Universität Stuttgart
IPVS – Institute for Parallel and Distributed Systems

Analytic Computing

# Advanced Topics in Machine Learning
# 10 Structure Learning

Prof. Dr. Steffen Staab
Dr. Rafika Boutalbi
Zihao Wang
https://www.ipvs.uni-stuttgart.de/departments/ac/

# Learning Objectives

Structure Learning

- Scoring
  - Maximum Likelihood – BIC
  - Bayesian Scoring
    - Bayesian Dirichilet Equivalance (BDE)
    - Gamma function
- Search
  - Heuristics
  - Decompositions

# Disclaimer

Figures and examples not marked otherwise are taken from the book by Koller & Friedman

# 1 Likelihood Score for Structure Learning

# Where does the graph come from?

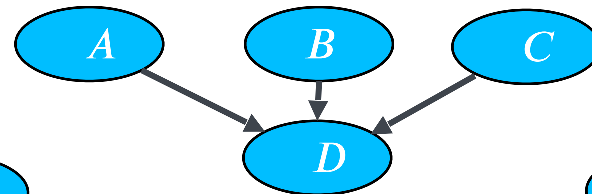
- Until now:
  - random variables defined by expert
  - priors defined by expert
  - structure defined by expert
- Now:
  - random variables defined by expert
  - priors defined by expert
  - **Bayesian network structure to be learned**

Structure may be unknown!

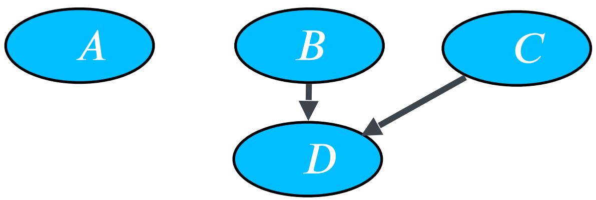
Finding the structure may be the goal!

# Issues with mismatching graph structures

"True" structure for distribution $P^*$

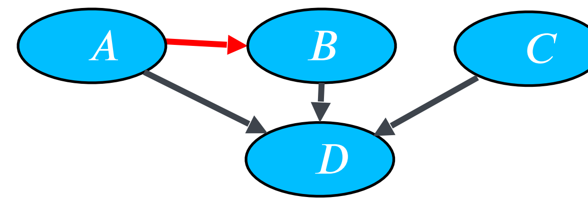



Missing arc




- Incorrect independencies
- $P^*$ cannot be learned
- Might generalize better

Additional arc




- Spurious dependencies
- Can learn $P^*$
- Increases # of parameters
- Possibly worse generalizations

# Principal approach for learning graph structures

- Score
  - quality of a graph structure
  - for representing a data set
- Search
  - over possible graph structures
  - using the score
  - reasonably limiting the search space

# Likelihood Score

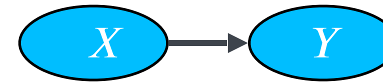
- Find $(G, \theta)$ that maximizes the likelihood

$$\mathrm{score}_{\mathrm{L}}(\mathscr{D};\mathscr{G}) = \ell\left(\mathscr{D};\mathscr{G},\hat{\theta}\right)$$

# Simple Example for Likelihood Score towards Mutual Information

$\mathscr{G}_0$


$\mathscr{G}_1$




$$\mathrm{score}_L(\mathscr{D};\mathscr{G}_0) = \sum_m \left(\log\hat{\theta}_{x[m]} + \log\hat{\theta}_{y[m]}\right)$$

$$\mathrm{score}_L(\mathscr{D};\mathscr{G}_1) = \sum_m \left(\log\hat{\theta}_{x[m]} + \log\hat{\theta}_{y[m]|x[m]}\right)$$

$$\mathrm{score}_L(\mathscr{D};\mathscr{G}_1) - \mathrm{score}_L(\mathscr{D};\mathscr{G}_0) = \sum_m \left(\log\hat{\theta}_{y[m]|x[m]} - \log\hat{\theta}_{y[m]}\right) =$$

$$= \sum_{x,y} k(x,y)\log\hat{\theta}_{y|x} - \sum_{y} k(y)\log\hat{\theta}_{y} = M\sum_{x,y}\hat{P}(x,y)\log\hat{P}(y\,|\,x) - M\sum_{y}\hat{P}(y)\log\hat{P}(y) =$$

$$= M\left(\sum_{x,y}\hat{P}(x,y)\log\hat{P}(y\,|\,x) - \sum_{x,y}\hat{P}(x,y)\log\hat{P}(y)\right) = M\left(\sum_{x,y}\hat{P}(x,y)\log\frac{\hat{P}(x,y)}{\hat{P}(x)\hat{P}(y)}\right) =$$

$$= \qquad M \bullet \mathbf{I}(X;Y)$$

# General Decomposition

- The likelihood score thus decomposes into

$$\mathrm{score}_L(\mathcal{D};\mathcal{G}) = M\sum_{i=1}^{n}\mathbf{I}_{\hat{P}}(X_i;\mathrm{Parents}(X_i;\mathcal{G})) - M\sum_{i=1}^{n}\mathbf{H}_{\hat{P}}(X_i)$$

score is higher if $X_i$ correlates with its parents

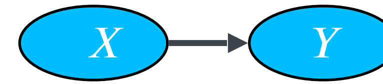
$$\mathbf{I}_{\hat{P}}(\boldsymbol{X};\boldsymbol{Y}) = \sum_{x,y}\hat{P}(x,y)\log\frac{\hat{P}(x,y)}{\hat{P}(x)\hat{P}(y)}$$

$$\mathbf{H}_{\hat{P}}(\boldsymbol{X}) = \sum_{x}\hat{P}(x)\log\hat{P}(x)$$

# Limitations of Likelihood Score

$\mathcal{G}_0$


$\mathcal{G}_1$




$$\mathrm{score}_L(\mathcal{D};\mathcal{G}_0)=\sum_m\left(\log\hat{\theta}_{x[m]}+\log\hat{\theta}_{y[m]}\right)$$

$$\mathrm{score}_L(\mathcal{D};\mathcal{G}_1)=\sum_m\left(\log\hat{\theta}_{x[m]}+\log\hat{\theta}_{y[m]|x[m]}\right)$$

$$\mathrm{score}_L(\mathcal{D};\mathcal{G}_1)-\mathrm{score}_L(\mathcal{D};\mathcal{G}_0)=M\bullet\mathbf{I}(X;Y)$$

- Mutual information is always $\geq 0$
- Equals $0$ iff $X, Y$ are independent
  - in empirical distribution $\hat{P}$: $\mathbf{I}_{\hat{P}}(X;Y)>0$ holds almost always
- Adding edges is not punished by the score, almost always improves the score
- Thus: score is maximized for fully connected network

# Avoiding overfitting

- Restrict the hypothesis space
  - restrict number of parents
  - restrict number of parameters
- Penalize complexity in the score
  - explicitly
  - **Bayesian score** averages over all possible parameter values

    (not only MLE of parameters)

# Minimum description length principle

- Algorithmic theory: *the description length of a data sequence is the length of the smallest program that outputs that data set (Kolmogorov complexity)*.
  - uncomputable
  - BUT: Given two models that output the data set MDL selects the „better“ model
- Applied to machine learning:
  - a smaller model is „better“, has “learned more“

# Bayesian information criterion

- Penalize complexity

$$\mathrm{score}_{\mathrm{BIC}}(\mathscr{D};\mathscr{G}) = \mathrm{score}_L(\mathscr{D};\mathscr{G}) - \frac{\mathrm{Dim}[\mathscr{G}]}{2}\log M$$

where

- $M$ is the number of training instances
- $\mathrm{Dim}[\mathscr{G}]$ are number of independent parameters (degrees of freedom)

Trade-off between fit to data and model complexity using minimum description length principle

# Asymptotic Behaviour of BIC

For small $M$,
BIC tends to underfit

$$\text{score}_{\text{BIC}}(\mathscr{D};\mathscr{G}) = \text{score}_L(\mathscr{D};\mathscr{G}) - \frac{\text{Dim}[\mathscr{G}]}{2}\log M =$$

$$= M\sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i;\text{Parents}(X_i;\mathscr{G})) - M\sum_{i=1}^{n} \mathbf{H}_{\hat{P}}(X_i) - \frac{\text{Dim}[\mathscr{G}]}{2}\log M$$

Mutual information is weighted with $M$,
model complexity is weighted with $(\log M)/2$,
thus as $M$ grows more emphasis is given to fit the data

**Consistency**: As $M \to \infty$, the true structure $\mathscr{G}^*$ maximizes the score
(or any I-equivalent structure)

- spurious edges will be penalized, required edges will be added

# 2 Bayesian Model Averaging

## Main principle of Bayesian approach

- Whenever we have uncertainty over anything,
  we place a distribution over it.

- In the context of structure learning:
  we have uncertainty both over structure and over parameters.

# Bayesian score (against overfitting)

marginal likelihood

prior over structures

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

$$\mathrm{score}_{\mathrm{B}}(\mathcal{D}; \mathcal{G}) = \log P(\mathcal{D} \mid \mathcal{G}) + \log P(\mathcal{G})$$

# Less optimistic than MLE for $\hat{\theta}$

Marginal likelihood of data given $\mathcal{G}$

- Averaging over all $\theta_{\mathcal{G}}$

$$\mathrm{score}_{\mathrm{B}}(\mathcal{D};\mathcal{G}) = \log P(\mathcal{D} \mid \mathcal{G}) + \log P(\mathcal{G})$$

likelihood

prior over parameters

$$P(\mathcal{D} \mid \mathcal{G}) = \int P(\mathcal{D} \mid \mathcal{G}, \theta_{\mathcal{G}}) P(\theta_{\mathcal{G}} \mid \mathcal{G}) d\theta_{\mathcal{G}}$$

# Marginal likelihood intuition

$$P(\mathscr{D} \mid \mathscr{G}) = P\left(x[1], \ldots, x[M] \middle| \mathscr{G}\right)$$

$$P(x[1] \mid \mathscr{G})$$

$$P\left(x[2] \mid x[1], \mathscr{G}\right)$$

$$\ldots$$

$$P\left(x[M] \mid x[1]\right.$$

Marginal likelihood: integrates generalization capability – how well can I predict $x[M]$ given $x\left[M-1\right]\ldots$ Natural way to punish for overfitting!

Contrast MLE: $\theta_{\mathscr{G}}$ depends on all of $\mathscr{D}$ – cannot break down MLE likewise

# Marginal Likelihood for <u>Multinomial</u> Bayesian Networks

Not to be memorized

$$P(\mathscr{D} \mid \mathscr{G}) = \int P(\mathscr{D} \mid \mathscr{G}, \theta_{\mathscr{G}}) P(\theta_{\mathscr{G}} \mid \mathscr{G}) d\theta_{\mathscr{G}} =$$

$$= \prod_{i} \prod_{\boldsymbol{u}_i \in \mathrm{Val}(\mathrm{Parents}(X_i; \mathscr{G}))} \frac{\Gamma(\alpha_{X_i|\boldsymbol{u}_i})}{\Gamma(\alpha_{X_i|\boldsymbol{u}_i} + M[\boldsymbol{u}_i])} \prod_{x_i^j \in \mathrm{Val}(X_i)} \left[ \frac{\Gamma\left(\alpha_{x_i^j|\boldsymbol{u}_i} + M[x_i^j, \boldsymbol{u}_i]\right)}{\Gamma\left(\alpha_{x_i^j|\boldsymbol{u}_i}\right)} \right]$$

Dirichlet prior

sufficient statistics

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt \qquad \Gamma(x) = x \bullet \Gamma(x-1)$$

# Marginal Likelihood Decomposition

$$\log P\left(\mathscr{D} \mid \mathscr{G}\right) = \sum_i \mathrm{FamilyScore}_B\left(X_i \mid \mathrm{Parents}(X_i; \mathscr{G}, \mathscr{D})\right)$$

# Struture Priors

$$\text{score}_B(\mathscr{D};\mathscr{G}) = \log P(\mathscr{D}\,|\,\mathscr{G}) + \log P(\mathscr{G})$$

Structure prior $P(\ \mathscr{G}\ )$

- uniform prior: make $P(\mathscr{G})$ constant
  - not bad; averaging in $P(\mathscr{D}\,|\,\mathscr{G})$ already avoids overfitting
  - prior penalizing of number of edges: $P(\mathscr{G}) \propto c^{|\mathscr{G}|}$, with $0 < c < 1$
  - prior penalizing of number of parameters
- Normalizing constant across network is similar and can be ignored

## Bayesian Dirichilet Equivalance Prior (BDe prior)

- Construction like in Learning Part II (unit 3)
  - equivalent sample size parameter $\alpha$
  - $B_0$ network representing prior probability of events
    - often a network without edges!
  - Set $\alpha_{x|\boldsymbol{u}} := \alpha \bullet P(x, u \mid B_0)$
    - with $u$ the value for parents in the target network (not in $B_0$)
- $B_0$ provides priors for all candidate networks
  - Unique prior with the property that I-equivalent networks have the same Bayesian score

# BDe and BIC

- As $M \to \infty$, a network $\mathcal{G}$ with Dirichlet priors satisfies

$$\log P(\mathcal{D} \mid \mathcal{G}) = \ell\left(\mathcal{D}; \mathcal{G}, \hat{\theta}\right) - \frac{\mathrm{Dim}[\mathcal{G}]}{2} \log M + \mathcal{O}(1)$$

$$\mathrm{score}_{\mathrm{BIC}}(\mathcal{D}; \mathcal{G}) = \mathrm{score}_L(\mathcal{D}; \mathcal{G}) - \frac{\mathrm{Dim}[\mathcal{G}]}{2} \log M$$

Thus, as BIC score is consistent, BDE is also consistent

# 3 Learning Forests

# Learnings Trees/Forests

- Trees
  - One root, at most one parent per variable
- Forests
  - at most one parent per variable, multiple roots allowed
- Why trees/forests?
  - elegant math
  - efficient optimization
  - sparse parametrization naturally avoids overfitting

# Learning Forests

- $p(i)$ is parent of $X_i$
  - $0$ if $X_i$ has no parent

$$\mathrm{score}\big(\mathscr{D};\mathscr{G}\big) = \sum_{i=1}^{n} \mathrm{score}(X_i \,|\, \mathrm{Parents}\big(X_i;\mathscr{G}\big)\ ;\ \mathscr{D}) =$$

$$= \sum_{i:p(i)>0} \mathrm{score}(X_i \,|\, X_{p(i)};\ \mathscr{D}) + \sum_{i:p(i)=0} \mathrm{score}(X_i;\ \mathscr{D}) =$$

$$= \sum_{i:p(i)>0} \left( \mathrm{score}\left(X_i \middle| X_{p(i)};\ \mathscr{D}\right) - \mathrm{score}(X_i;\ \mathscr{D}) \right) + \sum_{i=1}^{n} \mathrm{score}(X_i;\ \mathscr{D})$$

improvement over empty network

score of empty network

Score = Sum of edge scores + constant

# Learning Forests I

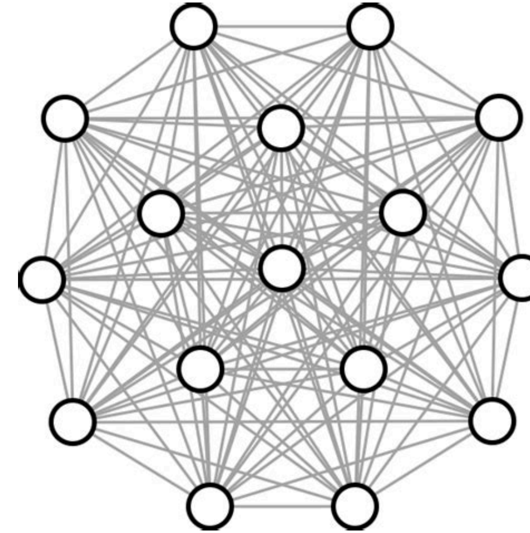
- Set $w(i \to j) = \left( \mathrm{score}\left( X_j \middle| X_i;\ \mathscr{D} \right) - \mathrm{score}(X_j;\ \mathscr{D}) \right)$

- Given likelihood score $w(i \to j) = M \bullet \mathbf{I}_{\hat{P}}(X_i; X_j)$, all edge weights are non-negative
  - **Optimal structure is always a tree**
- Given BIC or Bde score, weights can be negative
  - **Optimal structure might be a forest**

# Learning Forests II

- A score satisfies score equivalence if I-equivalent structures have the same scores
  - For example: likelihood, BIC, Bde
- Then, $w(i \rightarrow j) = w(j \rightarrow i)$, we can use an undirected graph

# Learning Forests III
# (for score-equivalence)

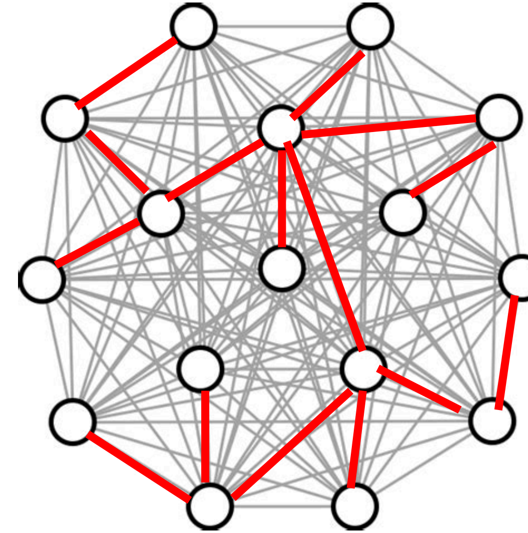


- Define undirected graph with nodes $\{1 \dots n\}$
- Set

$$w(i \to j) = \max\left(0, \left(\mathrm{score}\left(X_i \,\middle|\, X_j;\ \mathscr{D}\right) - \mathrm{score}(X_j;\ \mathscr{D})\right)\right)$$

- Find tree with maximal weight
  - Standard algorithms for max-weight spanning trees in $\mathcal{O}(n^2)$ time
    - Prim‘s or Kruskals‘s
    - Remove all edges of weight 0 to produce a forest

# Learning Forests III
# (for score-equivalent scores)



- Define undirected graph with nodes $\{1 \ldots n\}$
- Set

$$w(i \to j) = \max\left(0, \left(\text{score}\left(X_i \middle| X_j;\ \mathscr{D}\right) - \text{score}(X_j;\ \mathscr{D})\right)\right)$$

- Find tree with maximal weight
  - Standard algorithms for max-weight spanning trees in $\mathcal{O}(n^2)$ time
    - Prim‘s or Kruskals‘s
    - Remove all edges of weight 0 to produce a forest

# Summary

- Structure learning is an optimization over the combinatorial space of graph structures
- Decomposability
  - network score is a sum of terms for different families
- Optimal tree-structured network can be found using standard MST algorithms

# 4 Heuristic Search

# Optimization Problem

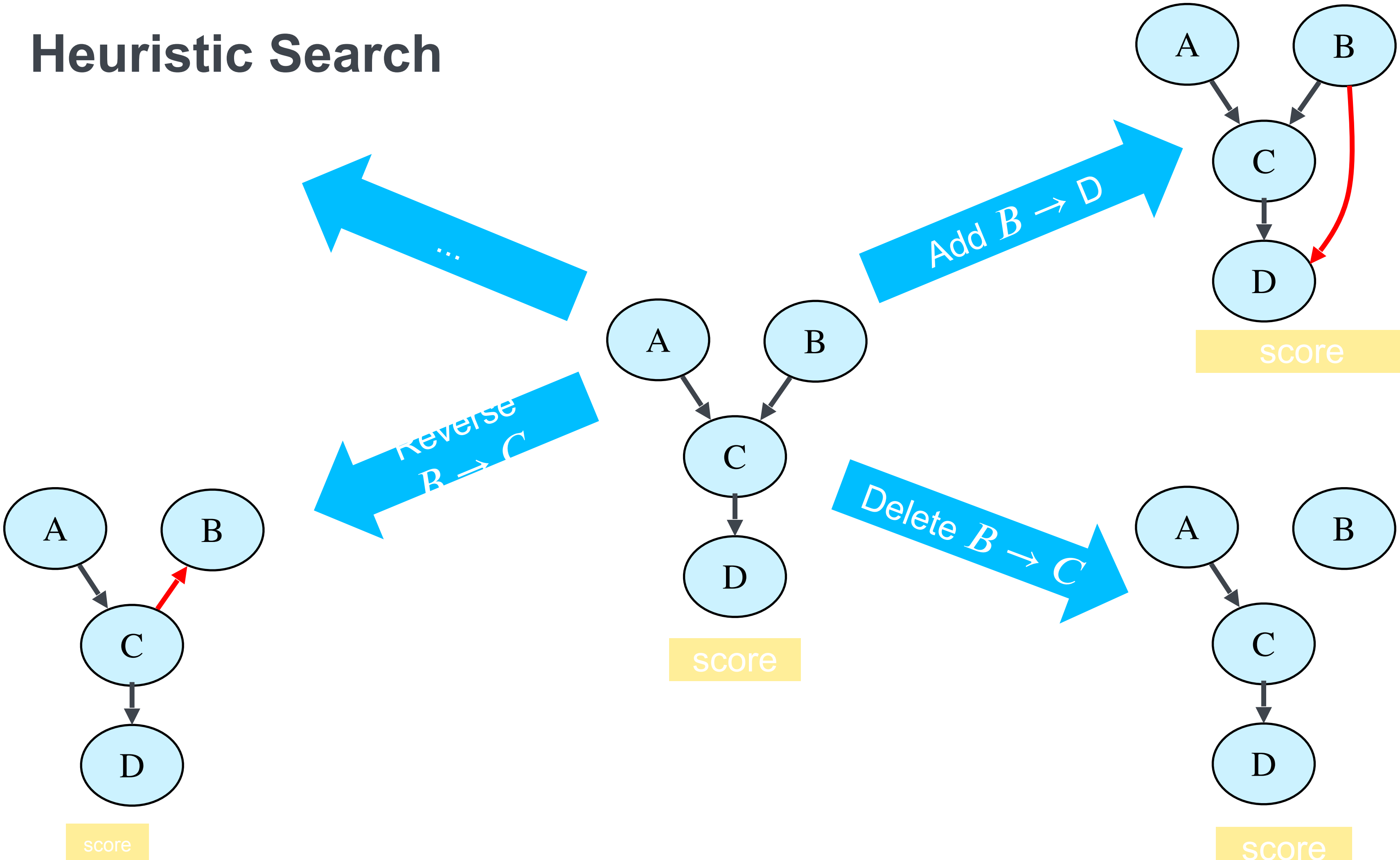
- Input
  - Training data
  - Scoring function
  - Set of possible structures
- Output
  - A network that maximizes the score

# Beyond Trees/Forests

- Problem is not obvious for general networks
  - Example: Allowing two parents, greedy algorithm is no longer guaranteed to find the optimal network

- Theorem
  - Finding maximal scoring network structure with at most k parents for each variable is NP-hard for k>1

# Heuristic Search



# Heuristic Search

- Search operators
  - local steps: edge addition, deletion, reversal
  - global steps

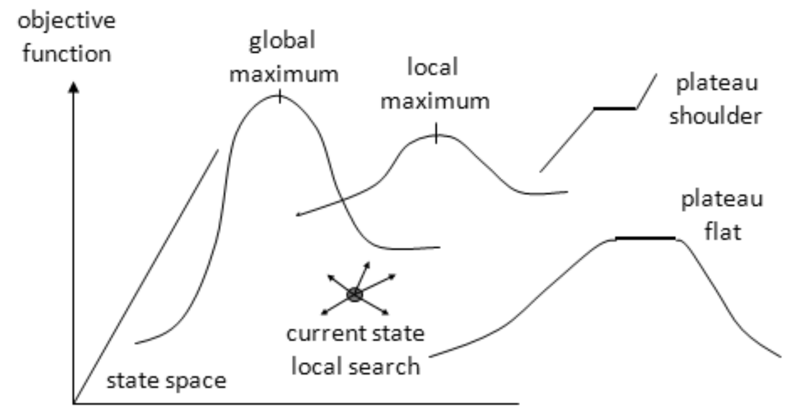
- Search techniques
  - greedy hill-climbing
  - best first search
  - simulated annealing
  - ...

# Heuristic Search: Greedy Hill Climbing

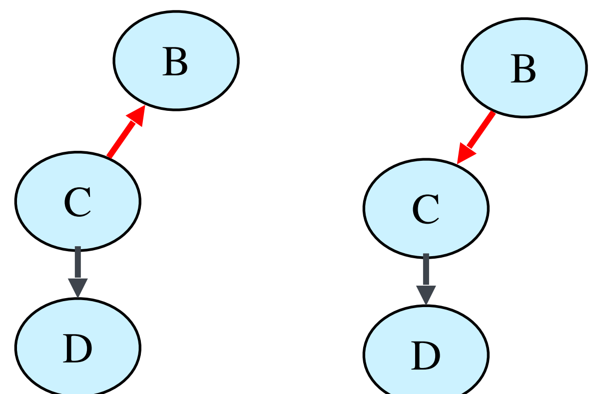
- Start with a given network
  - empty network
  - best tree
  - random network
  - prior knowledge
- At each iteration
  - Consider score for all possible one-step changes
  - Apply change that most improves the score
- Stop when no modification improves the score
  - local maximum

# Greedy Hill Climbing Pitfalls

- Getting stuck in
  - local maxima

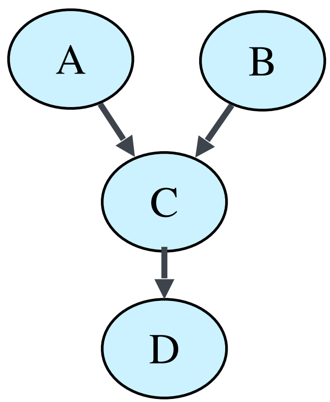



  - plateau
    - typically because equivalent networks are often neighbors in the search space

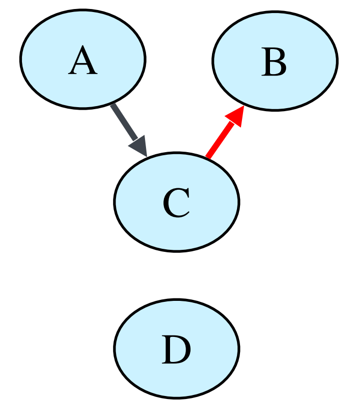


# Edge reversal: Get (conditional) independences right

$\mathscr{G}^*$



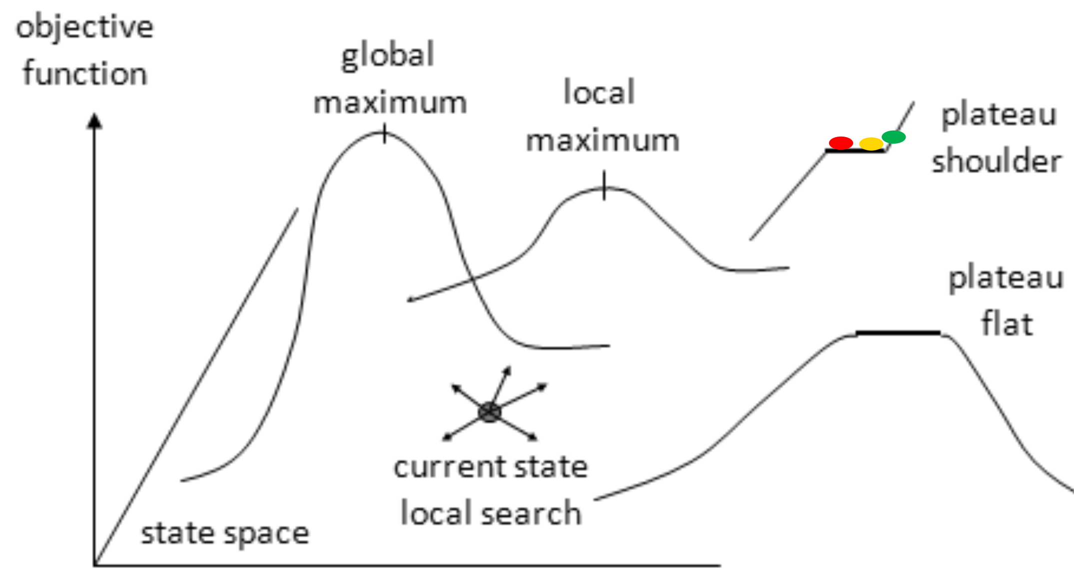

local maximum



# A pretty good, simple algorithm

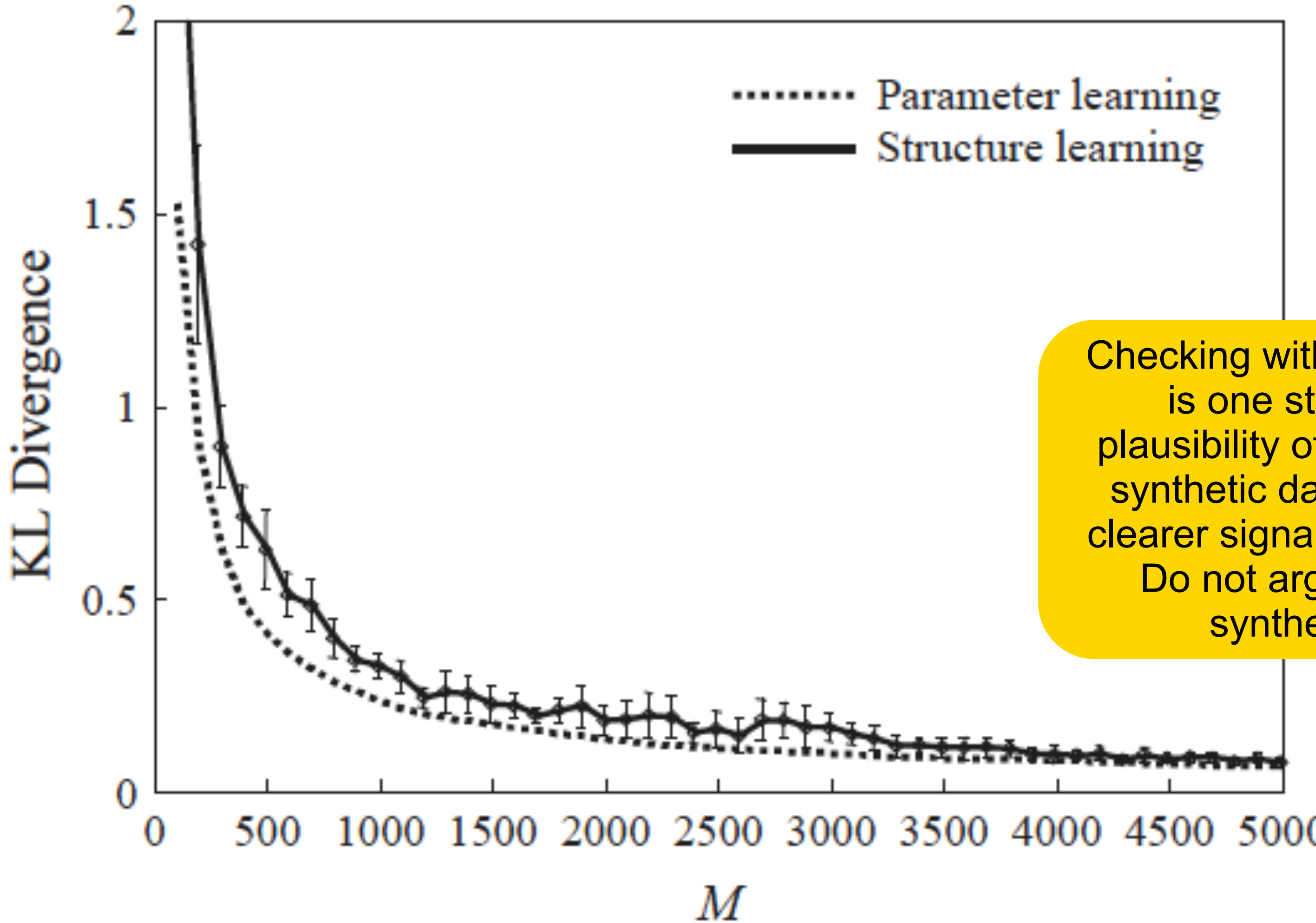
Greedy hill-climbing augmented with:

- Random restarts
  - when we get stuck, take some number of random steps and then start climbing again




- Tabu list
  - keep a list of K steps most recently taken
  - search cannot reverse any of these steps („momentum“)

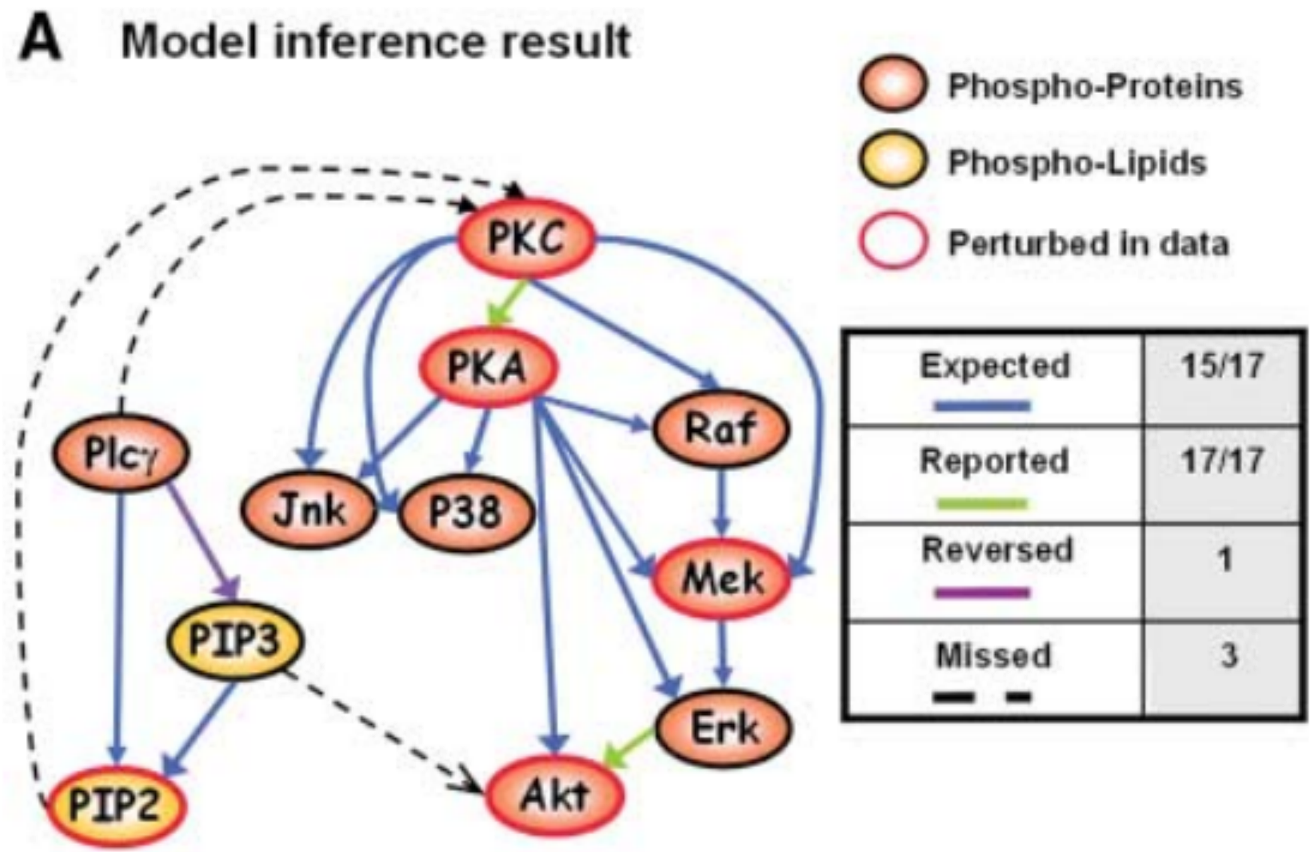
# Synthetic example from book (Intensive Care Unit alarm)




Checking with synthetic data is one step to show plausibility of approach, but synthetic data usually has clearer signal than real data. Do not argue only with synthetic data!

# Knowledge Discovery Example

Sachs, Karen, et al. "Causal protein-signaling networks derived from multiparameter single-cell data." *Science* 308.5721 (2005): 523-529.



# Summary

- Useful for building better predictive models:
  - when domain experts don‘t know the structure
  - for knowledge discovery
- Finding highest-scoring structure is NP-hard
- Typically solved using simple heuristic search
  - local steps: edge addition, deletion, reversal
  - hill climbing with tabu lists and random restarts
- But there are more advanced algorithms!

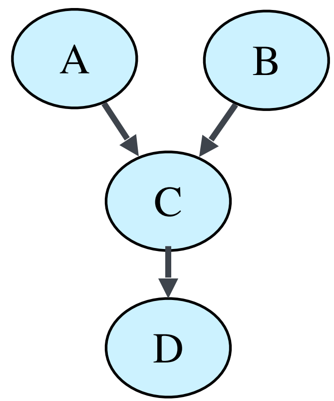
# 5 Learning General Graphs: Search and Decomposability

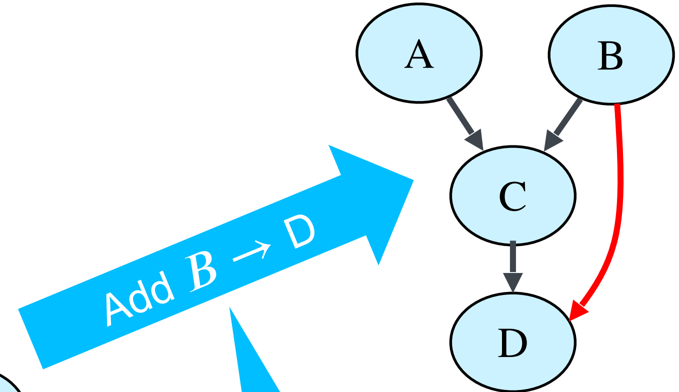
# Naive Computational Analysis of Structure Learning

- Operations per search step:
  - $\mathcal{O}\big(n(n-1)\big) = \mathcal{O}\big(n^2\big)$ possible edges to consider
  - per edge:
    - if present, consider delete or reverse
    - if absent, consider addition
  - Costs per network evaluation
    - components in score: $n$ edges
    - compute sufficient statistics: scan $M$ instances
    - Acyclicity check: $\mathcal{O}(m)$ number of present edges

    $\mathcal{O}(Mn + m)$
- Total: $\mathcal{O}\big(n^2(Mn+m)\big)$, usually $< \mathcal{O}\Big(Mn^3\Big)$ per search step

# Exploiting Decomposability



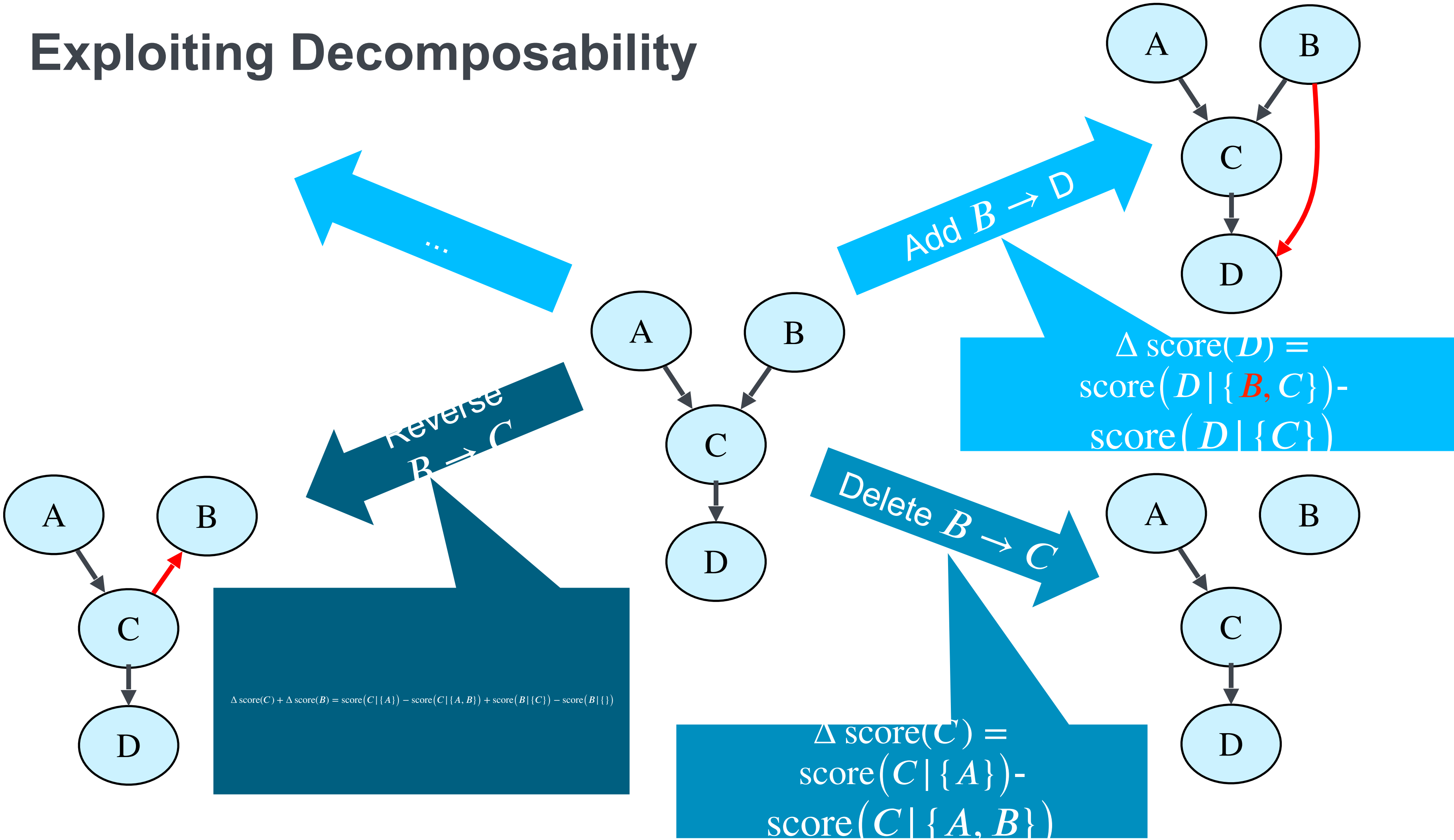




$$\Delta\,\mathrm{score}(D) = \mathrm{score}\left(D \mid \{B, C\}\right) - \mathrm{score}\left(D \mid \{C\}\right)$$

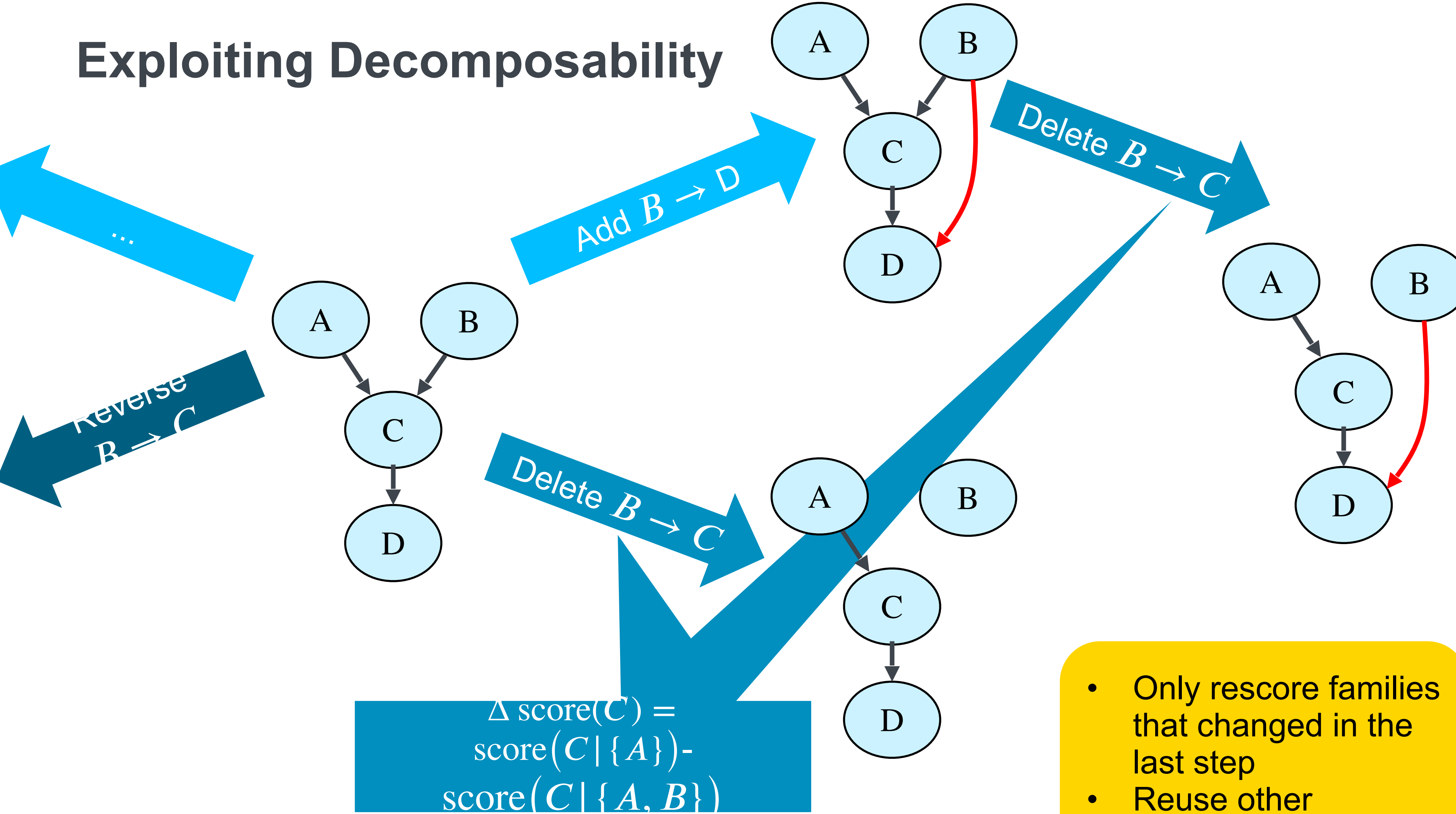
$$\mathrm{score}\left(A \mid \{\,\}\right) + \mathrm{score}\left(B \mid \{\,\}\right) + \mathrm{score}\left(C \mid \{A, B\}\right) + \mathrm{score}\left(D \mid \{C\}\right)$$

$$\mathrm{score}\left(A \mid \{\,\}\right) + \mathrm{score}\left(B \mid \{\,\}\right) + \mathrm{score}\left(C \mid \{A, B\}\right) + \mathrm{score}\left(D \mid \{B, C\}\right)$$

# Exploiting Decomposability



# Exploiting Decomposability




- Only rescore families that changed in the last step
- Reuse other computations

# Computational Cost

- Having picked a step
  - 1 or 2 families affected by step
  - Compute $\mathcal{O}(n)$ delta-scores
  - Each one takes $\mathcal{O}(M)$ time
- Priority queue of operators sorted by delta-score

$\mathcal{O}(Mn)$

$\mathcal{O}(n \log n)$

$\mathcal{O}(Mn + n \log n)$

# More computational efficiency

- Most plausible families are variations on a theme
  - $A$ might depend on $B_1, \ldots, B_k$ with a small $k$
- Re-use and adapt previously computed sufficient statistics
  - $A \rightarrow B$ and $B \rightarrow A$ require the same count $M[A, B]$
- Restrict in advance the set of operators considered in search
  - changes what can be found!

Universität Stuttgart
IPVS

# Thank you!

**Steffen Staab**

E-Mail Steffen.staab@ipvs.uni-stuttgart.de
Telefon +49 (0) 711 685- To be defined
www. ipvs.uni-stuttgart.de/departments/ac/

Universität Stuttgart
Analytic Computing, IPVS
Universitätsstraße 32, 50569 Stuttgart