# ベイジアンネットワークの確率計算とZDD

大阪大学大学院情報科学研究科
准教授 浜口清治

# ベイジアンネットワーク

- ベイジアンネットワークの処理技術
  - 確率推論(確率計算)，確率学習，構造推定

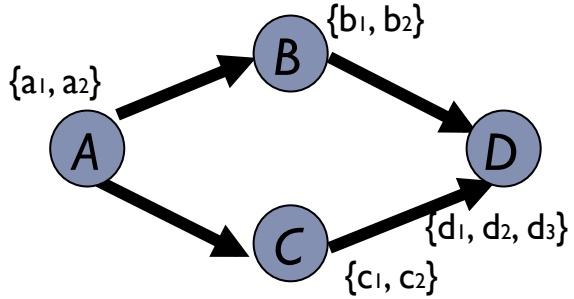
- 確率推論(確率計算)

$$\Pr(D=d1)=\sum_{A,B,C}\Pr(A,B,C,D=d1)$$

$$\Pr(B=b2,D=d1)=\sum_{A,C}\Pr(A,B=b2,C,D=d1)$$

| $AB$ | $\Pr(B\|A)$ |
|---|---|
| a1b1 | 0.2 |
| a1b2 | 0.8 |
| a2b1 | 0.8 |
| a2b2 | 0.2 |

| $BCD$ | $\Pr(D\|B,C)$ |
|---|---|
| b1c1d1 | 0.0 |
| b1c1d2 | 0.5 |
| b1c1d3 | 0.5 |
| b1c2d1 | 0.2 |
| b1c2d2 | 0.3 |
| b1c2d3 | 0.5 |
| b2c1d1 | 0.0 |
| b2c1d2 | 0.0 |
| b2c1d3 | 1.0 |
| b2c2d1 | 0.2 |
| b2c2d2 | 0.3 |
| b2c2d3 | 0.5 |




| $A$ | $\Pr(A)$ |
|---|---|
| a1 | 0.4 |
| a2 | 0.6 |

| $AC$ | $\Pr(C\|A)$ |
|---|---|
| a1c1 | 0.5 |
| a1c2 | 0.5 |
| a2c1 | 0.5 |
| a2c2 | 0.5 |

Mark Chavira and Adnan Darwiche:
"Compiling Bayesian Networks Using Variable Elimination"
20th International Joint Conference on Artificial Intelligence (IJCAI), 2007, pp. 2443-2449.

# 多重線形関数 (multi-linear function, MLF)

- 確率分布表を表現
- $\lambda$: インディケータ変数
  - 0 か 1
- $\theta$: パラメータ変数
  - 確率 [0,1]
- 各項が確率分布表の1行に相当

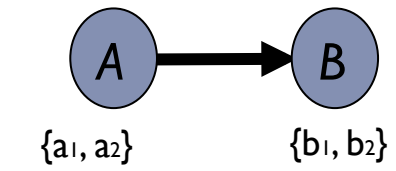
| *ABCD* | P(*A,B,C,D*) |
|---|---|
| a1b1c1d1 | P(A=a1,B=b1,C=c1,D=d1) |
| a1b1c1d2 | P(A=a1,B=b1,C=c1,D=d2) |
| a1b1c1d3 | P(A=a1,B=b1,C=c1,D=d3) |
| … | … |
| a2b2c2d3 | P(A=a2,B=b2,C=c2,D=d3) |

$$MLF^{all} =$$

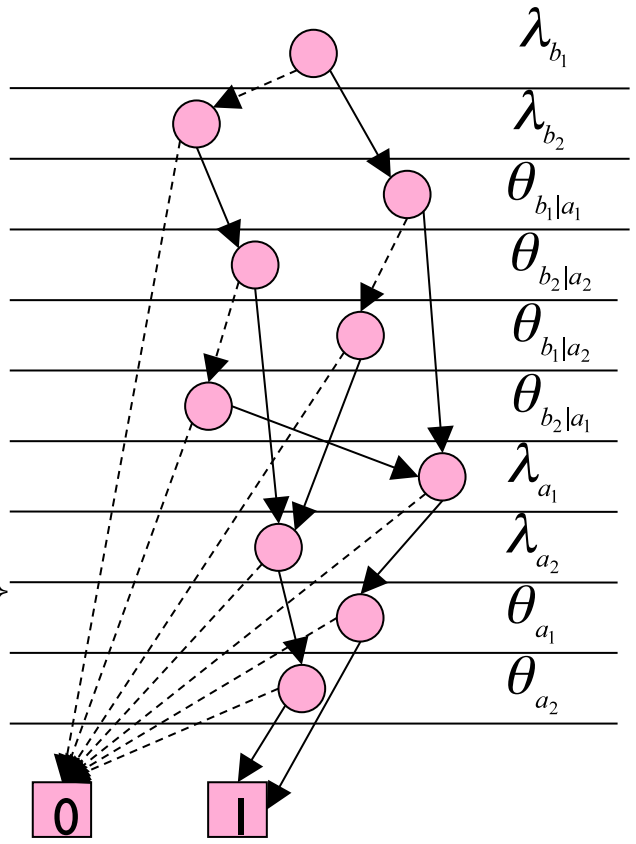
$$\lambda_{a_1}\lambda_{b_1}\lambda_{c_1}\lambda_{d_1}\theta_{a_1}\theta_{b_1|a_1}\theta_{c_1|a_1}\theta_{d_1|b_1c_1} +$$
$$\lambda_{a_1}\lambda_{b_1}\lambda_{c_1}\lambda_{d_2}\theta_{a_1}\theta_{b_1|a_1}\theta_{c_1|a_1}\theta_{d_2|b_1c_1} +$$
$$\lambda_{a_1}\lambda_{b_1}\lambda_{c_1}\lambda_{d_3}\theta_{a_1}\theta_{b_1|a_1}\theta_{c_1|a_1}\theta_{d_3|b_1c_1} +$$
$$\vdots$$
$$\lambda_{a_2}\lambda_{b_2}\lambda_{c_2}\lambda_{d_3}\theta_{a_2}\theta_{b_2|a_2}\theta_{c_2|a_2}\theta_{d_3|b_2c_2}$$

# ZDD による MLF (multi-linear function)表現

- ZDD 表現
  - 要素: $\lambda$ と $\theta$
  - 集合 : MLF の項
  - 族 : MLF 全体
- 確率計算
  - グラフサイズに線形

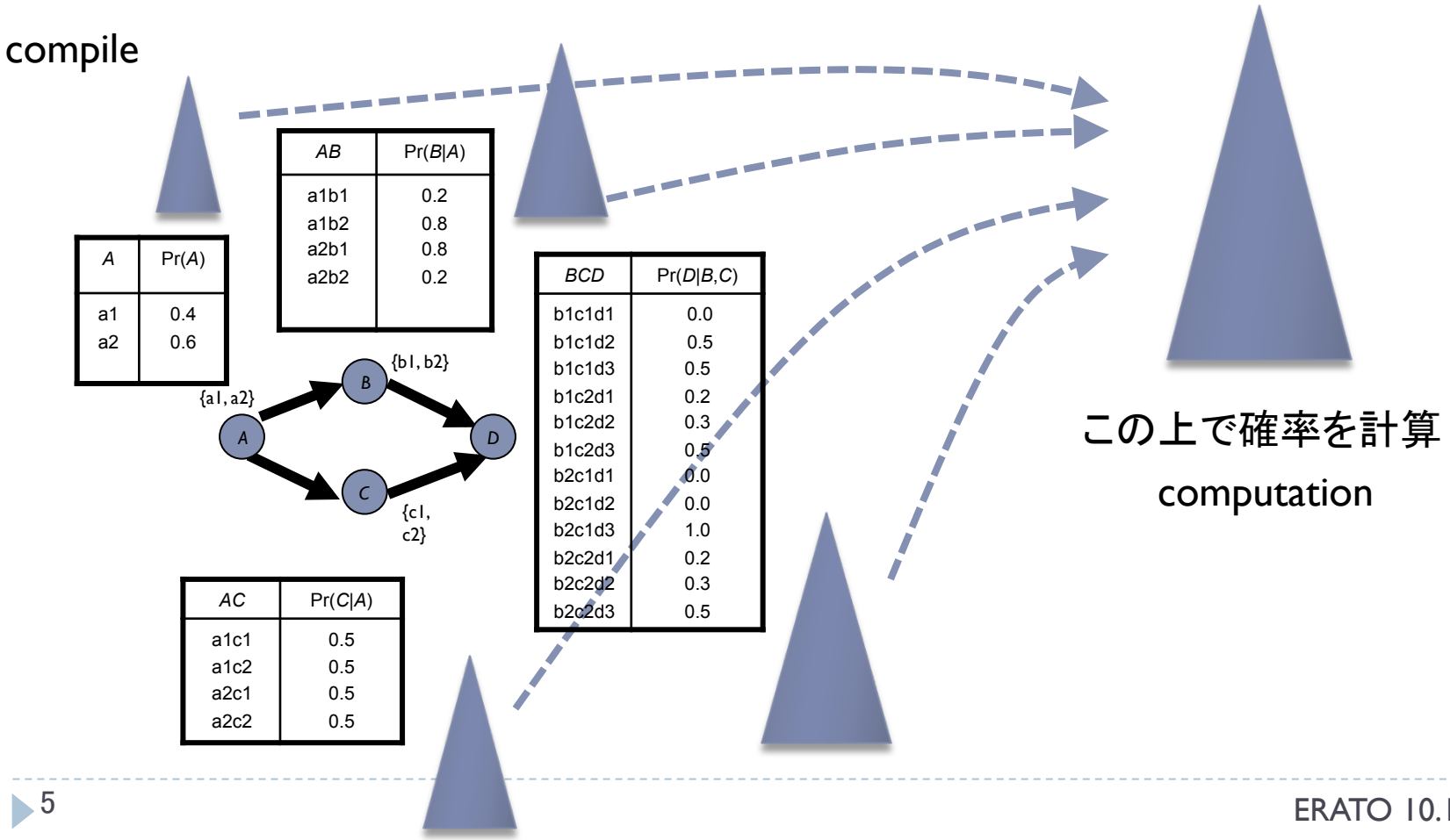



$$\{\{\lambda_{a_1},\lambda_{b_1},\theta_{a_1},\theta_{b_1|a_1}\},$$
$$\{\lambda_{a_1},\lambda_{b_2},\theta_{a_1},\theta_{b_2|a_1}\},$$
$$\{\lambda_{a_2},\lambda_{b_1},\theta_{a_2},\theta_{b_1|a_2}\},$$
$$\{\lambda_{a_2},\lambda_{b_2},\theta_{a_2},\theta_{b_2|a_2}\}\}$$



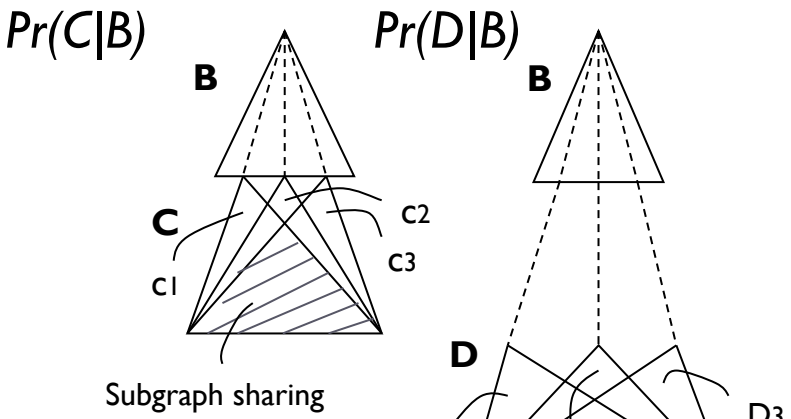
# ZBDDベース構築

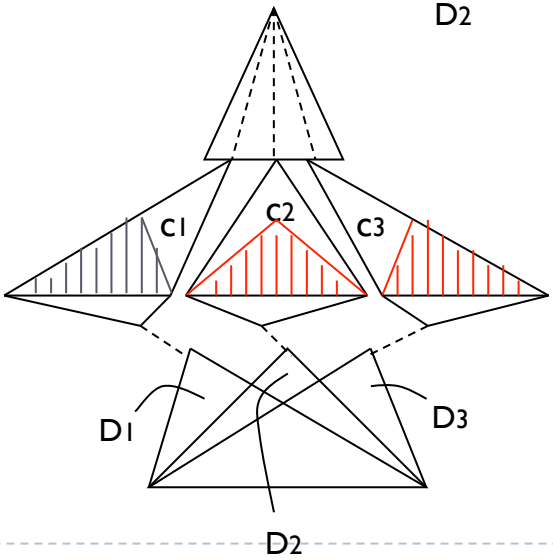
S. Minato, K. Sato and T. Sato, "Compiling Bayesian Networks by Symbolic Probability Calculation Based on Zero-suppressed BDDs," In Proc. of 20th International Joint Conference of Artificial Intelligence(IJCAI-2007), pp.2550-2555, 2007.



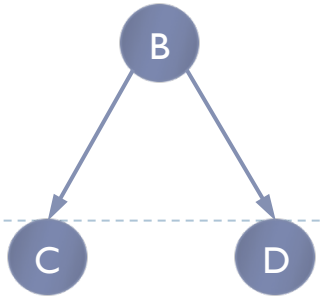
# 手法

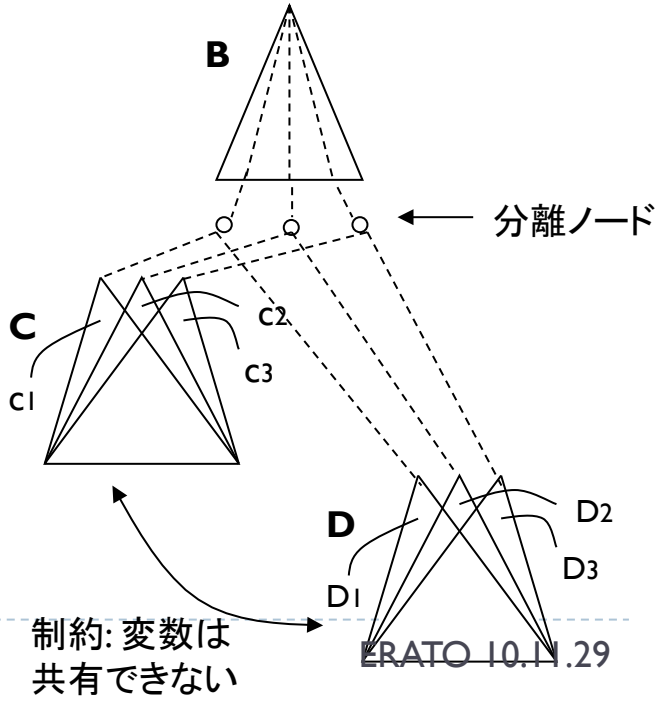
D. Tokoro, A. Fukunaga, K. Hamaguchi, T. Kashiwabara, S. Minato: ``Exploiting Global Structures in Bayesian Network Compilation by Zero-suppressed BDDs'', 19th International Conference on Inductive Logic Programming, (poster presentation), July 2009.




B による
分解構造
を利用




制約: 変数は
共有できない

# 確率推論：結果

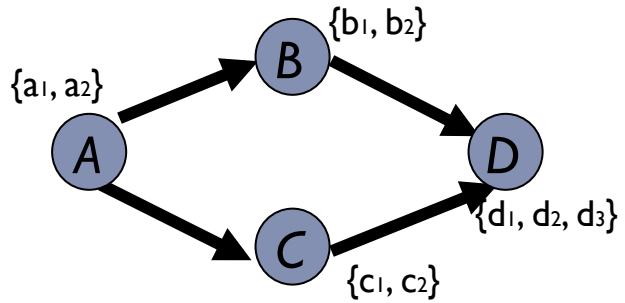
| BN name | # of BN nodes. | Compilation Time (s) | | AC size (edges) | | Prob. Computation Time (s) | |
|---|---|---|---|---|---|---|---|
| | | AceT | ZDD | AceT | ZDD | AceT | ZDD |
| Pigs | 441 | 0.602 | 2.68 | 1267412 | 2033640 | 0.072 | 0.17 |
| Diabetes | 413 | 3.911 | 32.85 | 15476258 | 25024900 | 0.631 | 1.60 |
| Munin2 | 1003 | 1.876 | 5.71 | 4222134 | 4465700 | 0.200 | 0.42 |
| Munin3 | 1044 | 1.184 | 7.14 | 2652334 | 4547260 | 0.134 | 0.54 |
| Munin4 | 1041 | 3.853 | 26.43 | 4643186 | 8181150 | 0.210 | 0.72 |

- Environments: Redhat Linux, Pentium 4, 2.4GHz, 2GB of memory, ZDD nodes < 21M
- AceT: Tabular method, ZDD: Ours
- AC size for ZDD: Evaluated from obtained ZDD size
- Inference: Average over random two instantiations for 100 times

D. Tokoro, A. Fukunaga, K. Hamaguchi, T. Kashiwabara, S. Minato: ``Exploiting Global Structures in Bayesian Network Compilation by Zero-suppressed BDDs", 19th International Conference on Inductive Logic Programming, (poster presentation), July 2009.

# BN上のMAP計算への適用

- 問題
  - (正確な) 周辺MAP (maximum a posteriori) 計算
    - ベイジアンネットワークのノードには離散変数を仮定
  - ***P(X)*** と証拠 ***e*** に対して:
    - ***P(m, e)*** を最大化する MAP 変数 $M \subseteq X$ への値の割り当て ***m*** を求める
    - ***(m, e)*** は $X$ への完全な割り当てではない
    - cf. MAP (MPE) 問題 : ***(m, e)*** は完全な割り当て




$M = \{A,B\}$, $e = (D=d2)$

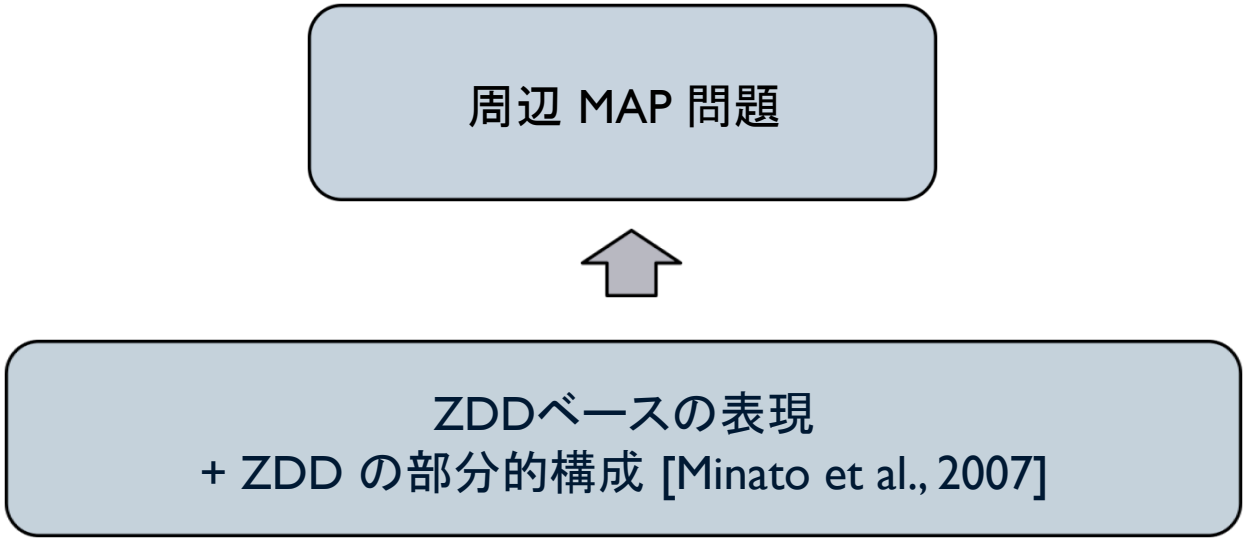
**次の中から最大の値の割り当てをみつければよい**

***P(A=a1,B=b1,D=d2), P(A=a1,B=b2,D=d2),***
***P(A=a2,B=b1,D=d2), P(A=a2,B=b2,D=d2).***

# 計算量など

- **計算量**
  - $NP^{PP}$– 完全 [Park, UAI 2002]
  - cf. MAP (MPE) 問題 : $NP$– 完全
- 先行研究
  - あまりたくさんは研究されていない
    - Jointree-based [Park&Darwiche, UAI 2003]
    - Compilation-based [Huang et al., AAAI 2006]
      - AceMAP : 規模の大きな BNs に有利
- アプローチ
  - MAP変数への値の割り当てを，分枝限定で探索
  - ZDDの組み立て方を工夫

# 方法と結果のまとめ

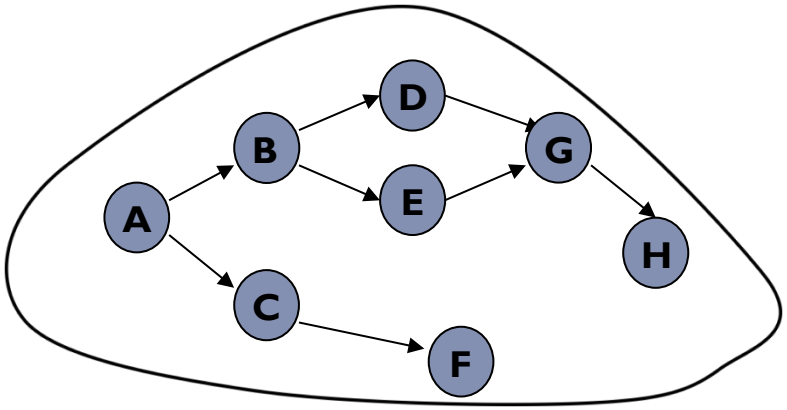



- 結果のまとめ
  - 部分的構成は全体構成よりも悪くはならない
    - 1–300 倍速い
  - 最良時には，AC–ベースの手法 (AceMAP) より 2桁程度高速になる.

# 部分的構成

- 2つのアプローチ

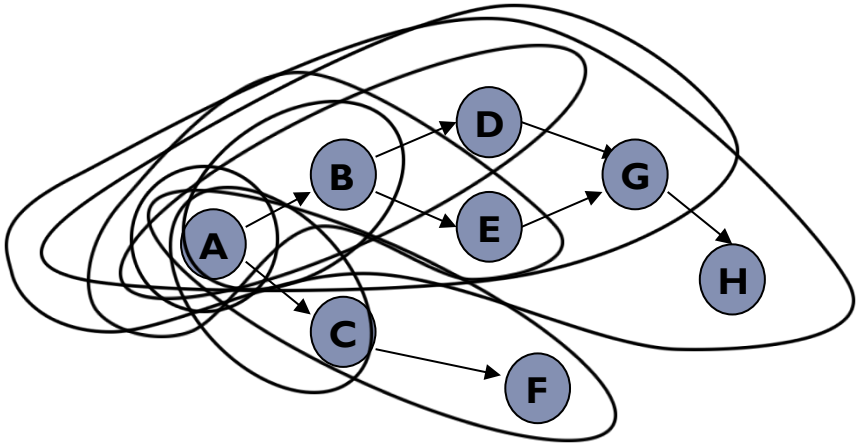
全体構成法




1個の ZDD を構成

確率推論:
サイズに線形

部分的構成法




8 個のZDD を構成

確率推論:
- さらなる ZDD 操作が必要
  - P(C=c1, H=h1) は 2つの ZDD を組み合わせる必要
  - サイズが小さければ，効率改善を期待できる

# Experimental results

- Results for randomly generated 10 instances with 20 MAP variables
- CPU: AMD Opteron 8350 (2 GHz), Mem: 64G, RedHat Linux, C++

| name | #nd | monolithic | | | partial | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ZDD #node | compile (sec) | MAP (sec) | ZDD #node | compile (sec) | Ave. #node | MAP (sec) | accer. ratio |
| alarm | 37 | 1792 | 0.007 | 1.1 | 6955 | 0.99 | 1580 | 0.10 | 11.0 |
| water | 32 | 86571 | 2.05 | 141.1 | 79178 | 0.38 | 79704 | 126.15 | 1.12 |
| hailfinder | 56 | 34423 | 0.12 | 3295.1 | 70903 | 0.14 | 22592 | 1597.84 | 2.06 |
| pigs | 441 | 693755 | 1.88 | 13663.7 | 45227 | 0.095 | 10868 | 43.23 | 316.07 |

"Partial" construction is not worse than "monolithic" construction

- In the best case, 300 times faster

As an approach, "monolithic" construction method $\doteqdot$ AceMAP : similar performance

- In the best case, two orders of magnitude faster in the partial construction approach

# 今後の展開 (1)

- (重み付き)モデルカウンティング
  - 与えられた条件をみたす解の個数を数える問題，あるいは
  - 与えられた条件をみたす解に付加されている全ての重みの総和を求める問題
    - BDD, ZBDDの場合：1に至るパスの数を数える問題
    - BN の確率計算も例のひとつ

- アプローチ
  - SAT ソルバーの拡張 vs. グラフ表現へのコンパイル
    - 全空間を探索，各解を評価する問題を解いているのは同じだが．．．
    - どうも，探索の途中結果を全て覚えておいて，再利用した方が効率がよさそう

グラフへのコンパイルによるアプローチ

# 今後の展開 (2)

- 方向
  - 従来：論理式ベース(0,1が対等な世界)
    - BDD <-> SAT, モデルカウンティング
  - 族集合ベース(0,1が対等でない世界)で行うとどうなる？
    - ZDD <-> ZSAT, Zモデルカウンティング

- 現在
  - 族集合の計算式から直接的に分解構造を抽出する方法を検討中
    - ベイジアンネットワークのコンパイルに限らない
    - 分解ノードによる，ZDD の爆縮をめざす

$$\{\{\lambda_{a_1},\theta_{a_1}\},\{\lambda_{a_2},\theta_{a_2}\}\}*$$
$$\{\{\lambda_{a_1},\lambda_{b_1},\theta_{b_1|a_1}\},\{\lambda_{a_1},\lambda_{b_2},\theta_{b_2|a_1}\},$$
$$\{\lambda_{a_2},\lambda_{b_1},\theta_{b_1|a_2}\},\{\lambda_{a_2},\lambda_{b_2},\theta_{b_2|a_2}\}\}*$$
…