Sub-quadratic Markov tree mixture models for probability density estimation

Sourour Ammar[1], Ph. Leray[1], L. Wehenkel[2]

[1] Equipe COnnaissances et Décision, LINA UMR 6241
Ecole Polytechnique de l'Université de Nantes
[2] Department of EECS and GIGA-Research, University of Liège

COMPSTAT'2010 - Paris - 22-27 August 2010

A simple idea

Proposition

Develop density estimation techniques that could scale to very high-dimensional spaces, by exploiting the **Perturb & Combine** *idea with probabilistic graphical models.*

Outline

- Background
- Our proposal
- Some results
- Conclusions and Further works

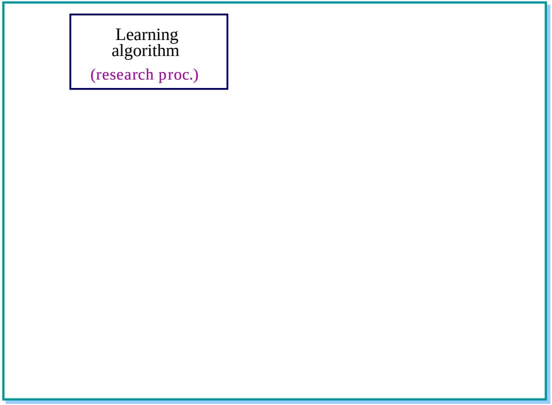
P&C principle in supervised learning

- Principle : (Bagging, Random forests, Extremely randomized trees)

How can we apply this idea to density estimation with Bayesian networks (BN)?

P&C principle in supervised learning

- Principle : (Bagging, Random forests, Extremely randomized trees)



How can we apply this idea to density estimation with Bayesian networks (BN)?

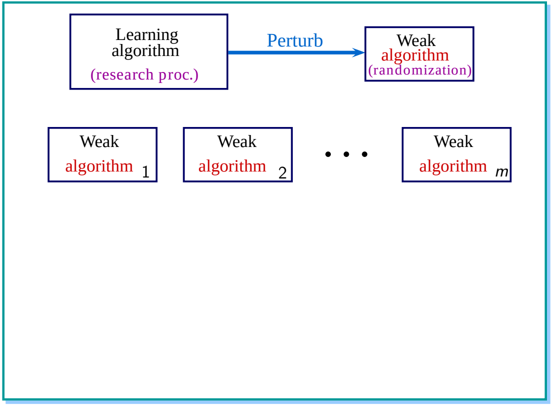
P&C principle in supervised learning

- Principle : (Bagging, Random forests, Extremely randomized trees)

How can we apply this idea to density estimation with Bayesian networks (BN)?

P&C principle in supervised learning

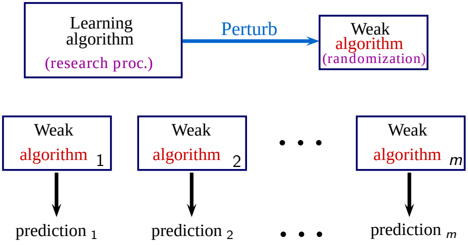
- Principle : (Bagging, Random forests, Extremely randomized trees)



How can we apply this idea to density estimation with Bayesian networks (BN)?

P&C principle in supervised learning

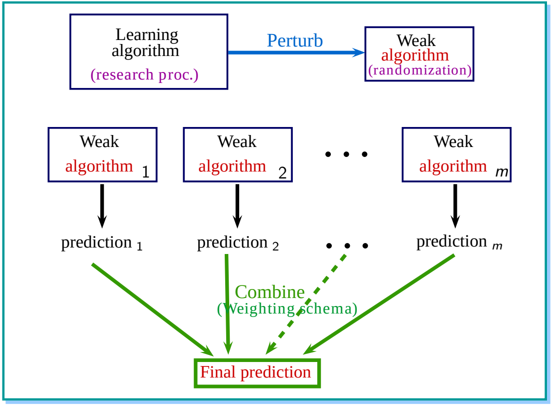
- Principle : (Bagging, Random forests, Extremely randomized trees)



How can we apply this idea to density estimation with Bayesian networks (BN)?

P&C principle in supervised learning

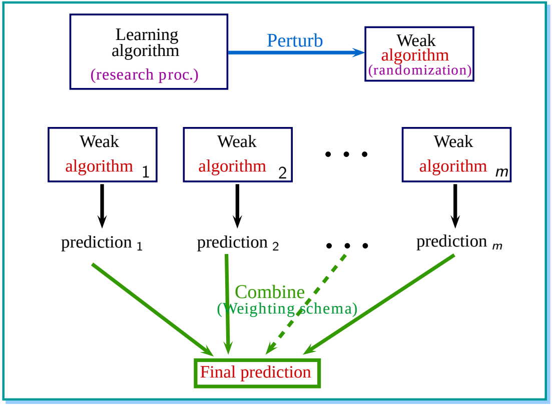
- Principle : (Bagging, Random forests, Extremely randomized trees)



How can we apply this idea to density estimation with Bayesian networks (BN)?

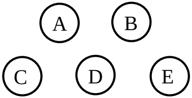
P&C principle in supervised learning

- Principle : (Bagging, Random forests, Extremely randomized trees)

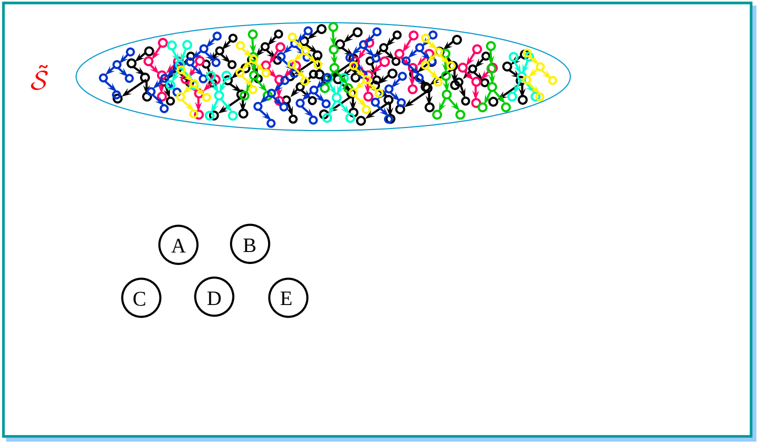


How can we apply this idea to density estimation with Bayesian networks (BN)?

Density estimation with BN

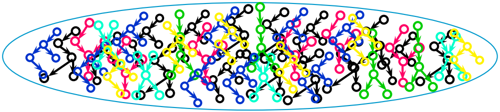


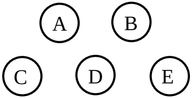
Density estimation with BN



Density estimation with BN

$\tilde{S}$

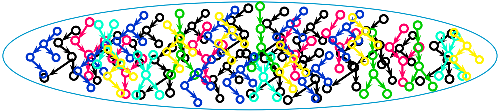


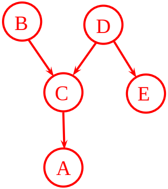


A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Density estimation with BN

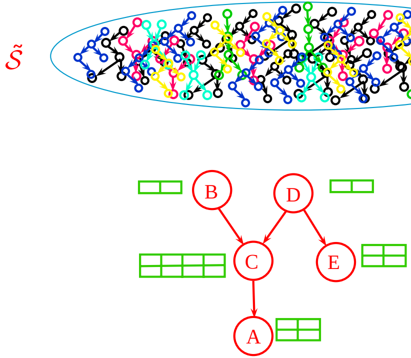
$\tilde{S}$





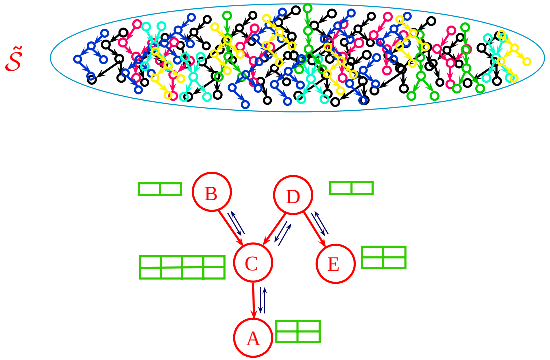
A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Density estimation with BN



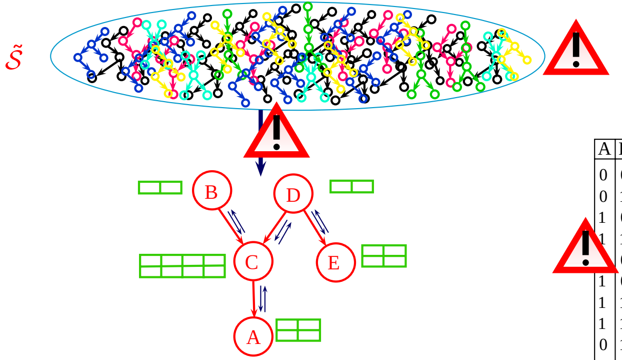
A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Density estimation with BN



A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Density estimation with BN



A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
[illegible]	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Bayesian averaging

- Instead of searching for an optimal model (structure + parameters):
  - Assume prior probabilities over the space of structures
  - Determine posterior probabilities of each model given a dataset
  - approach the target distribution by avereaging the different models wighted by their posterior probabilities

- Caveats : Exact Bayesian averaging over large space of models is not 'scalable'
  $\Rightarrow$ requires to strongly constrain the space of structures

Outline

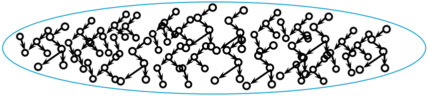
- Background
- Our proposal
- Some results
- Conclusions and Further works

Strategy

- Use simple spaces of graphical structures $\tilde{\mathcal{S}}$ (e.g. chains, trees, poly-trees etc.)
- Do not assume that target distribution is representable by one of these structures
- Rather, assume that target distribution may be approximated well by a mixture of a reasonable number of (S, θ^*) pairs, $S \in \tilde{\mathcal{S}}$

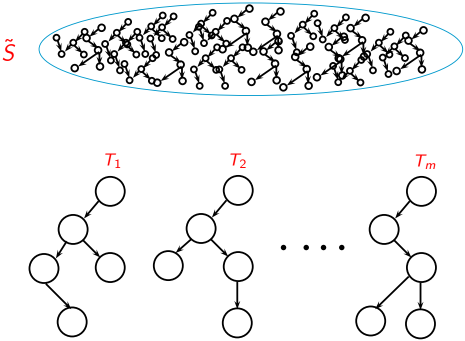
Generic algorithm principle

$\tilde{S}$



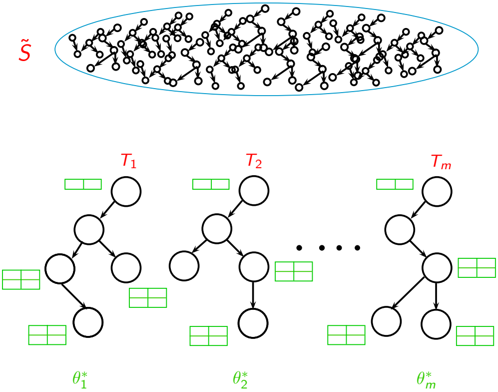
A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Generic algorithm principle



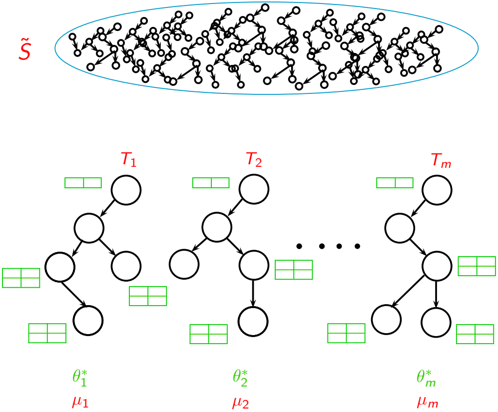
A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Generic algorithm principle



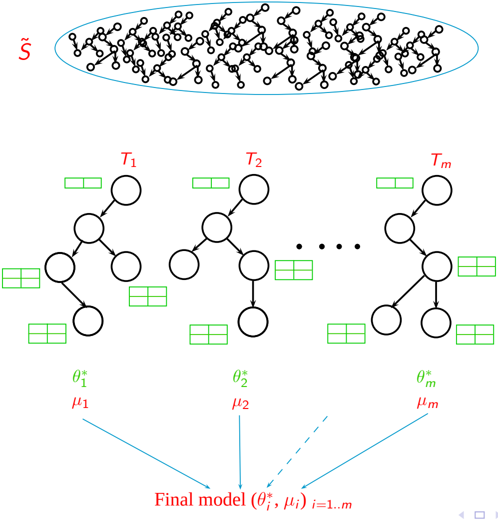
A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Generic algorithm principle



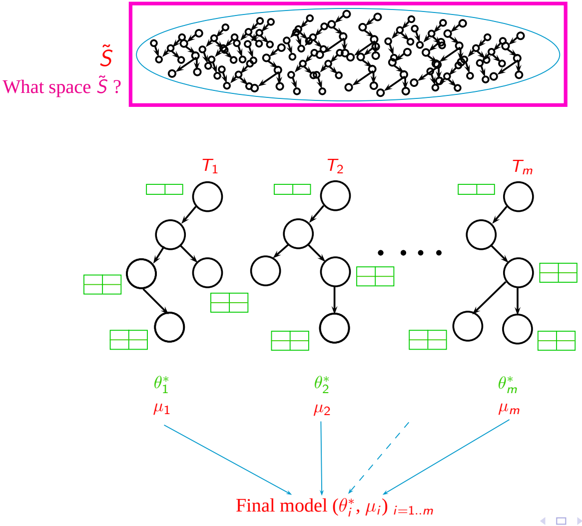
A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Generic algorithm principle



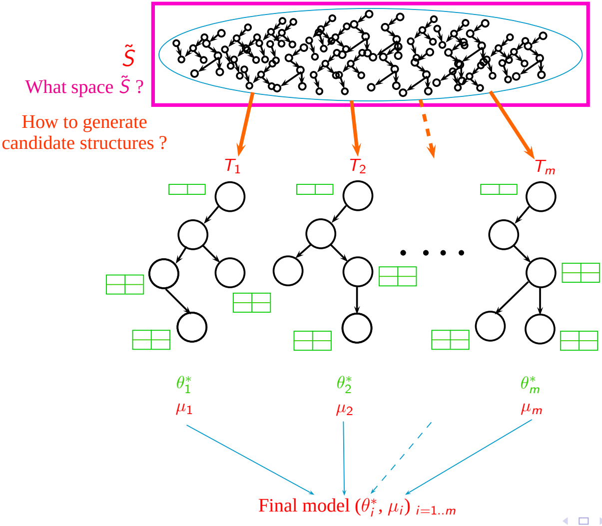
A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Degrees of freedom



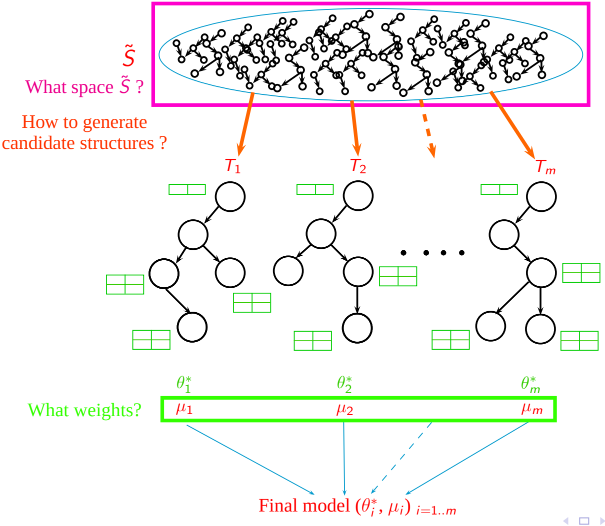
A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Degrees of freedom



A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Degrees of freedom



A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

Why Markov tree space ?

- Polytrees, although more expressive, do not yield more accurate ensemble models than undirected trees [Ammar et al.2008].

- Markov trees
  - (-) "poor" independancy models

  But
  - (+) Inference and parameters learning are scalable (linear complexity)
  - (+) Uniform sampling of trees (linear complexity)
  - (+) Optimal tree structure learning is polynomial (MWST)

How to generate candidate structures ?

- Random uniform sampling [Ammar et al.2009] : linear complexity $O(n)$ and good results
- Build the optimal tree (MWST) over a bootstrap replica of original dataset D [Ammar et al.2009] : quadratic complexity $O(n^2 log(n^2))$ and better results

Goal : *How can we improve the complexity of our quadratic methods and keep the same accuracy ?*

How to generate candidate structures ?

- Random uniform sampling [Ammar et al.2009] : linear complexity $O(n)$ and good results
- Build the optimal tree (MWST) over a bootstrap replica of original dataset D [Ammar et al.2009] : quadratic complexity $O(n^2 log(n^2))$ and better results

Goal : *How can we improve the complexity of our quadratic methods and keep the same accuracy ?*

Principle of MWST

- Step 1 : fill a $(n \times n)$ mutual information matrix
  complexity $O(n^2)$
- Step 2 : build optimal tree (Kruskal)
  complexity $E\ log(E);\ E = n^2$
- Step 3 : learn parameters
  linear complexity

$\Rightarrow$ **Solution :** reduce the first step complexity by applying the Perturb & Combine principle

Principle of MWST

- Step 1 : fill a $(n \times n)$ mutual information matrix

  complexity $O(n^2)$
- Step 2 : build optimal tree (Kruskal)

  complexity $E\ log(E);\ E = n^2$
- Step 3 : learn parameters

  linear complexity

$\Rightarrow$ **Solution :** reduce the first step complexity by applying the Perturb & Combine principle

sub-quadratic research heuristics

- Partial matrix *MI*

MI_K

	X					X		
			X					
				X				
		X				X		X
X							X	
		X						
		X		X				
						X		

- $K = n\ log(n) \Rightarrow$

  Step 1: $n\ log(n)$
  Etape 2: $n\ log(n)\ log(n \log(n))$

$\Rightarrow$ Total complexity $= n\ log(n)\ log(n \log(n))$
$\prec\prec$ quadratic and $\propto$ quasilinear

sub-quadratic research heuristics

- Partial matrix *MI*

MI_K

	X					X		
			X					
				X				
		X				X		X
X							X	
		X						
		X		X				
						X		

K = number of considered terms in MI

- $K = n\ log(n) \Rightarrow$

  Step 1: $n\ log(n)$
  Etape 2: $n\ log(n)\ log(n \log(n))$

$\Rightarrow$ Total complexity $= n\ log(n)\ log(n \log(n))$
$\prec\prec$ quadratic and $\propto$ quasilinear

sub-quadratic research heuristics

- Partial matrix *MI*

MI_K

	X					X		
			X					
				X				
		X				X		X
X							X	
		X						
		X		X				
						X		

K = number of considered terms in MI

$K = ?$

- $K = n\ log(n) \Rightarrow$

  Step 1: $n\ log(n)$
  Etape 2: $n\ log(n)\ log(n \log(n))$

$\Rightarrow$ Total complexity $= n\ log(n)\ log(n \log(n))$
$\prec\prec$ quadratic and $\propto$ quasilinear

sub-quadratic research heuristics

- Partial matrix *MI*

MI_K

	X					X		
			X					
				X				
		X				X		X
X							X	
		X						
		X		X				
						X		

K = number of considered terms in MI

$$K = n\ log(n)$$

- $K = n\ log(n) \Rightarrow$

  Step 1: $n\ log(n)$
  Etape 2: $n\ log(n)\ log(n \log(n))$

$\Rightarrow$ Total complexity $= n\ log(n)\ log(n \log(n))$
$\prec\prec$ quadratic and $\propto$ quasilinear

sub-quadratic research heuristics

- Partial matrix *MI*

MI_K

	X					X		
			X					
				X				
		X				X		X
X							X	
		X						
		X		X				
						X		

K = number of considered terms in MI

$$K = n\ log(n)$$

- $K = n\ log(n) \Rightarrow$

  Step 1: $n\ log(n)$
  Etape 2: $n\ log(n)\ log(n \log(n))$

$\Rightarrow$ Total complexity $= n\ log(n)\ log(n \log(n))$
$\prec\prec$ quadratic and $\propto$ quasilinear

sub-quadratic research heuristics

- Partial matrix *MI*

MI_K

	X					X		
			X					
				X				
		X				X		X
X							X	
		X						
		X		X				
						X		

K = number of considered terms in MI

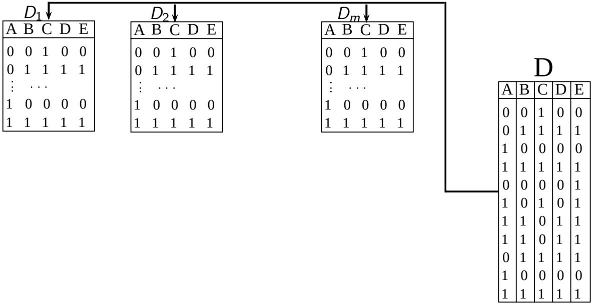
$$K = n\ log(n)$$

- $K = n\ log(n) \Rightarrow$

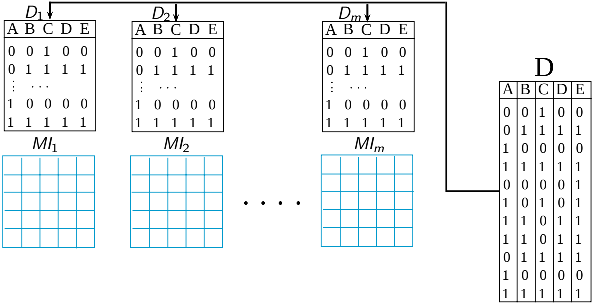
  Step 1: $n\ log(n)$
  Etape 2: $n\ log(n)\ log(n \log(n))$

$\Rightarrow$ Total complexity $= n\ log(n)\ log(n \log(n))$
$\prec\prec$ quadratic and $\propto$ quasilinear

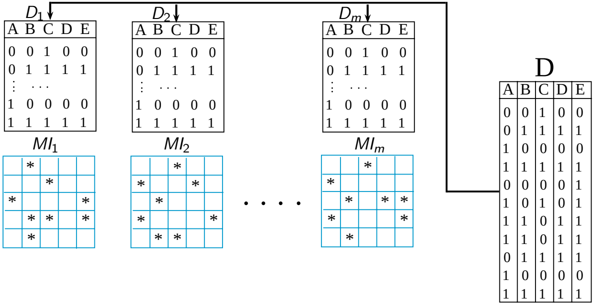
General algorithm : Naive Heuristic



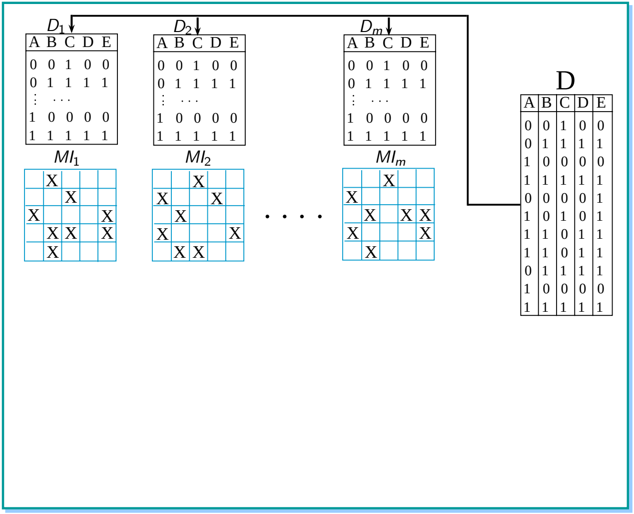
General algorithm : Naive Heuristic



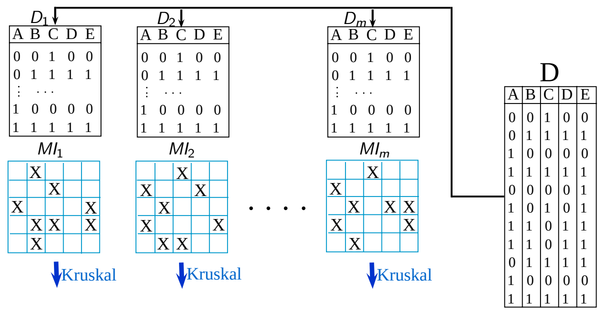
General algorithm : Naive Heuristic



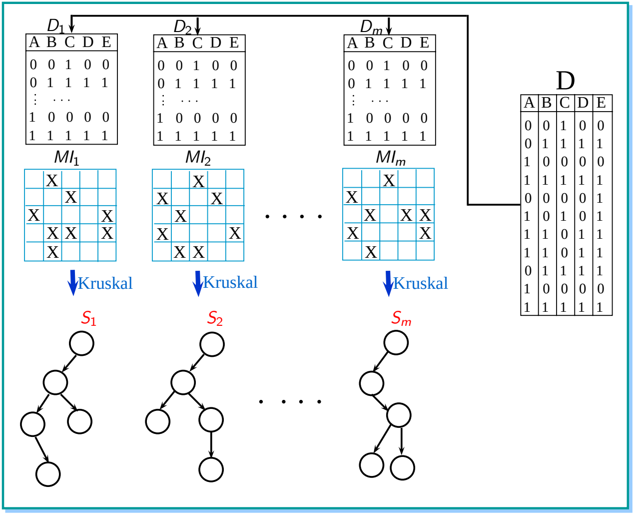
General algorithm : Naive Heuristic



General algorithm : Naive Heuristic



General algorithm : Naive Heuristic



General algorithm : Inertial Heuristic

D_1

A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
⋮	…			
1	0	0	0	0
1	1	1	1	1

D

A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

General algorithm : Inertial Heuristic

D_1

A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
⋮	⋯			
1	0	0	0	0
1	1	1	1	1

MI_1

D

A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

General algorithm : Inertial Heuristic

D_1

A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
⋮	⋯			
1	0	0	0	0
1	1	1	1	1

MI_1

		*		
*		*		
				*
	*			
*	*		*	

D

A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

General algorithm : Inertial Heuristic

D_1

A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
⋮	…			
1	0	0	0	0
1	1	1	1	1

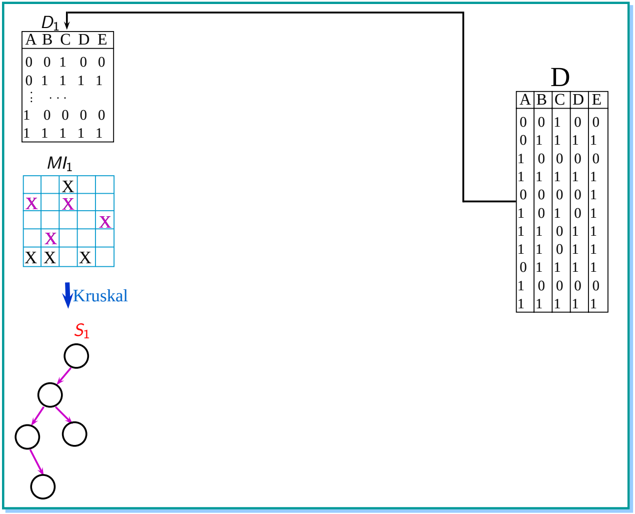
MI_1

		X		
X		X		
				X
	X			
X	X		X	

D

A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

General algorithm : Inertial Heuristic



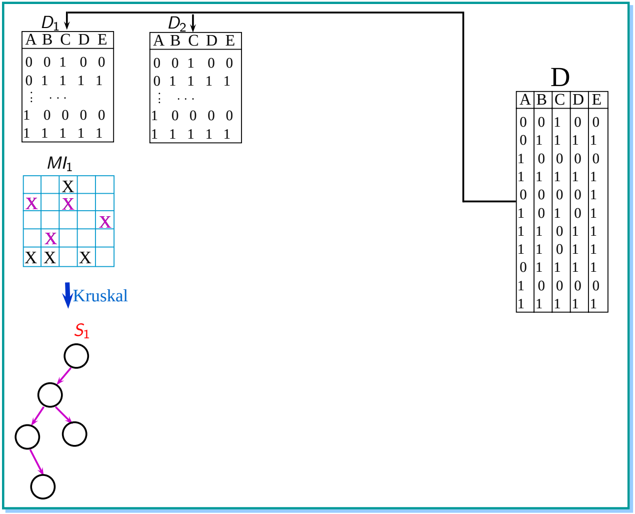
D_1

A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
⋮	⋯			
1	0	0	0	0
1	1	1	1	1

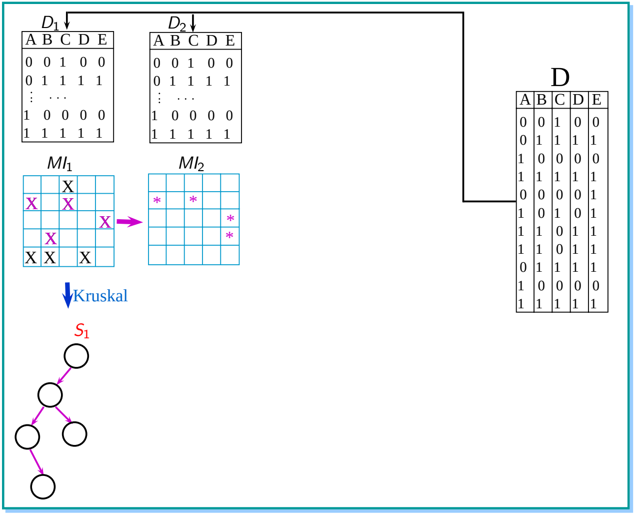
D

A	B	C	D	E
0	0	1	0	0
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1
0	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	1	1

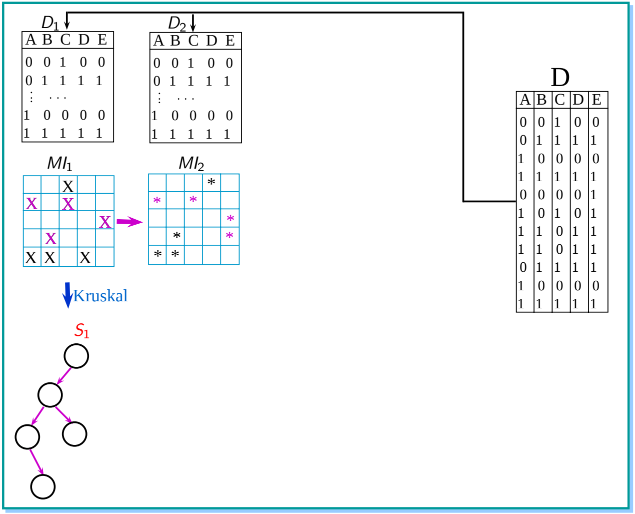
General algorithm : Inertial Heuristic



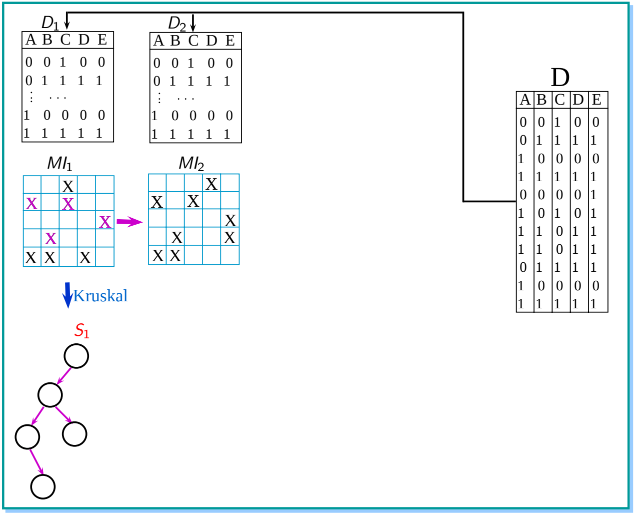
General algorithm : Inertial Heuristic



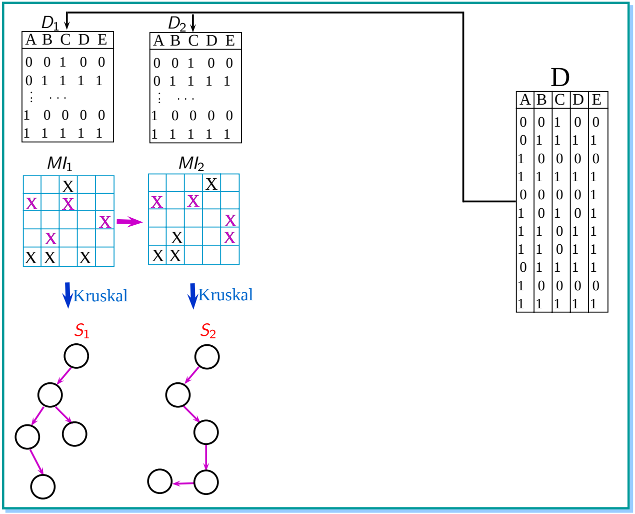
General algorithm : Inertial Heuristic



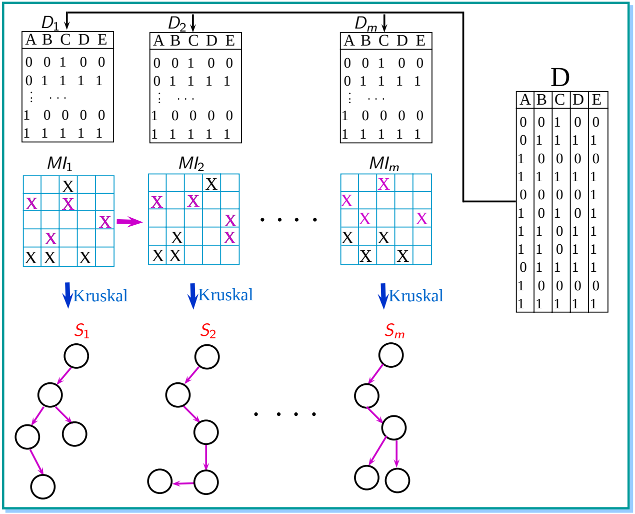
General algorithm : Inertial Heuristic



General algorithm : Inertial Heuristic



General algorithm : Inertial Heuristic



Experimental protocol

- Test problems
  - 1000 binary variables;
  - 10 target distributions : DAG (1000 vars)
  - 10 repeated experiments, for $|D| = 1000$
- Algorithm variants
  - Mixtures with growing sizes ($m = 1, 10, 20, \ldots, 150$) of Markov trees, two variants of the generation of these trees :
    - Sbquadratic Heuristics over the initial dataset D
    - Sbquadratic Heuristics over a bootstrap replica of D
  - parameter learning : MAP with uniform priors
  - Different weighting schemas :
    - Uniform weighting (ie. $\mu_i = 1/m$)
    - Bayesian averaging (ie. $\mu_i \propto \text{score}_{BDeu}$)
- Accuracy evaluation
  - asymmetric approached Kullback-Leibler divergence
  - Accuracy is better when $\hat{KL}$ divergence is lower.

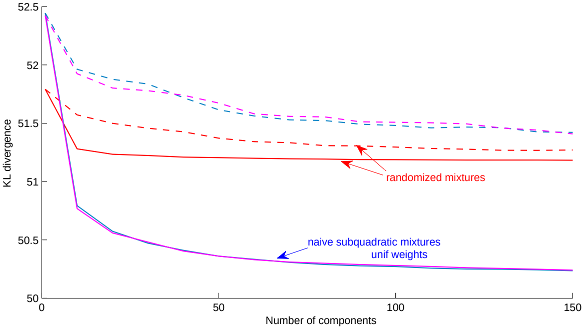
Experimental protocol

- Test problems
  - 1000 binary variables;
  - 10 target distributions : DAG (1000 vars)
  - 10 repeated experiments, for $|D| = 1000$
- Algorithm variants
  - Mixtures with growing sizes ($m = 1, 10, 20, \ldots, 150$) of Markov trees, two variants of the generation of these trees :
    - Sbquadratic Heuristics over the initial dataset D
    - Sbquadratic Heuristics over a bootstrap replica of D
  - parameter learning : MAP with uniform priors
  - Different weighting schemas :
    - Uniform weighting (ie. $\mu_i = 1/m$)
    - Bayesian averaging (ie. $\mu_i \propto \text{score}_{BDeu}$)
- Accuracy evaluation
  - asymmetric approached Kullback-Leibler divergence
  - Accuracy is better when $\hat{KL}$ divergence is lower.

Experimental protocol

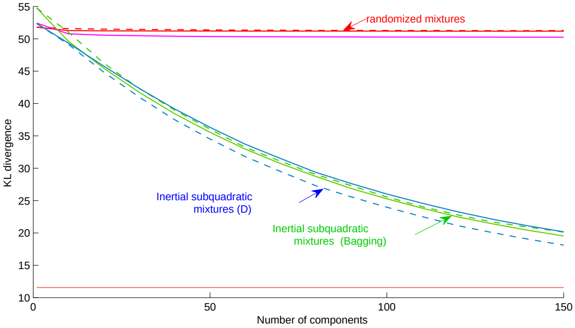
- Test problems
  - 1000 binary variables;
  - 10 target distributions : DAG (1000 vars)
  - 10 repeated experiments, for $|D| = 1000$
- Algorithm variants
  - Mixtures with growing sizes ($m = 1, 10, 20, \ldots, 150$) of Markov trees, two variants of the generation of these trees :
    - Sbquadratic Heuristics over the initial dataset D
    - Sbquadratic Heuristics over a bootstrap replica of D
  - parameter learning : MAP with uniform priors
  - Different weighting schemas :
    - Uniform weighting (ie. $\mu_i = 1/m$)
    - Bayesian averaging (ie. $\mu_i \propto \text{score}_{BDeu}$)
- Accuracy evaluation
  - asymmetric approached Kullback-Leibler divergence
  - Accuracy is better when $\hat{KL}$ divergence is lower.

Naive Heuristic results



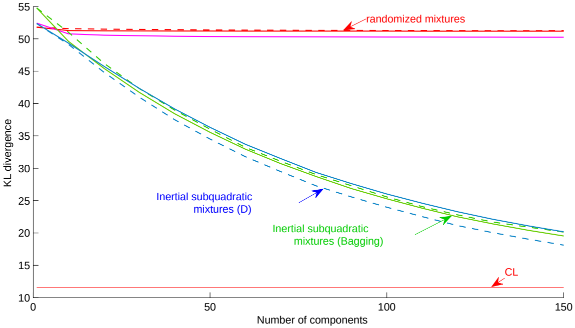
- Our naive subquadratic mixtures are clearly outperforming the randomized methods with uniform weighting

Inertial Heuristic results



- Our inertial methods outperform the randomized ones (Linear complexity)
- Our inertial methods approach the base method *CL* (Quadratic complexity)

Inertial Heuristic results



- Our inertial methods outperform the randomized ones (Linear complexity)
- Our inertial methods approach the base method *CL* (Quadratic complexity)

Conclusion

- Our subquadratic mixtures (quasilinear) outperform the randomized methods (linear)
- Our inertial methods approach the base method *CL* (quadratic)
- Bagging does not allow an improvement in the estimation quality

Further work

- Improve our methods : linear approximation of the optimal tree [Chazelle2000]
- Comparison with scalable optimal structure learning algorithms (sparse candidate [Friedman et al.1999], MMHC [Tsamardinos et al.2006] ...)
- Consider sequential methods of combination (Boosting, MCMC)

Bibliography

S. Ammar, Ph. Leray, B. Defourny, and L. Wehenkel.
2008.
High-dimensional probability density estimation with randomized ensembles of tree structured bayesian networks.
In *Proceedings of the fourth European Workshop on Probabilistic Graphical Models (PGM08)*, pages 9–16.

S. Ammar, Ph. Leray, B. Defourny, and L. Wehenkel.
2009.
Probability density estimation by perturbing and combining tree structured markov networks.
In *the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU09)*, pages 156–167.

B. Chazelle.
2000.
A minimum spanning tree algorithm with inverse-ackermann type complexity.
J. ACM, 47(6):1028–1047.

N. Friedman, I. Nachman, and D. Peér.
1999.
Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm.
In *Proc. Fifteenth Conf. on Uncertainty in Artificial Intelligence*, pages 206–215.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis.
2006.
The max-min hill-climbing bayesian network structure learning algorithm.
Machine Learning, 65(1):31–78.