电子科技大学
University of Electronic Science and Technology of China

Chap. 9-10
第II部分 推断

张众

Data Mining Lab,
Big Data Research Center, UESTC
Email：zhangzhong0512@163.com

目录

- 我是目录，还没来得及写...

一点点回顾

有向图中的独立性

➢局部独立性

- 给定父节点，每个节点X_i独立于其非后代节点。

➢全局独立性

- 令**X**, **Y**, **Z**是图G的三个节点集。给定**Z**的条件下，假如任意节点X ∈ **X**与Y ∈ **Y**之间不存在有效迹，那么**X**与**Y**在给定**Z**时时d-分离的，记作d-sep(**X**;**Y**|**Z**)。
- 与d-分离相对应的独立性的集合用I(G)表示：

$$I(G) = \{(X \perp Y|Z): d - sep(\boldsymbol{X}; \boldsymbol{Y}|\boldsymbol{Z})\}$$

这个集合称为**全局马尔科夫独立性**集

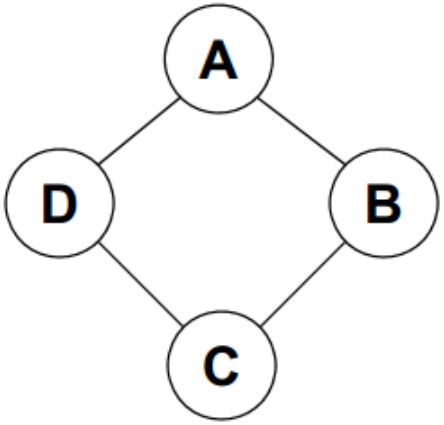
➢可靠性与完备性

- 图中的独立性包含于分布中的独立性，但分布中蕴含的某些独立性不一定会在图中呈现

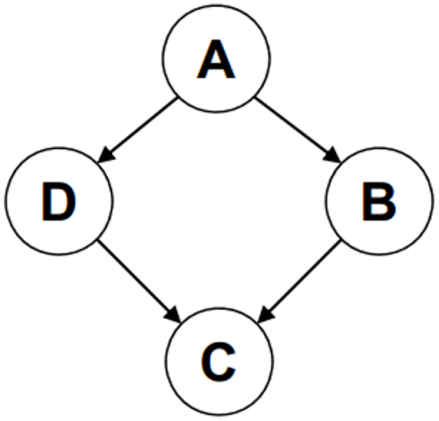
无向图中的独立性

➢马尔科夫毯（Markov Blanket）

- $\mathrm{MB}_{\mathrm{MRF}}(\boldsymbol{X})$={ X的所有邻居 }
- $\mathrm{MB}_{\mathrm{DAG}}(\boldsymbol{X})$={ X的父节点, X的子节点， X的子节点的父节点 }



MRF



BN1

$$MB(D) = \{A, C\}_{UGM} = \{A, C, B\}_{DAG}$$

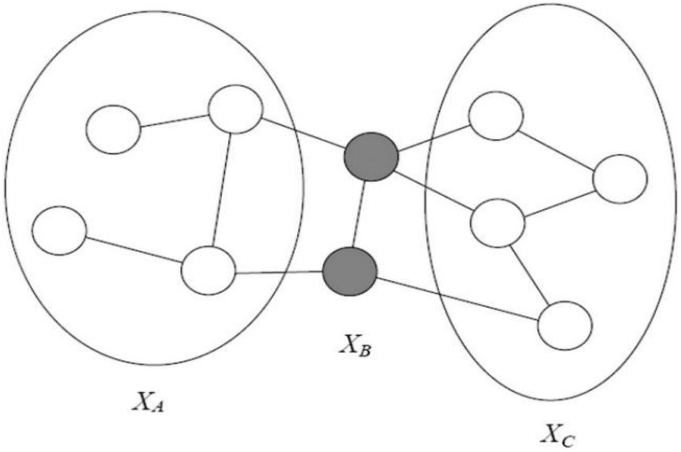
➢局部马尔科夫独立性

$$\mathrm{I}_{\iota}(H): \{X_i \perp V - \{X_i\} - \boldsymbol{MarkovBlanket}(X_i)\}$$

无向图中的独立性

➢全局马尔科夫独立性

- 对于点集A,B,C，B **分离(separate)** A and C, 如果每一条从A到C的路都经过B



$$Sep(A;C|B)$$

- *可靠性+完备性

图上因子分解的Gibbs分布满足与图相关的独立性
图中的非独立性，在部分分布上可能也非独立

因子分解

➢ 贝叶斯网

- 令G为定义在变量X1 ,…, Xn上的一个贝叶斯网。可以将分布因子分解如下面的形式：

$$P(X_1, X_2, \dots, X_n) = \prod_i P(X_i | Parents(X_i))$$

- 定义因子：

$$\phi_{X_i}\big(X_i, Parents(X_i)\big) = P(X_i | Parents(X_i))$$

- 则分布可以写成下面形式：

$$P(X_1, X_2, \dots, X_n) = \prod_i \phi_{X_i}\big(X_i, Parents(X_i)\big)$$

因子分解

➢ 马尔科夫随机场

- 令H为定义在变量X1 ,…, Xn上的一个马尔科夫网。令$\boldsymbol{C}_1$,…, $\boldsymbol{C}_k$为H中的团，对每个团覆盖的辖域指定一系列因子$\phi_1(\boldsymbol{C}_1), \dots, \phi_l(\boldsymbol{C}_l)$

$$P_\Phi(X_1, X_2, \dots, X_n) = \frac{1}{Z}\tilde{P}_\Phi(X_1, X_2, \dots, X_n)$$

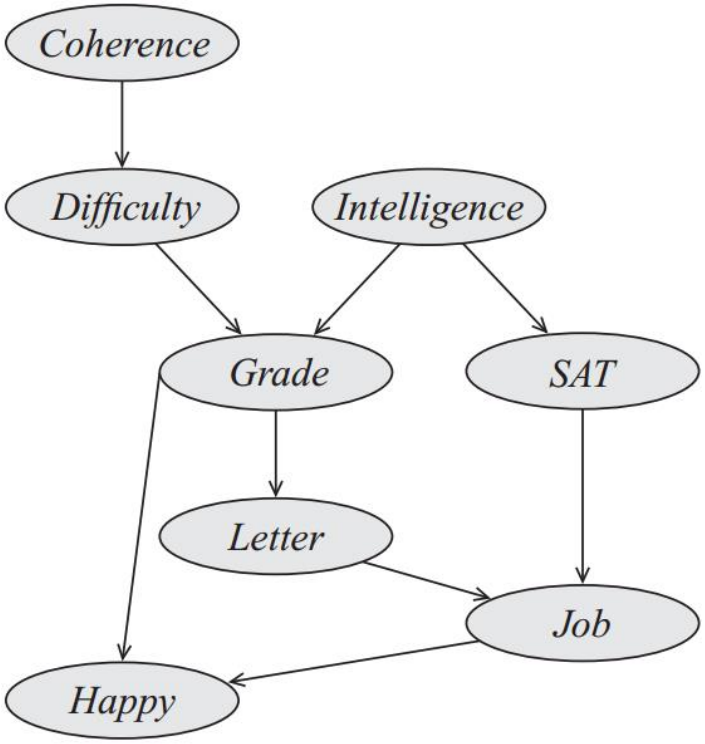
- 其中：

$$\tilde{P}_\Phi(X_1, X_2, \dots, X_n) = \prod_i \phi_i(\boldsymbol{C}_i)$$

- 归一化因子：

$$\mathrm{Z} = \sum_{X_1,\dots,X_n} \tilde{P}_\Phi(X_1, X_2, \dots, X_n)$$

一个例子



BN

$$
\begin{aligned}
P(C,D,I,G,S,L,J,H) &= P(C)P(D \mid C)P(I)P(G \mid I,D)P(S \mid I) \\
&\quad P(L \mid G)P(J \mid L,S)P(H \mid G,J) \\
&= \phi_C(C)\phi_D(D,C)\phi_I(I)\phi_G(G,I,D)\phi_S(S,I) \\
&\quad \phi_L(L,G)\phi_J(J,L,S)\phi_H(H,G,J).
\end{aligned}
$$

一个例子

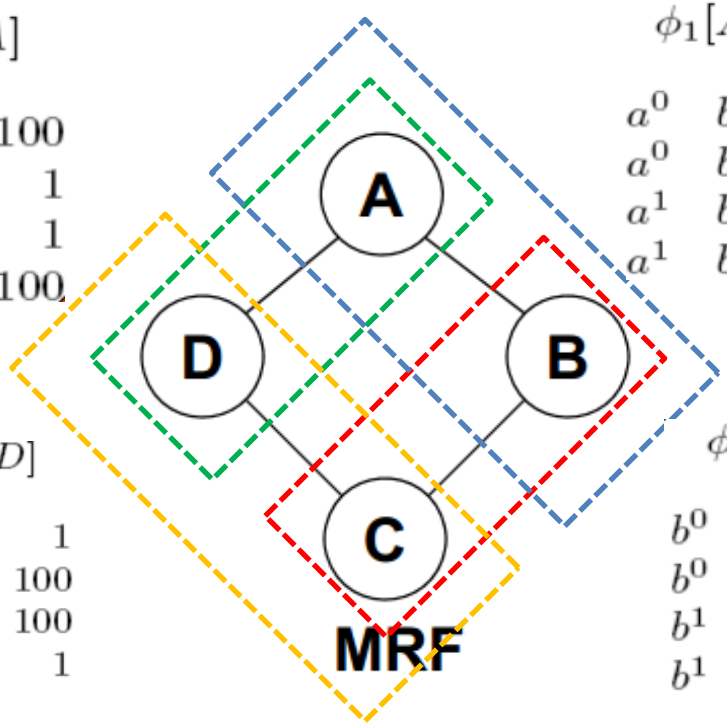
数据挖掘实验室
Data Mining Lab

$\phi_4[D, A]$

d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100

$\phi_1[A, B]$

a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10



$\phi_3[C, D]$

c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

$\phi_2[B, C]$

b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

$$P(a, b, c, d) = \frac{1}{Z}\phi_1(a, b) \cdot \phi_2(b, c) \cdot \phi_3(c, d) \cdot \phi_4(d, a),$$

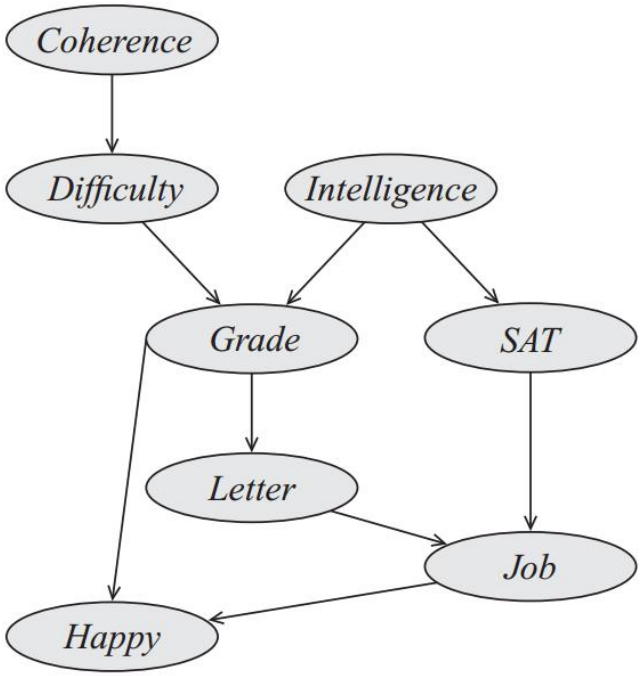
where

$$Z = \sum_{a,b,c,d} \phi_1(a, b) \cdot \phi_2(b, c) \cdot \phi_3(c, d) \cdot \phi_4(d, a)$$

推断的初步认识

三种查询任务

- 有了联合概率分布后，我们希望从概率分布中查询特定变量的分布或概率，可以将查询任务概括为以下三种：
  - Query 1：似然 Likelihood
  - Query 2：后验概率 Conditional Probability
  - Query 3：最大后验概率 Most Probable Assignment



BN

- $P(H, J) = \sum_C \dots \sum_L P(C, \dots, L, H, J)$
- $P(J|S = s^1) = \frac{P(J, S=s^1)}{P(S=s^1)} = \frac{P(J, S=s^1)}{\sum_j P(J=j, S=s^1)}$
- $MAP(L|J = j^1) = \text{argmax}_l \, P(L|J = j^1)$

Query 1：Likelihood

数据挖掘实验室
Data Mining Lab

- 带有证据的查询
  - 令集合**E** ={X_k+1, … , X_n}
  - **e**是**E**中变量的特定取值，称为证据
- 最简单的查询：计算证据的概率

$$P(\boldsymbol{e}) = \sum_{x_1} \dots \sum_{x_k} P(x_1, \dots, xk, \boldsymbol{e})$$

  - 称$P(\boldsymbol{e})$为证据**e**的似然概率

Query 2：Conditional Probability

数据挖掘实验室
Data Mining Lab

- 最常见的查询，在给定条件**e**的情况下，查询变量的条件分布

$$P(X|\boldsymbol{e}) = \frac{P(X,\boldsymbol{e})}{P(\boldsymbol{e})} = \frac{P(X,\boldsymbol{e})}{\sum_x P(X=x,\boldsymbol{e})}$$

  - 以上称为给定证据**e**的情况下，X的**后验分布**

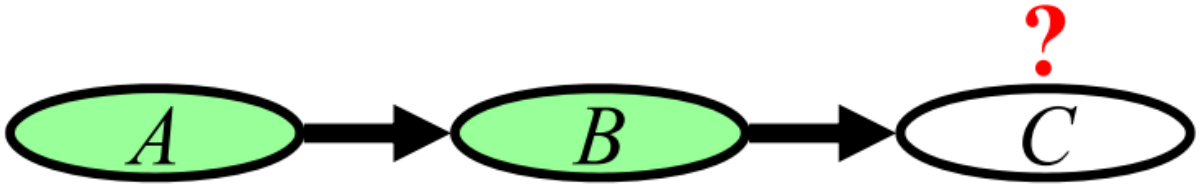
- 通常我们关心一个对变量子集的查询，令变量集合**X** = {**Y**, **Z**}，其中**Y**是关心的变量集，**Z**是无关的变量集

$$P(\boldsymbol{Y}|\boldsymbol{e}) = \sum_{\boldsymbol{Z}} P(\boldsymbol{Y},\boldsymbol{Z}=z\,|\boldsymbol{e})$$

  - 求和消除无关变量**Z**的过程，称为**边缘化**（marginalization）

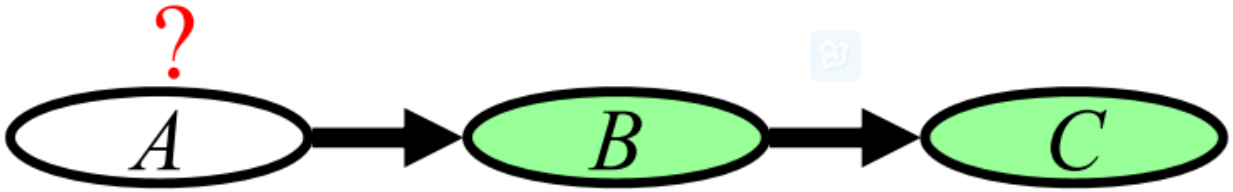
后验分布的例子

- 预测：给定条件，推理结果



  - 查询结点是给定证据的后代结点

- 诊断：给定结果，推理条件



  - 查询结点是给定证据的祖先结点

Query 3：MPA

- 通常我们希望找到对于某些变量的最可能的联合取值

- 给定证据**e**，无关变量集**Z**，关心变量集**Y**

$$MPA(\boldsymbol{Y}|\boldsymbol{e}) = argmax_{\boldsymbol{y}\in\boldsymbol{y}}\, P(\boldsymbol{y}|\boldsymbol{e}) = argmax_{\boldsymbol{y}\in\boldsymbol{y}} \sum_{z} P(\boldsymbol{y},\boldsymbol{Z},|\boldsymbol{e})$$

- 以上称为**y**的最大后验配置

- 例子
  - MPA(Y_1) ?　　y* = 1
  - MPA(Y_1, Y_2) ?　　y* = 0，0

y_1	y_2	$P(y_1, y_2)$
0	0	0.35
0	1	0.05
1	0	0.3
1	1	0.3

推断方法

➢精确推理

- The elimination algorithm
- Message-passing algorithm (sum-product, belief propagation)
- The junction tree algorithms

➢近似推理

- Stochastic simulation / sampling methods
- Markov chain Monte Carlo methods
- Variational algorithms

推理复杂度

- 给定PGM P_Φ，变量X，值x∈Val(X)，证据e∈Val(E)，以下问题是NP-Hardness：

- 计算$P_\Phi(X = x)$
- 判断$P_\Phi(X = x) ?> 0$
- 计算$P_\Phi(X = x \mid e)$
- 给定$\varepsilon < 0.5$，找到一个ρ使得$|P_\Phi(X = x) - \rho| < \varepsilon$
- 给定$\varepsilon < 0.5$，找到一个ρ使得$|P_\Phi(X = x \mid e) - \rho| < \varepsilon$

Quiz

- 给定n个变量，每个变量有k个可能的取值，对于这样一个全联合分布，在描述该分布的表中，有多少个表项？

  A、k^n

  B、nk^2

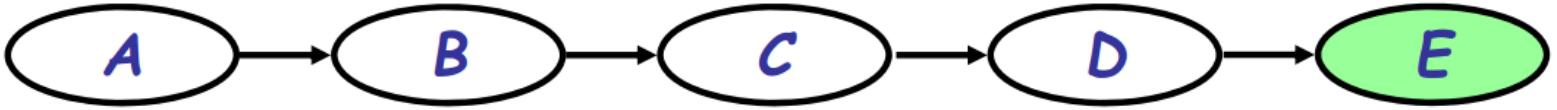
  C、kn^2

  D、n^k

- 联合分布表项随着变量个数指数增长，直观的反映了推理任务本质上就很难有效地求解

当然可以！
NP-Hardness不代表不能解决推断，只是对于任意图模型，没有一般的有效推断方法。对于特定图模型，存在一些高效的方法。

举个栗子



假如E是我们感兴趣的变量，如何查询？

边缘化无关变量：

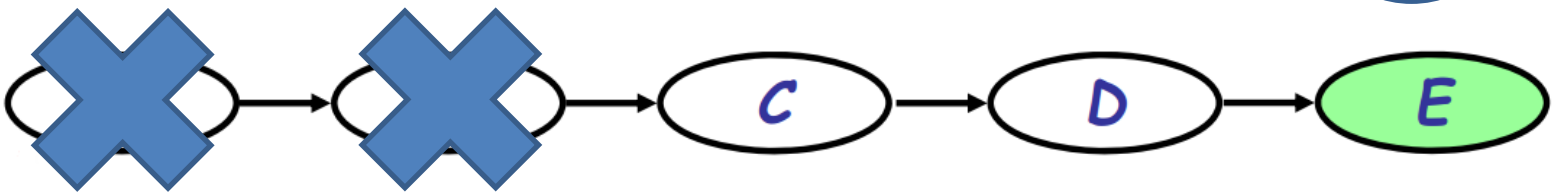
$$P(e) = \sum_{d}\sum_{c}\sum_{b}\sum_{a} P(a,b,c,d,e)$$

指数爆炸，需要遍历完整个分布表！

利用贝叶斯网络的链式规则：

$$P(e) = \sum_{d}\sum_{c}\sum_{b}\sum_{a} P(a)P(b|a)P(c|b)P(d|c)P(e|d)$$

举个栗子



利用贝叶斯网络的链式规则：

$$P(e) = \sum_d \sum_c \sum_b \sum_a P(a)P(b|a)P(c|b)P(d|c)P(e|d)$$

重新安排求和顺序：

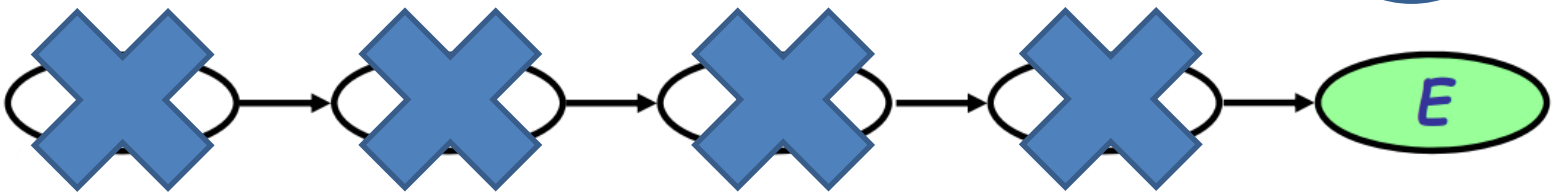
$$= \sum_d \sum_c \sum_b P(c|b)P(d|c)P(e|d) \sum_a P(a)P(b|a)$$

$$= \sum_d \sum_c \sum_b P(c|b)P(d|c)P(e|d)p(b)$$

$$= \sum_d \sum_c P(d|c)P(e|d) \sum_b P(c|b)p(b)$$

$$= \sum_d \sum_c P(d|c)P(e|d)p(c)$$

举个栗子



$$P(e) = \sum_d \sum_c \sum_b \sum_a P(a)P(b|a)P(c|b)P(d|c)P(e|d)$$

$$= \sum_d \sum_c \sum_b P(c|b)P(d|c)P(e|d) \sum_a P(a)P(b|a)$$

$$= \sum_d \sum_c \sum_b P(c|b)P(d|c)P(e|d)p(b)$$

$$= \sum_d \sum_c P(d|c)P(e|d) \sum_b P(c|b)p(b)$$

$$= \sum_d \sum_c P(d|c)P(e|d)p(c)$$

$$= \sum_d P(e|d) \sum_c P(d|c)p(c)$$

$$= \sum_d P(e|d)p(d)$$

举个栗子

➢ 对于一般的链，分析复杂性：

- 假定n个变量的一条链X_1 -> ... -> X_n，每个变量有k个取值。
- 算法根据$P(X_i)$计算$P(X_{i+1})$：

$$P(X_{i+1}) = \sum_{x_i} P(X_{i+1}|x_i)P(x_i)$$

- 其中每一步计算的代价是$O(k^2)$：

  共有 k^2 个乘法运算，$k \times (k-1)$ 个加法运算。

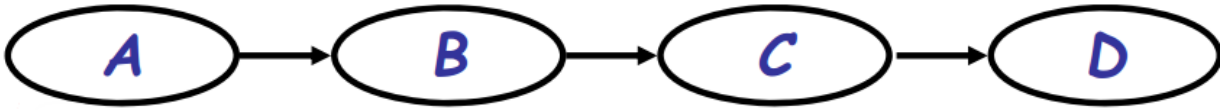
  x_i有k个取值，对于x_i的每个取值，必须用CPD的表值$P(x_{i+1}|x_i)$去乘$P(x_i)$（k^2个乘法运算）；对于每个x_{i+1}，有k x (k – 1)个加法运算。

- 对n个变量执行该过程，总的代价是$O(nk^2)$，意味着在联合分布是指数规模的情况下（k^n），我们却得到了一个线性时间内的推理！

Why ?

- 由于贝叶斯网的结构，联合分布中一些子表达式只依赖于少量的变量

再举个栗子



根据贝叶斯网络的链式规则，联合分布可以分解为：

$$P(A,B,C,D) = P(A)P(B|A)P(C|B)P(D|C)$$

假设所有变量是二值变量。为了计算P(D)，必须分别对D = d^1以及D = d^2的所有表值求和。计算过程如下：

$$
\begin{array}{rcccc}
 & P(a^1) & P(b^1 \mid a^1) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & P(a^1) & P(b^1 \mid a^1) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^2 \mid b^2) & P(d^1 \mid c^2) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^2 \mid b^2) & P(d^1 \mid c^2)
\end{array}
$$

计算P(D = d^1)

$$
\begin{array}{rcccc}
 & P(a^1) & P(b^1 \mid a^1) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & P(a^1) & P(b^1 \mid a^1) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^2 \mid b^2) & P(d^2 \mid c^2) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^2 \mid b^2) & P(d^2 \mid c^2)
\end{array}
$$

计算P(D = d^2)

再举个栗子

$$
\begin{array}{llll}
 & P(a^1) & P(b^1 \mid a^1) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & P(a^1) & P(b^1 \mid a^1) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^2 \mid b^2) & P(d^1 \mid c^2) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^2 \mid b^2) & P(d^1 \mid c^2)
\end{array}
$$

$$
\begin{array}{llll}
 & P(a^1) & P(b^1 \mid a^1) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & P(a^1) & P(b^1 \mid a^1) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & P(a^2) & P(b^1 \mid a^2) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & P(a^1) & P(b^2 \mid a^1) & P(c^2 \mid b^2) & P(d^2 \mid c^2) \\
+ & P(a^2) & P(b^2 \mid a^2) & P(c^2 \mid b^2) & P(d^2 \mid c^2)
\end{array}
$$

提出关于A的公共项

$$
\begin{array}{llll}
 & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^2 \mid b^2) & P(d^1 \mid c^2)
\end{array}
$$

$$
\begin{array}{llll}
 & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & (P(a^1)P(b^1 \mid a^1) + P(a^2)P(b^1 \mid a^2)) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & (P(a^1)P(b^2 \mid a^1) + P(a^2)P(b^2 \mid a^2)) & P(c^2 \mid b^2) & P(d^2 \mid c^2)
\end{array}
$$

令 $\tau_1(B) = \sum_A P(A)P(B|A)$

$$
\begin{array}{llll}
 & \tau_1(b^1) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\
+ & \tau_1(b^2) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\
+ & \tau_1(b^1) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\
+ & \tau_1(b^2) & P(c^2 \mid b^2) & P(d^1 \mid c^2)
\end{array}
$$

$$
\begin{array}{llll}
 & \tau_1(b^1) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\
+ & \tau_1(b^2) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\
+ & \tau_1(b^1) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\
+ & \tau_1(b^2) & P(c^2 \mid b^2) & P(d^2 \mid c^2)
\end{array}
$$

再举个栗子

$$\begin{array}{lll} & \tau_1(b^1) & P(c^1 \mid b^1) & P(d^1 \mid c^1) \\ + & \tau_1(b^2) & P(c^1 \mid b^2) & P(d^1 \mid c^1) \\ + & \tau_1(b^1) & P(c^2 \mid b^1) & P(d^1 \mid c^2) \\ + & \tau_1(b^2) & P(c^2 \mid b^2) & P(d^1 \mid c^2) \end{array}$$

$$\begin{array}{lll} & \tau_1(b^1) & P(c^1 \mid b^1) & P(d^2 \mid c^1) \\ + & \tau_1(b^2) & P(c^1 \mid b^2) & P(d^2 \mid c^1) \\ + & \tau_1(b^1) & P(c^2 \mid b^1) & P(d^2 \mid c^2) \\ + & \tau_1(b^2) & P(c^2 \mid b^2) & P(d^2 \mid c^2) \end{array}$$

提出关于B的公共项

$$\begin{array}{lll} & (\tau_1(b^1)P(c^1 \mid b^1) + \tau_1(b^2)P(c^1 \mid b^2)) & P(d^1 \mid c^1) \\ + & (\tau_1(b^1)P(c^2 \mid b^1) + \tau_1(b^2)P(c^2 \mid b^2)) & P(d^1 \mid c^2) \end{array}$$

$$\begin{array}{lll} & (\tau_1(b^1)P(c^1 \mid b^1) + \tau_1(b^2)P(c^1 \mid b^2)) & P(d^2 \mid c^1) \\ + & (\tau_1(b^1)P(c^2 \mid b^1) + \tau_1(b^2)P(c^2 \mid b^2)) & P(d^2 \mid c^2) \end{array}$$

令$\tau_2(C) = \sum_B P(B)P(C|B)$

$$\begin{array}{lll} & \tau_2(c^1) & P(d^1 \mid c^1) \\ + & \tau_2(c^2) & P(d^1 \mid c^2) \end{array}$$

$$\begin{array}{lll} & \tau_2(c^1) & P(d^2 \mid c^1) \\ + & \tau_2(c^2) & P(d^2 \mid c^2) \end{array}$$

注意到，通过改变计算顺序，得到和式中的公共项，这些公共项保存为变量τ后，可以重复使用，避免了指数级的重复运算

Summary

- 帮助我们解决联合分布中指数爆炸问题的两个观点：

- 根据贝叶斯网络的结构，联合分布中的一些子表达式只依赖于少量的变量
- 通过计算这些表达式并存储起来，可以避免多次对其指数地生成

变量消除算法

一些概念

- 辖域*Scope*[ϕ] = X，因子（factor）ϕ: Val (**X**) $\rightarrow$ $\boldsymbol{R}$

  在贝叶斯网中，因子往往对应变量的分布（概率）或条件分布（条件概率），而在马尔科夫网中，不一定有这样的对应关系。

- $\phi_1(X,Y)$和$\phi_2(Y,Z)$是两个因子，那么两者的乘积可以定义为一个新的因子：

$$\psi(X,Y,Z) = \phi_1(X,Y)\phi_2(Y,Z)$$

- **X**是变量的一个集合，Y是满足Y $\notin$ **X**的变量，令$\psi(\boldsymbol{X},Y)$是一个因子，通过边缘化变量Y可以得到一个关于**X**的因子：

$$\tau(\boldsymbol{X}) = \sum_{Y} \psi(\boldsymbol{X},Y)$$

和-积变量消除算法

Algorithm 9.1 Sum-product variable elimination algorithm

Procedure Sum-Product-VE (
  Φ, // Set of factors
  $\boldsymbol{Z}$, // Set of variables to be eliminated
  $\prec$ // Ordering on $\boldsymbol{Z}$
)
Let $Z_1, \ldots, Z_k$ be an ordering of $\boldsymbol{Z}$ such that
  $Z_i \prec Z_j$ if and only if $i < j$
for $i = 1, \ldots, k$
  $\Phi \leftarrow$ Sum-Product-Eliminate-Var(Φ, Z_i)
$\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$
return ϕ^*

Procedure Sum-Product-Eliminate-Var (
  Φ, // Set of factors
  Z // Variable to be eliminated
)
$\Phi' \leftarrow \{\phi \in \Phi : Z \in Scope[\phi]\}$
$\Phi'' \leftarrow \Phi - \Phi'$
$\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
$\tau \leftarrow \sum_Z \psi$
return $\Phi'' \cup \{\tau\}$

算法思想：每次对一个变量求和。

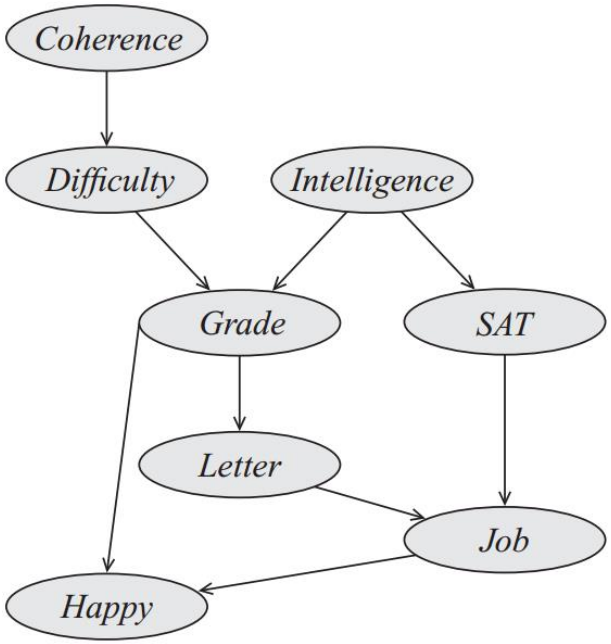
当对任意变量求和时，让所有与该变量有关的因子相乘，生成一个乘积因子，再对该乘积因子求和。进而生成一个新的关于其他待处理变量的因子。

一般来说，可以把任务看做计算以下表达式的值：

$$\sum_{Z} \prod_{\phi \in \mathcal{F}} \phi$$

例子

$$
\begin{aligned}
P(C,D,I,G,S,L,J,H) &= P(C)P(D \mid C)P(I)P(G \mid I,D)P(S \mid I) \\
&\quad P(L \mid G)P(J \mid L,S)P(H \mid G,J) \\
&= \phi_C(C)\phi_D(D,C)\phi_I(I)\phi_G(G,I,D)\phi_S(S,I) \\
&\quad \phi_L(L,G)\phi_J(J,L,S)\phi_H(H,G,J).
\end{aligned}
$$



下面利用变量消除算法计算P(J)。消除顺序为C, D, I, H, G, S, L

1. 消除C：计算因子

$$\psi_1(C,D) = \phi_C(C)\phi_D(D,C)$$

$$\tau_1(D) = \sum_C \psi_1(C,D)$$

2. 消除D：注意，已经消除了包含D的边缘因子——$\phi_D(D,C) = P(D|C)$。另一方面，引入了包含D的因子 $\tau_1(D)$。因此，计算

$$\psi_2(G,I,D) = \phi_G(G,I,D)\tau_1(D)$$

$$\tau_2(G,I) = \sum_D \psi_2(G,I,D)$$

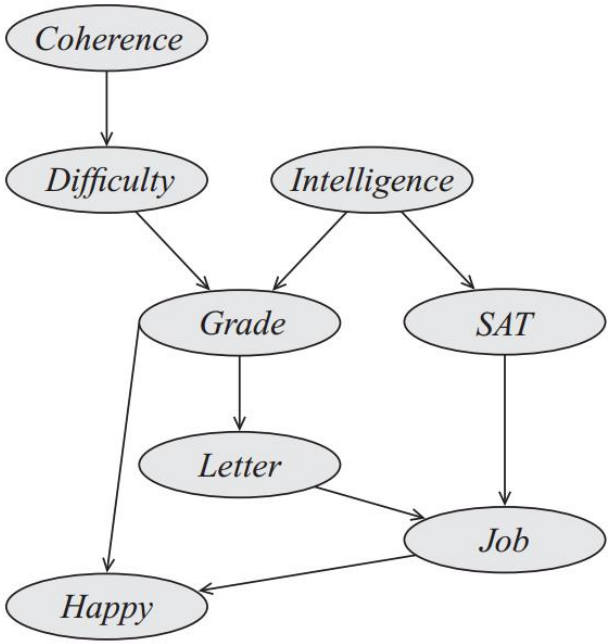
3. 消除I：计算因子

$$\psi_3(G,I,S) = \phi_I(I)\phi_S(S,I)\tau_2(G,I)$$

$$\tau_3(G,S) = \sum_I \psi_3(G,I,S)$$

例子

$$
\begin{aligned}
P(C,D,I,G,S,L,J,H) &= P(C)P(D \mid C)P(I)P(G \mid I,D)P(S \mid I) \\
&\quad P(L \mid G)P(J \mid L,S)P(H \mid G,J) \\
&= \phi_C(C)\phi_D(D,C)\phi_I(I)\phi_G(G,I,D)\phi_S(S,I) \\
&\quad \phi_L(L,G)\phi_J(J,L,S)\phi_H(H,G,J).
\end{aligned}
$$



下面利用变量消除算法计算P(J)。消除顺序为C, D, I, H, G, S, L

4. 消除H：计算因子

$$\psi_4(G,J,H) = \phi_H(H,G,J)$$

$$\tau_4(G,J) = \sum_H \psi_4(G,J,H) \equiv 1$$

5. 消除S：计算因子

$$\psi_6(J,L,S) = \tau_5(J,L,S)\phi_J(J,L,S)$$

$$\tau_6(J,L) = \sum_S \psi_6(J,L,S)$$

3. 消除L：计算因子

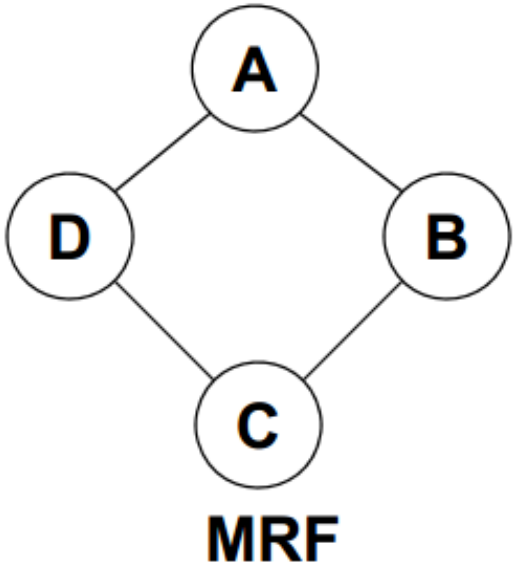
$$\psi_7(J,L) = \tau_6(J,L)$$

$$\tau_7(J) = \sum_L \psi_7(J,L)$$

又一个例子

$$P(A,B,C,D) = \frac{1}{Z}\phi_1(A,B)\phi_2(B,C)\ \ \phi_3(C,D)\phi_4(A,D).$$

$$Z = \sum_{X_1,\ldots,X_n} \tilde{P}_\Phi(X_1,\ldots,X_n)$$



查询目标P(D)。消除顺序为A, B, C

$$\tilde{P}(D) = \sum_{A,B,C} \phi_1(A,B)\phi_2(B,C)\phi_3(C,D)\phi_4(A,D)$$

1. 消除A：计算因子

$$\psi_1(A,B,D) = \phi_1(A,B)\phi_4(A,D)$$

$$\tau_1(B,D) = \sum_A \psi_1(A,B,D)$$

5. 消除B：计算因子

$$\psi_2(B,C,D) = \tau_1(B,D)\phi_2(B,C)$$

$$\tau_2(C,D) = \sum_B \psi_2(B,C,D)$$

3. 消除C：计算因子

$$\psi_3(C,D) = \tau_2(C,D)\phi_3(C,D)$$

$$\tau_3(D) = \sum_C \psi_3(C,D) = \tilde{P}(D)$$

证据处理

➢ 存在观测变量集**E**的情况下，定义如下变量：

- 证据势能：

$$\delta(E_i, \bar{e}) = \begin{cases} 1 \; if \; E_i == \bar{e} \\ 0 \; if \; E_i \; != \bar{e} \end{cases}$$

- 所有证据势能：

$$\delta(\boldsymbol{E}, \bar{\boldsymbol{e}}) = \prod_{i \in I_{\boldsymbol{E}}} \delta(E_i, \bar{e})$$

- 引入证据——受限的因子：

$$\tau(\boldsymbol{Y}, \bar{\boldsymbol{e}}) = \sum_{z,e} \prod_{\phi \in \mathcal{F}} \phi \times \delta(\boldsymbol{E}, \bar{\boldsymbol{e}})$$

变量消除算法的复杂度

- 变量消除算法的两个基本操作：

$$\psi_k(\boldsymbol{X}_k) = \prod_i \phi_i$$

$$\tau_k(\boldsymbol{X}_k - \{Z\}) = \sum_Z \psi_k(\boldsymbol{X}_k)$$

令N_i是因子ψ_i中表值的个数，$N_{max} = max_i N_i$

m + n个因子：m个初始因子ϕ，n个因子τ_i

乘法得到ψ_i的代价最多是N_i，即N_i个乘法运算

乘法总代价是$(n+m)N_i \leq (n+m)N_{max} = O(mN_{max})$

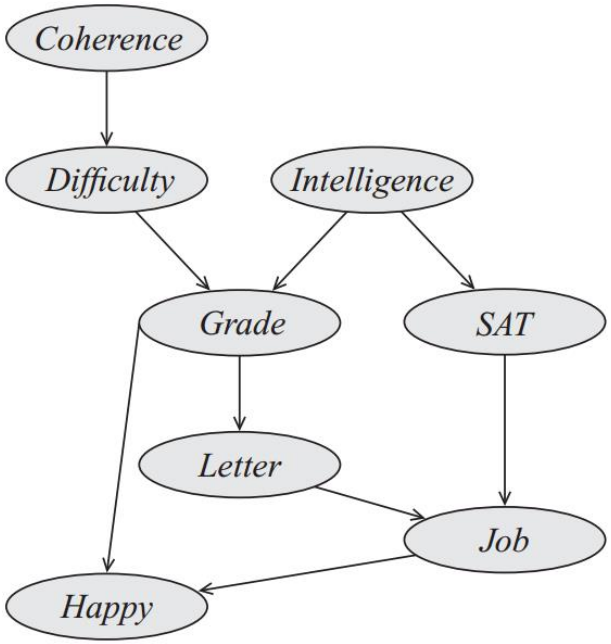
每次边缘化的代价是N_i，加法的总代价是nN_{max}

综上所述，总运算量是$O(mN_{max})$

但是！指数爆炸不可避免，是因为因子ψ_i的潜在规模。

如果每个变量值个数不超过v，并且ψ_i包含k_i个变量的辖域，那么$N_i \leq v^{k_i}$

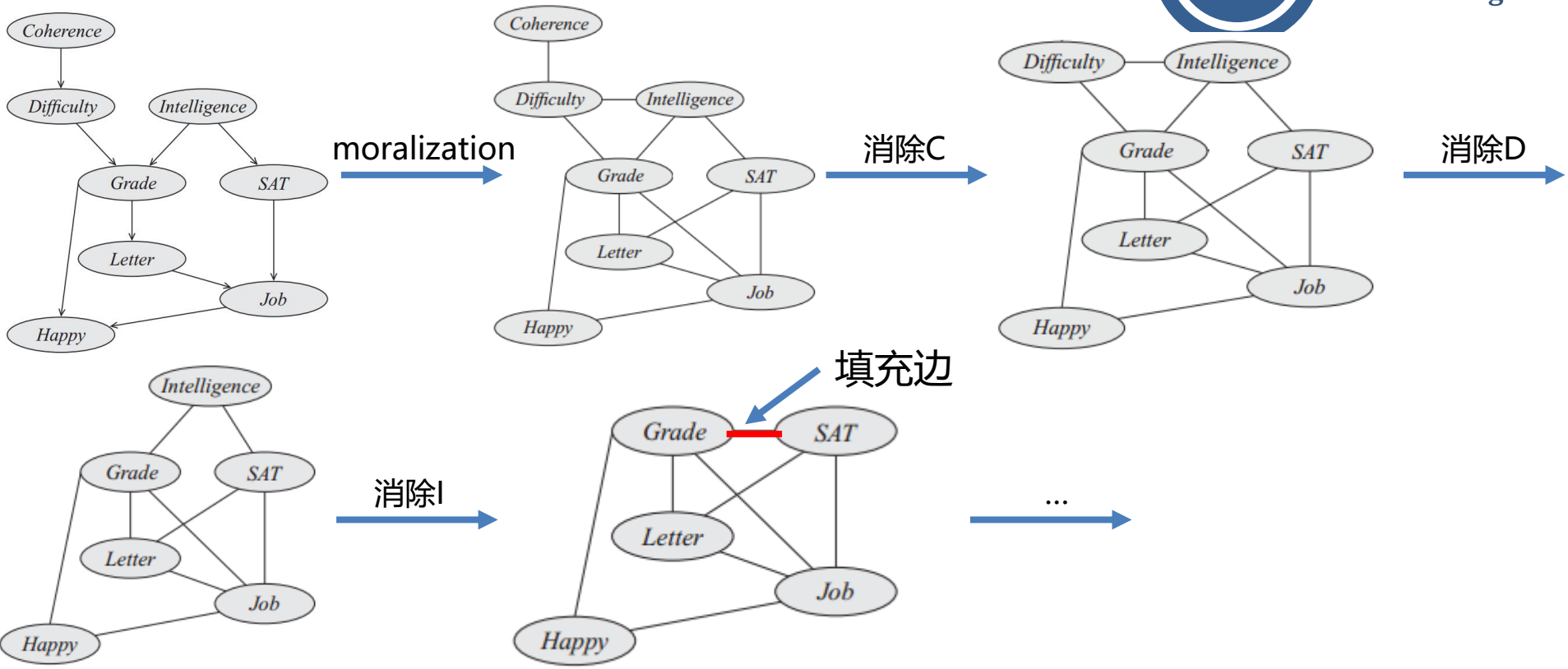
*消除顺序



Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C), \phi_D(D,C)$	C, D	$\tau_1(D)$
2	D	$\phi_G(G,I,D), \tau_1(D)$	G, I, D	$\tau_2(G,I)$
3	I	$\phi_I(I), \phi_S(S,I), \tau_2(G,I)$	G, S, I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H, G, J	$\tau_4(G,J)$
5	G	$\tau_4(G,J), \tau_3(G,S), \phi_L(L,G)$	G, J, L, S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S), \phi_J(J,L,S)$	J, L, S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J, L	$\tau_7(J)$

Step	Variable eliminated	Factors used	Variables involved	New factor
1	G	$\phi_G(G,I,D), \phi_L(L,G), \phi_H(H,G,J)$	G, I, D, L, J, H	$\tau_1(I,D,L,J,H)$
2	I	$\phi_I(I), \phi_S(S,I), \tau_1(I,D,L,S,J,H)$	S, I, D, L, J, H	$\tau_2(D,L,S,J,H)$
3	S	$\phi_J(J,L,S), \tau_2(D,L,S,J,H)$	D, L, S, J, H	$\tau_3(D,L,J,H)$
4	L	$\tau_3(D,L,J,H)$	D, L, J, H	$\tau_4(D,J,H)$
5	H	$\tau_4(D,J,H)$	D, J, H	$\tau_5(D,J)$
6	C	$\tau_5(D,J), \phi_C(C), \phi_D(D,C)$	D, J, C	$\tau_6(D,J)$
7	D	$\tau_6(D,J)$	D, J	$\tau_7(J)$

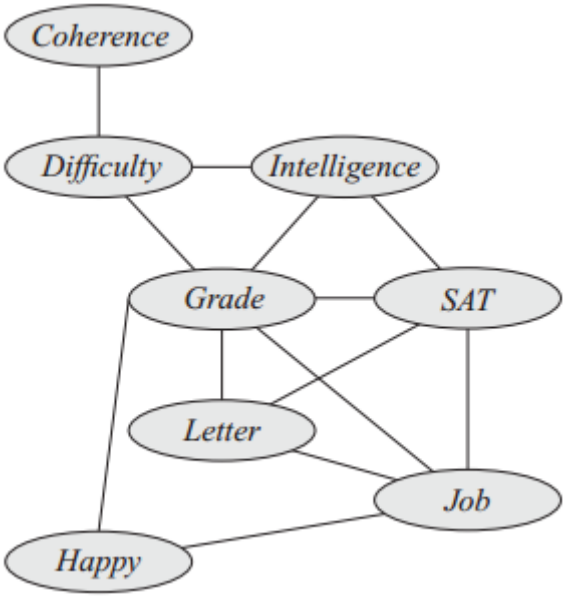
有趣现象



Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C)$, $\phi_D(D,C)$	C,D	$\tau_1(D)$
2	D	$\phi_G(G,I,D)$, $\tau_1(D)$	G,I,D	$\tau_2(G,I)$
3	I	$\phi_I(I)$, $\phi_S(S,I)$, $\tau_2(G,I)$	G,S,I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H,G,J	$\tau_4(G,J)$
5	G	$\tau_4(G,J)$, $\tau_3(G,S)$, $\phi_L(L,G)$	G,J,L,S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S)$, $\phi_J(J,L,S)$	J,L,S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J,L	$\tau_7(J)$

导出图

- 定义：令Φ是$\mathcal{X}$ = {X_1,...,X_n}上的一个因子集，$\prec$ 是某个子集X ∈ $\mathcal{X}$的消除顺序。**导出图**$I_{\Phi,\prec}$是$\mathcal{X}$上的一个无向图，其中，如果X_i和X_j同时出现在VE算法以$\prec$为消除顺序产生的中间因子ψ中，则它们由一条边链接。



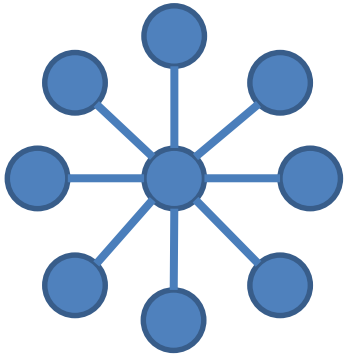
消除顺序为C, D, I, H, G, S, L的一个导出图

Moralization + 填充边 -> 导出图

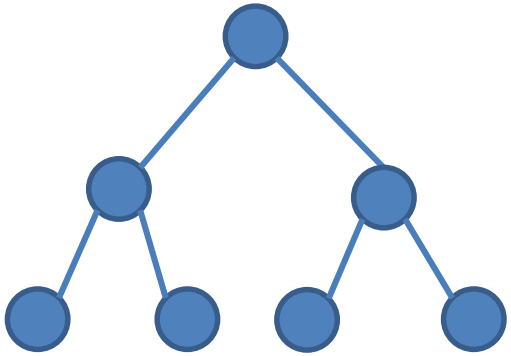
不同的消除顺序，导致不同的导出图（填充边）

导出图

- Star:

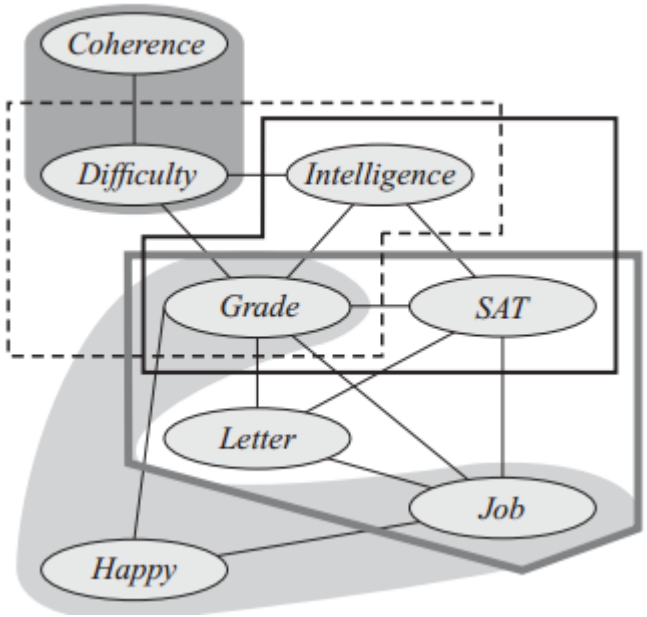


- Tree:



导出图

- 定理：令$I_{\Phi,\prec}$是因子集Φ和某个消除顺序$\prec$的导出图，那么
  1. 变量消除过程中产生的每个因子的辖域是$I_{\Phi,\prec}$中的一个团
  2. $I_{\Phi,\prec}$中的每个最大团在计算中是某个中间因子的辖域



导出图中的团

Step	Variable eliminated	Factors used	Variables involved	New factor
1	C	$\phi_C(C), \phi_D(D,C)$	C,D	$\tau_1(D)$
2	D	$\phi_G(G,I,D), \tau_1(D)$	G,I,D	$\tau_2(G,I)$
3	I	$\phi_I(I), \phi_S(S,I), \tau_2(G,I)$	G,S,I	$\tau_3(G,S)$
4	H	$\phi_H(H,G,J)$	H,G,J	$\tau_4(G,J)$
5	G	$\tau_4(G,J), \tau_3(G,S), \phi_L(L,G)$	G,J,L,S	$\tau_5(J,L,S)$
6	S	$\tau_5(J,L,S), \phi_J(J,L,S)$	J,L,S	$\tau_6(J,L)$
7	L	$\tau_6(J,L)$	J,L	$\tau_7(J)$

计算中使用的每个因子ψ，对应于图$I_{\Phi,\prec}$的一个完全子图

BP算法，团树

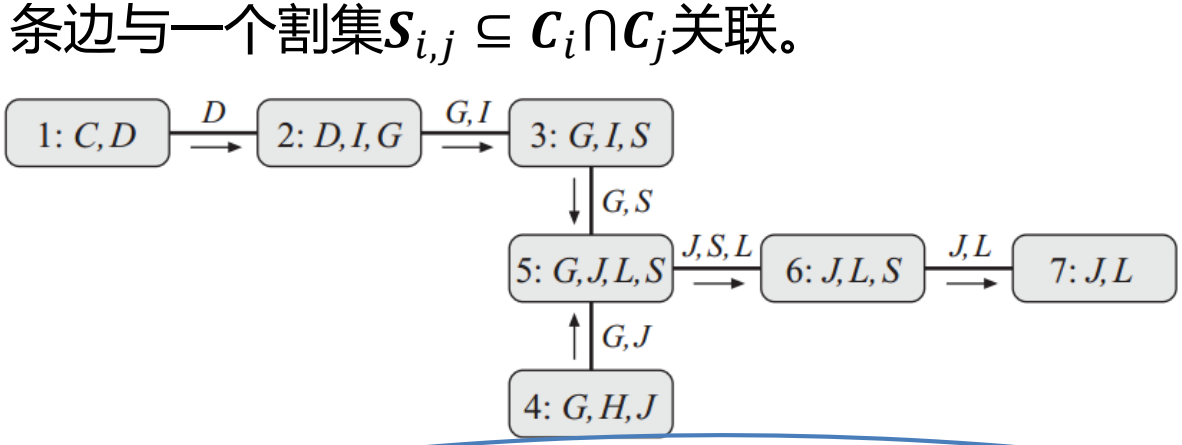
变量消除算法中，每一步通过已有的因子相乘来产生新的因子ψ_i，然后在ψ_i中消除一个变量，产生新因子τ_i，接着用τ_i产生下一个因子ψ_j

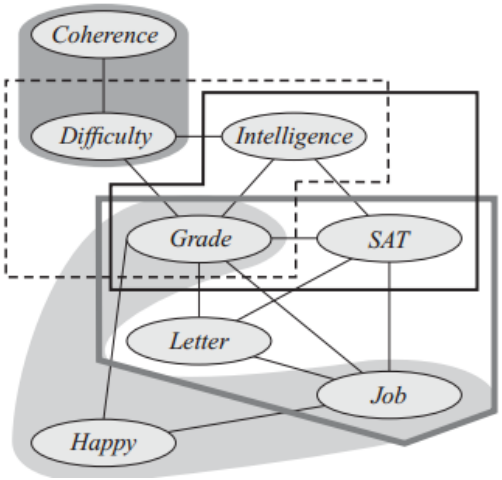
基于这种运算，接下来考虑另一种观点：

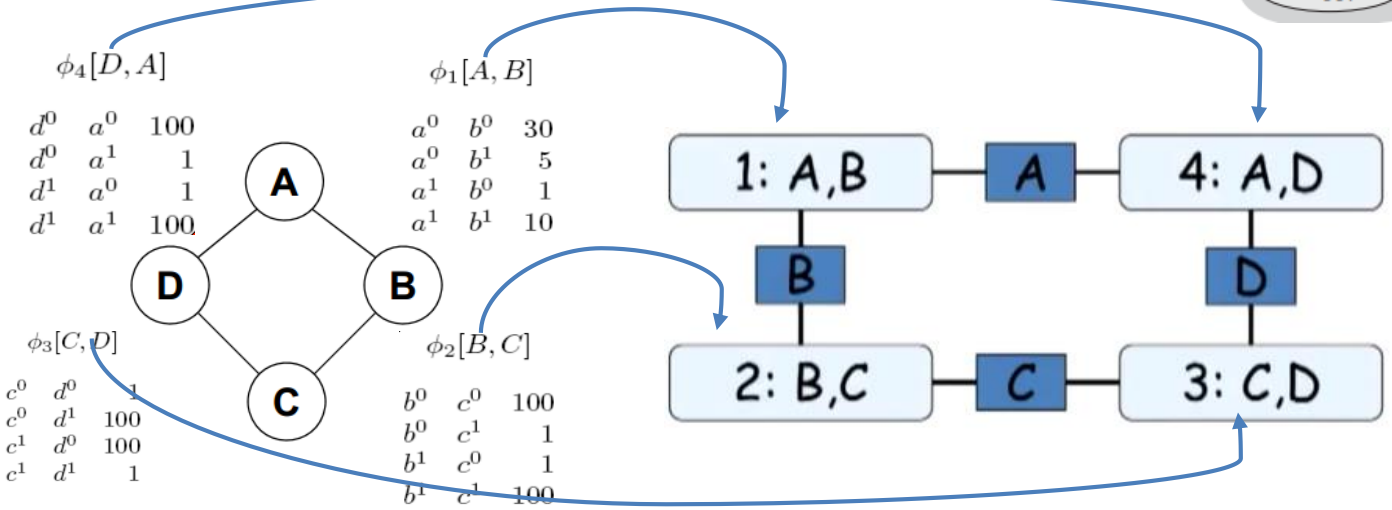
把因子ψ_i作为一种计算的数据结构考虑，它携带者由其他因子ψ_j产生的“消息”τ_j，并且产生用于其他因子ψ_l的消息τ_i

聚类图

- 定义：$\mathcal{X}$上的关于是因子集Φ 的**聚类图**$\mathcal{U}$是一个无向图，它的每个节点i与一个子集$\boldsymbol{C}_i \subseteq \mathcal{X}$关联。聚类图必须是族保存的——每个因子$\phi \subseteq \Phi$必须与一个聚类图$\boldsymbol{C}_i$关联，表示为$\alpha(\phi)$，使得$Scope[\phi] \subseteq \boldsymbol{C}_i$。一对聚类$\boldsymbol{C}_i$和$\boldsymbol{C}_j$之间的每条边与一个割集$\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$关联。

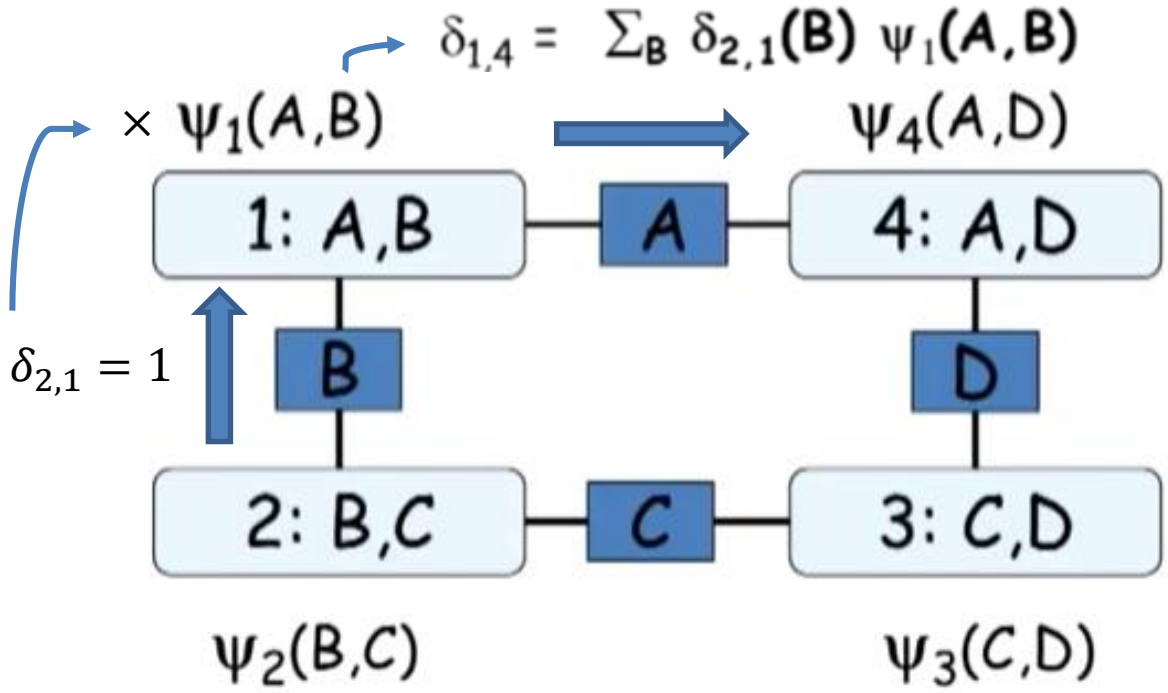






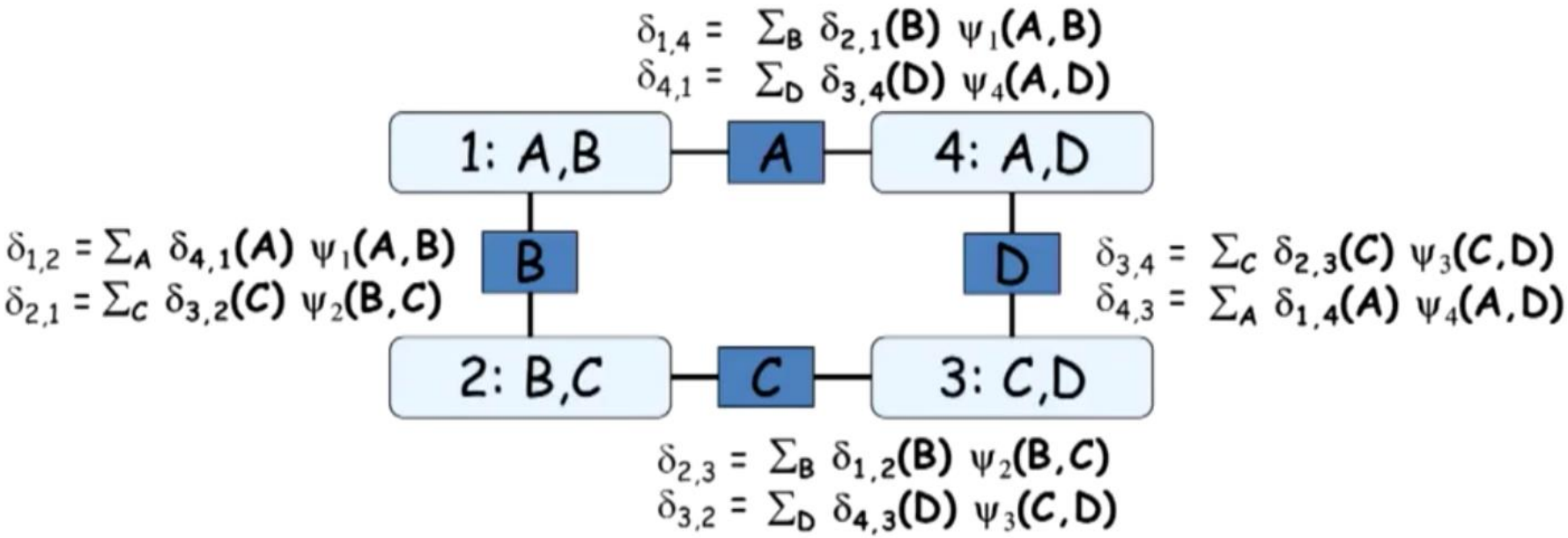
消息传递

- 初始化势能$\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$。在此例中，$\psi_i = \phi_i$
- 初始化消息$\delta_i = 1$
- 接收消息的团，接收其上游的消息，将消息与自己的势能相乘，并边缘化后形成新的消息，传给下游的团
- 传递消息直到收敛



消息传递

- 初始化势能$\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$。在此例中，$\psi_i = \phi_i$
- 初始化消息$\delta_i = 1$
- 接收消息的团，接收其上游的消息，将消息与自己的势能相乘，并边缘化后形成新的消息，传给下游的团
- 传递消息直到收敛



置信传播算法

- 将因子$\phi_k \in \Phi$指定给团$\boldsymbol{C}_{\alpha(k)}$
- 初始化势能$\psi_i(\boldsymbol{C}_i) = \prod_{k:\alpha(k)=i} \phi_k$
- 将所有消息δ_i初始化为1
- 重复：

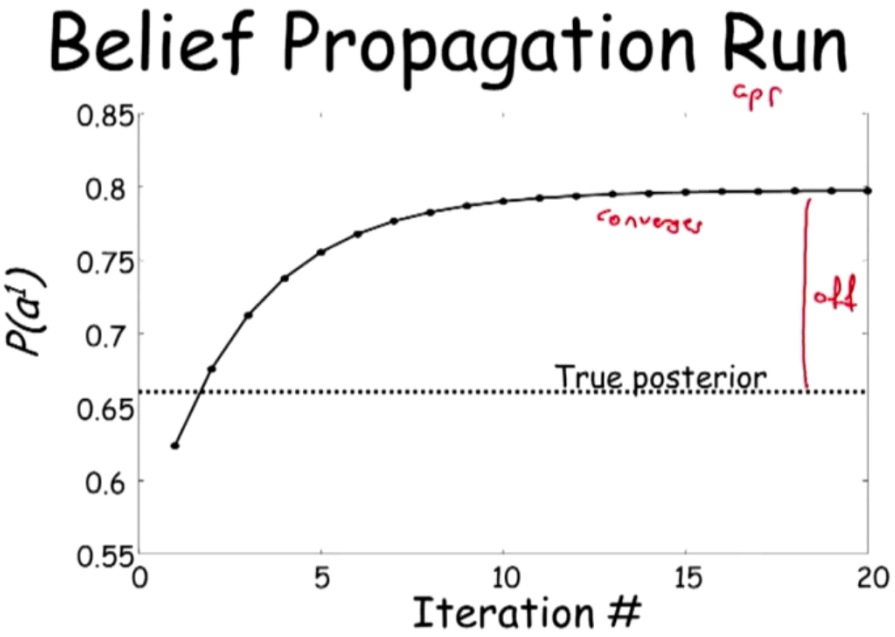
  -选择边(i,j)传播消息

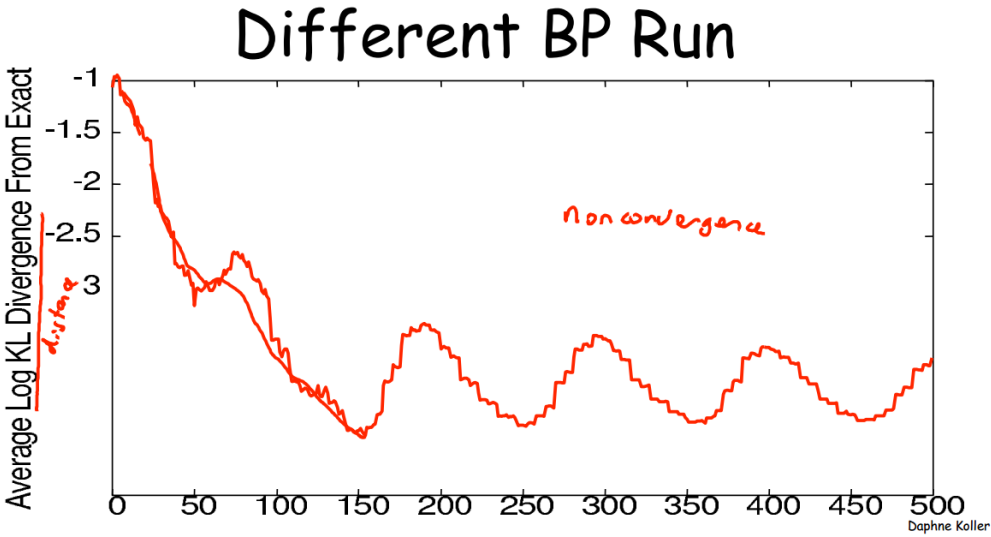
$$\delta_{i\to j}(\boldsymbol{S}_{i,j}) = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \times \prod_{k\in(\mathcal{N}_i - \{j\})} \delta_{k\to i}$$

- 计算团置信

$$\beta_i \leftarrow \psi_i \cdot \prod_{k\in \mathrm{Nb}_i} \delta_{k\to i}$$

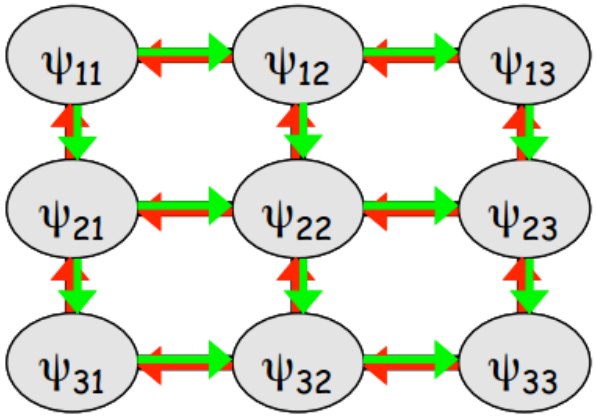
置信传播算法

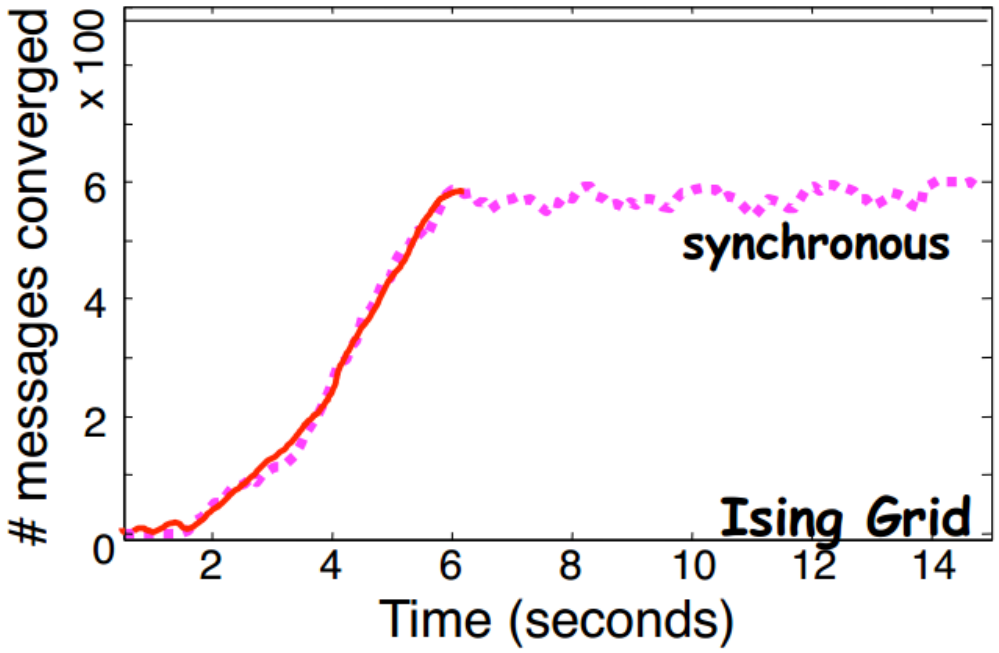

Belief Propagation Run
P(a^1)
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0
5
10
15
20
Iteration #
True posterior


Different BP Run
Average Log KL Divergence From Exact
-1
-1.5
-2
-2.5
-3
0
50
100
150
200
250
300
350
400
450
500
Daphne Koller

置信传播算法变种

Synchronous BP:
all messages are updated in parallel

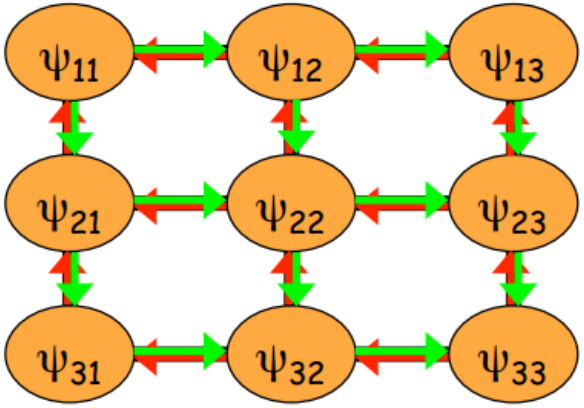


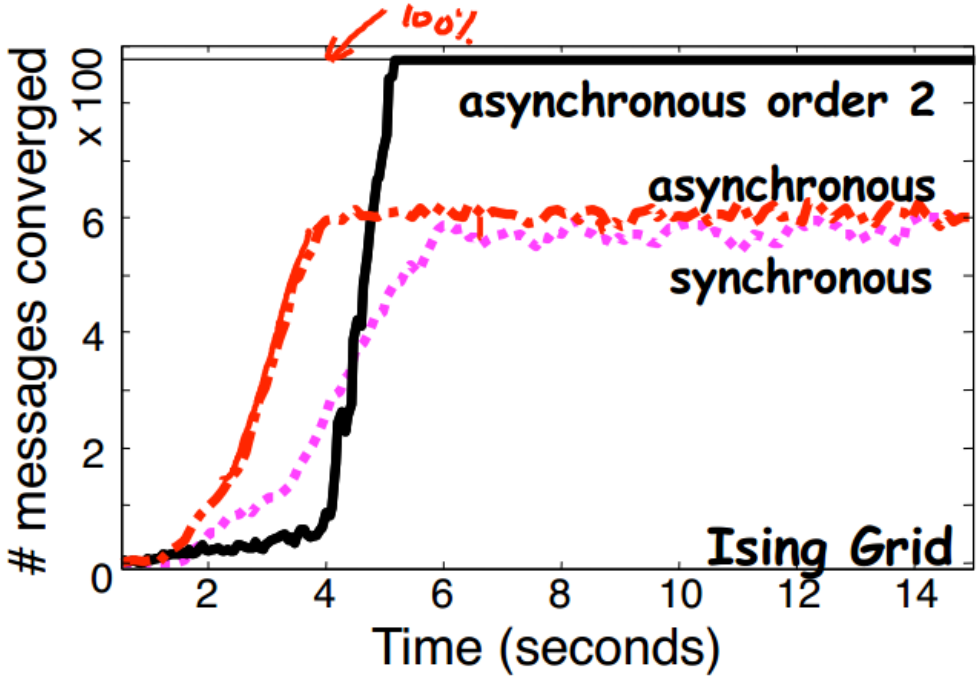


置信传播算法变种

Asynchronous BP:

Messages are updated one at a time

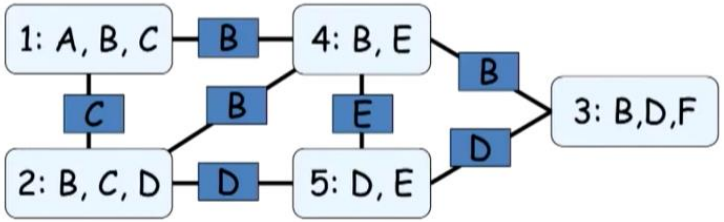




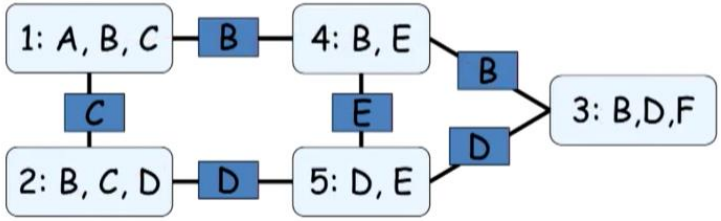
聚类图性质

- **族保存**：每个因子$\phi \subseteq \Phi$必须与一个聚类图$\boldsymbol{C}_i$关联，表示为$\alpha(\phi)$，使得$Scope[\phi] \subseteq \boldsymbol{C}_i$。一对聚类$\boldsymbol{C}_i$和$\boldsymbol{C}_j$之间的每条边与一个割集$\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$关联。

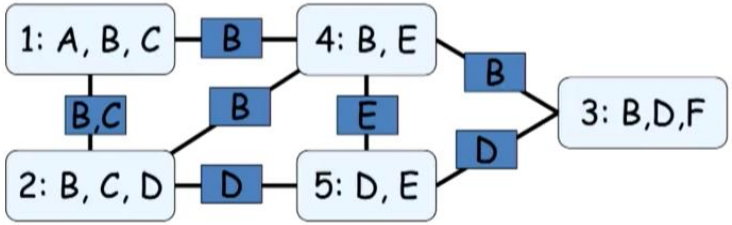
- **执行相交性**：对于一对簇$\boldsymbol{C}_i$与$\boldsymbol{C}_j$，以及变量X使得$X \in \boldsymbol{C}_i \cap \boldsymbol{C}_j$，在$\boldsymbol{C}_i$和$\boldsymbol{C}_j$之间存在一条唯一的路径，使得变量$X$存在于该路径上的所有簇中。



合法的聚类图



不符合“存在”



不符合“唯一”

BP算法性质

- 定义：如果

$$\sum_{C_i - S_{i,j}} \beta_i(C_i) = \sum_{C_j - S_{i,j}} \beta_j(C_j)$$

那么两个相邻的团C_i和C_j是**校准**的。

- 如果聚类图中，所有成对的相邻团都是校准的，那么该聚类图是**校准**的。

- 收敛 $\Longrightarrow$ 校准

收敛：$\delta_{i\to j}(S_{i,j}) = \delta'_{i\to j}(S_{i,j})$

$$\delta'_{i\to j}(S_{i,j}) = \sum_{C_i - S_{i,j}} \left(\psi_i \times \prod_{k\in(\mathcal{N}_i - \{j\})} \delta_{k\to i} \right) = \sum_{C_i - S_{i,j}} \frac{\beta_i(C_i)}{\delta_{j\to i}(S_{i,j})}$$

$$\beta_i = \psi_i \cdot \prod_{k\in \mathrm{Nb}_i} \delta_{k\to i}$$

$$\delta_{j\to i}(S_{i,j})\delta_{i\to j}(S_{i,j}) = \sum_{C_i - S_{i,j}} \beta_i(C_i)$$

$$\delta_{j\to i}(S_{i,j})\delta_{i\to j}(S_{i,j}) = \sum_{C_j - S_{i,j}} \beta_j(C_j)$$

$$\Longrightarrow \sum_{C_i - S_{i,j}} \beta_i(C_i) = \sum_{C_j - S_{i,j}} \beta_j(C_j)$$

割集置信 $\mu_{i,j}(S_{i,j}) = \delta_{j\to i}\delta_{i\to j} = \sum_{C_j - S_{i,j}} \beta_j(C_j)$

BP算法性质

➢ 再参数化

在**校准**的聚类图的算法**收敛**处，下式成立

$$\tilde{P}_{\Phi}(\mathcal{X}) = \frac{\prod_{i\in\mathcal{V}_T} \beta_i(\boldsymbol{C}_i)}{\prod_{(i-j)\in\mathcal{E}_T} \mu_{i,j}(\boldsymbol{S}_{i,j})}.$$

证明：

根据 $\beta_i = \psi_i \cdot \prod_{k\in \mathrm{Nb}_i} \delta_{k\to i}$

原始分子可以写成

$$\prod_{i\in\mathcal{V}_T} \psi_i(\boldsymbol{C}_i) \prod_{k\in \mathrm{Nb}_i} \delta_{k\to i}.$$

根据

$$\mu_{i,j}(\boldsymbol{S}_{i,j}) = \delta_{j\to i}\delta_{i\to j}$$

分母改写为

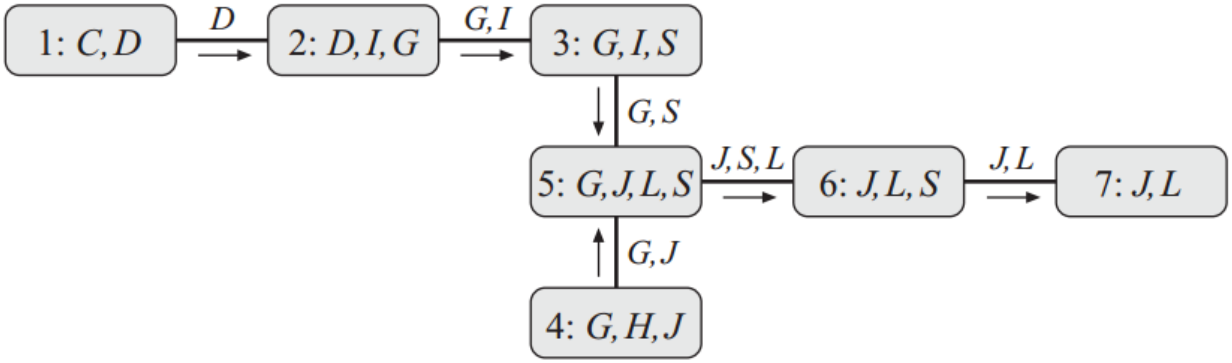
$$\prod_{(i-j)\in\mathcal{E}_T} \delta_{i\to j}\delta_{j\to i}.$$

$$\Longrightarrow \quad \prod_{i\in\mathcal{V}_T} \psi_i(\boldsymbol{C}_i) = \tilde{P}_{\Phi}.$$

聚类树

- 特殊地，变量消除算法的一个执行，得到的聚类图一定是一棵树。
- 定理：令$\mathcal{T}$是由变量消除算法在某个因子集Φ上产生的聚类树，则$\mathcal{T}$满足**执行相交性**。

- 命题：令$\mathcal{T}$是变量消除算法在某个因子集Φ上产生的聚类树。令$\boldsymbol{C}_i$和$\boldsymbol{C}_j$是两个相邻的簇，并且$\boldsymbol{C}_i$向$\boldsymbol{C}_j$发送消息τ_i，则消息τ_i的辖域恰好是$\boldsymbol{C}_i \cap \boldsymbol{C}_j$

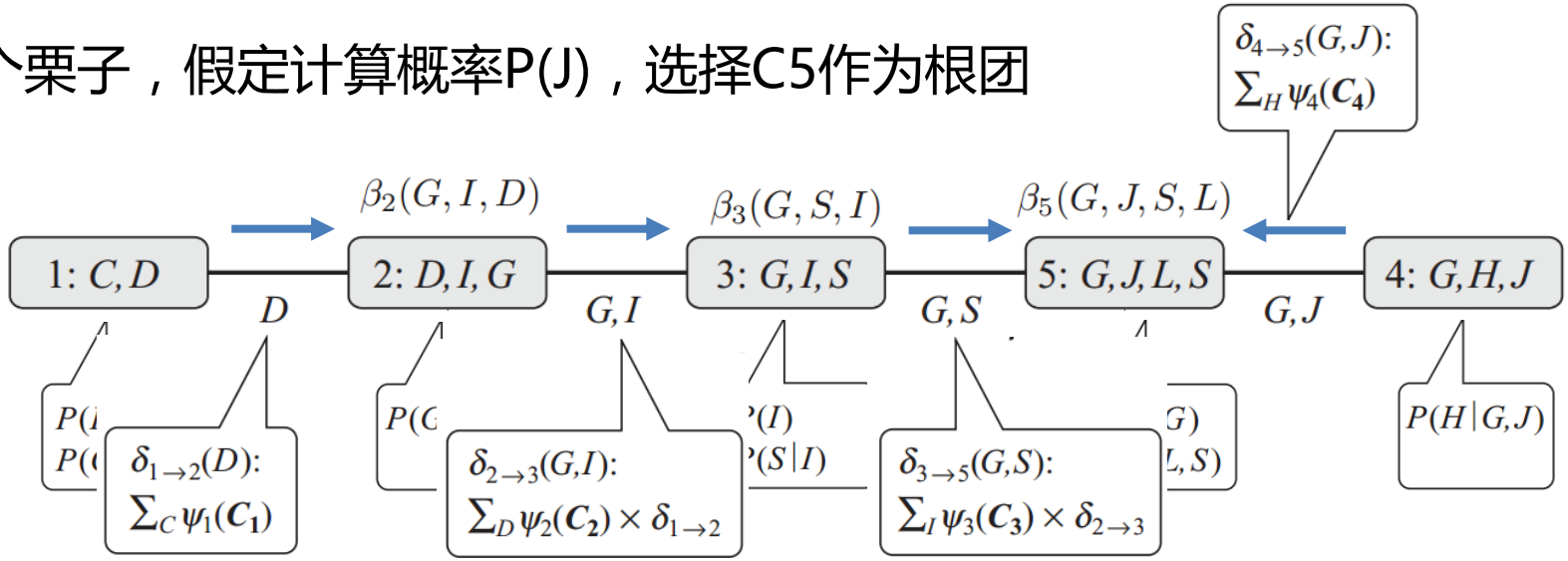


团树

- 定义：设Φ是$\mathcal{X}$上的因子集， Φ上满足**执行相交性**的聚类树，称为**团树**（clique tree）（也成为连接树或联合树），在团树的情况下，簇（cluster）也成为**团**（clique）

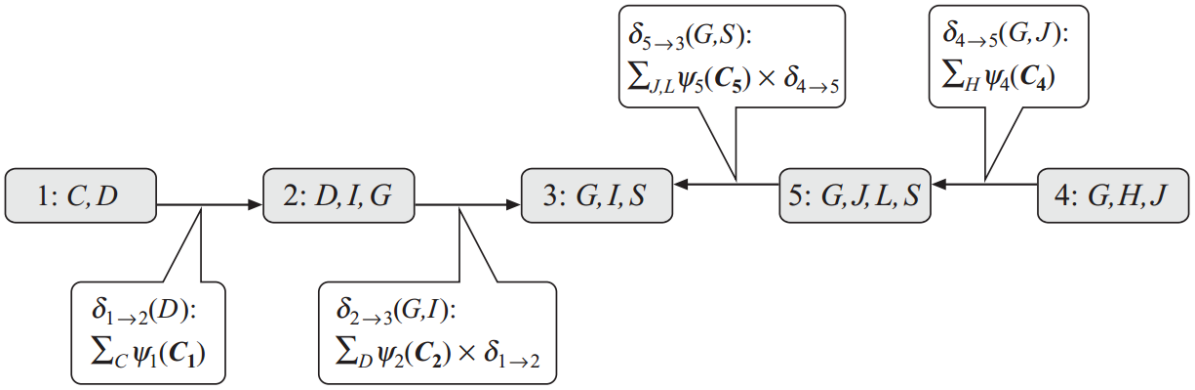
- 使用团树的好处：
  - 保证BP算法的**收敛性**（就绪—等待）
  - 保证BP算法的**正确性**（满足族保持性和执行相交性，自己下来证明）

- 举个栗子，假定计算概率P(J)，选择C5作为根团



团树

- 另一个顺序的执行。C3作为根团



- 注意到：

  在C1，C2，C4中，计算和消息都不变。

  在C5中，定义 $\beta_5(G,J,S,L) = \delta_{4\to5}(G,J)\cdot\psi_5(G,J,S,L).$

  并消除J和L后，将$\delta_{5\to3}(G,S)$作为消息发给C3

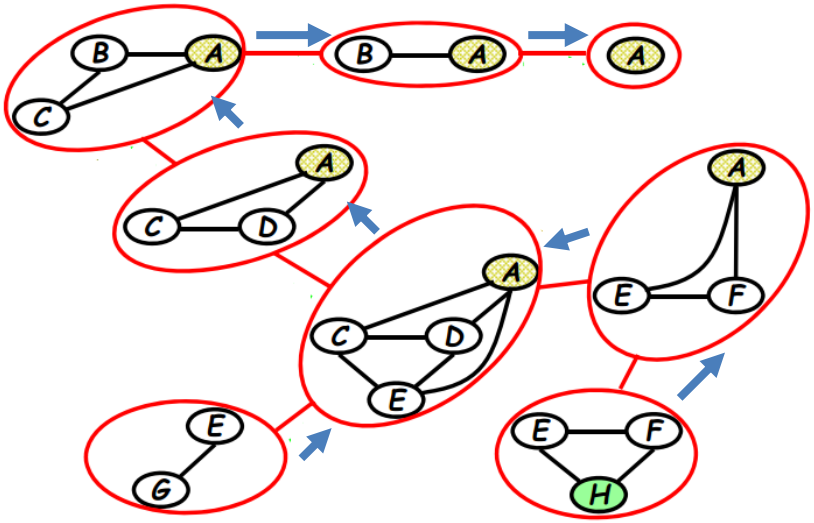
  C3接收来自于其邻居发送的消息，与自己的初始势能相乘，计算出团置信

团树消息传递算法

- 在团树消息传递过程中，不一定是同步的，消息可以异步的传播，只要满足约束：

团在给其上游近邻发出消息之前，必须收集所有来自其下游近邻传入的消息

- 如果一个团已经完成了所需消息的收集，那么该团是**就绪**的
- 叶子结点是就绪的



Algorithm 10.1 Upward pass of variable elimination in clique tree

Procedure CTree-SP-Upward (
  Φ, // Set of factors
  $\mathcal{T}$, // Clique tree over Φ
  α, // Initial assignment of factors to cliques
  C_r // Some selected root clique
)
1 Initialize-Cliques
2 **while** C_r is not ready
3 Let C_i be a ready clique
4 $\delta_{i \to p_r(i)}(S_{i,p_r(i)}) \leftarrow$ SP-Message$(i, p_r(i))$
5 $\beta_r \leftarrow \psi_r \cdot \prod_{k \in \text{Nb}_{C_r}} \delta_{k \to r}$
6 **return** β_r

Procedure Initialize-Cliques (
)
1 **for** each clique C_i
2 $\psi_i(C_i) \leftarrow \prod_{\phi_j : \alpha(\phi_j)=i} \phi_j$
3

Procedure SP-Message (
  i, // sending clique
  j // receiving clique
)
1 $\psi(C_i) \leftarrow \psi_i \cdot \prod_{k \in (\text{Nb}_i - \{j\})} \delta_{k \to i}$
2 $\tau(S_{i,j}) \leftarrow \sum_{C_i - S_{i,j}} \psi(C_i)$
3 **return** $\tau(S_{i,j})$

团树校准

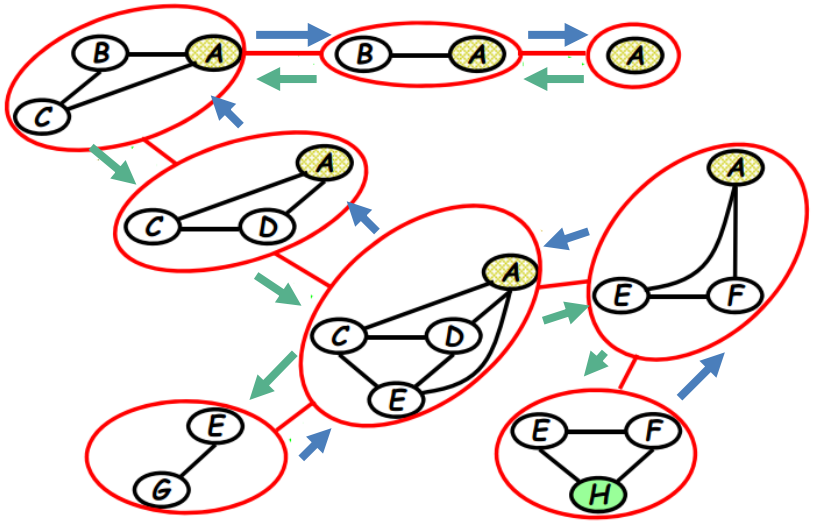
- 有时我们不只关注一个团的查询，而是多个团。为方便日后分布的表示和查询，我们希望团树是校准的。即求得所有团的置信因子β。算法如下：

Algorithm 10.2 Calibration using sum-product message passing in a clique tree

Procedure CTree-SP-Calibrate (
  Φ, // Set of factors
  $\mathcal{T}$ // Clique tree over Φ
)

1 Initialize-Cliques
2 **while** exist i, j such that i is ready to transmit to j
3 $\delta_{i \to j}(\boldsymbol{S}_{i,j}) \leftarrow$ SP-Message(i, j)
4 **for** each clique i
5 $\beta_i \leftarrow \psi_i \cdot \prod_{k \in \text{Nb}_i} \delta_{k \to i}$
6 **return** $\{\beta_i\}$

- 算法异步定义
- 过程由**向上传递**和**向下传递**组成



校准的团树作为分布

- 由于**校准**团树能保证收敛性和正确性，其**再参数化**性质成立：

$$\tilde{P}_\Phi(\mathcal{X}) = \frac{\prod_{i \in \mathcal{V}_\mathcal{T}} \beta_i(\boldsymbol{C}_i)}{\prod_{(i-j) \in \mathcal{E}_\mathcal{T}} \mu_{i,j}(\boldsymbol{S}_{i,j})}.$$

- 由此，团树可以看作是联合分布的另一种表示形式
- 一个例子：

A	B	D	$\max_C$
a^0	b^0	d^0	600,000
a^0	b^0	d^1	300,030
a^0	b^1	d^0	5,000,500
a^0	b^1	d^1	1,000
a^1	b^0	d^0	200
a^1	b^0	d^1	1,000,100
a^1	b^1	d^0	100,010
a^1	b^1	d^1	200,000

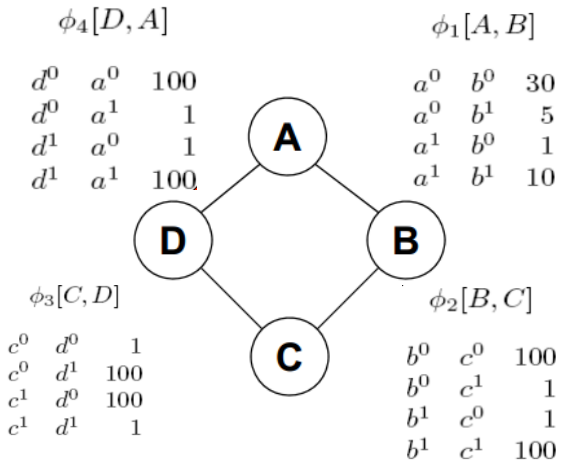
$\beta_1(A, B, D)$

Assignment		$\max_{A,C}$
b^0	d^0	600,200
b^0	d^1	1,300,130
b^1	d^0	5,100,510
b^1	d^1	201,000

$\mu_{1,2}(B, D)$

Assignment			$\max_A$
b^0	c^0	d^0	300,100
b^0	c^0	d^1	1,300,000
b^0	c^1	d^0	300,100
b^0	c^1	d^1	130
b^1	c^0	d^0	510
b^1	c^0	d^1	100,500
b^1	c^1	d^0	5,100,000
b^1	c^1	d^1	100,500

$\beta_2(B, C, D)$



$$\tilde{P}_\Phi(a^1, b^0, c^1, d^0) = \frac{\beta_1(a^1, b^0, d^0)\beta_2(b^0, c^1, d^0)}{\mu_{1,2}(b^0, d^0)} = \frac{200 \cdot 300,100}{600,200} = 100$$

使用除法的消息传递

- 见概率图模型10.3.1节

$$\delta_{i\to j} = \frac{\sum \boldsymbol{C}_i - \boldsymbol{S}_{i,j}\ \beta_i}{\delta_{j\to i}}.$$

回答查询

- 查询的变量都在一个团中：

  so easy！毕竟在校准团树中，$\beta_i(\boldsymbol{C}_i) = \tilde{P}(\boldsymbol{C}_i)$，do whatever you want！

- 增量更新

  假设分布：$\tilde{P}_\Phi(\mathcal{X}) = \prod_{\phi \in \Phi} \phi$

  存在证据z，那么：$\tilde{P}_\Phi(\mathcal{X}, Z = z) = \mathbf{1}\{Z = z\} \prod_{\phi \in \Phi} \phi$

  令：$\tilde{P}'_\Phi(\mathcal{X}) = \tilde{P}_\Phi(\mathcal{X}, Z = z)$

  假设有一个用团树不变量表示该分部的团树（校准或非校准的）

$$\tilde{P}_\Phi(\mathcal{X}) = \frac{\prod_{i \in \mathcal{V}_\mathcal{T}} \beta_i(\boldsymbol{C}_i)}{\prod_{(i-j) \in \mathcal{E}_\mathcal{T}} \mu_{i,j}(\boldsymbol{S}_{i,j})}.$$

  $\tilde{P}'_\Phi(\mathcal{X})$可表示为：$\tilde{P}'_\Phi(\mathcal{X}) = \mathbf{1}\{Z = z\} \frac{\prod_{i \in \mathcal{V}_\mathcal{T}} \beta_i(\boldsymbol{C}_i)}{\prod_{(i-j) \in \mathcal{E}_\mathcal{T}} \mu_{i,j}(\boldsymbol{S}_{i,j})}$

- 如果团树在引入证据前已经校准，那么C_i已经吸收了图中所有信息。为了获得$\tilde{P}'_\Phi(\boldsymbol{C}_i)$不需要额外的消息传递。然而，其他团仍需要用新的消息来更新。

回答查询

- 团外查询：考虑查询P(**Y**|e)，其中变量**Y**不会一起出现在一个团中
  naïve方法：强行构建一个团树，其中团包含**Y**的所有变量
  another way：在校准的团树上执行变量消除

- 如果一棵团树是校准的，那么该团树的任意子树也是校准的

- 例子：考虑如下简单团树，并假定该团树已经得到校准，所以可以再参数化表示该分布。现计算$\tilde{P}_\Phi(B,D)$

$$\tilde{P}_\Phi(B,C,D) = \frac{\beta_2(B,C)\beta_3(C,D)}{\mu_{2,3}(C)}$$

$$\begin{aligned}\tilde{P}_\Phi(B,D) &= \sum_C \tilde{P}_\Phi(B,C,D) \\ &= \sum_C \frac{\beta_2(B,C)\beta_3(C,D)}{\mu_{2,3}(C)} \\ &= \sum_C \tilde{P}_\Phi(B \mid C)\tilde{P}_\Phi(C,D),\end{aligned}$$

构建一棵团树

- 源自变量消除的团树（见概率图模型10.4.1节）
- 源自弦图的团树（见概率图模型10.4.2节）

Thanks

Q & S