Algorithms for Exact Structure Discovery in Bayesian Networks

Pekka Parviainen

Helsinki Institute for Information Technology HIIT
Department of Computer Science
University of Helsinki

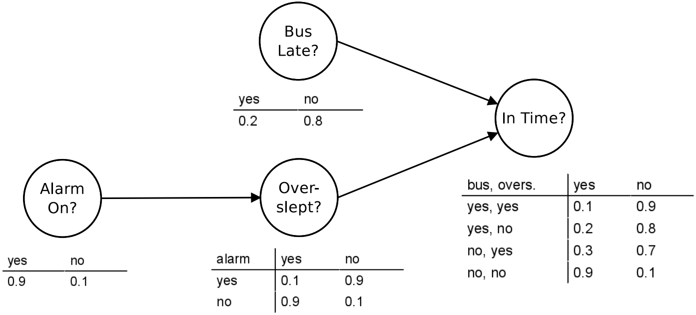
Algodan Seminar
28.10.2011

Outline

- Structure Discovery Problems
- Time–Space Tradeoffs
- Extensions and Future Work

Bayesian networks

- Representations of joint probability distributions
- Consist of:
  - The structure is a directed acyclic graph (DAG) that represents conditional independencies between variables.
  - The local conditional probability distributions that are specified by parameters.



yes	no
0.2	0.8

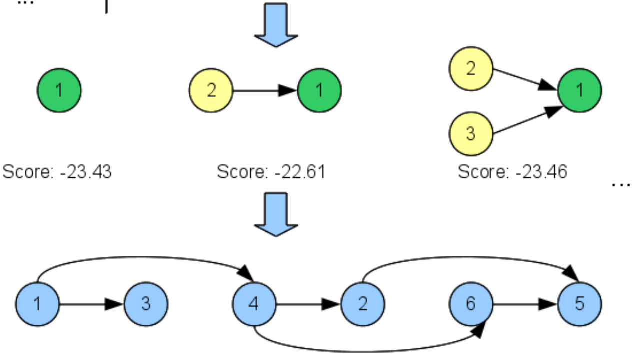
yes	no
0.9	0.1

alarm	yes	no
yes	0.1	0.9
no	0.9	0.1

bus, overs.	yes	no
yes, yes	0.1	0.9
yes, no	0.2	0.8
no, yes	0.3	0.7
no, no	0.9	0.1

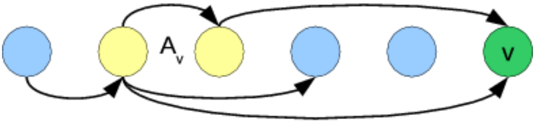
Score-based Structure Discovery

	Var. 1	Var. 2	Var. 3	Var. 4	...
Person A	1	1	2	1	
Person B	2	2	1	1	
Person C	1	2	2	2	
Person D	2	1	2	1	
...					



Optimal Structure Discovery (OSD) Problem

- The score of a DAG is the sum of the local scores.
- Problem:
  - Input: Local scores for each node and possible parent set.
  - Output: A DAG that maximizes the score.



Feature Probability (FP) Problem

- Problem:
  - Input: Local scores for each node and possible parent set (computed from the data), a structural prior and a structural feature.
  - Output: Posterior probability of the feature given the data.
- Bayesian averaging.
- Assumptions: Order-modular prior, modular feature (for example an arc).

Why Time–Space Tradeoffs?

- An exact algorithm is guaranteed to learn an optimal Bayesian network from data $\rightarrow$ no uncertainty on the quality of the output.
- Many exact methods use dynamic programming
- **Time** and **space** complexities are within a polynomial factor of $\mathbf{2^n}$, where n is the number of nodes.
- Space requirement is the bottleneck
  - For example Silander–Myllymäki implementation requires 89 GB of space (memory + disk), when $n = 29$ and 784 GB, when $n = 32$.
- If we save space, how much more time do we need?

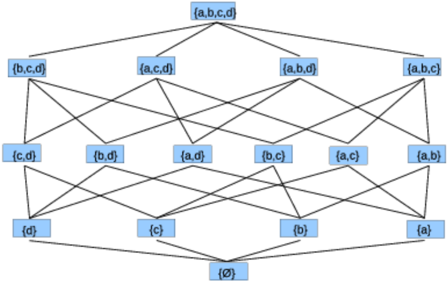
Partial Order Approach [Parviainen & Koivisto UAI'09]

- Idea:
  1. Fix a set of partial orders to "cover" all possible linear orders.
  2. Choose a partial order from the set.
  3. Find an optimal DAG compatible with the chosen partial order.
  4. Repeat steps 2 and 3 for all partial orders in the set.
- Step 3 can be computed in time and space proportional to the number of ideals.
  - An ideal of a partial order P is a set that can start a linear extension of P.
- Space: the number of ideals (per partial order)
- Time: the number of ideals multiplied by the number of partial orders.

Linear Orders and Ideals

$N = \{a, b, c, d\}$

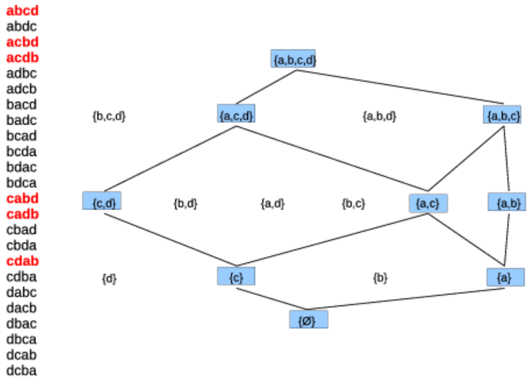
abcd
abdc
acbd
acdb
adbc
adcb
bacd
badc
bcad
bcda
bdac
bdca
cabd
cadb
cbad
cbda
cdab
cdba
dabc
dacb
dbac
dbca
dcab
dcba



Number of linear orders $= 4! = 24$
Number of ideals $= 2^4 = 16$
Space $= 16$, Time $= 16$

Partial Orders and Ideals

$N = \{a, b, c, d\}$, partial order $a \prec b, c \prec d$ fixed.

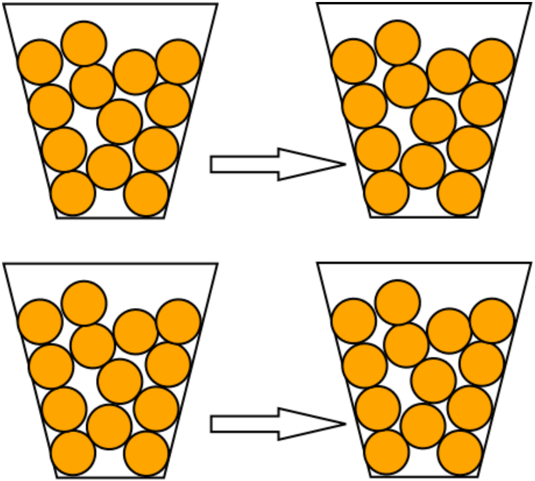


Number of ideals $= 3^2 2^0 = 9$
Partial orders needed to cover all linear orders $= 2^2 = 4$
Space $= 9$, Time $= 9 \times 4 = 36$

Space–Time Tradeoffs for Permutation Problems [Koivisto & Parviainen SODA'10]

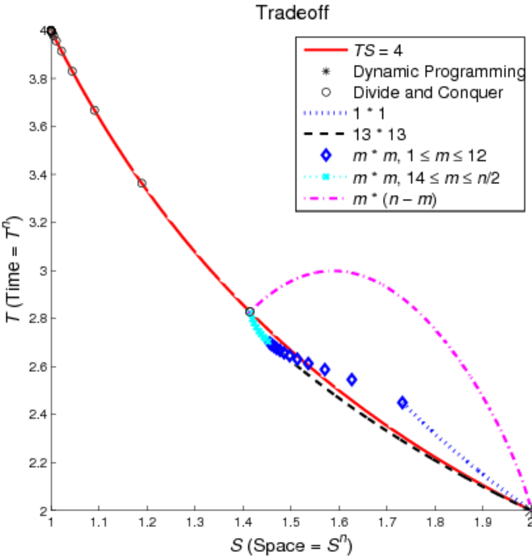
- Find a permutation of n elements so as to minimize a given cost function.
- Examples:
  - Travelling Salesman
  - Feedback Arc Set
  - Cutwidth
  - Treewidth
  - Scheduling
  - OSD
- Sum-product problems

Parallel Bucket Orders



Parallel $13 * 13$ bucket orders are optimal with respect to time–space product.

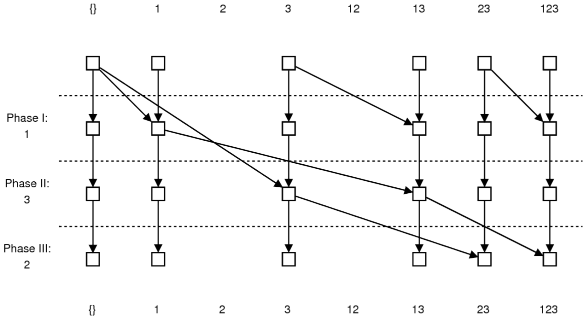
Tradeoffs



Tradeoff
TS = 4
Dynamic Programming
Divide and Conquer
1 * 1
13 * 13
m * m, 1 ≤ m ≤ 12
m * m, 14 ≤ m ≤ n/2
m * (n − m)
T (Time = T^n)
S (Space = S^n)
4
3.8
3.6
3.4
3.2
3
2.8
2.6
2.4
2.2
2
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2

Space–Time Tradeoffs for the FP Problem [Parviainen & Koivisto AISTATS'10]

- In similar fashion as for the OSD problem.
- Requires a fast sparse zeta transform algorithm (a special case of zeta transform for lattices, see [Björklund, Husfeldt, Kaski, Koivisto, Nederlof & Parviainen SODA'12]).



Extensions

- Use exact algorithms as building blocks to develop better heuristics [Niinimäki, Parviainen & Koivisto UAI'11].
- FP problem with nonmodular features $\rightarrow$ learning ancestor relations [Parviainen & Koivisto ECML PKDD'11].

Future Work

- Unobserved variables in score-based structure discovery
- Local learning
- Learning under structural constraints (e.g. treewidth)

Thank you!