Introduction aux modèles graphiques **2007/2008**

Cours 7 — 21 novembre 2007

Enseignant: Francis Bach *Scribe: Joan Fruitet, Zohra Cherfi*

7.1 Arbre de jonction

7.1.1 Le cadre

Soit $G = (V, E)$ un graphe non orienté. On suppose toujours que G n'a qu'une composante connexe. Soit $\mathcal{C}$ l'ensemble des cliques maximales de G.

7.1.2 Definitions

Définition 7.1 (Arbre de cliques) *$\mathcal{T} = (\mathcal{C}', \mathcal{E})$ est un arbre de cliques si :*

- *Il est couvrant, c'est à dire $\mathcal{C}' = \mathcal{C}$.*
- *Les arrêtes vérifient la propriété : $(C, D) \in \mathcal{E} \Rightarrow C \cap D \neq \emptyset$*

Définition 7.2 (Arbre de jonction) *C'est un arbre de cliques $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ vérifiant :*

1. *$\forall v \in V$, le sous-arbre des cliques contenant v est connexe.*
2. *$\forall (C, D) \in \mathcal{C} \times \mathcal{C}$, $C \cap D$ est inclus dans chaque clique de l'unique chemin de C à D.*

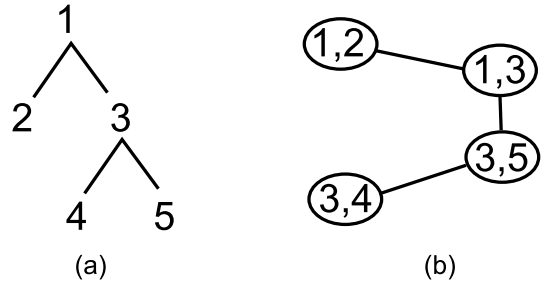

Fig. 7.1. (a) Graphe. (b) Arbre de jonction

Proposition 7.3 *Les propriétées 1 et 2 de la définition sont équivalentes.*

Tout graphe G admet au moins un (et souvent plusieurs) arbre de cliques, mais pas forcément un arbre de jonctions (voir sections suivantes sur la triangulation).

Définition 7.4 (Séparateur) *L'interséction entre deux cliques voisines est appelée séparateur.*

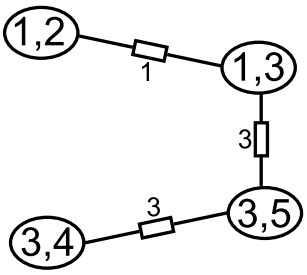

Fig. 7.2. Séparateur.

7.2 Algorithme Somme-Produit

L'algorithme somme-produit a été définit précdemment sur un arbre, il s'étend comme suit sur un arbre de jonctions.

7.2.1 L'algorithme

On suppose que l'on a un arbre de jonction $\mathcal{T} = (\mathcal{C}, \mathcal{E})$.

Définition 7.5 *Pour un couple $(C_i, C_j) \in \mathcal{E}$, le message $m_{C_iC_j}$ de $i \to j$ est égal à :*

$$m_{C_iC_j}(x_{C_i \cap C_j}) = \sum_{x_{C_i \setminus (C_i \cap C_j)}} \psi_{C_i}(x_{C_i}) \prod_{C_k \in \mathcal{N}(C_i)} m_{C_kC_i}(x_{C_k \cap C_i})$$

Le protocole de passage des messages consiste à transmettre le message de $i \to j$ après avoir reçu les messages de tous les voisins de i.

Théorème 7.6 *Si on passe les $2(|\mathcal{C}| - 1)$ messages en respectant le "protocole", alors :*

$$\forall C_i, \quad u(x_{C_i}) = \psi_{C_i}(x_{C_i}) \prod_{C_k \in \mathcal{N}(C_i)} m_{C_kC_j}(x_{C_i})$$

7.2.2 Preuve de l'algorithme

Soit $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ un arbre de jonction. On introduit des copies locales et un potentiel :

$$\forall C \in \mathcal{C} \quad \tilde{x}_C = \{\tilde{x}_{C,i}, \quad i \in C\}$$

$$\forall (C, D) \in \mathcal{C}^2 \quad \psi_{C,D}(\tilde{x}_C, \tilde{x}_D) = \prod_{i \in C \cap D} \delta(\tilde{x}_{i,C} = \tilde{x}_{i,D})$$

On a alors :

$$u(x) = \prod_{C \in \mathcal{C}} \psi_C(x_C)$$

$$\tilde{u}(\tilde{x}) = \prod_{C \in \mathcal{C}} \psi_C(\tilde{x}_C) \prod_{(C,D)\in\mathcal{E}} \underbrace{\prod_{i \in C \cap D} \delta(\tilde{x}_{i,C} = \tilde{x}_{i,D})}_{\psi_{C,D}(\tilde{x}_C, \tilde{x}_D)}$$

Lemme 7.7 *Si toutes les copies locales de chaque x_i sont cohérentes ($\tilde{x}_{C,i} = \tilde{x}_{D,i} = x_i$) alors $\tilde{u}(\tilde{x}) = u(x)$ sinon $\tilde{u}(\tilde{x}) = 0$*

Démonstration

- Si toutes les copies locales de chaque x_i sont cohérentes le résultat est trivial.
- Sinon : Supposons qu'il existe i, C, D tel que $i \in C \cap D$ et $\tilde{x}_{C,i} \neq \tilde{x}_{D,i}$. Comme on a un arbre de jonction :

$$\begin{aligned} &\exists C_1..C_K, \quad C_k \in \mathcal{N}(C_{k-1}) \text{ et } \forall k, i \in C_k \\ &\Rightarrow \exists k \leq K, \quad \tilde{x}_{i,k} \neq \tilde{x}_{i,k+1} \\ &\Rightarrow \tilde{u}(\tilde{x}) = 0 \end{aligned}$$

■

Fin de la preuve de l'algorithme :

$$m_{C_i,C_j}(\tilde{x}_{C_j}) = \sum_{\tilde{x}_{C_i}} \psi_{C_i,C_j}(\tilde{x}_{C_i}, \tilde{x}_{C_j}) \underbrace{\psi_{C_i}(\tilde{x}_{C_i}) \prod_{C_k \in \mathcal{N}(C_i)} m_{k,i}(\tilde{x}_{C_i})}_{=F(\tilde{x}_{C_i})}$$

$$\begin{aligned} m_{C_i,C_j}(\tilde{x}_{C_i\cap C_j, C_j}, \tilde{x}_{C_i\setminus(C_i\cap C_j), C_j}) &= \sum_{\tilde{x}_{C_i}} F(\tilde{x}_{C_i}) \psi_{C_i,C_j}(\tilde{x}_{C_i}, \tilde{x}_{C_j}) \\ &= \sum_{\tilde{x}_{C_i\cap C_j, C_j}} F\begin{pmatrix} \tilde{x}_{C_i\cap C_j, C_j} \\ \tilde{x}_{C_i\setminus(C_i\cap C_j), C_i} \end{pmatrix} \end{aligned}$$

Comme on a un arbre de jonction, tous les $\tilde{x}$ sont consistants et donc :

$$\begin{aligned} m_{C_i,C_j}(x_{C_i\cap C_j}) &= \sum_{\tilde{x}_{C_i}} F(\tilde{x}_{C_i}) \psi_{C_i,C_j}(\tilde{x}_{C_i}, \tilde{x}_{C_j}) \\ &= \sum_{x_{C_i\setminus(C_i\cap C_j)}} F\begin{pmatrix} \tilde{x}_{C_i\cap C_j} \\ \tilde{x}_{C_i\setminus(C_i\cap C_j)} \end{pmatrix} \end{aligned}$$

Faire passer les messages pour $\tilde{u}(\tilde{x})$ revient à faire passer les messages dans l'arbre de jonction. L'algorithme Somme-Produit est exact pour $\tilde{u}(\tilde{x})$ si et seulement si il l'est pour $u(x)$.

Proposition 7.8 *Une fois les messages passés :*

$$u(x) = \prod_{C_i \in \mathcal{C}} \psi_{C_i}(x_{C_i}) = \frac{\prod_{Cliques} u(x_C)}{\prod_{Separateurs} u(x_S)}$$

Exemple

Soit G le graphe suivant :

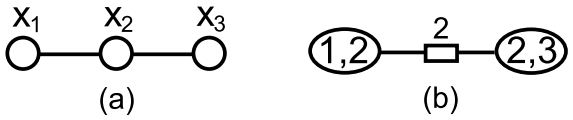

Fig. 7.3. Exemple (a) Graphe. (b) Abre de jonction.

D'après la proposition précédente : $u(x) = \frac{u(x_1,x_2)u(x_2,x_3)}{u(x_2)}$.

7.2.3 Complexité

Soit r le nombre de valeur que prend chaque variable x_i.

1. Algorithme naif : $u(x_C) = \sum_{x_{V\setminus C}} u(x) \Rightarrow O(r^{|V|})$
2. Algorithme Somme-Produit :
   - Complexité du calcul de $m_{C_iC_j}$: $\sum_{k\in\mathcal{N}(i)} r^{|C_k\cap C_j|}$
   - Soit une complexité totale : $\sum_{i,j}\sum_{k\in\mathcal{N}(i)} r^{|C_k\cap C_j|} = O(|\mathcal{C}|r^{\max_{C\in\mathcal{C}}|C|})$

L'algorithme Somme-Produit est donc exponentiel en la taille de la clique maximale, mais seulement linéaire dans le nombre de cliques.

7.3 Triangulation

Comme vu précédemment, tout graphe n'admet pas forcément un arbre de jonction. Néanmoins, en rajoutant potentiellement des arêtes pour le rendre triangulé, tout graphe admet un arbre de jonction.

Définition 7.9 *Soit $G = (V, E)$ un graphe non orienté.*
Un cycle sans corde (figure7.4) est défini comme un cycle C d'ordre ≥ 4 tel que :

$$\forall (u, v) \in C, (u, v) \in E \Rightarrow (u, v) \ \ adjacents \ \ dans \ \ C.$$

Autrement dit, deux sommets non adjacents (au niveau du cycle) ne sont pas voisins (au niveau du graphe).

Définition 7.10 *Soit $G = (V, E)$ un graphe non orienté. G est triangulé s'il ne contient pas de cycle sans corde (figure7.5).*

Théorème 7.11 *G est non triangulé $\Rightarrow$ G reconstitué après élimination est triangulé.*

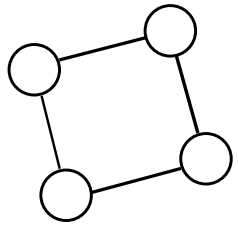

Fig. 7.4. Cycle sans corde.

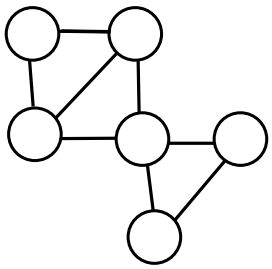

Fig. 7.5. Graphe triangulé.

Démonstration La preuve se fait par récurrence sur le nombre de sommets ($n = \#V$). Pour le cas de base $n = 1$, le graphe constitué d'un nœud unique est nécessairement triangulé. On suppose l'hypothèse vraie pour un graphe à n sommets et on montre que c'est toujours vrai pour un graphe à $n + 1$ sommets.
Prenons un graphe $G = (V, E)$ à $n + 1$ sommets, la première itération de l'algorithme d'élimination élimine le sommet $\{n + 1\}$ et relie tous ses voisins. Le résultat est un graphe à n sommets $\tilde{G} = (\tilde{V}, \tilde{E})$ où $\tilde{V} = V \backslash \{n + 1\}$ et $\tilde{E} = (E \cap (\tilde{V} \times \tilde{V})) \cup (N(n + 1) \times N(n + 1))$.
D'après l'hypothèse de récurrence, en applicant l'algorithme d'élimination sur $\tilde{G}$, on obtient un graphe reconstitué $rec(\tilde{G})$ triangulé.
En ajoutant le sommet $\{n+1\}$ au graphe $rec(\tilde{G})$, on obtient le graphe $rec(G)$ qui ne contient aucun cycle sans corde car tous les voisins du nœud $\{n + 1\}$ ont été connectés. Par conséquent, le graphe reconstitué $rec(G)$ est triangulé (figure7.6). ■

Théorème 7.12 *G est triangulé $\Leftrightarrow$ Il existe un ordre d'élimination laissant G invariant.*

Pour le théorème précédent :
- *G est triangulé $\Rightarrow$ il existe un ordre d'élimination laissant G invariant* est un corollaire du théorème 7.11.
- *Il existe un ordre d'élimination laissant G invariant $\Rightarrow$ G est triangulé* est admis.

Théorème 7.13 *G est triangulé $\Leftrightarrow$ G admet un arbre de jonction JT (Junction Tree).*

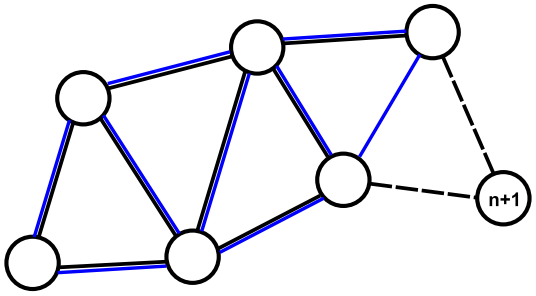

Fig. 7.6. Graphe reconstitué triangulé.

Démonstration

1. *G est triangulé* $\Rightarrow$ *G admet un arbre de jonction* (preuve par récurrence) :
   Le cas de base où G est un graphe à un seul nœud est trivial.
   On considère que le graphe $G = (V, E)$ à $n+1$ nœuds est triangulé. Selon le théorème 7.3, il existe un ordre d'élimination qui le laisse invariant.
   Soit $\{\alpha\}$ le premier nœud éliminé. Le graphe $G\backslash\{\alpha\}$ obtenu est triangulé. Ainsi, par hypothèse de récurrence, le graphe triangulé $G\backslash\{\alpha\}$ admet un arbre de jonction T.
   Soit C_α la clique contenant α et ses voisins.
   Si $C = C_\alpha\backslash\{\alpha\}$ est une clique dans T, alors en ajoutant $\{\alpha\}$ à cette clique, l'arbre de jonction T contenant la clique augmentée est un arbre de jonction pour G.
   Si $C = C_\alpha\backslash\{\alpha\}$ n'est pas une clique dans T, alors c'est un sous ensemble d'une clique D dans T. Dans ce cas C_α peut être ajoutée comme une feuille de T, relié à D et un ensemble séparateur $S = C$. Le résultat est un arbre de jonction.
2. *G admet un arbre de jonction* $\Rightarrow$ *G est triangulé* (pour cette preuve se référer au livre de M.Jordan : 17.14.2).

■

7.4 Construction de l'arbre de jonction

Il est clair à présent que tout graphe triangulé admet un arbre de jonction. Néanmoins, tout arbre de clique obtenu à partir d'un graphe triangulé n'est pas un arbre de jonction. Si l'on considère le graphe de la figure7.7(a), l'arbre de cliques de la figure7.7(b), n'est pas un arbre de jonction. Ce graphe admet comme arbre de jonction l'arbre de la figure7.7(c).

Chaque arbre de cliques associé à un graphe triangulé est caractérisé par un poids $w(T)$ relatif à la somme des cardinaux des ensembles séparateurs. Un arbre de cliques sera un arbre de jonction si et seulement s'il est de poids maximal.

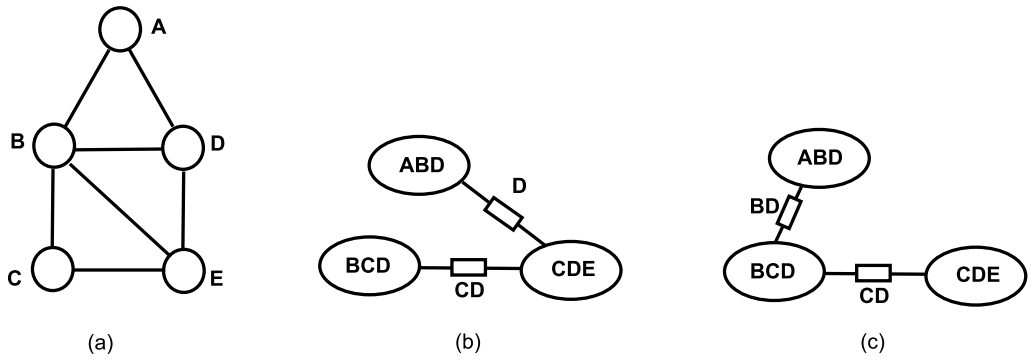

Fig. 7.7. (a) Graphe triangulé. (b) arbre de cliques. (c) arbre de jonction.

Soit le graphe $G = (V, E)$ associé à un arbre de cliques T, où S représente un ensemble séparateur ($S \subset V$ et $S \in \mathcal{S}$) avec $w(S) = \#S$ et C une clique de T ($C \in \mathcal{C}$) avec $w(C) = \#C$.

Lemme 7.14 $\forall$ *T arbre de cliques ($\mathcal{S}$ ensemble des séparateurs) :*

$$\forall x_v \in V; \quad \sum_{S \in \mathcal{S}} \delta(x_v \in S) \leq \sum_{C \in \mathcal{C}} \delta(x_v \in C) - 1. \tag{7.1}$$

Démonstration Considérons le sous arbre T_v (de cliques) des cliques contenant x_v. Il est clair que : (nombre d'arêtes de T_v) $\leq$ (nombre de sommets de T_v) -1. ■

Lemme 7.15 $\forall$ *T arbre de cliques :*

$$\sum_{S \in \mathcal{S}} w(S) \leq \sum_{C \in \mathcal{C}} w(C) - (\#V). \tag{7.2}$$

avec égalité ssi l'arbre T est un arbre de jonction.

Démonstration

$$\begin{aligned}
\sum_{S \in \mathcal{S}} w(S) &= \sum_{S \in \mathcal{S}} \sum_{x_v \in V} \delta(x_v \in S) && (7.3)\\
&= \sum_{x_v \in V} \left(\sum_{S \in \mathcal{S}} \delta(x_v \in S) \right) && (7.4)\\
&\leq \sum_{x_v \in V} \left(\sum_{C \in \mathcal{C}} \delta(x_v \in C) - 1 \right) && (7.5)\\
&= \sum_{C \in \mathcal{C}} w(C) - (\#V). && (7.6)
\end{aligned}$$

Il y a égalité ssi pour tout v, le sous-arbre T_v est connexe, i.e., exactement si on a un arbre de jonctions. ■

Dans la preuve, l'équation (7.6) est indépendante de T. Ceci signifie que, pour n'importe quel arbre, on ne peut obtenir un poids $w(T)$ supérieur à $\sum_{C\in\mathcal{C}} w(C) - (\#V)$.

Si T est un arbre de jonction, l'inégalité (7.5) devient une égalité. Ainsi, il s'agit de trouver l'arbre couvrant de poids maximum pour l'arbre de cliques (Problème de l'arbre couvrant de poids maximum). Il s'agit donc de trouver T tel que : $T \in \arg\max \left(\sum_{(C,D)\in\mathcal{C}} w(C,D)\right)$. Où C et D sont des cliques de T et :

$$\begin{aligned} w : \mathcal{C}\times\mathcal{C} &\rightarrow \mathbf{R}_+ \\ w(C,D) &= w(C\cap D)\ (=0\ si\ C\cap D=\emptyset). \end{aligned}$$

Le problème de l'arbre couvrant de poids maximum peut être résolu de manière exacte par l'algorithme de Kruskal (algorithme glouton).

Algorithme de Kruskal
Pour $i = 1, ..., \#\mathcal{C}$:
Choisir l'arête de poids maximum ne créant pas de cycle (figure7.8).

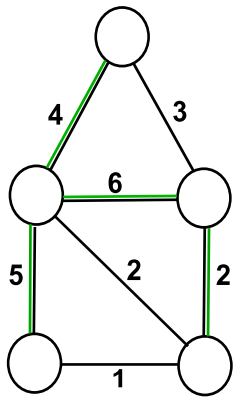

Fig. 7.8. Arbre couvrant de poids maximum (6,5,4,2).

Complexité $O\left((\#\mathcal{C})^2 \log(\#\mathcal{C})\right)$.

7.5 Inférence exacte

Résumé des étapes de l'algorithme de l'arbre de jonction :

1. Moralisation (si un graphe est orienté).

2. Introduction des observations (évidence).

3. Construction de l'arbre de jonction :

   – Sélectionner un ordre d'élimination
   – Triangulation par élimination et extraction des cliques qui formeront les nœuds du futur arbre ;
   – Création d'un arbre couvrant maximum (Algorithme de Kruskal).

4. Propagation des messages sur l'arbre de jonction.

L'algorithme de l'arbre de jonction a une complexité exponentielle en $\#C_{max}$ (taille de la clique maximale après triangulation). Trouver l'ordre d'élimination (ou la triangulation) qui permet d'obtenir la plus petite clique de taille maximale est un problème NP-difficile.