

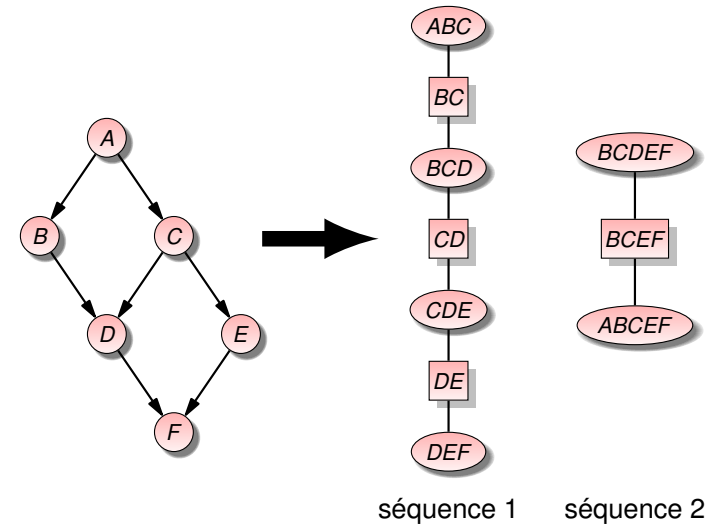
cours 6 Construction de l'arbre de jonction

Aix-Marseille université
Initiative d'excellence
Master SID — Raisonement dans l'incertain

©CG(2023)

Toutes les séquences d'élimination ne sont pas égales

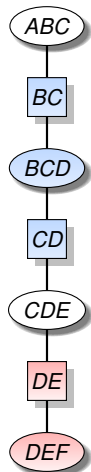
Séquence 1 : A, B, C, F, D, E Séquence 2 : D, C, A, E, B, F



cours 6 Construction de l'arbre de jonction

1/21

De la propriété d'intersection courante



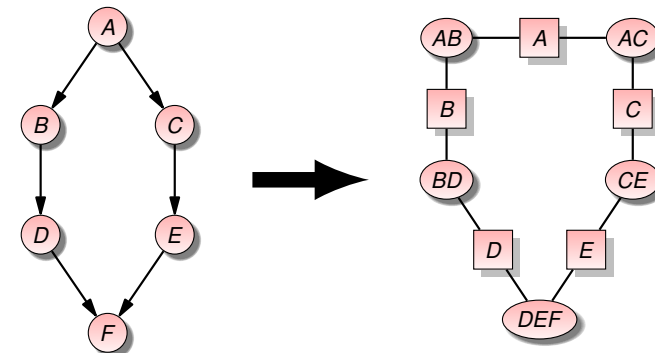
Propriété d'intersection courante

Soient C_1 et C_2 deux cliques quelconques de l'arbre de jonction et soit $S = C_1 \cap C_2 \neq \emptyset$. Alors sur toute chaîne reliant C_1 et C_2 , les cliques et séparateurs contiennent S

Théorème

L'algorithme de Shafer-Shenoy fonctionne avec n'importe quel graphe sans cycle vérifiant la propriété d'intersection courante (ce que l'on appelle un *join tree*)

Un graphe de jonction avec cycles



⚠ Si l'on n'y prend garde, des cycles peuvent exister bien que la propriété d'intersection courante soit vérifiée

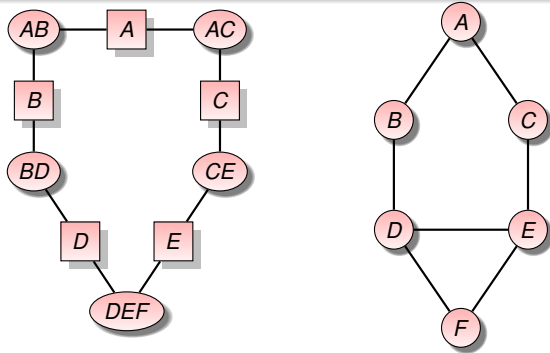
cours 6 Construction de l'arbre de jonction

2/21

cours 6 Construction de l'arbre de jonction

3/21

Graphe de jonction et triangulation (1/2)

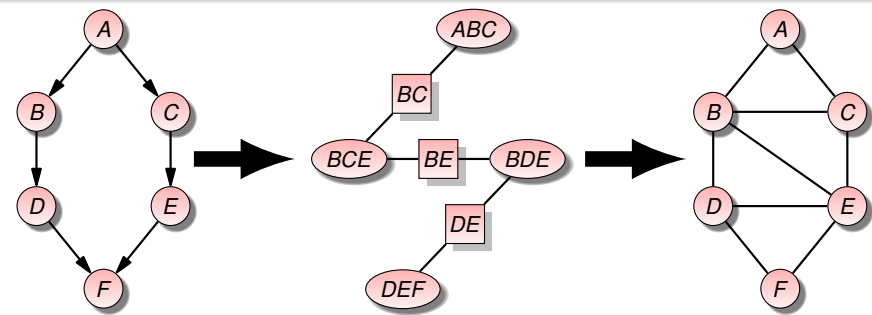


Triangulation

Un graphe non orienté est triangulé si et seulement si, pour tout cycle de longueur 4 ou plus, il existe une corde, c'est-à-dire une arête reliant deux nœuds non consécutifs du cycle

Exemple : le graphe ci-dessus n'est pas triangulé car le cycle A, B, D, E, C, A ne comporte pas de corde

Graphe de jonction et triangulation (2/2)



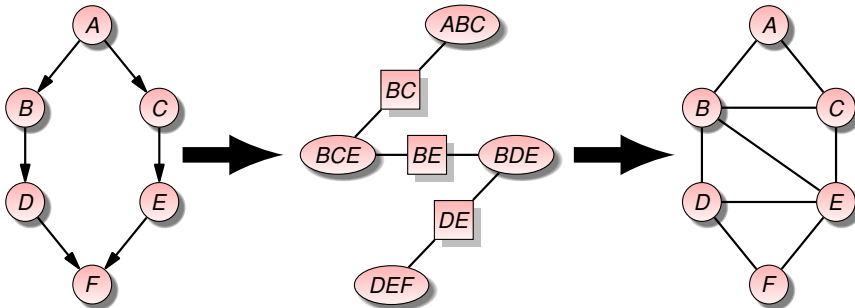
Proposition

il y a équivalence entre les deux assertions :

- 1 Le graphe de jonction est acyclique
- 2 Le graphe non orienté correspondant est triangulé

⇒ pour trouver un « bon » arbre de jonction, il faut trouver une « bonne » triangulation

Un peu de morale, ça ne fait pas de mal



Moralisation

relier tous les parents d'un même nœud, puis supprimer les orientations ⇒ le graphe moral.

⇒ les cliques pourront contenir l'ensemble des probabilités conditionnelles de la décomposition de la loi jointe

Recherche des triangulations optimales (1/2)

Proposition

[Rose 1970]

Un graphe non orienté est triangulé si et seulement si l'application des deux règles suivantes permet d'éliminer tous les nœuds X_i du graphe sans rajouter une seule arête :

- 1 on rajoute des arêtes entre tous les voisins du nœud X_i que l'on veut éliminer (on forme une clique)
- 2 on supprime X_i ainsi que les arêtes qui lui sont adjacentes du graphe

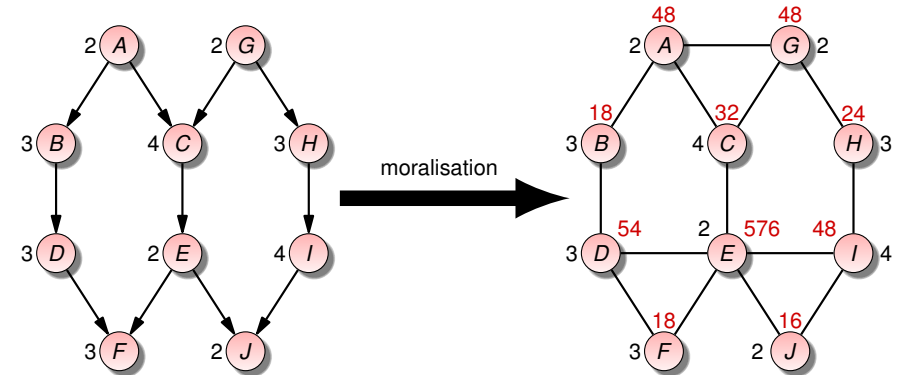
⇒ pour créer un join tree, il suffit de partir d'un graphe non orienté et d'appliquer, avec une certaine séquence d'élimination, les deux points ci-dessus

[Mellouli (87)] : Tout join tree « optimal » peut être construit à partir d'une séquence d'élimination

Recherche des triangulations optimales (2/2)

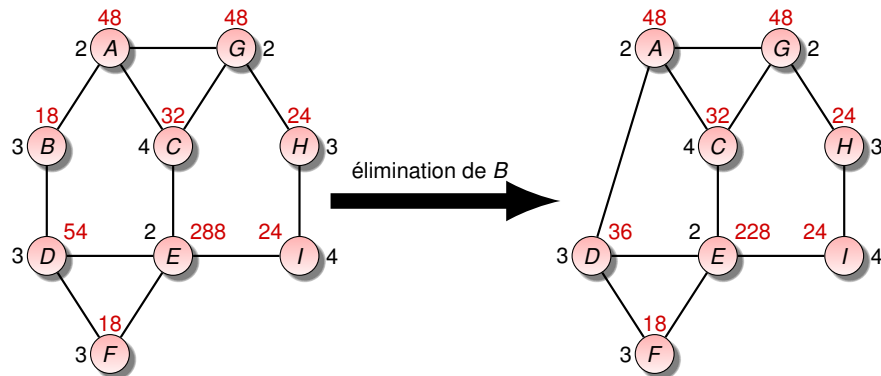
- *Arnborg et al. (87)* : trouver la triangulation optimale est NP-difficile \implies essayer de trouver des heuristiques
- ▶ *Kjærulff (90)* : un algorithme glouton rapide et efficace :
Soit un graphe non orienté (moral) $G = (X, E)$, $X = \{X_1, \dots, X_n\}$
 - 1 Associer à chaque X_i un « poids » égal au produit des modalités de X_i et de ses voisins
 - 2 éliminer le nœud X_i dont le poids est minimal (i.e., relier tous ses voisins de manière à former une clique C_i puis éliminer X_i et ses arêtes adjacentes)
 - 3 mettre à jour les poids des nœuds restants \implies les C_i sont les cliques (ellipses) du join tree
- ▶ *van den Eijkhof & Bodlaender (2002)* : “safe reductions”
 \implies élimination de variables avec garantie d’optimalité
- ▶ Autres algorithmes : *Becker & Geiger (96)* ; *Shoiket & Geiger (87)*

Exemple de création de join tree (1/5)



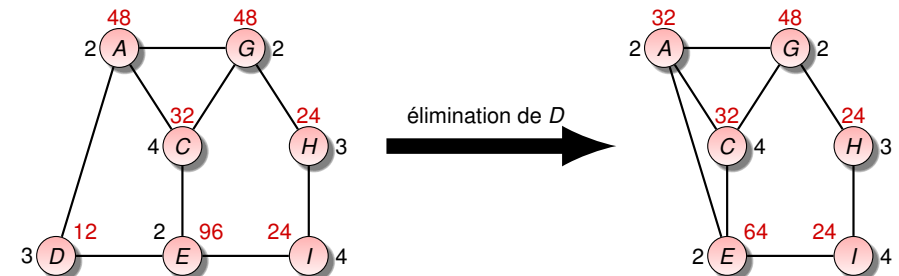
- ▶ Variable à éliminer : $J \implies$ clique EIJ

Exemple de création de join tree (2/5)



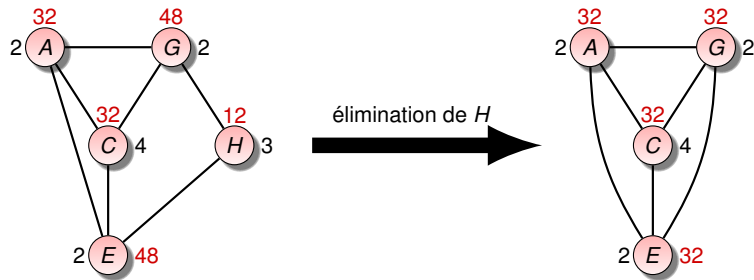
- ▶ première variable à éliminer : $B \implies$ clique ABD
- ▶ deuxième variable à éliminer : $F \implies$ clique DEF

Exemple de création de join tree (3/5)



- ▶ première variable à éliminer : $D \implies$ clique ADE
- ▶ deuxième variable à éliminer : $I \implies$ clique EHI

Exemple de création de join tree (4/5)

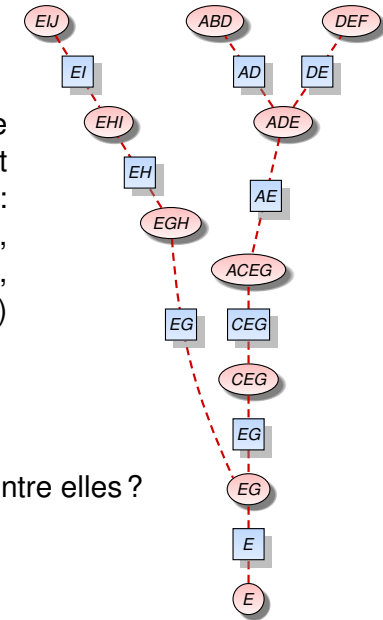


- ▶ première variable à éliminer : $H \Rightarrow$ clique EGH
- ▶ puis les autres variables peuvent être éliminées dans n'importe quel ordre puisqu'elles appartiennent toutes à la même clique :
 - $A \Rightarrow$ clique $ACEG$
 - $C \Rightarrow$ clique CEG
 - $G \Rightarrow$ clique EG
 - $E \Rightarrow$ clique E

Exemple de création de join tree (5/5)

Ensemble des cliques selon leur ordre de création (avec la variable dont l'élimination a créé la clique) :

EIJ (J), ABD (B), DEF (F), ADE (D),
 EHI (I), EGH (H), $ACEG$ (A),
 CEG (C), EG (G), E (E)



Problème : comment relier les cliques entre elles ?

Des cliques vers l'arbre d'élimination (1/2)

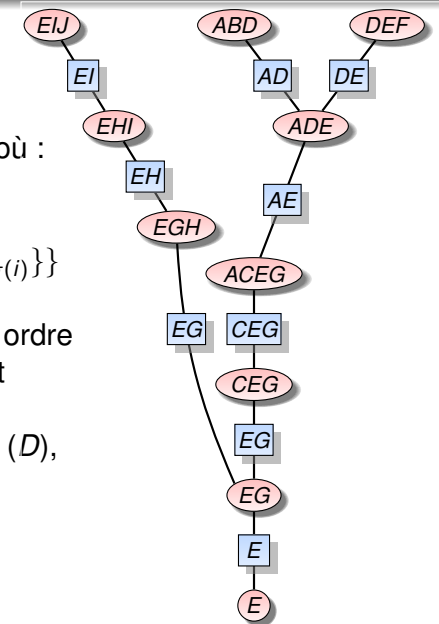
Définition de l'arbre d'élimination

- ▶ Soit $\sigma : \{1, \dots, n\} \mapsto \{1, \dots, n\}$ la permutation telle que les variables X_i sont éliminées dans l'ordre $X_{\sigma(1)}, \dots, X_{\sigma(n)}$
- ▶ Pour tout i , soit $D_{\sigma(i)}$ la clique créée au moment où $X_{\sigma(i)}$ est éliminée
- ▶ Arbre d'élimination : graphe $\mathcal{G} = (\mathcal{D}, \mathcal{E})$, où :
 - ▶ $\mathcal{D} = \{D_{\sigma(i)} : i \in \{1, \dots, n\}\}$,
 - ▶ $\mathcal{E} = \{(D_{\sigma(i)}, D_{\sigma(j)}) : 1 \leq i < n, j = \min\{k \neq i : X_{\sigma(k)} \in D_{\sigma(i)}\}\}$

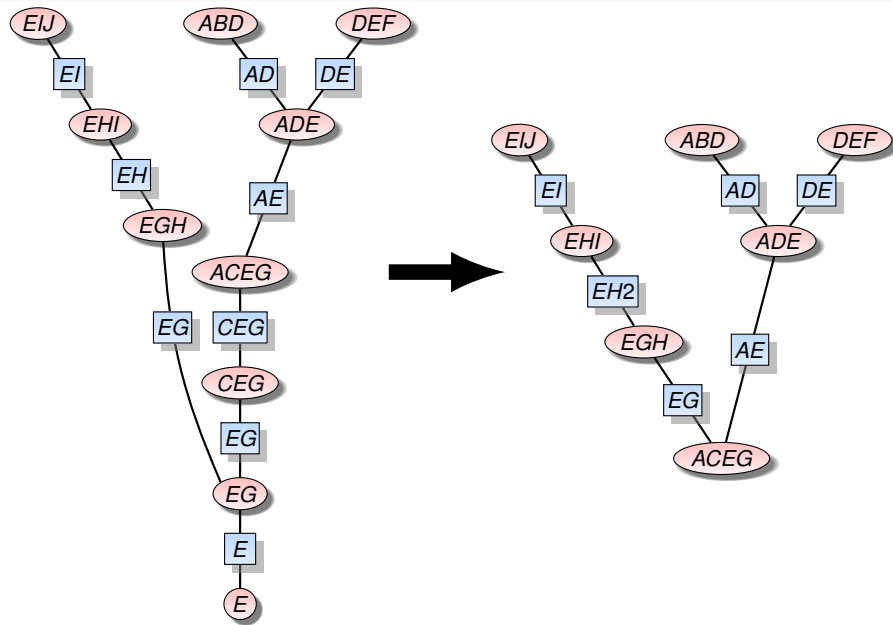
\Rightarrow Si l'on trie les nœuds X_i à l'intérieur des cliques selon leur ordre d'élimination, alors :
 on relie $D_{\sigma(i)} = \{X_{\sigma(i)}, X_{\sigma(j)}, \dots\}$ à $D_{\sigma(j)}$.

Des cliques vers l'arbre d'élimination (2/2)

- ▶ Arbre d'élimination : $\mathcal{G} = (\mathcal{D}, \mathcal{E})$, où :
 - ▶ $\mathcal{D} = \{D_{\sigma(i)} : i \in \{1, \dots, n\}\}$,
 - ▶ $\mathcal{E} = \{(D_{\sigma(i)}, D_{\sigma(j)}) : 1 \leq i < n, j = \min\{k \neq i : X_{\sigma(k)} \in D_{\sigma(i)}\}\}$
- ▶ Ensemble des cliques selon leur ordre de création (avec la variable dont l'élimination a créé la clique) :
 EIJ (J), ABD (B), DEF (F), ADE (D),
 EHI (I), EGH (H), $ACEG$ (A),
 CEG (C), EG (G), E (E)



De l'arbre d'élimination vers l'arbre de jonction (1/2)



Propriétés de l'arbre d'élimination

Propriétés

$$\mathcal{D} = \{D_{\sigma(i)} : i \in \{1, \dots, n\}\},$$

$$\mathcal{E} = \{(D_{\sigma(i)}, D_{\sigma(j)}) : 1 \leq i < n, j = \min\{k \neq i : X_{\sigma(k)} \in D_{\sigma(i)}\}\}$$

- 1 L'arbre d'élimination est un arbre
- 2 Il vérifie la propriété d'intersection courante
- 3 Soit $D_{\sigma(j)}$ un enfant de $D_{\sigma(i)}$, alors $|D_{\sigma(j)}| \geq |D_{\sigma(i)}| - 1$
- 4 Soient $D_{\sigma(i)}$ et $D_{\sigma(j)}$ les parents de $D_{\sigma(k)}$, alors $D_{\sigma(i)} \not\subset D_{\sigma(j)}$ et $D_{\sigma(j)} \not\subset D_{\sigma(i)}$
- 5 Soit $D_{\sigma(j)}$ un enfant de $D_{\sigma(i)}$, alors $D_{\sigma(j)} \subset D_{\sigma(i)} \iff |D_{\sigma(j)}| = |D_{\sigma(i)}| - 1$
- 6 Soit $D_{\sigma(j)}$ un enfant de $D_{\sigma(i)}$ tel que $D_{\sigma(j)} \not\subset D_{\sigma(i)}$, alors il n'existe pas d'ancêtre $D_{\sigma(k)}$ de $D_{\sigma(i)}$ tel que $D_{\sigma(j)} \subset D_{\sigma(k)}$

De l'arbre d'élimination vers l'arbre de jonction (2/2)

Algorithme pour obtenir un arbre de jonction

```

01 créer l'arbre d'élimination  $\mathcal{G} = (\mathcal{D}, \mathcal{E})$ 
02 marquer à false tous les arcs de  $\mathcal{E}$ 
03 pour  $i$  variant de  $n$  à 1 faire
04   si il existe  $D_{\sigma(j)}$  parent de  $D_{\sigma(i)}$  tel que l'arc
       $(D_{\sigma(j)}, D_{\sigma(i)})$  est non marqué et  $|D_{\sigma(i)}| = |D_{\sigma(j)}| - 1$  alors
05     pour tous les autres parents  $D_{\sigma(k)}$  de  $D_{\sigma(i)}$  faire
06       créer dans  $\mathcal{G}$  un arc  $(D_{\sigma(k)}, D_{\sigma(j)})$ 
07       marquer cet arc à true
08   fait
09   si  $D_{\sigma(i)}$  a un enfant  $D_{\sigma(k)}$  alors
10     créer dans  $\mathcal{G}$  un arc  $(D_{\sigma(j)}, D_{\sigma(k)})$ 
11   finsi
12   supprimer  $D_{\sigma(i)}$  ainsi que ses arcs adjacents
13 fait
    
```

A la fin de l'algorithme ci-dessus, \mathcal{G} est un arbre de jonction.

Bibliographie

- ▶ Becker A. et Geiger D. (1996) « A sufficiently fast algorithm for finding close to optimal junction trees », Proceedings of Uncertainty in Artificial Intelligence
- ▶ Flores J., Gámez J. et Olesen, K. (2003) « Incremental compilation of Bayesian networks », Proceedings of Uncertainty in Artificial Intelligence
- ▶ Jensen F.V. et Jensen F. (1994) « Optimal junction trees », Proceedings of Uncertainty in Artificial Intelligence
- ▶ Kjærulff U. (1990) « Triangulation of graphs – Algorithms giving small total state space », technical report, Aalborg University
- ▶ Kjærulff U. (1991) « Optimal decomposition of probabilistic networks by simulated annealing », Statistics and Computing, 2 :7–17
- ▶ Leimer H.-G. (1993) « Optimal decomposition by clique separators », Discrete Mathematics, 113 :99–123

- ▶ **Lepar V. et Shenoy P.P. (1998)** « A Comparison of Lauritzen-Spiegelhalter, Hugin and Shenoy-Shafer Architectures for Computing Marginals of Probability Distributions », Proceedings of Uncertainty in Artificial Intelligence, 328–337
- ▶ **Olesen K. et Madsen A. (1999)** « Maximal prime decomposition of Bayesian networks », technical report.
- ▶ **Shoikhet K. et Geiger D. (1997)** « Finding optimal triangulations via minimal vertex separators », Proceedings of the International Conference on Artificial Intelligence
- ▶ **van den Eijkhof F. et Bodlaender A. (2002)** « Safe reduction rules for weighted treewidth », Proceedings of the 28th International Workshop on Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, 2573 :176–185