UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Bayesian Networks

Learning the topology from a database of cases using the K2 algorithm

L. Belli S. Di Lucia

30/06/2025

Table of Contents

Table of Contents

Goal

Use the K2 algorithm to learn the **topology** of a Bayesian Network, i.e. the probabilistic structure of a BN, given a dataset.

This task is done by:

1. implementing the K2 algorithm
2. using the `bnstruct` R library

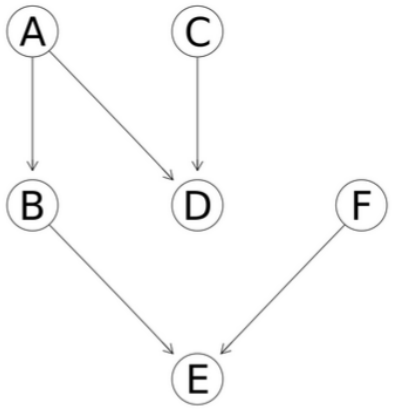

Figure: Example of a BN structure for the learning.test dataset

A Bayesian Network (BN) is a probabilistic graphical model that represents a set of variables and their conditional dependencies via a **directed acyclic graph** (**DAG**). [1]

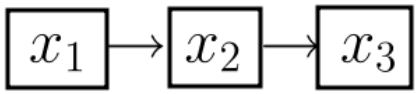

Figure: BN structure for Ruiz dataset

- Each node represents a **random variable**
- Each edge represents a **conditional dependency**

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \pi_i)$$

Introduction - K2 Algorithm

The K2 algorithm is a greedy search algorithm used to learn the structure of a Bayesian network from data.[6]
The assumptions are:

1. Nodes are **ordered**: $X_1, X_2, ..., X_n$
2. All structures are considered equally likely
3. Data are **complete** (no missing values)
4. Each node can have a **limited** number of parents

Introduction - K2 Pseudocode

1: **for** $i := 1$ to n **do**
2: $\pi_i := \emptyset$
3: $P_{\text{old}} := f(i, \pi_i)$
4: OKToProceed := TRUE
5: **while** OKToProceed **and** $|\pi_i| < u$ **do**
6: $P_{\text{new}} := f(i, \pi_i \cup \{z\})$
7: **if** $P_{\text{new}} > P_{\text{old}}$ **then**
8: $P_{\text{old}} := P_{\text{new}}$
9: $\pi_i := \pi_i \cup \{z\}$
10: **else**
11: OKToProceed := FALSE
12: **end if**
13: **end while**
14: **write**("Node: ", x_i, "Parents of this node: ", π_i)
15: **end for**

Table of Contents

Implementation: Data

For testing the K2 algorithm we used four different datasets:

1. **Ruiz Dataset**:
   - contains 3 variables (x_1, x_2, x_3, i.e. the nodes) and 10 items
2. **learning.test**:
   - synthetic dataset included in the `bnlearn` R package
   - contains 6 variables (A, B, C, D, E, F) and 5000 items
3. **Asia Dataset**:
   - data about lung diseases (tuberculosis, lung cancer or bronchitis) and visits to Asia from `bnstruct` [2]
   - 8 variables and 10000 items
4. **Child Dataset**:
   - 20 variables and 5000 items related to health conditions and symptoms from `bnstruct` [2]
   - contains random missing values [3]

Implementation: K2

We wrote the K2 algorithm in `Rbase` along with some `tidyverse` packages, to better deal with datasets as whole entities and improve efficiency.

Dealing with numerical over/underflow

We adopted log-form for the $f(i, \pi_i)$ as it would diverge for bigger datasets.

$$f(i, \pi_i) = \prod_{j_i=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} a_{ijk}! \tag{1}$$

We assigned **BIC score** using `bnlearn::score` to each detected BN structure.

Implementation: Nodes order

Practical issues

K2 needs as input the correct **nodes order**, but this is not available for every dataset.

To deal with this we searched for the order that optimizes the BIC score

- For smaller datasets (less than 9 variables) we **bruteforced** the right permutation
- As the number of variables increase, it is more efficient to adopt optimization algorithms. We used **Simulated Annealing**, implemented in `base::optim`, as it suits well with combinatorial problems.

Implementation with `bnstruct`

We use the `learn.network()` function from the `bnstruct` R package using different combinations of learning algorithms and scoring functions:

- **MMHC** with **BDeu**
- **MMHC** with **BIC**
- **SEM** with **BDeu** (used specifically for Child dataset - before imputation - which contains missing values)

We also apply **bootstrap** on Asia dataset and **imputation** (kNN algo) on Child dataset.

Table of Contents

Results: Visual Comparisons

In the following BN structures [4] obtained by `bnlearn::graphviz.compare`:

1. the `bnlearn`'s network is taken as the true network [5];
2. true positive arcs are in black;
3. false positive arcs (which are missing or have different directions in the true network) are in red;
4. false negative arcs are in blue and drawn using a dashed line.

Instead, for the `bnstruct` comparison:

1. the `bnstruct`'s network is taken as the true network;
2. the algorithm used in the `learning.function` is MMHC and the scoring function is BIC.

Results: Ruiz Dataset

Figure: Topology learned using K2 algorithm and bruteforcing the correct nodes order.

Results: `learning.test` Dataset

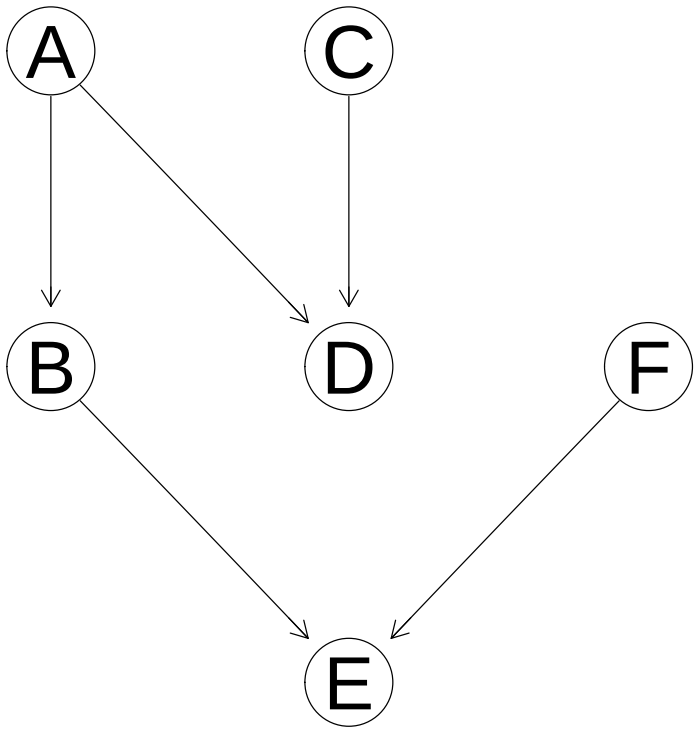

Figure: Topology learned using K2 algorithm and bruteforcing the correct nodes order.

Results: Asia Dataset

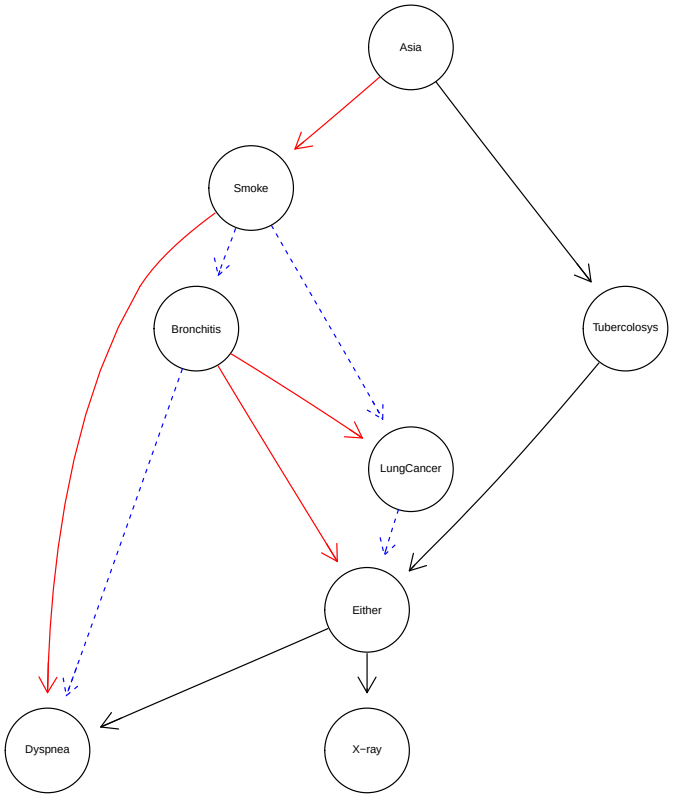

Figure: Solution by bruteforce. Four false positives and four false negatives.

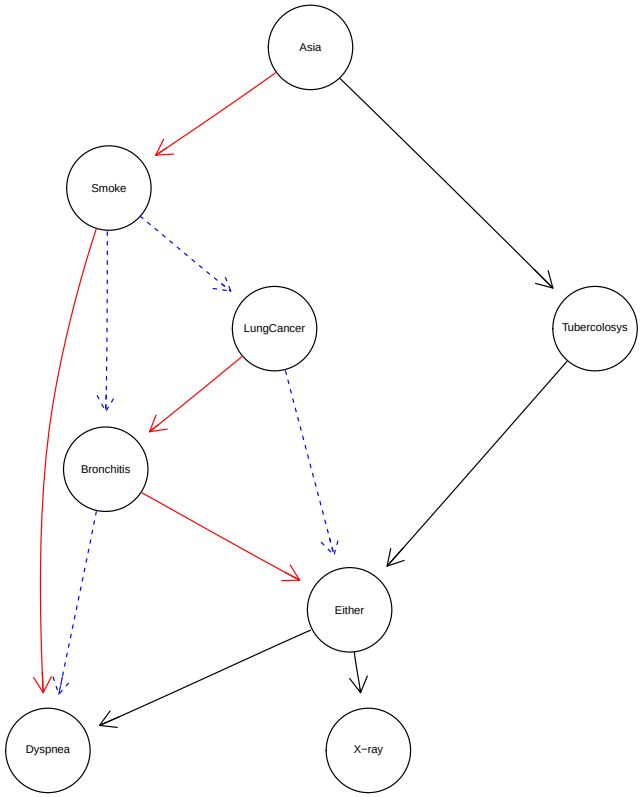

Figure: Solution by SANN. Four false positives and four false negatives.

Results: Child Dataset

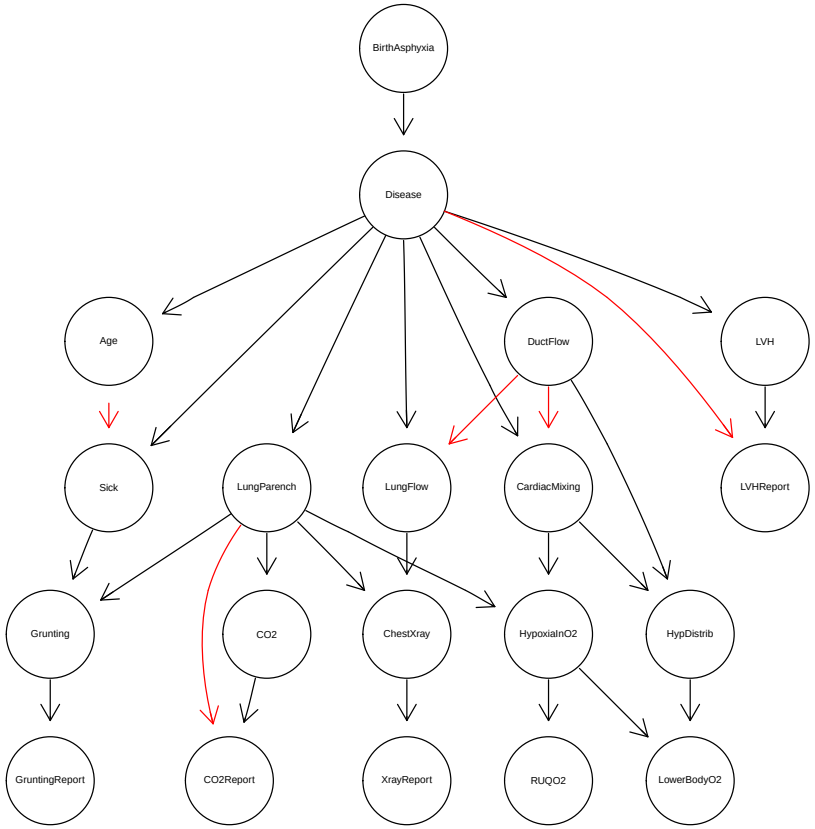

Figure: Solution by SANN. Five false positives (Age-Sick connection has opposite direction).

Compare with `bnstruct`: Ruiz Dataset

Figure: BN structure learned from the Ruiz dataset. The edge connections given by `bnstruct` are opposite to that obtained with K2 algorithm (false positives).

Compare with `bnstruct`: `learning.test` Dataset

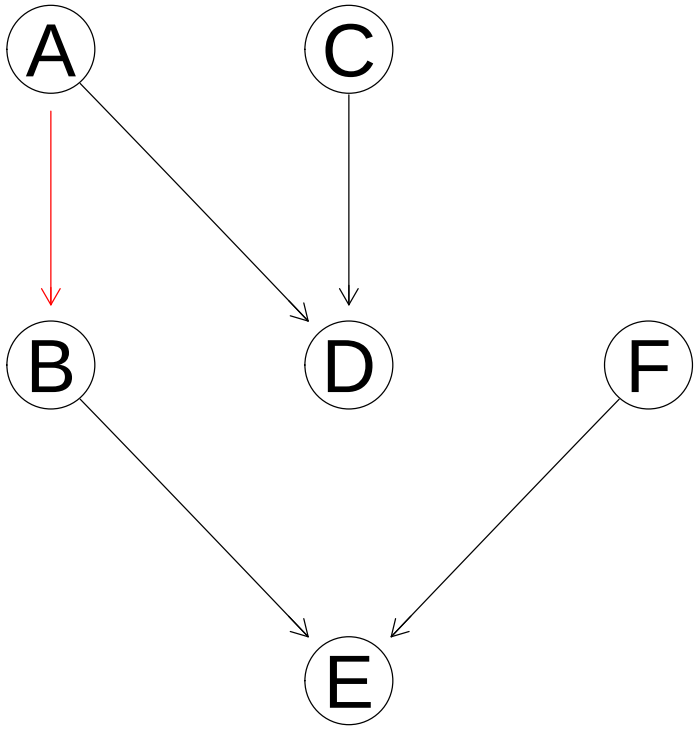

Figure: BN structure learned from the `learning.test` dataset. Only one edge connection given by `bnstruct` is opposite to that obtained with K2 algorithm (false positive).

Compare with bnstruct: Asia Dataset

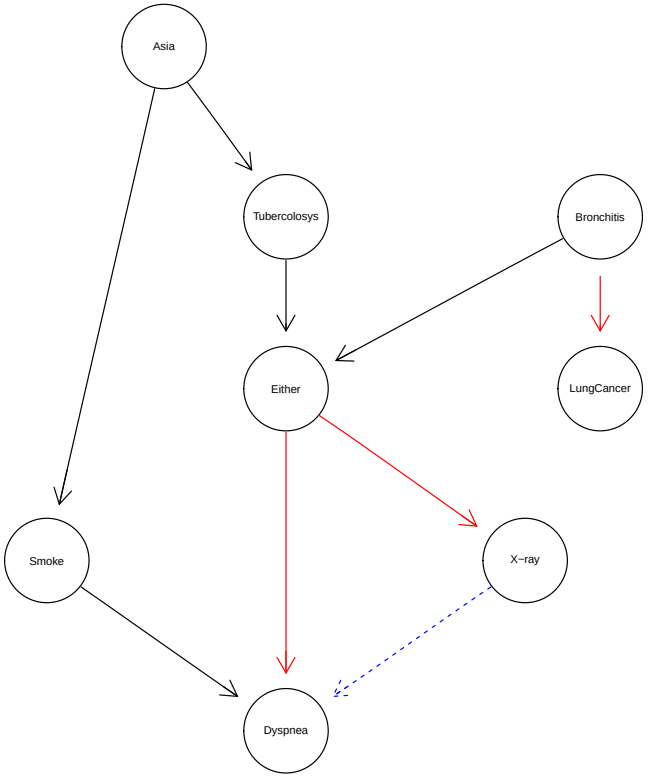

Figure: BN structure learned from the Asia dataset, solution by bruteforce. Three false positives (B-L for the opposite direction) and one false negative.

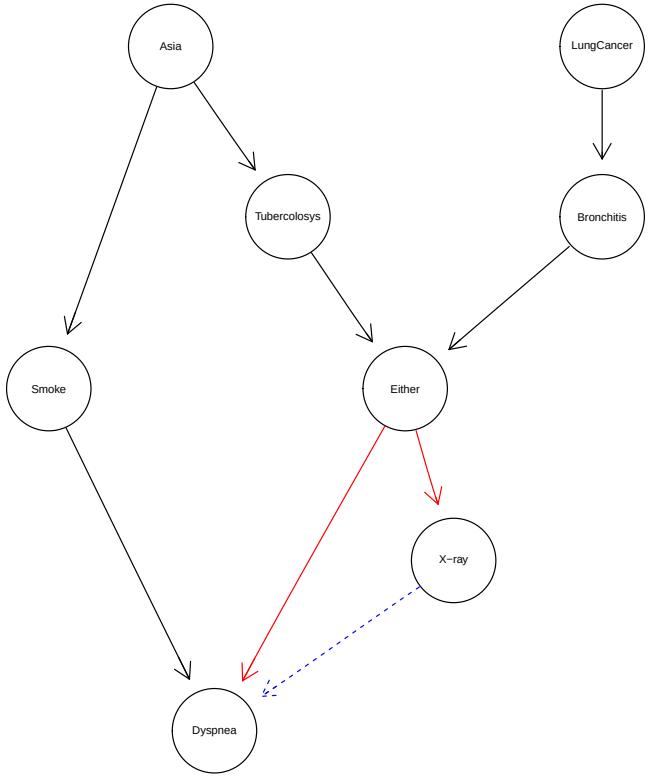

Figure: BN structure learned from the Asia dataset, solution by SANN. Two false positives and one false negative.

Compare with bnstruct: Child Dataset

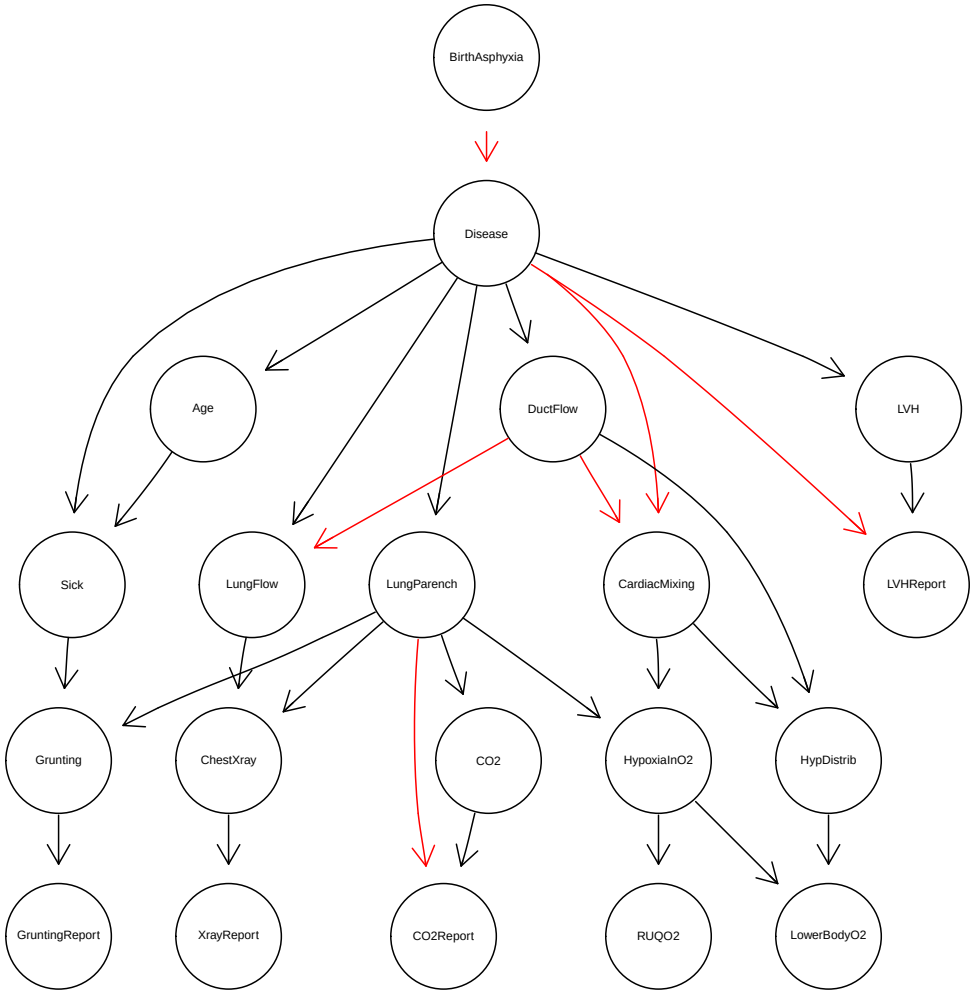

Figure: BN structure learned from the Child dataset. No false negatives and the Disease-LVHReport has opposite direction.

Table of Contents

Conclusions

- The implemented **K2 algorithm** effectively learns BN structures when a correct node order is provided.
- Combined with automatic node ordering, it achieves results **comparable to `bnstruct`**.
- For **small datasets**, **brute-force** search is simple and effective.
- For larger datasets, **SANN** is more scalable but has a **high computational cost** (up to 10^4 evaluations).
- The `learn.network()` function from `bnstruct` (MMHC + BIC) remains **efficient and competitive**.

Future Work

- **Parallelize** the K2 implementation.
- Explore **alternative scoring functions** (e.g., BDeu, AIC).
- Integrate a custom K2 algorithm into `learn.network()`.
- Investigate **alternative order search** strategies (e.g., genetic algorithms).
- Evaluate on additional datasets (e.g. `ALARM`, `Hailfinder`, `Insurance`).

References I

Gregory F. Cooper and Edward Herskovits.
A bayesian method for the induction of probabilistic networks from data.
Machine Learning, 9(4):309–347, 10 1992.

Alberto Franzin and Francesco Sambo.
bnstruct: an r package for bayesian network structure learning with missing data.
`https://cran.r-project.org/web/packages/bnstruct/vignettes/bnstruct.pdf`, 2016.

Alberto Franzin, Francesco Sambo, and Barbara Di Camillo.
bnstruct: an r package for bayesian network structure learning in the presence of missing data.
Bioinformatics, 38(8):1250–1252, 2017.

Marco Scutari.
bnlearn - an r package for bayesian network learning and inference. comparing bayesian network structures.
`https://www.bnlearn.com/examples/compare-dags/`, 2024.

References II

Marco Scutari.
bnlearn - an r package for bayesian network learning and inference. small synthetic networks, for testing purposes.
`https://www.bnlearn.com/documentation/networks/`, 2024.

Worcester Polytechnic Institute.
The k2 algorithm for bayesian network structure learning.
`https://web.cs.wpi.edu/~cs539/s05/Projects/k2_algorithm.pdf`, 2005.
Accessed: 2025-07-01.

Thank you

Bayesian Information Criterion score

Is a network score, so it focus on the Directed Acyclic Graph (DAG) as a whole and provides a statistical measurement of how well the BN structure mirrors the dependence structure of the data.

$$BIC = \log P(X_1, X_2, \ldots, X_n) - \frac{d}{2} \log n \tag{2}$$

with n sample size, d is the number of paramenters of the network, that depends of the number of parents, number of states of the single node and of its parents.

BDeu score

The Bayesian Dirichlet equivalent score it's a scoring metric to evaluate how well the BN structure mirrors the dependence structure of the data.
It computes the **posterior probability** of a BN structure given the data, integrating over all possible parameter values.
It takes a non-informative, uniform prior, the Dirichlet distribution $Dir(\alpha)$ with $\alpha = 1$.

Simulated Annealing

1. From a random generated solution, it computes the score from a defined function, `cold`
2. Generates a random neighbouring solution and it computes its cost, `cnew`
3. Compare:
   - If `cnew` $<$ `cold`, moves to the new solution
   - Else: *maybe* move to the new solution
4. Repeat 1-3 until acceptable solution is found or reached maximum number of iterations.

Step 3 is based on a parameter called *temperature*, that is function of which iteration we are on. At the start the algorithm is more prone to accept worse solutions, the probability follows

$$e^{\frac{c_{new} - c_{old}}{T}}$$

MMHC

The Max-Min Hill-Climbing heuristic ('mmhc') performs a statistical sieving of the search space followed by a greedy evaluation, by combining the MMPC (Min-Max Parent-Children) and the HC (Hill-Climbing) algorithms. As for MMPC, the computational time depends on the density of the network, the number of observations and the tuning of the parameters.

SEM

The Structural Expectation-Maximization (SEM) algorithm learns a network from a dataset with missing values.
It iterates a sequence of Expectation-Maximization (in order to *fill in* the holes in the dataset) and structure learning from the guessed dataset, until convergence.
The structure learning used inside SEM, due to computational reasons, is MMHC.

- Expectation: from current BN structure and parameters estimates missing data
- Maximization and structure search: uses completed data to search for a better BN structure and estimate parameters that maximize the expected Bayesian score.