UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Bayesian Networks in R

A Comparison of the K2 and MMHC search algorithms

Filippo Bezzi, Cristian Tigan

July 14, 2025

Table of Contents

Table of Contents

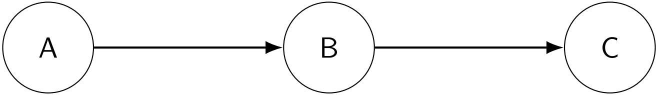
1. Introduction
2. K2 Analysis
3. bnstruct Methods
4. Results

Theory and Motivation 1

Decomposition of joint probability in a product of conditional probabilities:

$$
\begin{aligned}
P(X_1, X_2, \ldots, X_n) &= P(X_1)\, P(X_2, \ldots, X_n \mid X_1) \\
&= P(X_1)\, P(X_2 \mid X_1) \cdots P(X_3, \ldots, X_n \mid X_1, X_2) \\
&= \cdots \\
&= P(X_1)\, P(X_2 \mid X_1) \cdots P(X_n \mid X_1, \ldots, X_{n-1}).
\end{aligned}
$$

Theory and Motivation 2



NODE = *random variable*
EDGE = *conditional dependence* (*direct* relationships)
NO EDGE = *conditional independence* (*indirect* relationships)

$$P(A, B, C) = P(A) \cdot P(B \mid A) \cdot P(C \mid B).$$

Theory and Motivation 3

Definition (Bayesian Network)

A Bayesian Network uses a directed acyclic graph (**DAG**) to build a **probabilistic graphical model** of the joint probability distribution of a set of **random variables** and their **conditional dependencies**[1].

A graph of n nodes is a Bayesian Network representing the variables $X_1, X_2, \ldots, X_n$ if:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \text{Parents}(X_i))$$

where Parents(X_j) denotes the set of all variables X_i , such that there is an arc from node i to node j in the graph.

K2 algorithm - Introduction

Introduction

The K2 algorithm is a **greedy algorithm** (i.e search for the local optimum) that uses a scoring function to compare different parent set configurations and pick the one with the highest score.

- **Input:**
    - The maximum number of parents per node
    - The node ordering
    - The dataset of cases
- **Output:** The most probable set of parents for each node

K2 algorithm - Scoring function

For a node X_i with parents π_i, the score is:

$$g(i,\pi_i) = \prod_{j=1}^{q_i} \frac{(r_i-1)!}{(N_{ij}+r_i-1)!} \prod_{k=1}^{r_i} N_{ijk}!$$

- $r_i =$ number of possible values for X_i
- $q_i =$ number of possible parent configurations
- $N_{ij} =$ number of cases where parents take their j-th configuration
- $N_{ijk} =$ number of cases where $X_i = k$ and parents in j-th configuration

K2 algorithm - Scoring function

We can build it from a contingency table, for example:

Parents:

- Season = {Winter, Summer}
- Windy = {yes, no}

Child:

- Weather = {Sunny, Rainy}

j	**Parent Config**	**k = 1 (N_{ijk})**	**k = 2 (N_{ijk})**	N_{ij}
		Weather = Sunny	**Weather = Rainy**	
1	(Winter, yes)	3	1	4
2	(Winter, no)	2	4	6
3	(Summer, yes)	5	2	7
4	(Summer, no)	6	1	7

Figure: Example of contingency table for Weather variable with Season and binary parent variable

K2 algorithm - Logarithmic scale

For larger datasets, the results from the scoring function can become really high really quickly. To better handle the output for R, we chose to rewrite the scoring function in the logarithmic scale, such that the product becomes a sum.

$$\begin{aligned}
\log(f_j) &= \log\left[\frac{(r_i-1)!\prod_{k=1}^{r_i}\alpha_{ijk}!}{(N_{ij}+r_i-1)!}\right] = \\
&= \log\left[(r_i-1)!\right] + \log\left[\prod_{k=1}^{r_i}\alpha_{ijk}!\right] - \log\left[(N_{ij}+r_i-1)!\right] = \\
&= \sum_{x=1}^{r_i-1}\log(x) + \sum_{k=1}^{r_i}\sum_{y=1}^{\alpha_{ijk}}\log(y) - \sum_{z=1}^{N_{ij}+r_i-1}\log(z) = \\
&= A + B - C
\end{aligned}$$

Figure: The log-form scoring function

Greedy Hill-Climbing Algorithm

```r
for (i in seq_along(nodes)) {
  node <- nodes[i]
  prev.nodes <- if(i==1) character(0) else nodes[1:(i-1)]
  parents <- character(0)
  P_old <- f(node, parents, dataset)
  OK <- TRUE
  score.log[[node]] <- list()

  while (OK && length(parents) < u && length(prev.nodes) > 0) {
    scores <- sapply(prev.nodes, function(p)
                     f(node, c(parents, p), dataset))
    score.log[[node]][[paste(sort(parents), collapse=",")]] <-
      setNames(as.list(scores), prev.nodes)
    z <- prev.nodes[which.max(scores)]
    P_new <- max(scores)

    if (P_new > P_old && z != node) {
      P_old <- P_new
      parents <- c(parents, z)
      prev.nodes <- setdiff(prev.nodes, z)
      net.dag <- set.arc(net.dag, z, node)
    } else {
      OK <- FALSE
    }
  }
  parent.list[[node]] <- parents
}
return(list(dag = net.dag, parents = parent.list, scores = score.log))
```

Table of Contents

Università degli Studi di Padova

1. Introduction

2. K2 Analysis

3. bnstruct Methods

4. Results

Using Mutual Information to obtain node ordering

We want to define node ordering using **Mutual Information** between nodes and **condition independence tests**. [2]

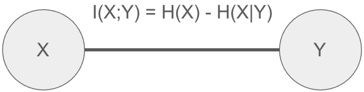


Figure: Mutual Information between two random variables

- $H(X)$ is the **Shannon Entropy** on the variable X.
- $H(X|Y)$ is the conditional entropy, i.e. how much uncertainty there is in X given Y.

Step 1 Construct an undirected network

The algorithm is divided in **four parts**.

1. Creates an **UDN** using **Mutual Information** between variables using the `infotheo` library.

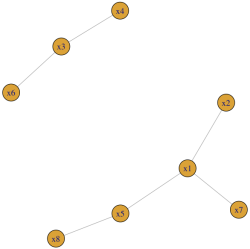


Figure: Undirected network of random variables. Edges are created when $I(X; Y) \geq \alpha \times MMI(X)$

Step 2 Achieve complete connectivity

The algorithm is divided in **four parts**.

2. Leverages the adjacency matrix proprieties to ensure **complete connectivity** of the UDN.

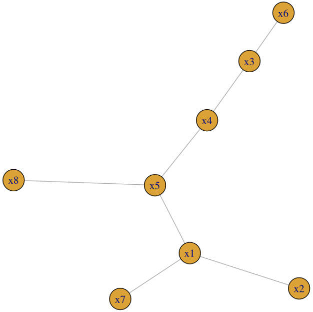


Figure: The undirected network now is fully connected.

Step 3 Refine the graph

The algorithm is divided in **four parts**.

3. **Removes** possible false edges and add true edges using conditions independence test.

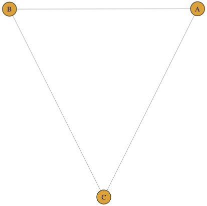


Figure: Independence test on triangles. $P(A \mid C) = P(A \mid B, C)$ $P(A \mid B) = P(A \mid C, B)$ $P(B \mid A) = P(B \mid C, A)$

Step 4 Add orientation to the network

The algorithm is divided in **four parts**.

4. **Adds** orientation to the network using again condition indipendence tests

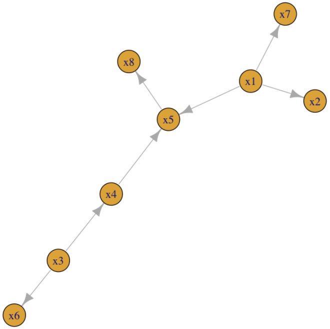


Figure: The network becomes directed

The final node ordering

Node ordering

The topological order of the directed network is the node order to give to the K2 algorithm.

Benchmark

Dataset	Time
asia	0.676s
sachs	2.900s
child	4.950s

Table: Total computational time of getting the node order and compute the K2 algorithm per dataset.

Table of Contents

Università degli Studi di Padova

1. Introduction

2. K2 Analysis

3. bnstruct Methods

4. Results

Implementation: The Data

We use four different datasets:

1. **Ruiz Dataset**[3]:
   - dummy dataset of 0s and 1s
   - 3 variables (x_1, x_2, x_3) and 10 items per variable.
2. **Asia Dataset**[4]:
   - data about lung diseases (tuberculosis, lung cancer or bronchitis) and visits to Asia
   - loaded from `bnstruct`
   - 8 variables and 10000 items per variable
3. **Sachs Dataset**[5]:
   - biological data
   - 11 variables and 5000 items per variable
4. **Child Dataset**[4]:
   - data containing a set of health conditions and symptoms
   - 20 variables and 5000 items per variable
   - loaded from `bnstruct`
   - contains random missing values

Structure Learning with bnstruct 1

The bnstruct[4] package is built for Bayesian Network Structure Learning.

We use two main classes of objects:

- BNDataset-class objects contain all the data required information relevant to learning its network structure
- BN-class objects construct different Bayesian Networks depending on a variety of parameters.

Structure Learning with bnstruct 2

```
# Define the BNDataset object
ruiz.bnd <- BNDataset(
  data = ruiz.df,
  discreteness = rep("d", 3),
  variables = c("x1", "x2", "x3"),
  node.sizes = rep(2, 3)
)
```

Structure Learning with bnstruct 3

Università degli Studi di Padova

```
# Learn the Bayesian network structure
ruiz.bns <- learn.network(
  dataset = ruiz.bnd,
  algo = "mmhc",
  scoring.func = "BDeu",
  layering = ruiz.order,
  max.parents = num.variables(ruiz.bnd) - 1
)
```

The Max-Min Hill-Climbing (mmhc[4]) heuristic search algorithm performs a **statistical sieving** of the **search space** followed by a **greedy evaluation**.
The BDeu[4] scoring function assumes a **flat prior**.

Search Algorithms Comparison: Visualization Methods

- For visualization and metric computation we used the `bnlearn`[6] package
- `Rgraphviz` for visualization of network structures. The `graphviz.compare()` method conveniently takes two `bn-class` objects and highlights:
  - True positive arcs
  - False positive arcs
  - False negative arcs

Search Algorithms Comparison: Metrics 1

- As **statistical score**:
  - the *Bayesian Dirichlet equivalent uniform* (BDeu) score[7].
- As **structural metrics**:
  - the Structural Hamming Distance (SHD)
  - the Edge Counts (TP, FP, FN)
  - **Precision / Recall / $\mathbf{F}_1$**

Search Algorithms Comparison: Metrics 2

Structural Hamming Distance (SHD)[7] The SHD between two DAGs (G_1, G_2) is

$$\mathrm{SHD}(G_1, G_2) = \left|\{(i,j) \,:\, A_1(i,j) \neq A_2(i,j)\}\right|,$$

that is, the distance between two adjacency matrices in terms of edges.

- The lower the SHD, the more similar the two network structures.
- The `shd()`[6] method evaluates this metric conveniently by matching nodes automatically.

Search Algorithms Comparison: Metrics 3

Università degli Studi di Padova

Edge Counts & Precision/Recall

Be $E_1, \quad E_2$ the arc sets belonging to two network structures. Then:

$$\mathrm{TP} = |E_1 \cap E_2|, \quad \mathrm{FP} = |E_2 \setminus E_1|, \quad \mathrm{FN} = |E_1 \setminus E_2|.$$

- The `compare()`[6] method prints a table with TP, FP, FN arcs.
- Precision, Recall and F1 Score:
  - Precision $\left(= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}\right)$
  - Recall $\left(= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}\right)$
  - F1 Score $\left(= 2\frac{PR}{P+R}\right)$

Table of Contents

1. Introduction

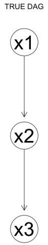
2. K2 Analysis

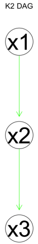
3. bnstruct Methods

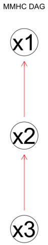
4. Results

Search Algorithms Comparison: K2 vs `bnstruct` MMHC 1

Ruiz Dataset



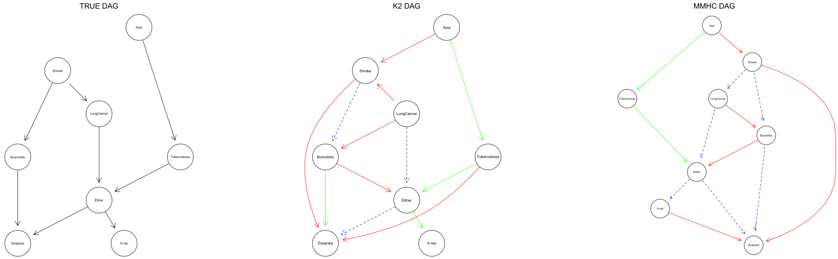




Method	TP	FP	FN
K2	2	0	0
MMHC	0	2	2

Search Algorithms Comparison: K2 vs bnstruct MMHC 2

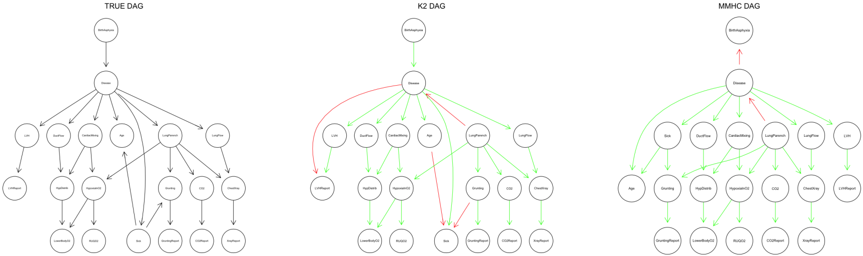
Asia Dataset



Method	TP	FP	FN
K2	4	6	4
MMHC	2	5	6

Search Algorithms Comparison: K2 vs bnstruct MMHC 3

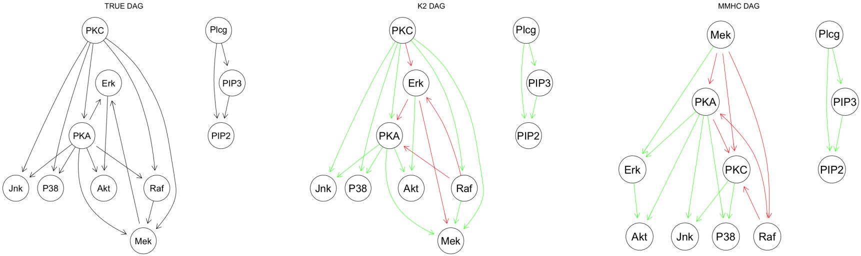
Child Dataset



Method	TP	FP	FN
K2	22	4	3
MMHC	23	2	2

Search Algorithms Comparison: K2 vs `bnstruct` MMHC 4

Sachs Dataset



Method	TP	FP	FN
K2	14	5	3
MMHC	11	6	6

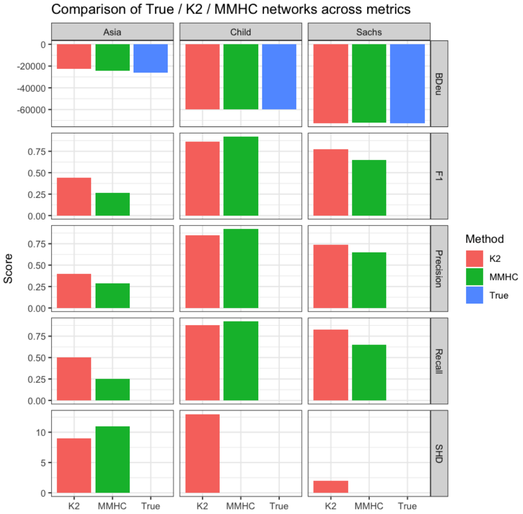
Search Algorithms Comparison: K2 vs bnstruct MMHC 5

Table: Summary of BDeu, SHD, and Precision/Recall/F1 Scores

Dataset	Method	BDeu	SHD	TP	FP	FN	Precision	Recall	F1
Ruiz	True	-19.653	NA	NA	NA	NA	NA	NA	NA
Ruiz	K2	-19.653	0	2	0	0	1.00	1.00	1.00
Ruiz	MMHC	-19.653	0	0	2	2	0.00	0.00	NA
Child	True	-59676	NA	NA	NA	NA	NA	NA	NA
Child	K2	-59762	13	22	4	3	0.85	0.88	0.86
Child	MMHC	-59676	0	23	2	2	0.92	0.92	0.92
Asia	True	-26093	NA	NA	NA	NA	NA	NA	NA
Asia	K2	-22517	9	4	6	4	0.40	0.50	0.44
Asia	MMHC	-24244	11	2	5	6	0.29	0.25	0.27
Sachs	True	-72448	NA	NA	NA	NA	NA	NA	NA
Sachs	K2	-72362	2	14	5	3	0.74	0.82	0.78
Sachs	MMHC	-72129	0	11	6	6	0.65	0.65	0.65

Search Algorithms Comparison: K2 vs `bnstruct` MMHC 6

Università degli Studi di Padova



- Similar BDeu score
- Manual K2 performs better on the Sachs dataset
- `bnstruct` MMHC performs better on Child dataset
- General poor performance on the Asia dataset

References 1

[1] G. F. Cooper and E. Herskovits. “A Bayesian Method for the Induction of Probabilistic Networks from Data”. In: *Machine Learning* 9 (1992), p. 309.

[2] Xue-Wen Chen, Gopalakrishna Anantha, and Xiaotong Lin. “Improving Bayesian Network Structure Learning with Mutual Information-Based Node Ordering in the K2 Algorithm”. In: *IEEE Transactions on Knowledge and Data Engineering* 20.5 (2008), pp. 628–640.

[3] C. Ruiz. *Illustration of the K2 Algorithm for learning Bayes Net Structures*. `http://web.cs.wpi.edu/~cs539/s11/Projects/k2_algorithm.pdf`. 2011.

[4] Alberto Franzin, Francesco Sambo, and Barbara Di Camillo. “bnstruct: an R package for Bayesian Network structure learning in the presence of missing data”. In: *Bioinformatics* 33.8 (2017), pp. 1250–1252.

References 2

[5] *Sachs Protein Dataset*. `https://paperswithcode.com/dataset/sachs`. Accessed: 2025-07. 2025.

[6] Marco Scutari. *bnlearn: Bayesian Network Structure Learning, Parameter Learning and Inference*. R package version 5.0.2. 2025. URL: `https://www.bnlearn.com/`.

[7] Marco Scutari and Jean-Baptiste Denis. *Bayesian Networks: With Examples in R*. Boca Raton, FL: CRC Press, 2021. ISBN: 978-0367136774.