A New Method for Vertical Parallelisation of TAN Learning Based on Balanced Incomplete Block Designs

Anders L. Madsen[12] Frank Jensen[1] Antonio Salmerón[3] Martin Karlsen[1] Helge Langseth[4] Thomas D. Nielsen[2]

HUGIN EXPERT A/S, Aalborg, Denmark

Department of Computer Science, Aalborg University, Denmark

Department of Mathematics, University of Almería , Spain

Department of Computer and Information Science, Norwegian University of Science and Technology, Norway

Outline

Introduction

A Bayesian network (BN) is an excellent knowledge representation, but learning a BN from data can be a challenge

- Data sets are increasing in size w.r.t. both variables and cases
- The computational power of computers is increasing

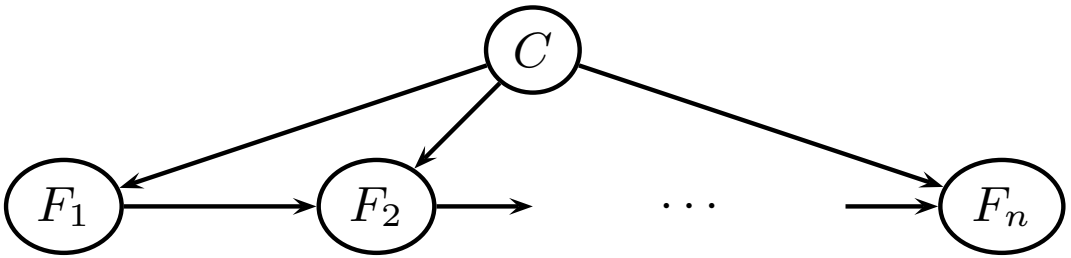
Research Problem

Learning the structure of a Tree-Augmented Naive Bayes model from massive amounts of data in parallel

This work is performed as part of the AMIDST project (no. 619209)

Learning a TAN model

Let $\mathcal{F}$ be a set of features and C be the class variable



The structure of a TAN can be constructed as ([11] and [3]):

1. Compute mutual information $I(F_i, F_j \,|\, C)$ for each pair, $i \neq j$.
2. Build a complete graph G over $\mathcal{F}$ with edges annotated by $I(F_i, F_j \,|\, C)$.
3. Build a maximal spanning tree T from G.
4. Select a vertex and recursively direct edges outward from it.
5. Add C as parent of each $F \in \mathcal{F}$.

Parallel TAN learning

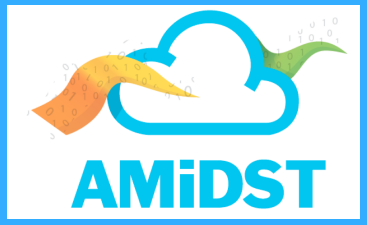
- Step 1 (scoring all pairs w.r.t. MI) is the most time-consuming element
- Both horizontal and **vertical** parallelization of learning is possible
- Data should be distributed, we assume each variable has its own file

Challenge

Given a set of features $\mathcal{F}$ we need to distribute MI-computations such that each pair is scored exactly once

We use Balanced Incomplete Block (BIB) Designs to achieve vertical parallelisation

Balanced Incomplete Block (BIB) Designs[23]



Definition (Design)

A *design* is a pair $(X, \mathcal{A})$ s. t. the following properties are satisfied:

1. X is a set of elements called *points*, and
2. $\mathcal{A}$ is a collection of nonempty subsets of X called *blocks*.

Definition (BIB design)

Let v, k and λ be positive integers s. t. $v > k \geq 2$. A (v, k, λ)-BIB design is a design $(X, \mathcal{A})$ s. t. the following properties are satisfied:

1. $|X| = \mathrm{v}$,
2. each block contains exactly k points, and
3. every pair of distinct points is contained in exactly λ blocks.

If $v = b$ where $b = |\mathcal{A}|$, then the design is symmetric

BIB Design Intuition

Let $\mathcal{F}$ be the features with $|\mathcal{F}| = 7$ and let $(v = 7, k = 3, \lambda = 1)$ be a symmetric BIB design, i.e., $v = b$, with blocks

$$\{013\}, \{124\}, \{235\}, \{346\}, \{450\}, \{561\}, \{602\}.$$

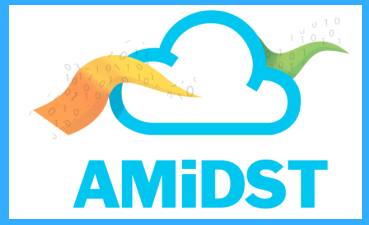
We observe

- $\lambda = 1$ - we compute mutual information exactly once
- a symmetric BIB design, i.e., $v = b$
- $k = 3$ is the number of (subset) of features in each block

We take the following approach

- create a process for each block to compute mutual information
- a point p represents $\mathcal{F}_p$ when $|\mathcal{F}| > v$.
- each process generates its block based on its rank

Difference Sets for Symmetric BIB designs



BIB design	Difference set	k/v
(3,2,1)	(0,1)	0.67
(7,3,1)	(0,1,3)	0.43
(13,4,1)	(0,1,3,9)	0.31
(21,5,1)	(0,1,4,14,16)	0.24
(31,6,1)	(0,1,3,8,12,18)	0.19
(57,8,1)	(0,1,3,13,32,36,43,52)	0.14
(73,9,1)	(0,1,3,7,15,31,36,54,63)	0.12
(91,10,1)	(0,1,3,9,27,49,56,61,77,81)	0.11
(133,12,1)	(0,9,10,12,26,30,67,74,82,109,114,120)	0.09
(183,14,1)	(0,12,19,20,22,43,60,71,76,85,89,115,121,168)	0.08

Difference sets for (v, k, λ)-BIB designs where v is the number of points and k is the number of points in a block

BIB Design and TAN Learning

All processes should compute the same number of $I(F_i, F_j \mid C)$ scores

- There is $\binom{n}{2} = n(n-1)/2$ pairs where $n = |\mathcal{F}|$
- We assume each variable is in one file

If we have p processes and each process reads data from k variables, then the following must be satisfied

$$p\binom{k}{2} \geq \binom{n}{2}.$$

Solving for k produces $k \geq n/\sqrt{p}$, and equality holds only when $p = 1$.

BIB Design Example

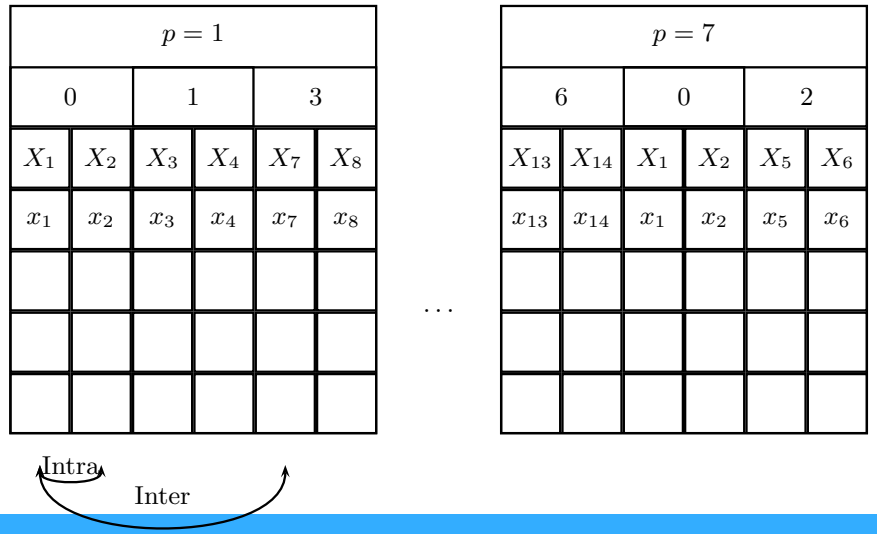
Consider the $(7, 3, 1)$-BIB design and assume $|\mathcal{F}| = 14$

- Each point represents two features
- Each process is assigned six features

The seven blocks ($b = 7$) are:

$$\{013\}, \{124\}, \{235\}, \{346\}, \{450\}, \{561\}, \{602\}$$

The pairwise scoring is performed as

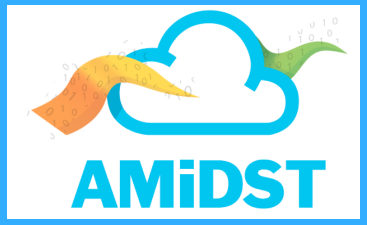


Experimental Setup

- Software implementation based on HUGIN software and using Message Passing Interface (MPI)
- Client-server architecture; a master and worker processes
- Average run time (wall-clock) is computed over ten runs of the algorithm on each dataset with specific number of processes
- Clock starts before reading data and ends after all results are received by master process

data set	$\lvert\mathcal{X}\rvert$	N
Munin1	189	750,000
Munin2	1,003	750,000
Bank	1,823	1,140,000

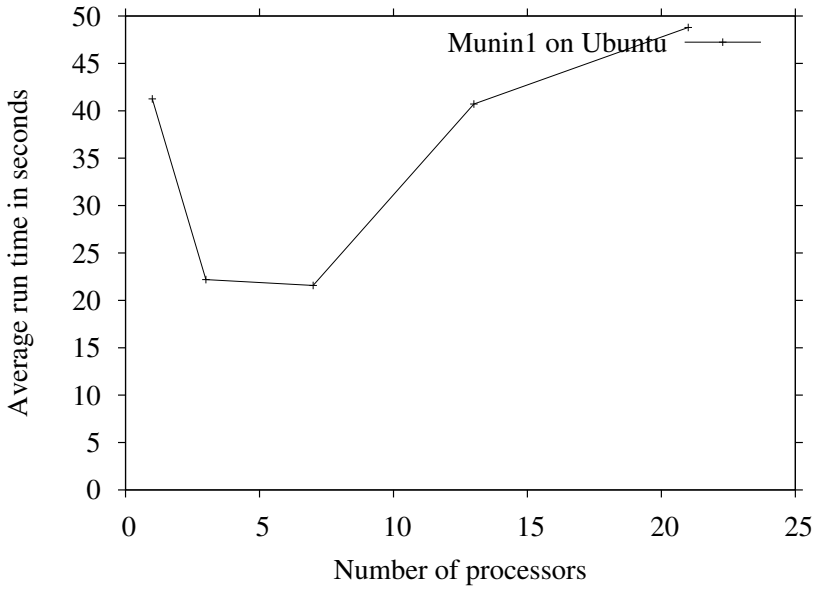
Three Different Test Computers

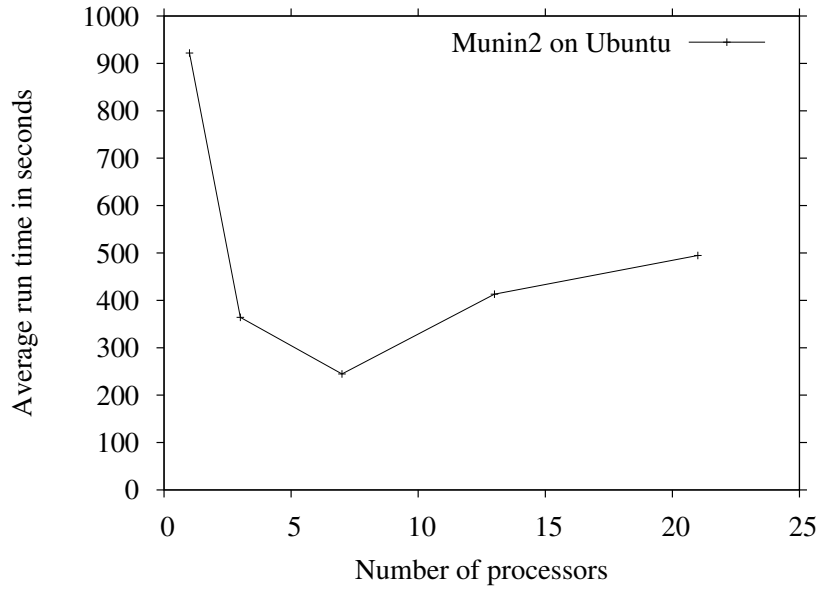


1. Ubuntu: a linux server running - 1 Intel Xeon(TM) E3-1270 Processor (4 cores) and 32 GB RAM
2. Fyrkat: cluster with 80 nodes - each has 2 Intel Xeon (TM) X5260 Processors (2 cores) and 16GB RAM
3. Vilje: cluster with 1404 nodes - each has 2 Intel Xeon E5-2670 Processors (8 cores) and 32GB

Test computers have different resource management systems
For 2. and 3. we allocate one worker node for each process

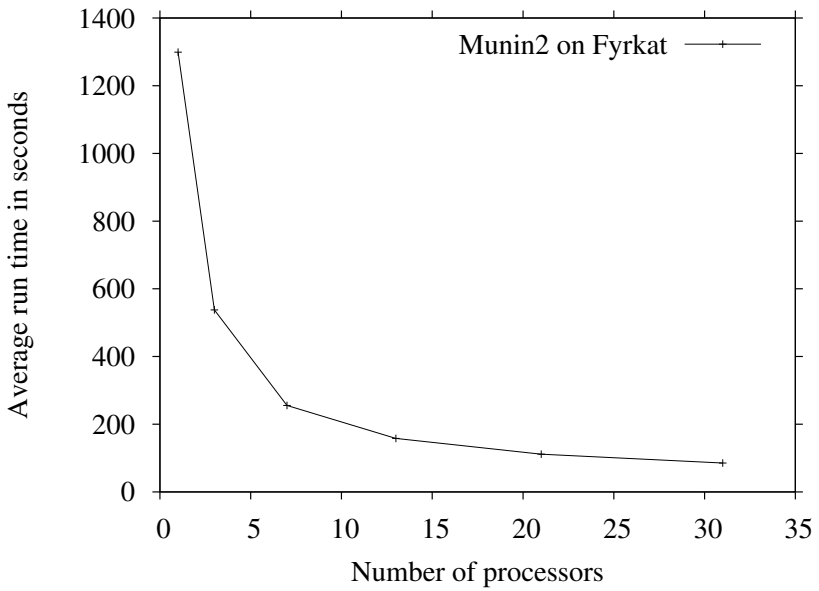
Ubuntu - Linux Server One Node

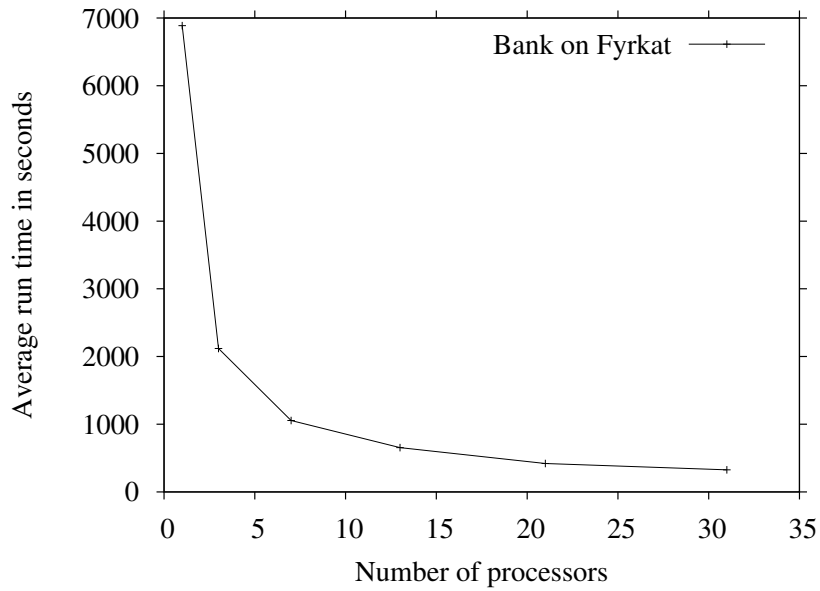




Ubuntu: A linux server running Ubuntu - Intel Xeon(TM) E3-1270 Processor (4 cores) and 32 GB RAM

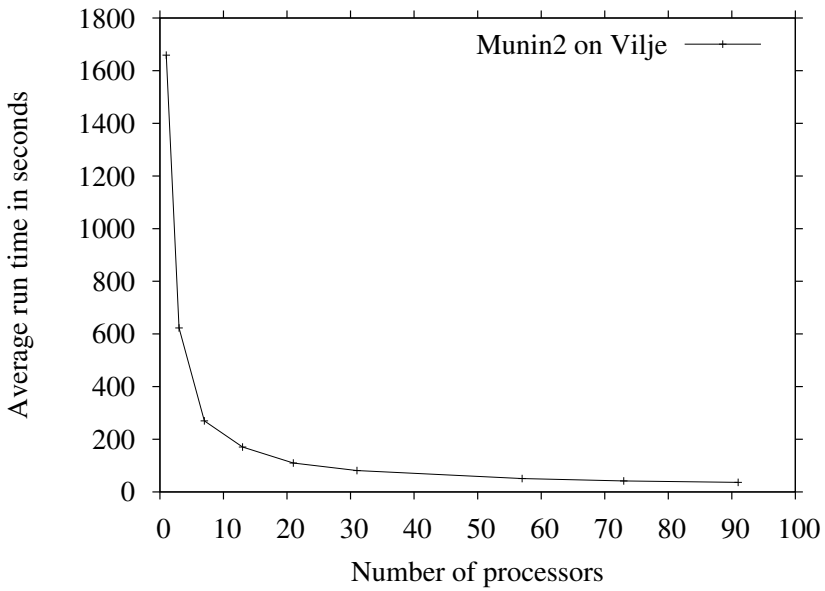
Fyrkat - Cluster 80 Nodes

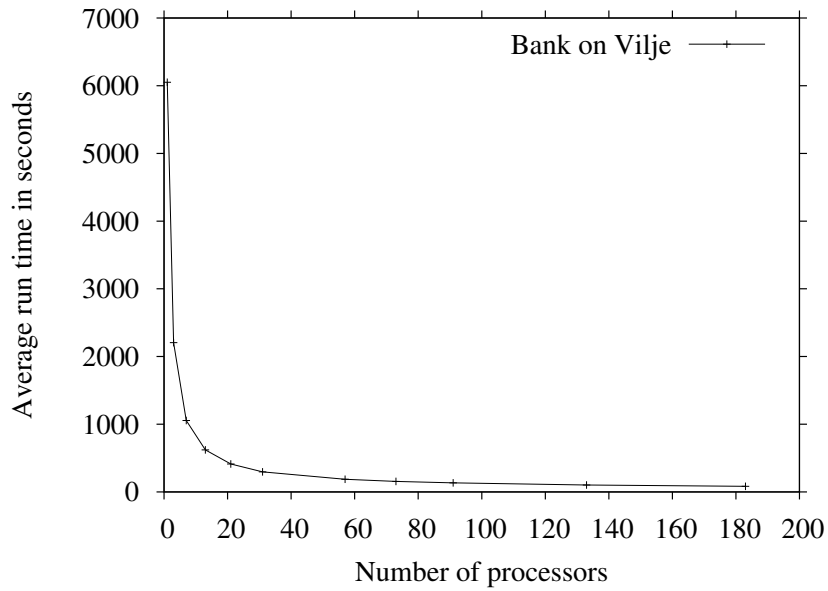




Fyrkat: 80 nodes - each has 2 Intel Xeon (TM) X5260 Processors (2 cores) and 16GB RAM

Vilje - Cluster 1404 Nodes

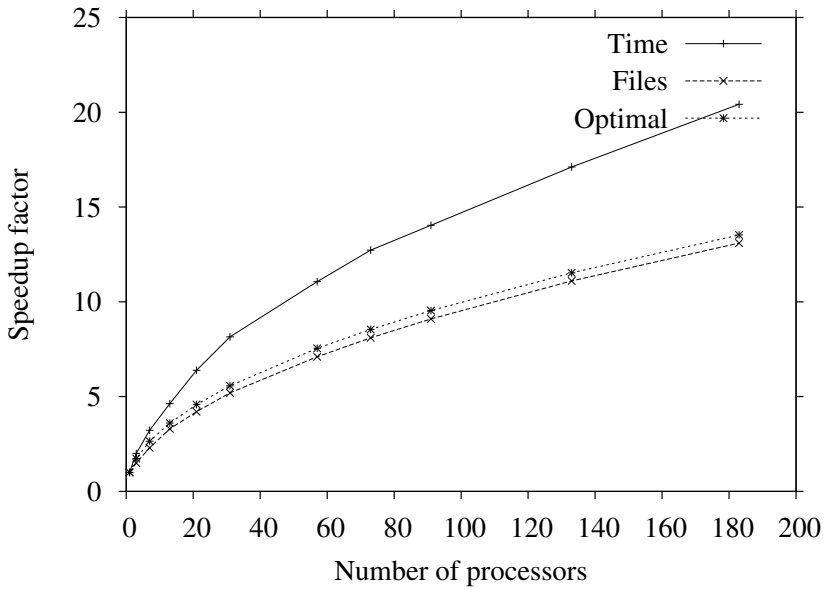


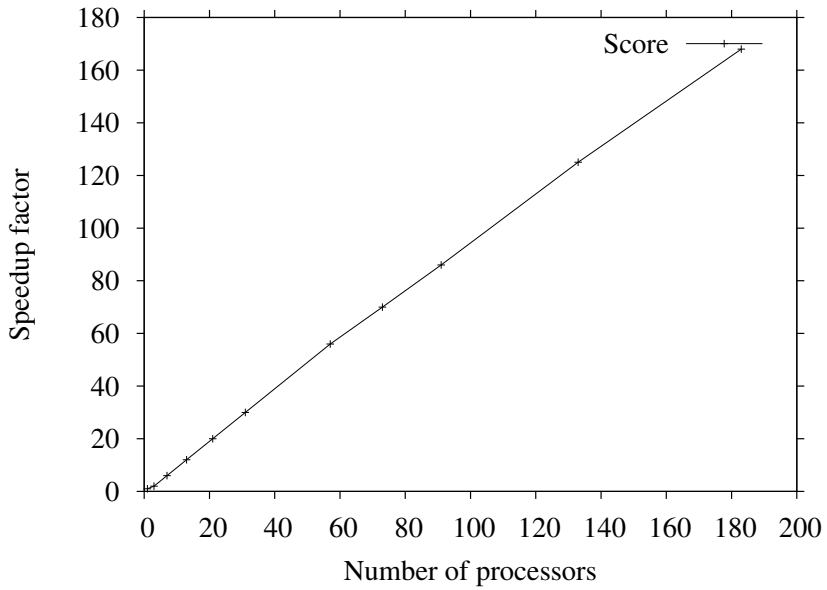


Vilje: 1404 nodes - each has 2 Intel Xeon E5-2670 Processors (8 cores) and 32GB

Speedup Factors - Vilje & Bank

Relative performance compared to case of using one processor





Time speed-up factors for reading data

Files *theoretical* number of files read by each process ($k/v \cdot |\mathcal{X}|$)

Optimal square root of the number of processors

Score speed-up factor for computing mutual information

Conclusion & Future Work

Conclusion

- Introduced a new algorithm for learning TAN structure using parallel computing
- Empirical evaluation shows that significant performance improvements are possible

Future Work

- Apply principles to structure learning in general <> TAN
- Multithread implementation <> Multiprocess
- Horizontal parallelization <> Vertical parallelization

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 619209