Parallelization of the PC Algorithm

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

[1]Hugin Expert A/S, Aalborg, Denmark

[2]Dept. Computer Science, Aalborg University, Denmark

[3]Dept. Mathematics, University of Almería, Spain

[4]Dept. Computer and Information Science. Norwegian University of Science and Technology, Trondheim, Norway

Introduction

- The AMiDST project: Analysis of Masslve Data STreams
  http://www.amidst.eu

Introduction

- The AMiDST project: Analysis of Masslve Data STreams
  http://www.amidst.eu
- Large number of variables
- Massive datasets
- Hybrid Bayesian networks (involving discrete and continuous variables)
  - Conditional linear Gaussian networks

Introduction

- The AMiDST project: Analysis of Masslve Data STreams
  http://www.amidst.eu
- Large number of variables
- Massive datasets
- Hybrid Bayesian networks (involving discrete and continuous variables)
  - Conditional linear Gaussian networks

Objectives

- Scale up the PC algorithm for learning CLG networks from large volumes of data.
- Take advantage of parallel computing environments with shared memory.

The PC algorithm

1. Determine pairwise (conditional) independence $I(X, Y; \mathcal{S})$.
2. Identify skeleton of G.
3. Identify v-structures in G.
4. Identify derived directions in G.
5. Complete orientation of G making it a DAG.

The PC algorithm

1. Determine pairwise (conditional) independence $I(X, Y; \mathcal{S})$.
2. Identify skeleton of G.
3. Identify v-structures in G.
4. Identify derived directions in G.
5. Complete orientation of G making it a DAG.

Remarks

- Step 1 takes most of the computing time
- Marginal independence ($\mathcal{S} = \emptyset$) is tested first
- Only potential neighbours are included in the conditioning set

Our proposal for parallelisation

We propose to parallelise Step 1 (pairwise c.i. tests)

1. Test all pairs X and Y for marginal independence.
   - Use BIB designs
2. Perform the most promising higher-order c.i. tests.
   - We create an edge index array, which the threads iterate over to select the next edge to evaluate for each iteration.
   - The edge index array contains all edges that has not been removed at an earlier step and it is sorted in decreasing order of the test score
   - Tests of size $|\mathcal{S}| = 1, 2, 3$ may be performed.
3. Remaining tests of conditional independence $(X, Y; \mathcal{S})$ where $|\mathcal{S}| = 1, 2, 3$.

Balanced Incomplete Block (BIB) designs

- It is a concept coming from statistical design of experiments that provides a way of arranging experimental units when testing the effectiveness of a treatment

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.

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.

BIB Design Example

Consider the $(7, 3, 1)$-BIB design for 14 variables

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

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

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

The pairwise scoring is performed as

$p = 1$					
0		1		3	
X_1	X_2	X_3	X_4	X_7	X_8
x_1	x_2	x_3	x_4	x_7	x_8

Intra

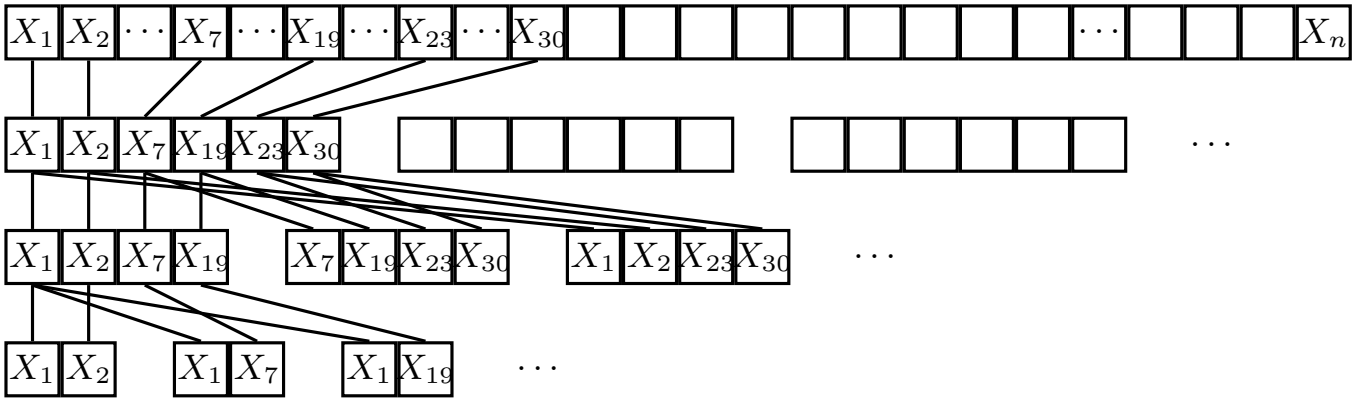
Inter

...

$p = 7$					
6		0		2	
X_{13}	X_{14}	X_1	X_2	X_5	X_6
x_{13}	x_{14}	x_1	x_2	x_5	x_6

Balanced Incomplete Block (BIB) designs

- The testing is divided into tasks of equal size such that we test exactly all pairs X, Y for marginal independence
- This is achieved using BIB designs on the form $(q, 6, 1)$ and then $(3, 2, 1)$ where q is at least the number of variables



Extra heuristics

- For each edge, we compute the set of most promising tests
  - For each edge (X, Y) the set of best candidate variables to include in $\mathcal{S}$ are identified using the weight of a candidate variable Z which is equal to the sum of the test scores for (X, Z) and (Y, Z):

$$w(Z \,|\, (X, Y)) = 2N(\mathrm{MI}(Z, X) + \mathrm{MI}(Z, Y))$$

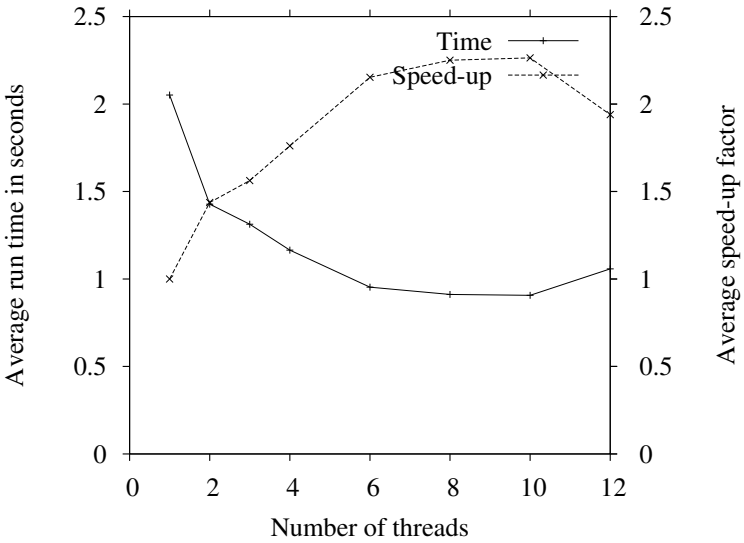
    where $\mathrm{MI}(\cdot, \cdot)$ is the mutual information.
- We create an array of best candidates with ≤ 7 vars (counts stored in memory) sorted by the sum of the edge weights
- The threads iterate over the edge index array. A thread performs all tests for a selected edge (with $|\mathcal{S}| = 1, 2, 3$) from the best candidate array. Testing stops as soon as an independence hypothesis is not rejected

Empirical evaluation

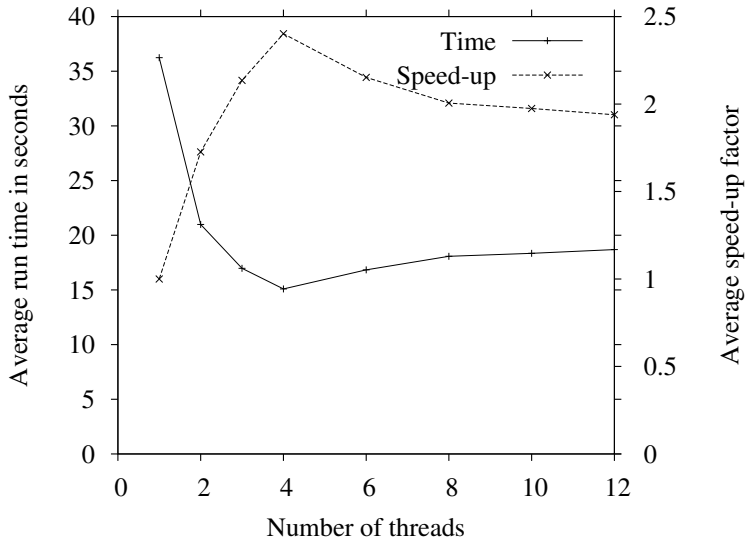
data set	$\lvert\mathcal{X}\rvert$	Total CPT size
ship-ship	50	130,478
Munin1	189	19,466
Diabetes	413	461,069
Munin2	1,003	83,920
sacso	2,371	44,274

- Software implementation based on HUGIN software
- Three data sets generated at random for each network with 100,000, 250,000, and 500,000 cases
- The empirical evaluation is performed on a Linux computer running Red Hat Enterprise Linux 7 with a six-core Intel (TM) i7-5820K 3.3GHz processor and 64 GB RAM
- The computer has 6 physical cores and 12 logical cores

Empirical evaluation

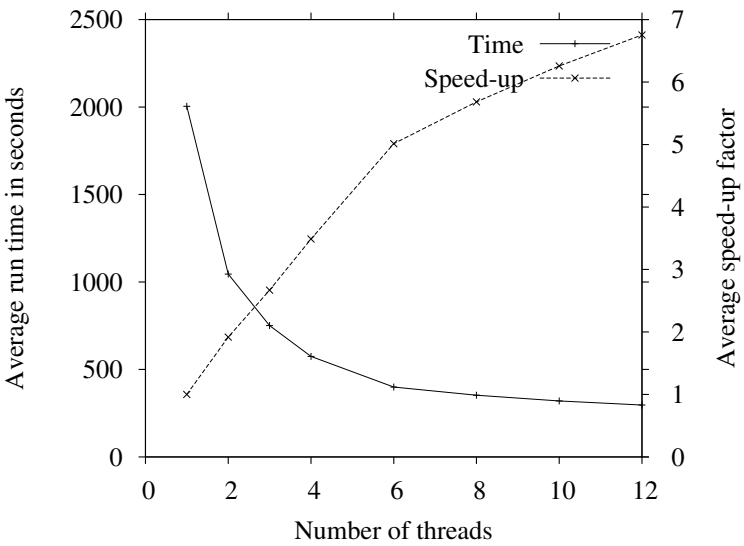


(a) ship-ship 500,000

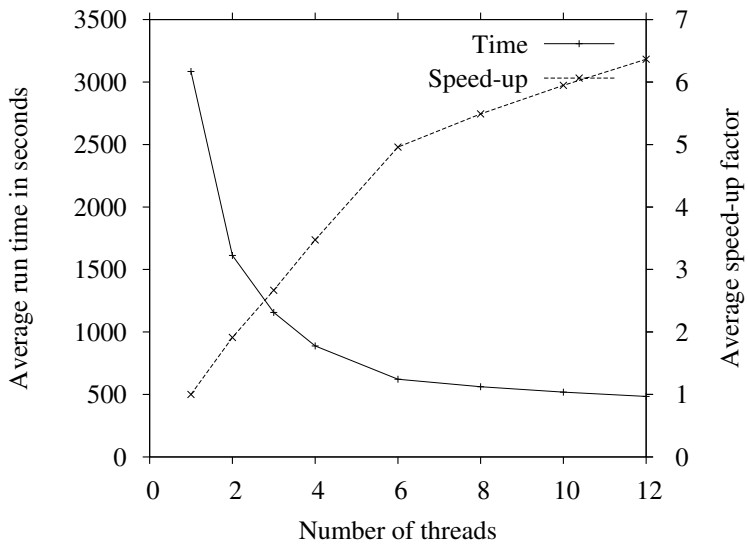


(b) Munin1 250,000

Empirical evaluation

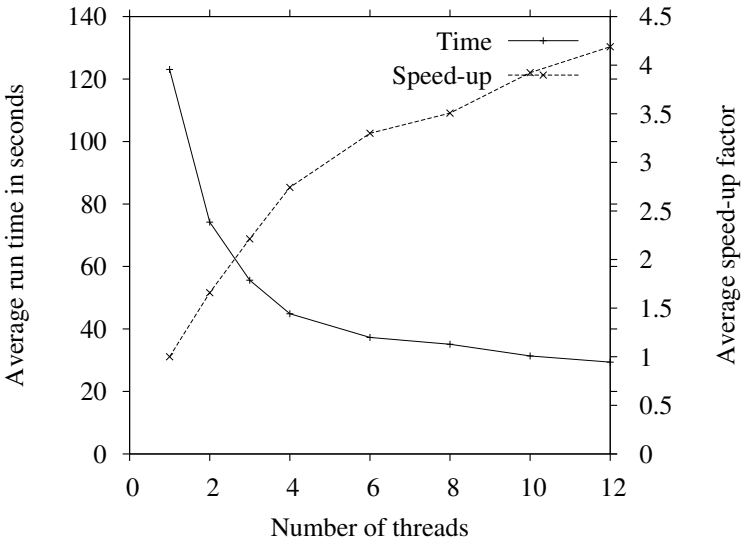


(c) Diabetes 250,000

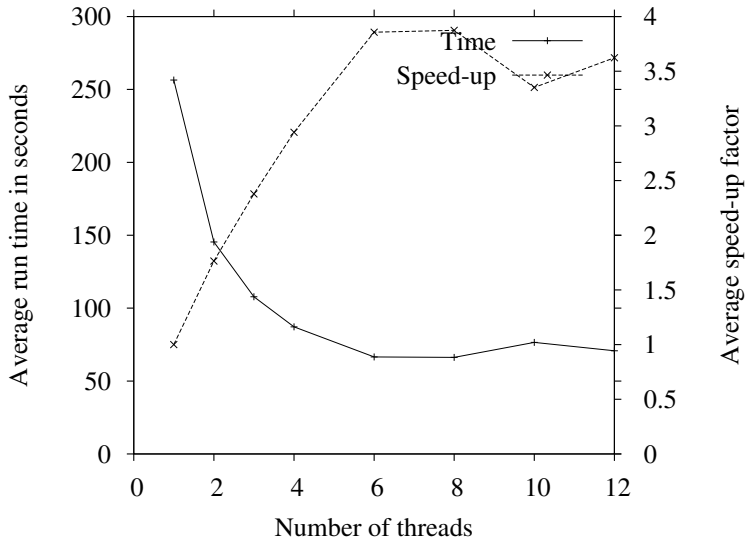


(d) Diabetes 500,000

Empirical evaluation

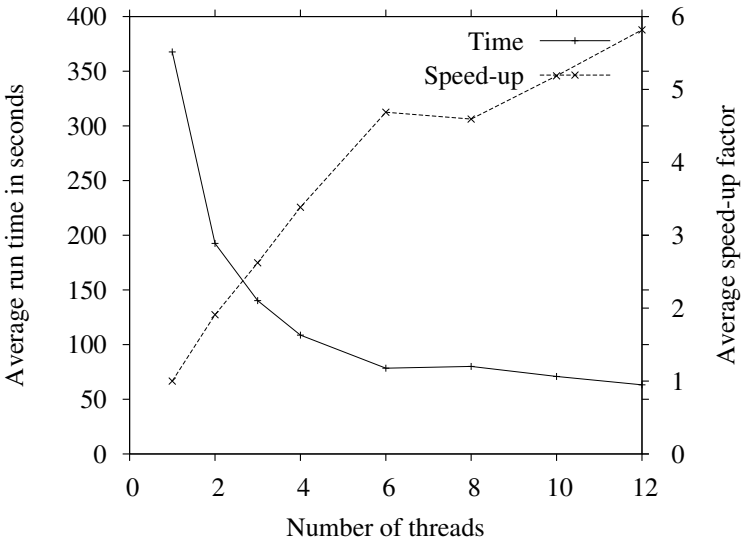


(e) Munin2 250,000

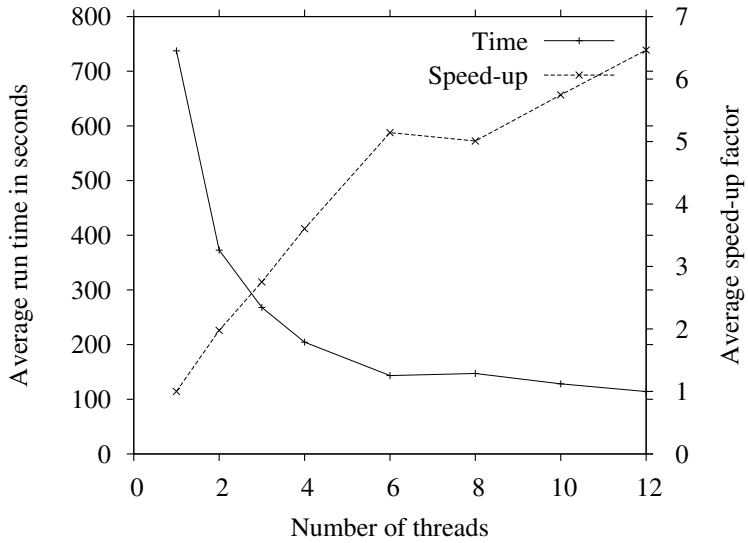


(f) Munin2 500,000

Empirical evaluation



(g) sacso 250,000



(h) sacso 500,000

Empirical evaluation

Data set	Skeleton (Step 2)	*v*-structures (Step 3)	Orientation (Steps 4 and 5)
ship-ship	0	0	0
Munin1	0.005	0	0.001
Diabetes	0.001	0.004	0.002
Munin2	0.006	0.002	0.034
sacso	0.051	5.692	0.502

Conclusions

- Parallelisation of structure learning using the PC algorithm
- The edge index array is the central bottleneck of the approach as it is the only element that requires synchronization
- The number of threads used by the algorithm may impact the result as the order of tests is not invariant under the number of threads used. This is a topic of future research.
- The results of the empirical evaluation show a significant time performance improvement over the pure sequential method.

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