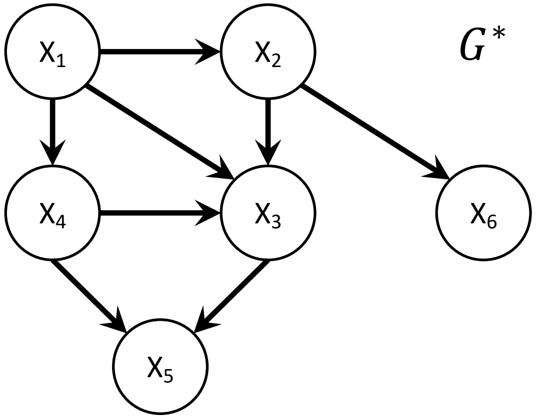


3. Orient using Meek rules

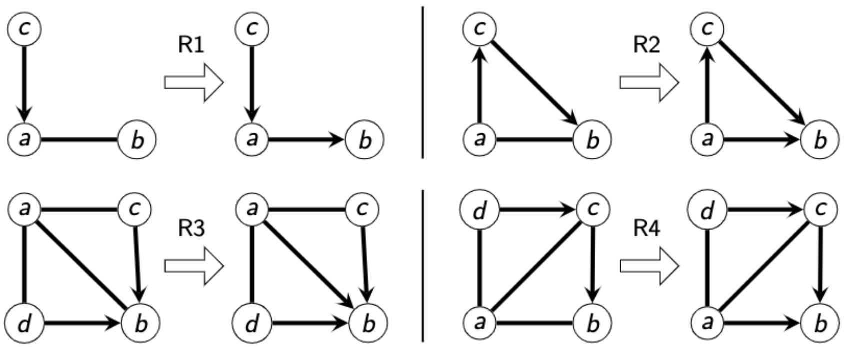




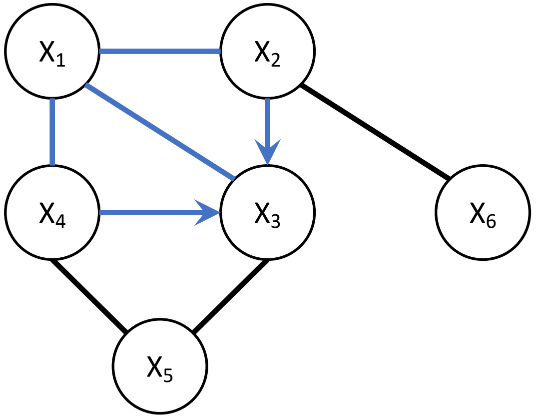
Example: PC algorithm



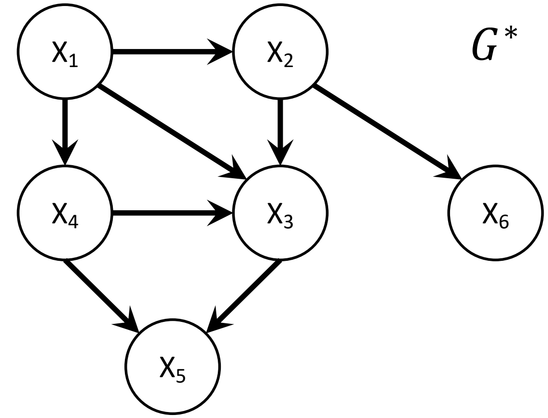
3. Orient using Meek rules



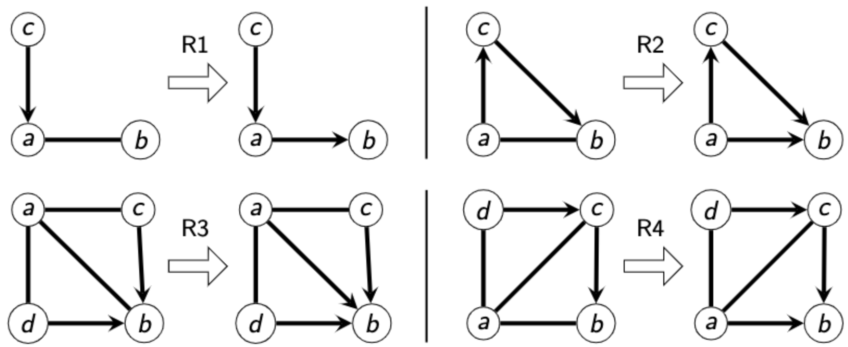
Meek R3

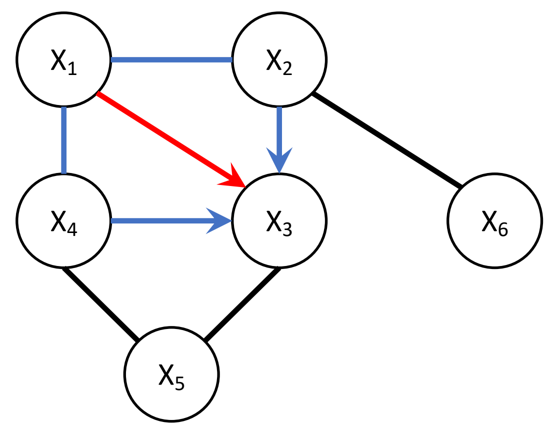


Example: PC algorithm



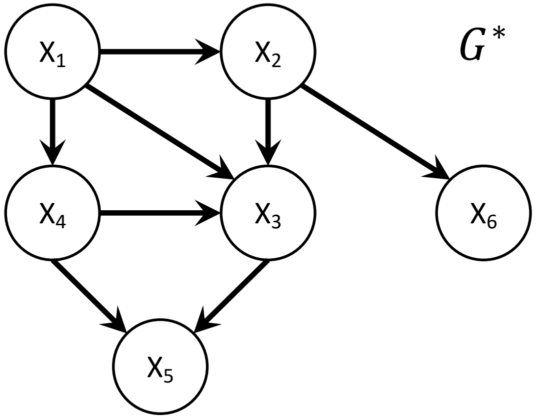
3. Orient using Meek rules



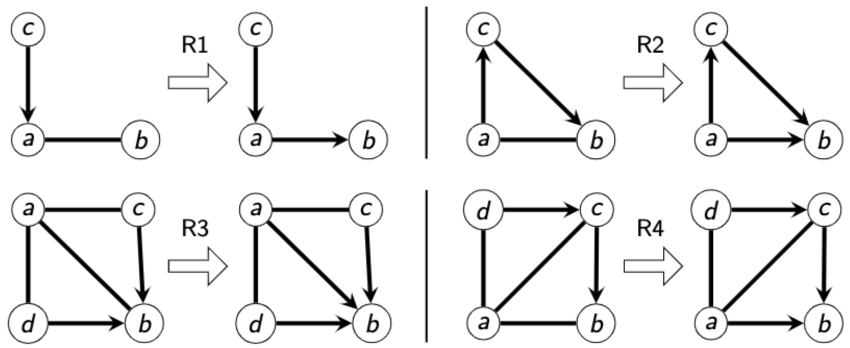


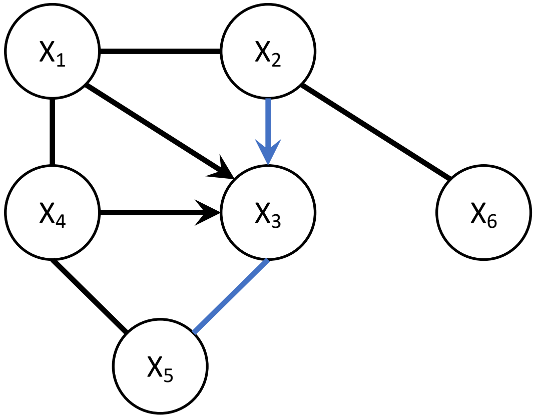
Meek R3

Example: PC algorithm



3. Orient using Meek rules

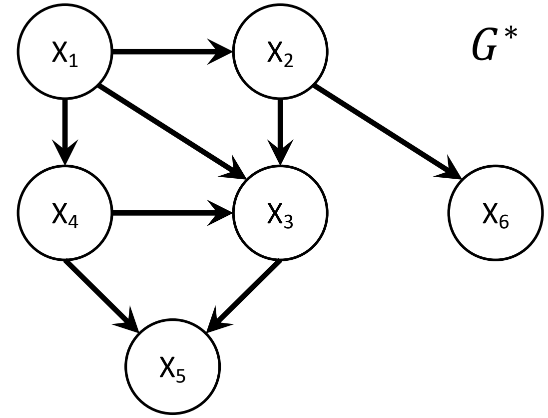




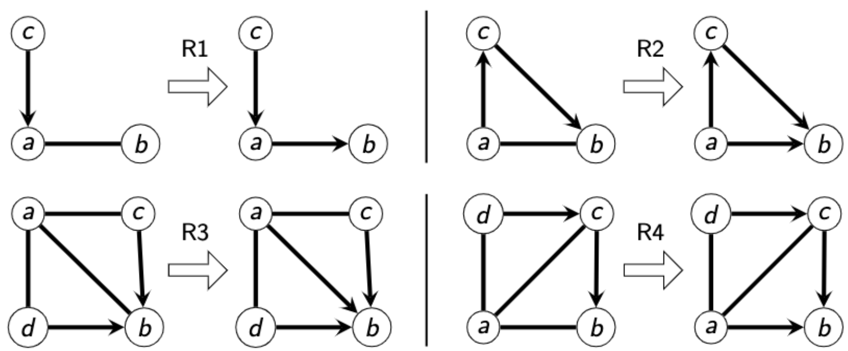
Meek R3

Meek R1

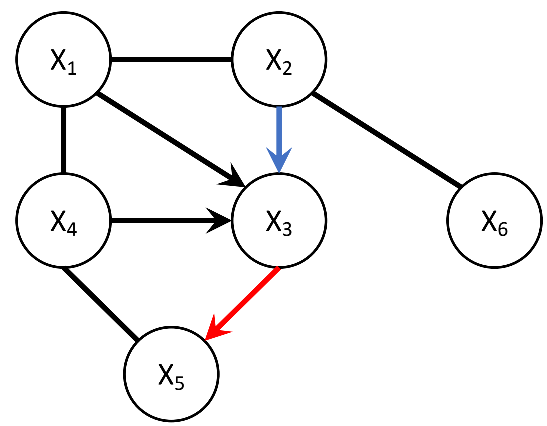
Example: PC algorithm



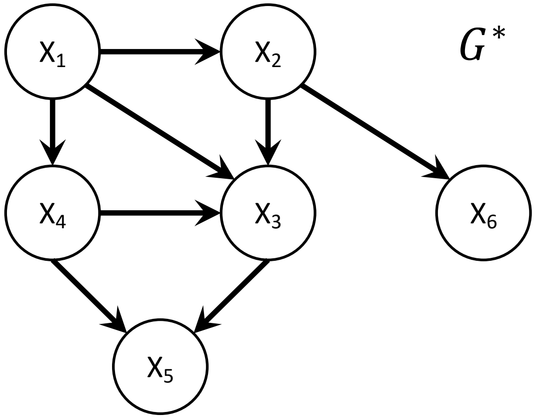
3. Orient using Meek rules



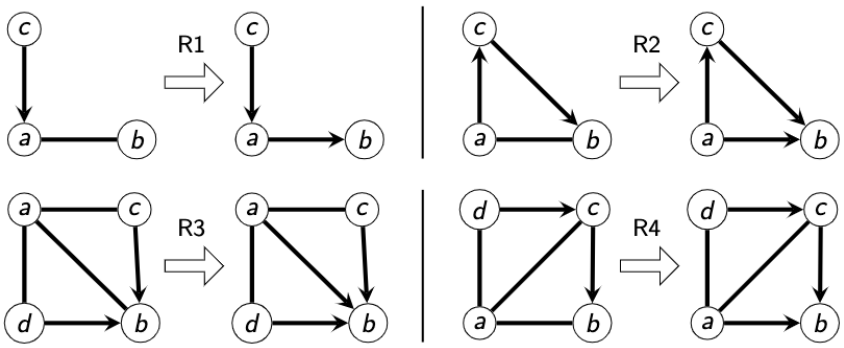
Meek R3
Meek R1

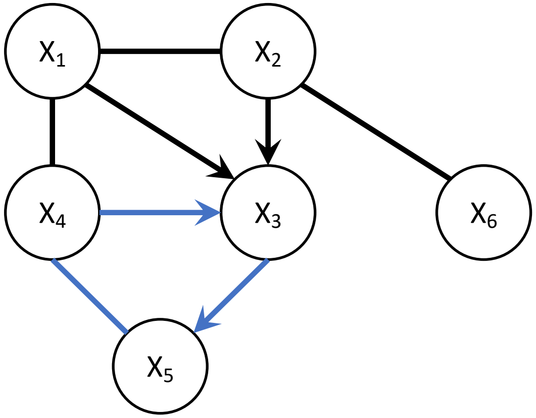


Example: PC algorithm



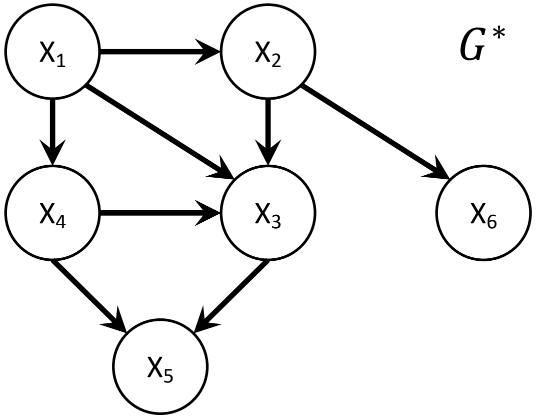
3. Orient using Meek rules



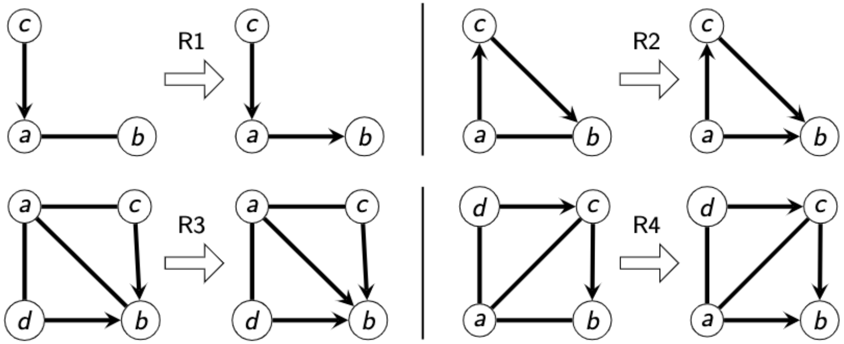


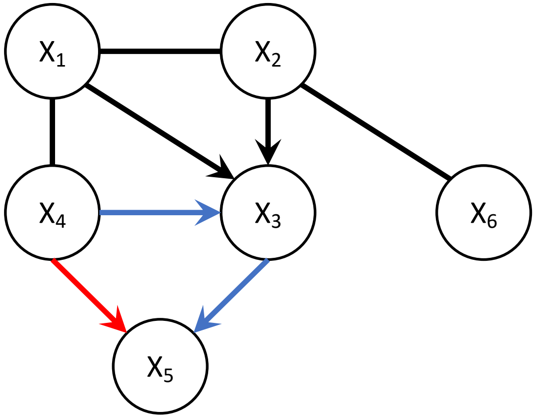
Meek R3
Meek R1
Meek R2

Example: PC algorithm



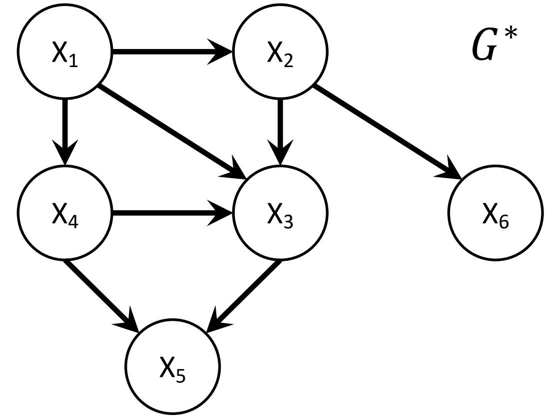
3. Orient using Meek rules



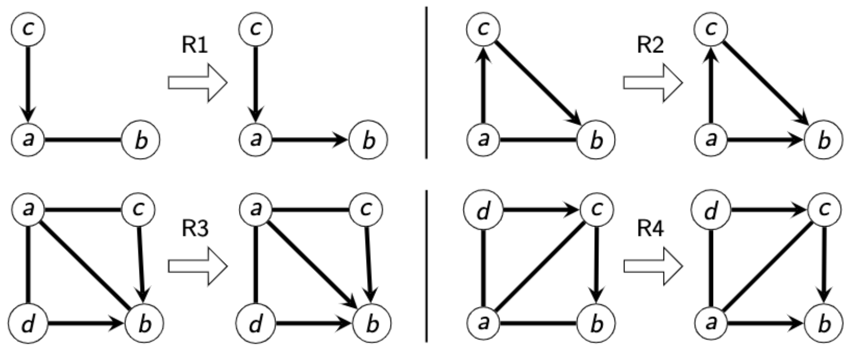


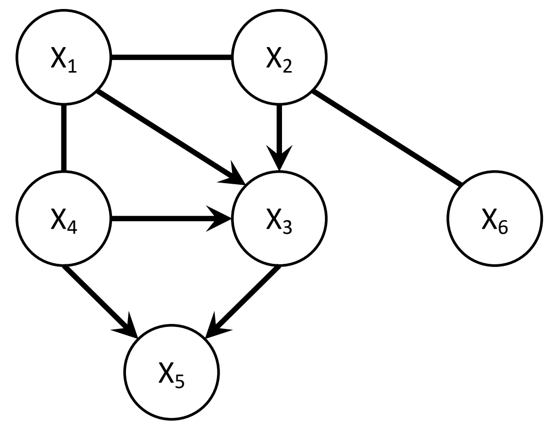
Meek R3
Meek R1
Meek R2

Example: PC algorithm



3. Orient using Meek rules





Meek R3
Meek R1
Meek R2

Output of PC: Essential graph of G^*