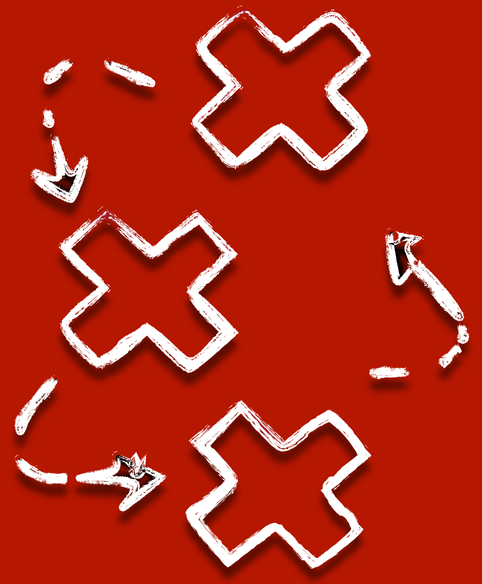


Causal Data Science

Lecture 10.1: Score-based causal discovery

Lecturer: Sara Magliacane

UvA - Spring 2024

Last class: PC algorithm - when does it fail?

- PC can use the results of any conditional independence test given α
  - Even non parametric tests like HSIC

Last class: PC algorithm - when does it fail?

- PC can use the results of any conditional independence test given α
  - Even non parametric tests like HSIC

- PC fails if the conditional independence tests give the wrong result
  - Too few samples, very weak dependence
  - Wrong parametric assumption (e.g. partial correlation on nonlinear data)
  - If there are unmeasured confounders or selection bias **(causal sufficiency)**

Last class: PC algorithm - when does it fail?

- PC can use the results of any conditional independence test given α
  - Even non parametric tests like HSIC

- PC fails if the conditional independence tests give the wrong result
  - Too few samples, very weak dependence
  - Wrong parametric assumption (e.g. partial correlation on nonlinear data)
  - If there are unmeasured confounders or selection bias **(causal sufficiency)**

- For unmeasured confounders, we can use more advanced constraint-based algorithms like Fast Causal Inference (FCI) Chapter 6 in [SGS book]

Last class: PC algorithm - when does it fail?

- PC can use the results of any conditional independence test given α
  - Even non parametric tests like HSIC

- PC fails if the conditional independence tests give the wrong result
  - Too few samples, very weak dependence
  - Wrong parametric assumption (e.g. partial correlation on nonlinear data)
  - If there are unmeasured confounders or selection bias **(causal sufficiency)**

- For unmeasured confounders, we can use more advanced constraint-based algorithms like Fast Causal Inference (FCI) Chapter 6 in [SGS book]

Causal discovery simplified overview

Constraint-based causal discovery

- Conditional independence tests
- Observational data
- Output: MEC
- SGS, PC, FCI

Score-based causal discovery

- Penalised likelihood
- Observational data
- Output: MEC
- GES, MMHC

Restricted models

- Nonlinear additive noise, Linear Non-Gaussianity
- Observational data
- Output: DAG
- RESIT, LINGAM

Interventional causal discovery / causal invariance

- Observational and Interventional data
- Output: parents of Y, I-MEC
- ICP, GIES, JCI

Causal discovery simplified overview

Constraint-based causal discovery

- Conditional independence tests
- Observational data
- Output: MEC
- SGS, PC, FCI

Score-based discovery

- Penalised
- Observational data
- Output: MI
- GES, MMH

causal
usal
al and
al data
ents of Y,
CI

Both constraint-based and score-based methods find a MEC, but identification strategies need a DAG

We can either use background knowledge to orient the remaining edges

Or we can use advanced methods like IDA (Maathuis et al 2009) that combine the identification of each possible DAG in MEC into bounds

Score-based causal discovery

- **Constraint-based causal discovery:** test conditional independences in the data and use them to constrain the possible causal graphs **(up to MEC)**

Score-based causal discovery

- **Constraint-based causal discovery:** test conditional independences in the data and use them to constrain the possible causal graphs **(up to MEC)**

- **Score-based causal discovery:** find the graph that maximises a **score** $S(G, D)$ (fit of graph G on data D)

$$\hat{G} = \text{argmax}_{G \in \mathcal{G}} S(G, D)$$

Model selection

Score-based causal discovery

- **Constraint-based causal discovery:** test conditional independences in the data and use them to constrain the possible causal graphs **(up to MEC)**

- **Score-based causal discovery:** find the graph that maximises a **score** $S(G, D)$ (fit of graph G on data D)

  - This is also called **Bayesian network structure learning**

Score-based causal discovery

- **Constraint-based causal discovery:** test conditional independences in the data and use them to constrain the possible causal graphs **(up to MEC)**

- **Score-based causal discovery:** find the graph that maximises a **score** $S(G, D)$ (fit of graph G on data D)

  - This is also called **Bayesian network structure learning**

  - We will use some specific properties for the score, e.g. **score equivalence:**
    - All DAGs in a MEC have the same score

Score-based causal discovery

- **Constraint-based causal discovery:** test conditional independences in the data and use them to constrain the possible causal graphs **(up to MEC)**

- **Score-based causal discovery:** find $S(G, D)$ (fit of graph G on data D)

We usually need to know the parametric model to compute the score - often based on the likelihood

  - This is also called **Bayesian network structure learning**

  - We will use some specific properties for the score, e.g. **score equivalence:**
    - All DAGs in a MEC have the same score

Score-based causal discovery

- For score-based discovery we assume we know the parametric form of SCM
  - **Linear functions with parameters $\mathbf{B}$ + Gaussian noises with variance Σ**

$X_1 = \varepsilon_1$

Score-based causal discovery

- For score-based discovery we assume we know the parametric form of SCM
  - **Linear functions with parameters $\mathbf{B}$ + Gaussian noises with variance Σ**

$X_1 = \varepsilon_1$

$X_2 = B_{21} X_1 + \varepsilon_2$

B_{21} is the coefficient of X_1 for the equation for X_2

Score-based causal discovery

- For score-based discovery we assume we know the parametric form of SCM
  - **Linear functions with parameters $\mathbf{B}$ + Gaussian noises with variance Σ**

$X_1 = \varepsilon_1$

$X_2 = B_{21} X_1 + \varepsilon_2$

$X_3 = B_{31} X_1 + B_{32} X_2 + \varepsilon_3$

B_{31} is the coefficient of X_1 for the equation for X_3

B_{32} is the coefficient of X_2 for the equation for X_3

Score-based causal discovery

- For score-based discovery we assume we know the parametric form of SCM
  - **Linear functions with parameters $\mathbf{B}$ + Gaussian noises with variance Σ**

$$\begin{cases} X_1 = \varepsilon_1 \\ X_2 = B_{21} X_1 + \varepsilon_2 \\ X_3 = B_{31} X_1 + B_{32} X_2 + \varepsilon_3 \end{cases}$$

$$\varepsilon_1 \sim N(0, \sigma_1^2) \; ; \varepsilon_2 \sim N(0, \sigma_2^2) \; ; \varepsilon_3 \sim N(0, \sigma_3^2)$$

Score-based causal discovery

- For score-based discovery we assume we know the parametric form of SCM
  - **Linear functions with parameters $\mathbf{B}$ + Gaussian noises with variance Σ**

$$\begin{cases} X_1 = \varepsilon_1 \\ X_2 = B_{21} X_1 + \varepsilon_2 \\ X_3 = B_{31} X_1 + B_{32} X_2 + \varepsilon_3 \end{cases}$$

Let's write it in a matrix form

$$\varepsilon_1 \sim N(0, \sigma_1^2) \; ; \varepsilon_2 \sim N(0, \sigma_2^2) \; ; \varepsilon_3 \sim N(0, \sigma_3^2)$$

$$B \in \mathbb{R}^{p \times p} \; : \quad B_{ij} = 0 \Rightarrow j \not\to i$$

B_{ij} is the coefficient of X_j for the equation for X_i

Score-based causal discovery

- For score-based discovery we assume we know the parametric form of SCM
  - **Linear functions with parameters $\mathbf{B}$ + Gaussian noises with variance Σ**

$$\begin{cases} X_1 = \varepsilon_1 \\ X_2 = B_{21} X_1 + \varepsilon_2 \\ X_3 = B_{31} X_1 + B_{32} X_2 + \varepsilon_3 \end{cases}$$

Let's write it in a matrix form

$$X = B \cdot X + \varepsilon$$

$$B \in \mathbb{R}^{p \times p} : \quad B_{ij} = 0 \Rightarrow j \not\to i$$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ B_{21} & 0 & 0 \\ B_{31} & B_{32} & 0 \end{bmatrix}$$

Score-based causal discovery

- For score-based discovery we assume we know the parametric form of SCM
  - **Linear functions with parameters $\mathbf{B}$ + Gaussian noises with variance Σ**

$$\begin{cases} X_1 = \varepsilon_1 \\ X_2 = B_{21} X_1 + \varepsilon_2 \\ X_3 = B_{31} X_1 + B_{32} X_2 + \varepsilon_3 \end{cases}$$

$B \in \mathbb{R}^{p \times p}$: $B_{ij} = 0 \Rightarrow j \not\to i$

Let's write it in a matrix form

$$X = B \cdot X + \varepsilon$$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ B_{21} & 0 & 0 \\ B_{31} & B_{32} & 0 \end{bmatrix}$$

$$X_2 = B_{21} X_1 + 0 \cdot X_2 + 0 \cdot X_3 + \dots$$

Score-based causal discovery

- For score-based discovery we assume we know the parametric form of SCM
  - **Linear functions with parameters $\mathbf{B}$ + Gaussian noises with variance Σ**

$$\begin{cases} X_1 = \varepsilon_1 \\ X_2 = B_{21} X_1 + \varepsilon_2 \\ X_3 = B_{31} X_1 + B_{32} X_2 + \varepsilon_3 \end{cases}$$

$$\varepsilon_1 \sim N(0, \sigma_1^2) \; ; \varepsilon_2 \sim N(0, \sigma_2^2) \; ; \varepsilon_3 \sim N(0, \sigma_3^2)$$

$$X = B \cdot X + \varepsilon$$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ B_{21} & 0 & 0 \\ B_{31} & B_{32} & 0 \end{bmatrix}$$

$$\varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} \sim N(0, \Sigma) \qquad \Sigma = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix}$$

Score-based causal discovery

- For score-based discovery we assume we know the parametric form of SCM
  - **Linear functions with parameters $\mathbf{B}$ + Gaussian noises with variance Σ**

$$\begin{cases} X_1 = \varepsilon_1 \\ X_2 = B_{21} X_1 + \varepsilon_2 \\ X_3 = B_{31} X_1 + B_{32} X_2 + \varepsilon_3 \end{cases}$$

$$X = B \cdot X + \varepsilon$$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ B_{21} & 0 & 0 \\ B_{31} & B_{32} & 0 \end{bmatrix}$$

$$\varepsilon_1 \sim N(0, \sigma_1^2) \; ; \varepsilon_2 \sim N(0, \sigma_2^2) \; ; \varepsilon_3 \sim N(0, \sigma_3^2)$$

$$\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} \sim N(0, \Sigma) \qquad \Sigma = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix}$$

We can define $\| B \|_0$ (Frobenius norm) as the number of non-zero elements in B

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of n **i.i.d.** samples, where each $x_{\mathbf{V}}^i = (x_1^i, \ldots, x_p^i)$ is a realisation for a unit i for all random variables $X_{\mathbf{V}}$

i.i.d. = independent and identically distributed

In other words, we sampled them independently one from each other and they come from the same P

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of n **i.i.d.** samples, where each $x_{\mathbf{V}}^i = (x_1^i, \ldots, x_p^i)$ is a realisation for a unit i for all random variables $X_{\mathbf{V}}$

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of n **i.i.d.** samples, where each $x_{\mathbf{V}}^i = (x_1^i, \ldots, x_p^i)$ is a realisation for a unit i for all random variables $X_{\mathbf{V}}$

- Attempt 1: Bayes theorem

$$P(G, \theta \mid D) = \frac{P(D \mid G, \theta) P(G, \theta)}{P(D)}$$

G graph structure, θ parameters, e.g. (B, ε) in linear SCM

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of n **i.i.d.** samples, where each $x_{\mathbf{V}}^i = (x_1^i, \ldots, x_p^i)$ is a realisation for a unit i for all random variables $X_{\mathbf{V}}$

- Attempt 1: Bayes theorem

$$P(G, \theta \mid D) = \frac{P(D \mid G, \theta) P(G, \theta)}{P(D)}$$

Prior (assumed to be uniform)

Normalisation constant

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of n **i.i.d.** samples, where each $x_{\mathbf{V}}^i = (x_1^i, \ldots, x_p^i)$ is a realisation for a unit i for all random variables $X_{\mathbf{V}}$

- Attempt 1: Bayes theorem

Likelihood, depends on G and θ

$$P(G, \theta \mid D) = \frac{P(D \mid G, \theta) P(G, \theta)}{P(D)}$$

- Find G, θ that maximise $P(D \mid G, \theta)$

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of n **i.i.d.** samples, where each $x_{\mathbf{V}}^i = (x_1^i, \ldots, x_p^i)$ is a realisation for a unit i for all random variables $X_{\mathbf{V}}$

- Attempt 1: Find G, θ that maximise $P(D \mid G, \theta)$

$$P(D \mid G, \theta) = \prod_{i=1}^{n} P(x_1^i, \ldots, x_p^i \mid G, \theta)$$

i.i.d. n samples

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of n **i.i.d.** samples, where each $x_{\mathbf{V}}^i = (x_1^i, \ldots, x_p^i)$ is a realisation for a unit i for all random variables $X_{\mathbf{V}}$

- Attempt 1: Find G, θ that maximise $P(D \mid G, \theta)$

$$P(D \mid G, \theta) = \prod_{i=1}^{n} P(x_1^i, \ldots, x_p^i \mid G, \theta) = \prod_{i=1}^{n} \prod_{j=1}^{p} P(x_j^i \mid x_{\mathrm{Pa}(j)_G}^i, \theta)$$

i.i.d. n samples

BN factorization

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of n **i.i.d.** samples, where each $x_{\mathbf{V}}^i = (x_1^i, \ldots, x_p^i)$ is a realisation for a unit i for all random variables $X_{\mathbf{V}}$

- Attempt 1: Find G, θ that maximise $P(D \mid G, \theta)$

$$P(D \mid G, \theta) = \prod_{i=1}^{n} P(x_1^i, \ldots, x_p^i \mid G, \theta) = \prod_{i=1}^{n} \prod_{j=1}^{p} P(x_j^i \mid x_{\mathrm{Pa}(j)_G}^i, \theta)$$

$$logP(D \mid G, \theta) = \sum_{i}^{n} \sum_{j}^{p} \log P(x_j^i \mid x_{\mathrm{Pa}(j)_G}^i, \theta)$$

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of n **i.i.d.** samples, where each $x_{\mathbf{V}}^i = (x_1^i, \ldots, x_p^i)$ is a realisation for a unit i for all random variables $X_{\mathbf{V}}$

- Attempt 1: Find G, θ that maximise $P(D \mid G, \theta)$

$$log P(D \mid G, \theta) = \sum_i^n \sum_j^p \log P(x_j^i \mid x^i_{\mathrm{Pa}(j)_G}, \theta)$$

Maximum likelihood estimation

$$(G^{MLE}, \theta^{MLE}) = \mathrm{argmax}_{G,\theta} \log P(D \mid G, \theta)$$

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of n **i.i.d.** samples, where each $x_{\mathbf{V}}^i = (x_1^i, \ldots, x_p^i)$ is a realisation for a unit i for all random variables $X_{\mathbf{V}}$

- Attempt 1: Find G, θ that maximise $P(D \mid G, \theta)$

$$log P(D \mid G, \theta) = \sum_i^n \sum_j^p \log P(x_j^i \mid x_{\mathrm{Pa}(j)_G}^i, \theta)$$

Maximum likelihood estimation

$$(G^{MLE}, \theta^{MLE}) = \mathrm{argmax}_{G,\theta} \log P(D \mid G, \theta)$$

- **Issue**: this will result in fully connected DAGs (*overfitting*)

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of n **i.i.d.** samples, where each $x_{\mathbf{V}}^i = (x_1^i, \ldots, x_p^i)$ is a realisation for a unit i for all random variables $X_{\mathbf{V}}$

- **Attempt 2:** G, θ maximise $P(D \mid G, \theta)$ + **encourage sparsity with penalty**

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of n **i.i.d.** samples, where each $x_{\mathbf{V}}^i = (x_1^i, \ldots, x_p^i)$ is a realisation for a unit i for all random variables $X_{\mathbf{V}}$

- **Attempt 2:** G, θ maximise $P(D \mid G, \theta)$ + **encourage sparsity with penalty**

- Typically we use **BIC (Bayesian information criterion)**

$$BIC(D, G) := 2 \cdot \log\, p(D \mid G, \theta_G^{MLE}) - \log(n) \cdot \#\text{parameters}$$

$$G^{BIC} = \text{argmax}_G\, \text{BIC(D, G)}$$

Number of edges in G + const

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of i.i.d. samples
- For each G, we can compute $BIC(D, G)$ as:

1. Given G we find

$$\theta_G^{MLE} = \text{argmax}_{\theta_G} \log P(D \mid G, \theta_G) = \sum_i^n \sum_j^p \log P(x_j^i \mid x_{\text{Pa}(j)_G}^i, \theta_G)$$

Bayesian Information Criterion (BIC)

- We have a dataset $D = \{x_{\mathbf{V}}^1, x_{\mathbf{V}}^2, \ldots, x_{\mathbf{V}}^n\}$ of i.i.d. samples
- For each G, we can compute $BIC(D, G)$ as:

1. Given G we find

$$\theta_G^{MLE} = \text{argmax}_{\theta_G} \log P(D \mid G, \theta_G) = \sum_i^n \sum_j^p \log P(x_j^i \mid x_{\text{Pa}(j)_G}^i, \theta_G)$$

2. Compute $BIC(D, G) = 2\log P(D \mid G, \theta_G^{\text{MLE}}) - \log(n) \cdot \#\text{edges in G}$

BIC in the linear Gaussian case

- Linear Gaussian case:

$$P(X_j \mid \mathbf{X}_{\mathrm{Pa}_G(j)} = \mathbf{x}_{\mathrm{Pa}_G(j)}, \theta_G) = N(\sum_{k \in \mathrm{Pa}_G(j)} B_{jk} x_k, \sigma_j^2)$$

1. For each possible G:
   1. We can learn θ_G^{MLE} with linear regression of each variable X_j on its parents
   2. We can compute $BIC(D, G) = 2\log P(D \mid G, \theta_G^{\mathrm{MLE}}) - \log n(p + \| B \|_0)$

Bayesian Information Criterion (BIC) - properties

- **Score equivalence:** all DAGs in a MEC get the same score

Bayesian Information Criterion (BIC) - properties

- **Score equivalence:** all DAGs in a MEC get the same score

- **Decomposable:** we can decompose the score as the sum of the contributions for each variable and its parents
  - If we want to compare two similar DAGs, we just need to check the variables with different edges, the rest stays the same

Bayesian Information Criterion (BIC) - properties

- **Score equivalence:** all DAGs in a MEC get the same score

- **Decomposable:** we can decompose the score as the sum of the contributions for each variable and its parents
  - If we want to compare two similar DAGs, we just need to check the variables with different edges, the rest stays the same

- **Local consistency :** adding edges that create **superfluous** d-connections (that are independences in the data) decreases score, instead adding an edge that adds a “correct” d-connection (also dependence in the data) increases it

Number of DAGs with n nodes ?

A003024 as a simple table

n	a(n)
0	1
1	1
2	3
3	25
4	543
5	29281
6	3781503
7	1138779265
8	783702329343
9	1213442454842881
10	4175098976430598143
11	316034593964189176074 25
12	52193965134382940502050 4063
13	18676600744432035186664816926721
14	14394281410443983349417907198395351 03

Number of DAGs grows super-exponentially!

https://oeis.org/A003024

Number of DAGs with n nodes ?

A003024 as a simple table

n	a(n)
0	1
1	1
2	3
3	25
4	543
5	29281
6	3781503
7	1138779265
8	783702329343
9	1213442454842881
10	4175098976430598143
11	31603459396418917607425
12	521939651343829405020504063
13	18676600744432035186664816926721
14	1439428141044398334941790719839535103

Number of DAGs grows super-exponentially!

$$G^{BIC} = \text{argmax}_G \text{ BIC}(\text{D}, \text{G})$$

We cannot do exhaustive search for BIC

https://oeis.org/A003024

Greedy algorithms (very informal)

- We are trying to maximise a score

```
While (true):
```

1. Start with a candidate C
2. Check similar solutions in the neighbourhood ("one step away") to see if there is any other candidate C' that is better than C.
3. Is there any?
   - No, return C
   - Yes, choose the best C' as the candidate, continue loop

Greedy search for graphs (informal)

- Given data D and score $S(D, G)$, iterate:
  - Candidate graph G with score S
  - **Get neighbouring graphs** $\mathcal{G}$
  - Find **best scoring neighbour** $G^* = \text{argmax}_{G' \in \mathcal{G}} S(D, G')$
  - If $S(D, G^*) > S(D, G)$
    - Take G^* as new candidate graph G and **iterate**
  - Otherwise return G

Greedy search for graphs (informal)

- Given data D and score $S(D, G)$, iterate:
  - Candidate graph G with score S
  - **Get neighbouring graphs $\mathcal{G}$** add/remove an edge
  - Find **best scoring neighbour** $G^* = \text{argmax}_{G' \in \mathcal{G}} S(D, G')$
  - If $S(D, G^*) > S(D, G)$
    - Take G^* as new candidate graph G and **iterate**
  - Otherwise return G

in general could find local optima

Greedy Equivalence Search (GES)

- Optimise **Bayesian Information Criterion (BIC)**
- Reduce search space by **searching over CPDAGs** instead of DAGs

Greedy Equivalence Search (GES)

- Optimise **Bayesian Information Criterion (BIC)**
- Reduce search space by **searching over CPDAGs** instead of DAGs
- BIC is score-equivalent, so DAGs in same Markov equivalence class (so represented by same CPDAG) have the same score *(so you can pick any)*
- It can be shown that searching over CPDAGs with local consistency allows us to find the **global optimum** *(in large sample limit)*

Naive Equivalence Search

1. Start with complete CPDAG
2. Remove edges one by one until local maxima in BIC

decomposability helps us to recompute only one factor

Computationally inefficient (especially sparse graphs)

Greedy Equivalence Search (GES)

1. Start with empty CPDAG
2. **Forward (phase 1):** Add edges one by one until local maxima in BIC
3. **Backward (phase 2):** Remove edges one by one until local maxima in BIC

We need to be careful when adding and removing edges **(valid moves)**, because we want to have a CPDAG at each step

Greedy Equivalence Search (GES)

1. Start with empty CPDAG
2. **Forward (phase 1):** Add edges one by one until local maxima in BIC
3. **Backward (phase 2):** Remove edges one by one until local maxima in BIC

Phase 1 neighbours ε^+:

Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$

Greedy Equivalence Search (GES)

1. Start with empty CPDAG
2. **Forward (phase 1):** Add edges one by one until local maxima in BIC
3. **Backward (phase 2):** Remove edges one by one until local maxima in BIC

ε' is only "one-edge away" from ε

Phase 1 neighbours ε^+:

Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$

Phase 2 neighbours ε^-: same with removing an edge

Greedy Equivalence Search (GES) example

Phase 1 neighbours ε^+:

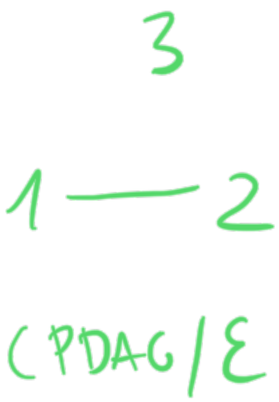
Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$

1. **Step 1:** Enumerate all DAGs $G \in \varepsilon$ in the initial MEC ε
2. **Step 2:** for each G, enumerate all DAGs G' that can be created by adding a single edge to G
3. **Step 3:** For each DAG G' add its MEC ε' to the list of neighbours ϵ^+
4. Return Phase 1 neighbours ε^+

Greedy Equivalence Search (GES) example

Phase 1 neighbours ε^+:

Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$

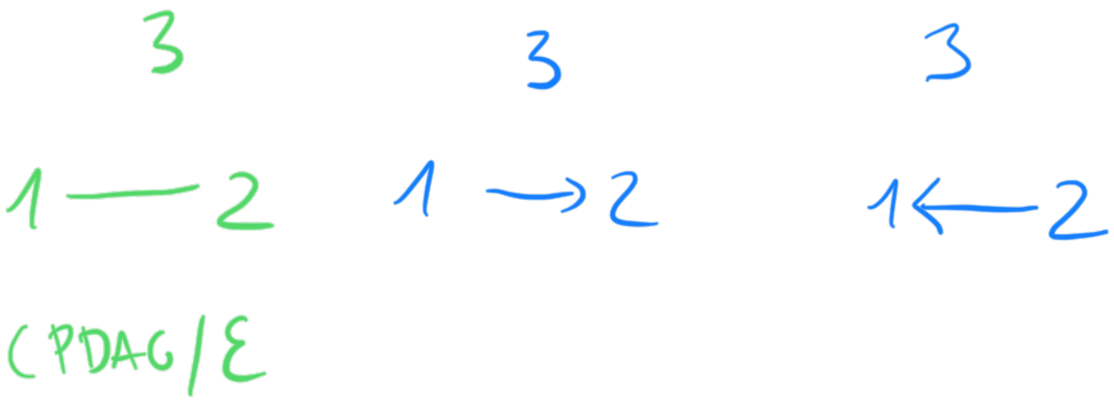


Step 1: Enumerate all DAGs $G \in \varepsilon$ in the initial MEC ε

Greedy Equivalence Search (GES) example

Phase 1 neighbours ε^+:

Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$

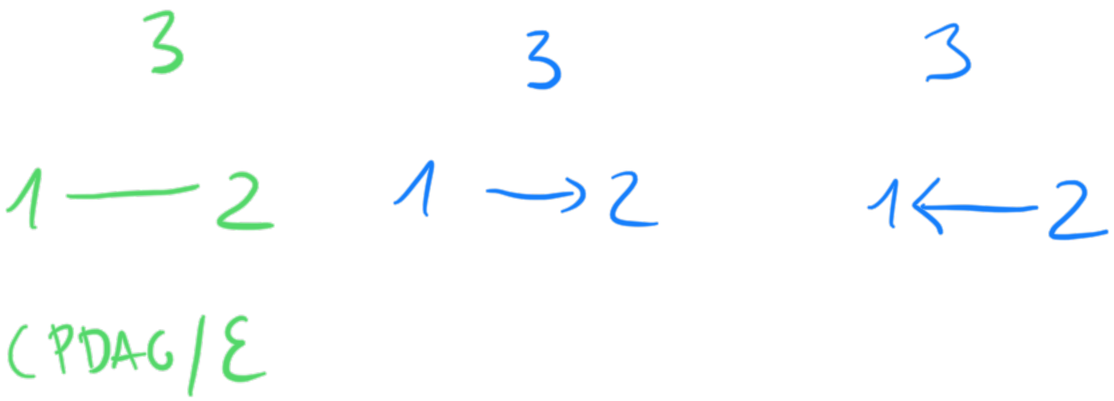


Step 1: Enumerate all DAGs $G \in \varepsilon$ in the initial MEC ε

Greedy Equivalence Search (GES) example

Phase 1 neighbours ε^+:

Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$



Add an edge between 3 and 1

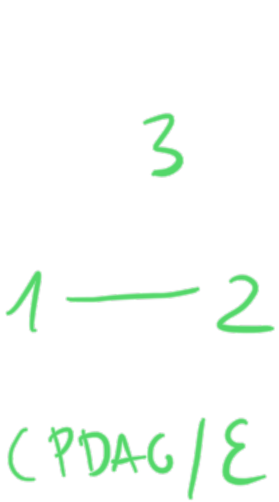
Add an edge between 3 and 2

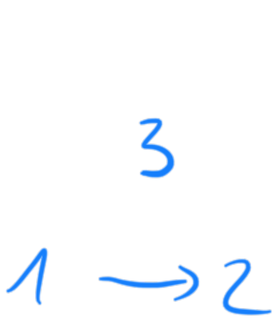
Step 2: for each G, enumerate all G' that can be created by adding a single edge to G

Greedy Equivalence Search (GES) example

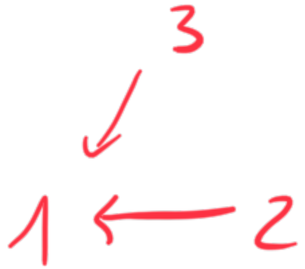
Phase 1 neighbours ε^+:

Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$





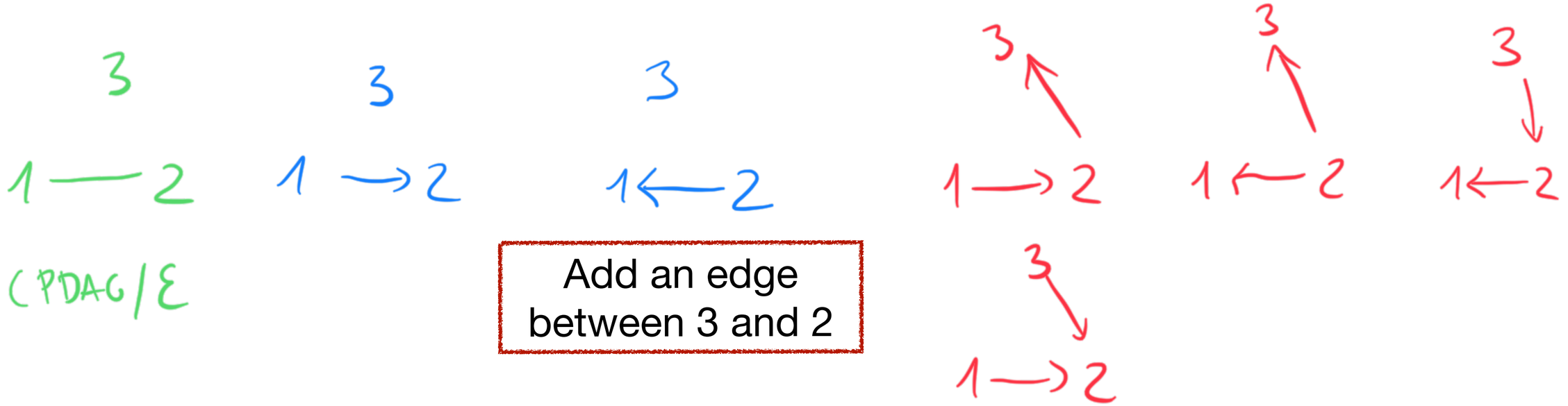
Add an edge between 3 and 1



Greedy Equivalence Search (GES) example

Phase 1 neighbours ε^+:

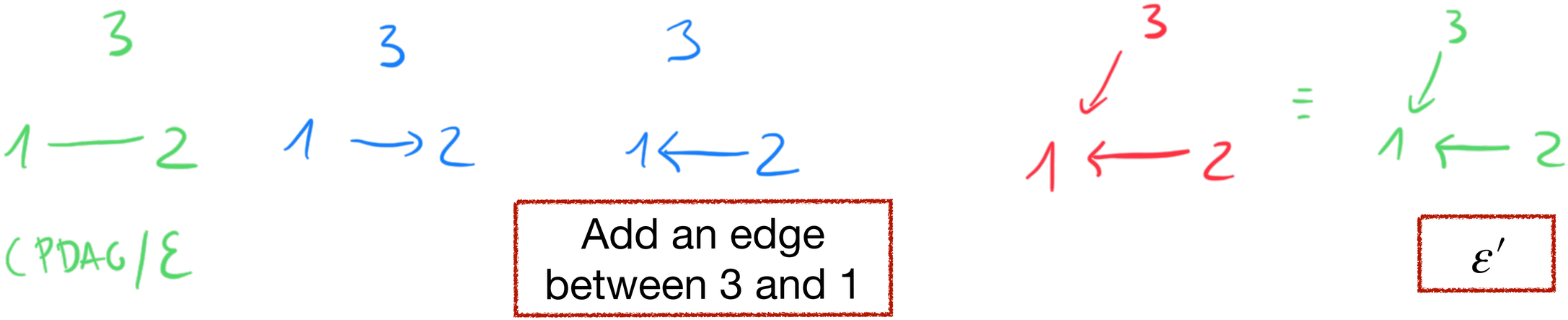
Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$



Greedy Equivalence Search (GES) example

Phase 1 neighbours ε^+:

Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$

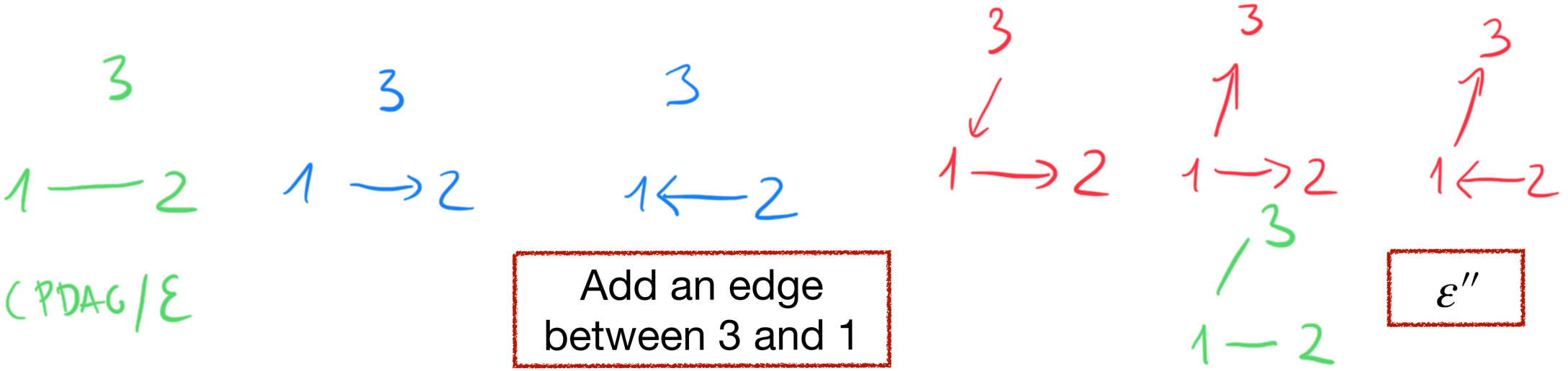


Step 3: For each DAG G' add its MEC ε' to the list of neighbours ϵ^+

Greedy Equivalence Search (GES) example

Phase 1 neighbours ε^+:

Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$

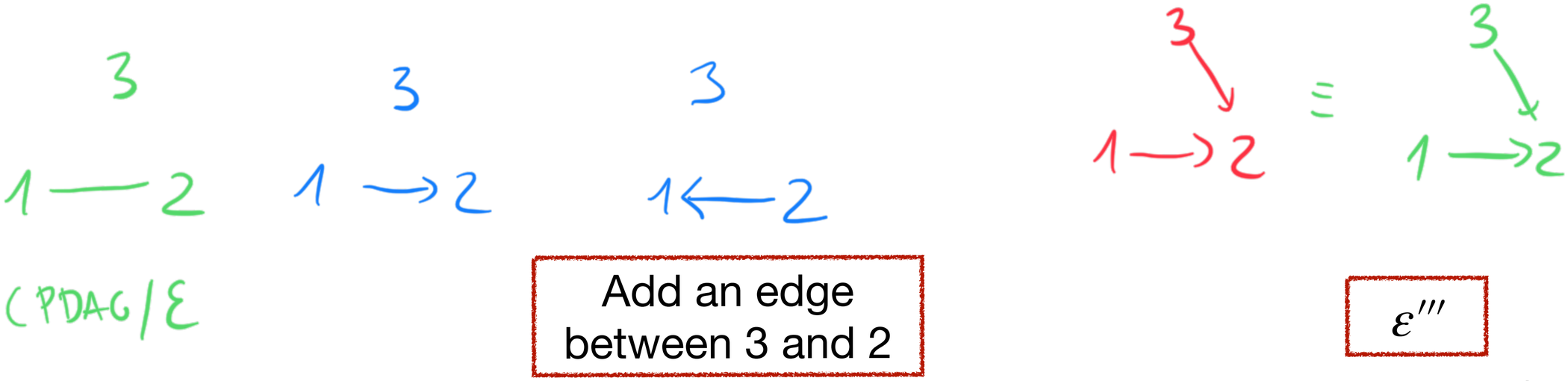


Step 3: For each DAG G' add its MEC ε' to the list of neighbours ϵ^+

Greedy Equivalence Search (GES) example

Phase 1 neighbours ε^+:

Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$

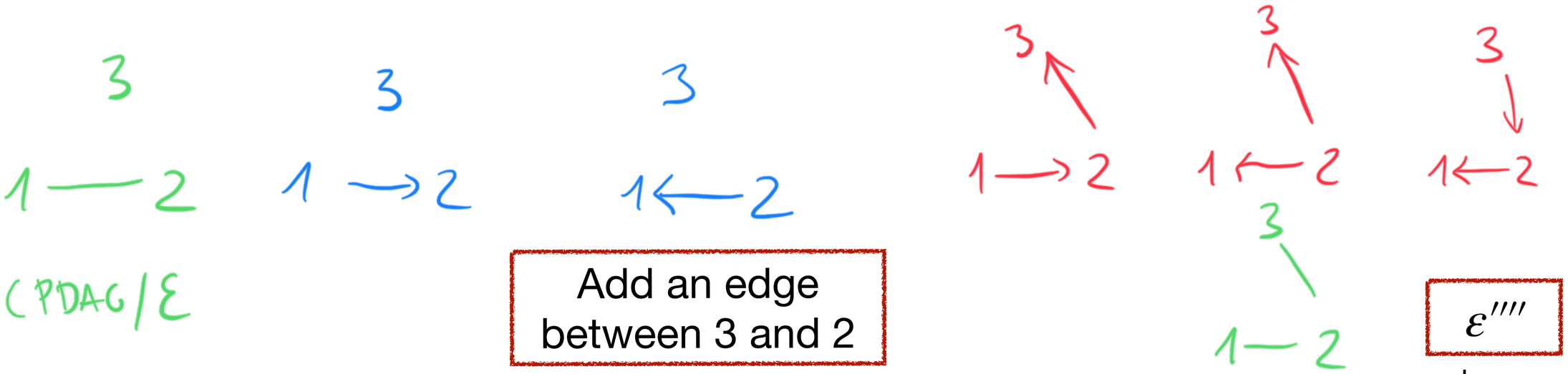


Step 3: For each DAG G' add its MEC ε' to the list of neighbours ϵ^+

Greedy Equivalence Search (GES) example

Phase 1 neighbours ε^+:

Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$

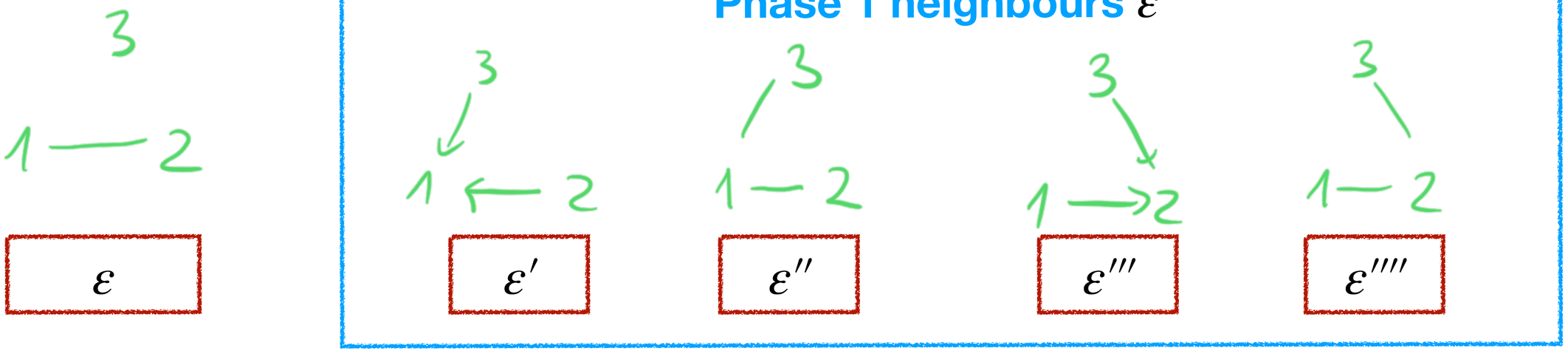


Step 3: For each DAG G' add its MEC ε' to the list of neighbours ϵ^+

Greedy Equivalence Search (GES) example

Phase 1 neighbours ε^+:

Given a a starting equivalence class ε, another class ε' is in the neighbours ε^+ if there exists a DAG $G \in \varepsilon$, such that adding an edge to G results in $G' \in \varepsilon'$



Greedy Equivalence Search - phase 1: example 2

$\mathcal{E}$: **Step 1:** Enumerate all DAGs $G \in \mathcal{E}$ in the initial MEC $\mathcal{E}$

$1 \rightarrow 2 \rightarrow 3$

Greedy Equivalence Search - phase 1: example 2

$\mathcal{E}$: **Step 1:** Enumerate all DAGs $G \in \mathcal{E}$ in the initial MEC $\mathcal{E}$

$1 \rightarrow 2 \rightarrow 3$ $1 - 2 - 3$ $1 \leftarrow 2 \rightarrow 3$ $1 \leftarrow 2 \leftarrow 3$

Greedy Equivalence Search - phase 1: example 2

$\mathcal{E}$:

$1 \rightarrow 2 \rightarrow 3$ $1 - 2 - 3$ $1 \leftarrow 2 \rightarrow 3$

$1 \leftarrow 2 \leftarrow 3$

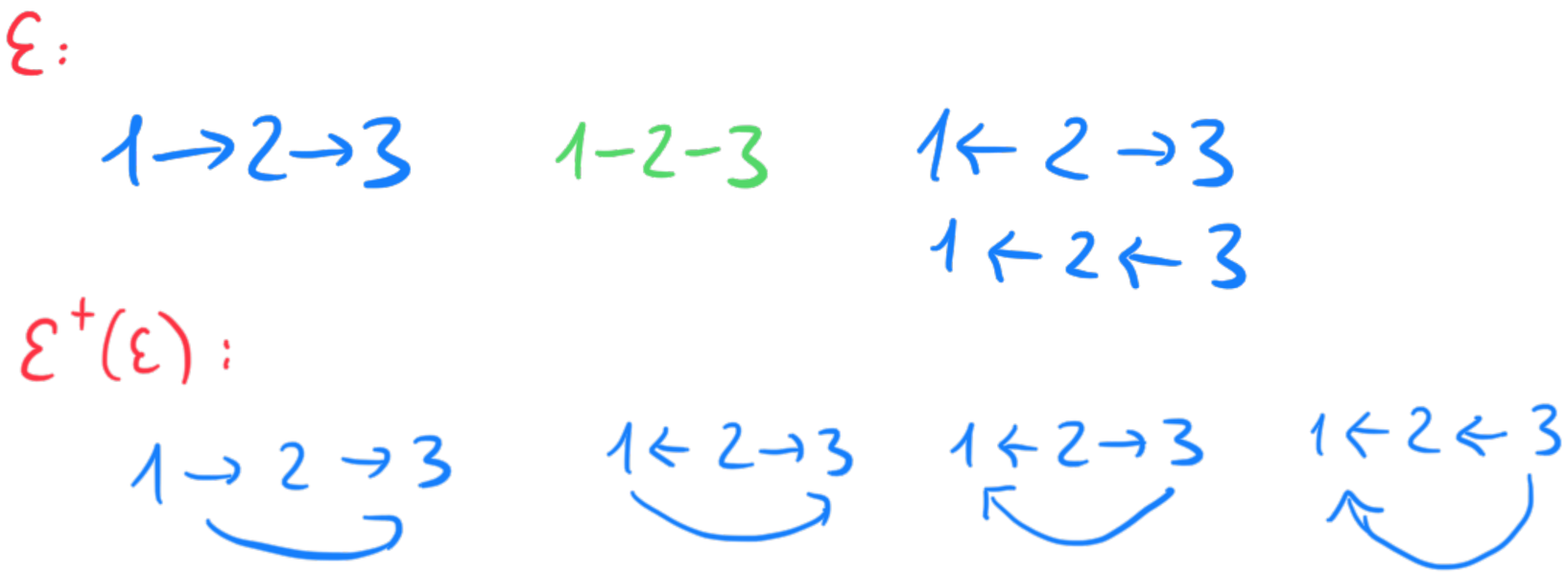
$\mathcal{E}^+(\mathcal{E})$:

$1 \rightarrow 2 \rightarrow 3$

Here we cannot add 3 ->1 because that would be a cycle!

Step 2: for each G, enumerate all G' that can be created by adding a single edge to G

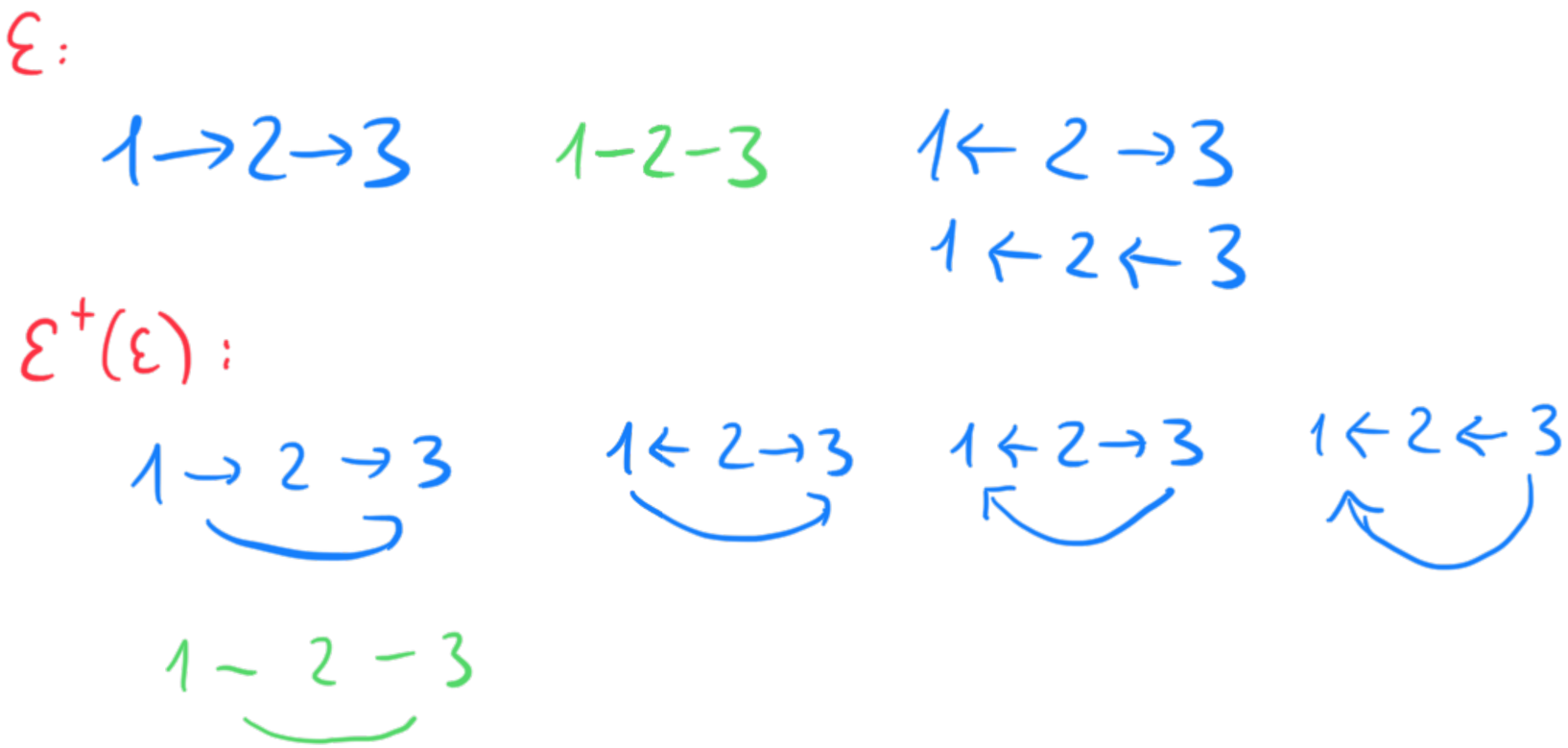
Greedy Equivalence Search - phase 1: example 2



Can we add 1->3 here?

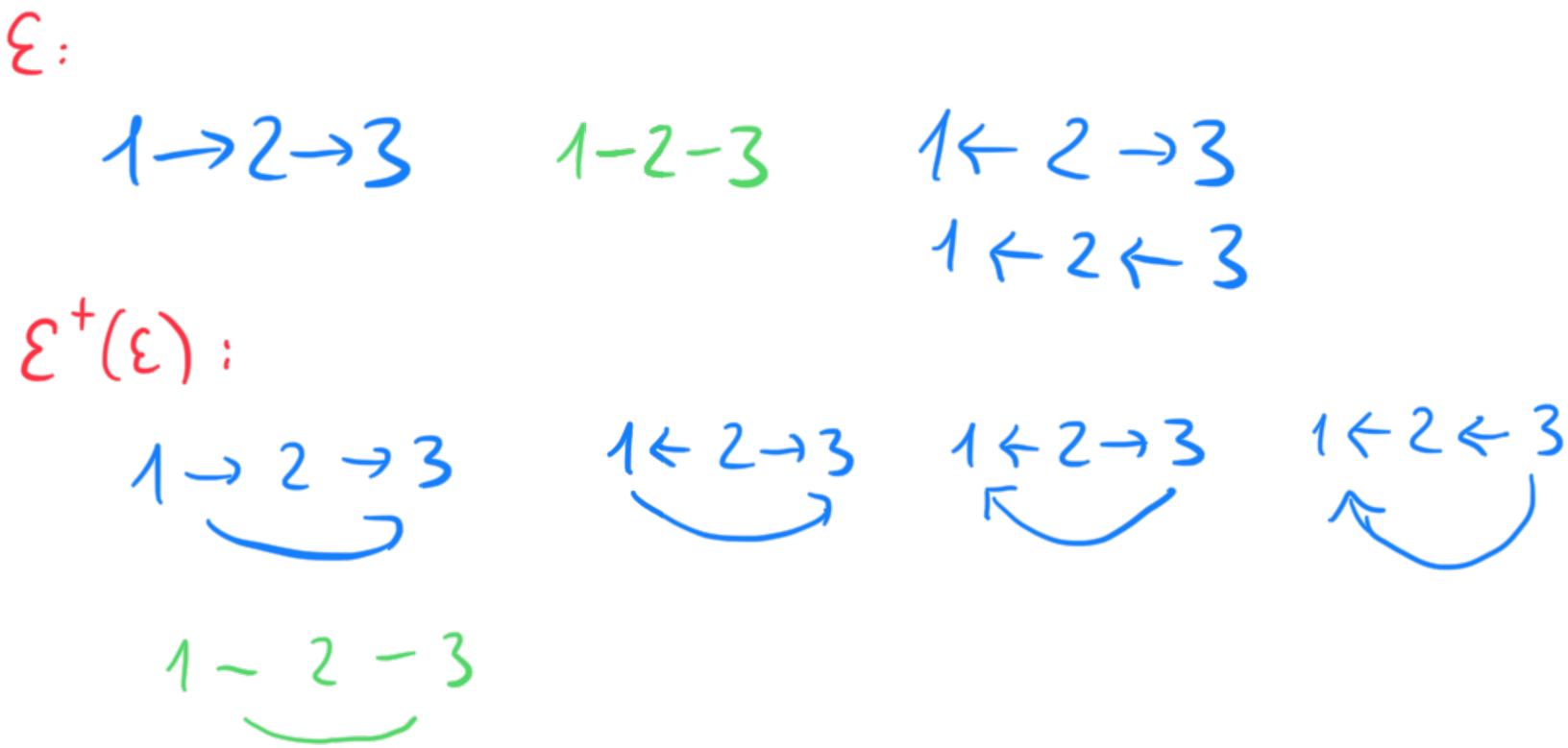
Step 2: for each G, enumerate all G' that can be created by adding a single edge to G

Greedy Equivalence Search - phase 1: example 2



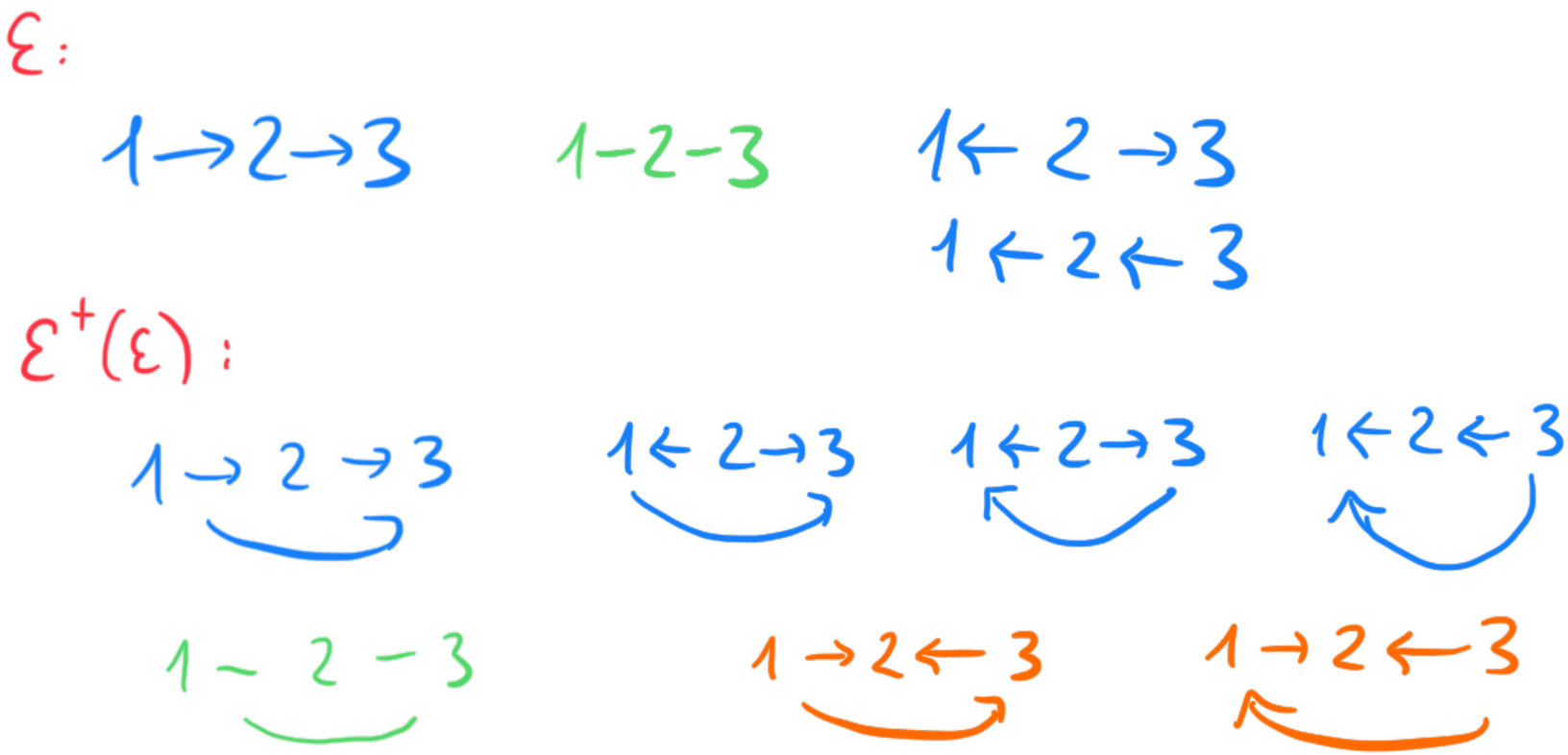
Step 3: For each DAG G' add its MEC ε' to the list of neighbours ϵ^+

Greedy Equivalence Search - phase 1: example 2



Is there any other graph besides the four above that fits this CPDAG?

Greedy Equivalence Search - phase 1: example 2



Greedy Equivalence Search - phase 2: example 2

$\mathcal{E}$:

$1 \rightarrow 2 \rightarrow 3$ $1 - 2 - 3$ $1 \leftarrow 2 \rightarrow 3$

$1 \leftarrow 2 \leftarrow 3$

$\mathcal{E}^{-}(\mathcal{E})$:

Step 1: Enumerate all DAGs $G \in \mathcal{E}$ in the initial MEC $\mathcal{E}$

Greedy Equivalence Search - phase 2: example 2

$\mathcal{E}$:

$1 \to 2 \to 3$ $1 - 2 - 3$ $1 \leftarrow 2 \to 3$

$1 \leftarrow 2 \leftarrow 3$

$\mathcal{E}^-(\mathcal{E})$:

$1 \quad 2 \to 3$ $1 \quad 2 \leftarrow 3$

$1 \to 2 \quad 3$ $1 \leftarrow 2 \quad 3$

Step 2: for each G, enumerate all G' that can be created by removing a single edge to G

Greedy Equivalence Search - phase 2: example 2

$\mathcal{E}$:

$1 \rightarrow 2 \rightarrow 3$ $1 - 2 - 3$ $1 \leftarrow 2 \rightarrow 3$

$1 \leftarrow 2 \leftarrow 3$

$\mathcal{E}^-(\mathcal{E})$:

$1 \quad 2 \rightarrow 3$ $1 \quad 2 \leftarrow 3$ $1 \quad 2 - 3$

$1 \rightarrow 2 \quad 3$ $1 \leftarrow 2 \quad 3$ $1 - 2 \quad 3$

Step 3: For each DAG G' add its MEC ε' to the list of neighbours ϵ^-

Greedy Equivalence Search - Canvas quiz