

Chap. 4 Arbres

Ce sont des structures fondamentales utilisées dans de nombreux domaines :

- Informatique
- Sciences sociales
- Classification et analyse de données
- Théorie des questionnaires
- Recherche opérationnelle
- Intelligence artificielle
- Optimisation combinatoire
- Théorie des réseaux électriques

Définitions et propriétés

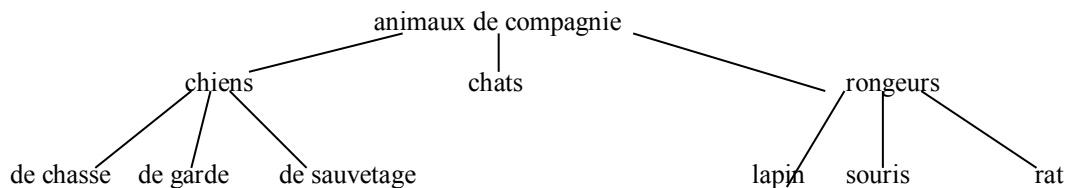
- La structure arbre fait appel au concept **non orienté**
- Un arbre est un graphe **connexe** sans **cycle**
- Un graphe sans cycle qui n'est **pas connexe** est une **forêt**
- Un arbre est toujours un graphe simple (au plus une arête entre 2 sommets)

Ex.1 : Arbre généalogique d'une famille ;

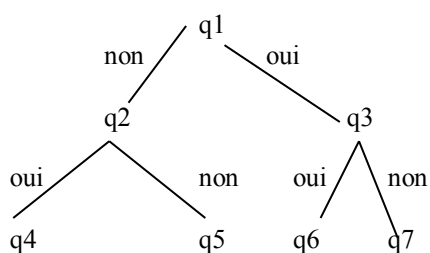
sommets – membres de la famille

arêtes – liens de parenté

Ex.2 : Arbre de classification



Ex.3 : Arbres et questionnaires



4.2 Le problème de l'arbre de poids minimum

- Soit $G = [S, A]$ un graphe. A chaque arc $u \in A$ correspond un poids $w(u)$.
- Soit $G' = [S, A']$ un graphe partiel de G
- On appelle poids de G' : $w(G') = \sum_{u \in A} w(u)$

- Le problème de l'arbre de poids minimum – rechercher un arbre τ^* de G tel que :

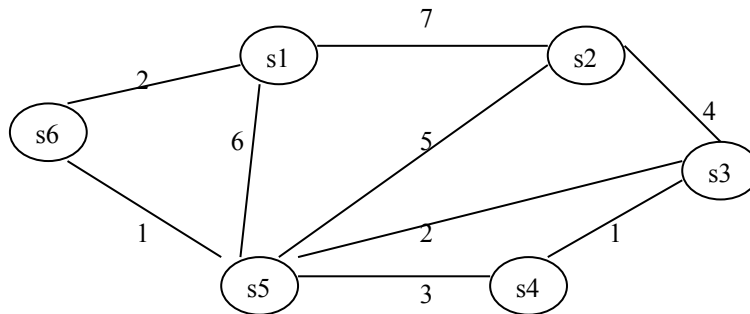
$$w(\tau^*) = \min_{\tau} \{w(\tau)\}$$

Le minimum est pris sur l'ensemble de tous les arbres de G possibles.

4.3 Algorithme de Kruskal (1956) – 1^{ère} version

Principe

- Soit $G = \langle S, A, C \rangle$ un graphe non orienté valué et connexe, de n sommets et p arêtes.
- On part d'un graphe T vide
- On ajoute à T des arêtes de G , une par une, en choisissant à chaque étape, parmi les arêtes qui ne sont pas dans T , une arête de coût minimum, qui ne forme pas de cycle avec les arêtes qui sont déjà dans T .
- Lorsqu'on a ajouté $n-1$ arêtes, sans créer de cycle, on a obtenu un arbre de poids minimum (arbre de recouvrement minimum)

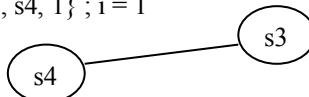


Initialisation

$U = \{ \{s1, s2, 7\}, \{s1, s5, 6\}, \{s1, s6, 2\}, \{s2, s3, 4\}, \{s2, s5, 5\}, \{s3, s4, 1\}, \{s3, s5, 2\}, \{s4, s5, 3\}, \{s5, s6, 1\} \}$

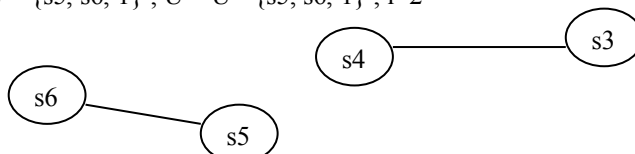
1) $\min(U) = \{s3, s4, 1\}$; $U = U - \{s3, s4, 1\}$; $i = 1$

$T :$



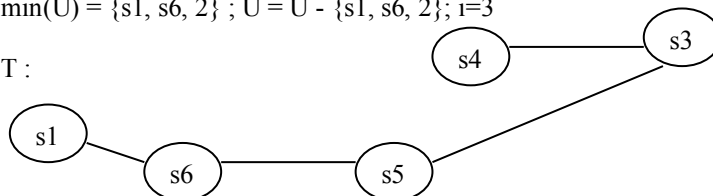
2) $\min(U) = \{s5, s6, 1\}$; $U = U - \{s5, s6, 1\}$; $i = 2$

$T :$



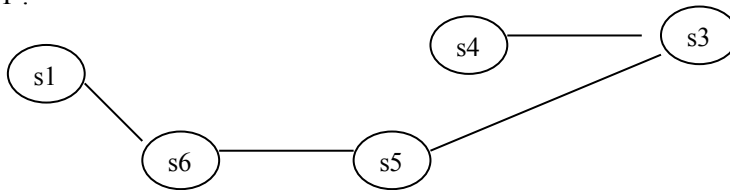
3) $\min(U) = \{s1, s6, 2\}$; $U = U - \{s1, s6, 2\}$; $i = 3$

$T :$



4) $\min(U) = \{s3, s5, 2\}$; $U = U - \{s3, s5, 2\}$; $i=4$

5) T :

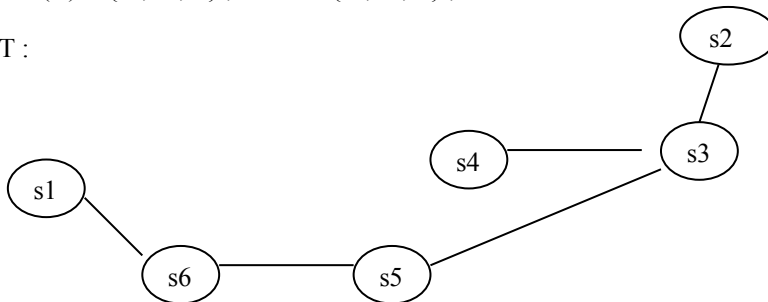


6) $\min(U) = \{s4, s5, 3\}$; $U = U - \{s4, s5, 3\}$; $i=4$

L'arête $\{s4, s5\}$ forme un cycle dans T : elle est rejetée.

7) $\min(U) = \{s2, s3, 4\}$; $U = U - \{s2, s3, 4\}$; $i=5$

T :



Lorsque $i = n - 1 = 5$, on obtient un arbre de recouvrement minimum.

4.4 Algorithme de Kruskal (1956) – 2^{ème} version

Principe

- Soit $G = \langle S, A, C \rangle$ un graphe non orienté valué et connexe, de n sommets et p arêtes.
- On part d'un graphe $T = G$
- On retire de T des arêtes de G , une par une, en choisissant à chaque étape, parmi les arêtes qui ne sont pas dans T , une arête de coût maximum, de façon que T reste connexe.

Lorsque T possède $n-1$ arêtes, sans créer de cycle, on a obtenu un arbre de poids minimum (arbre de recouvrement minimum)

4.5 Algorithme de Prim (1957)

Principe

Le principe de cet algorithme est voisin de celui de l'algorithme de Dijkstra : un minimum local est choisi à chaque étape selon des critères qui assurent qu'il fait partie de la solution globale.

- Soit $G = \langle S, A, C \rangle$ un graphe non orienté, valué et connexe.
- On va construire progressivement un sous-graphe T de G . Au départ, T est le graphe vide.
- On se donne un sommet s de S .
- On appelle CC l'ensemble des sommets reliés à s dans T , et M son complémentaire. (Au départ CC est réduit au sommet s).
- On modifie le graphe T , par l'adjonction, à chaque étape, d'une arête. L'arête ajoutée est choisie de sorte qu'elle soit de coût minimum parmi toutes les arêtes ayant une extrémité x dans M et l'autre y dans CC . On ne crée donc pas de cycle dans T : T est bien un arbre. On ajoute alors le sommet x à CC et on recommence.
- On s'arrête lorsque CC est l'ensemble S : T est un arbre de recouvrement de G .

Algorithme

On utilise une procédure de marquage. A chaque sommet $i \in S$ on associe :

- un nombre réel $\pi(i)$ /*marque ou potentiel*/
- un index $\alpha(i)$ indiquant le numéro de l'arête ayant permis de connecter s , sommet de départ, à i .

a) Initialisation :

$$\begin{aligned} \pi(s) &= 0 ; \pi(i) = +\infty \text{ pour } \forall i \neq s \\ \alpha(i) &= 0 \text{ pour } \forall i \\ CC &= \{s\} ; T = \{\emptyset\} ; M = S - CC \end{aligned}$$

b) A l'étape courante, on a déjà construit :

- un sous-ensemble de sommets $CC \subset S$, ensemble des sommets connectés à s .
- un arbre T minimum connectant tous les sommets de CC

On sélectionne $i \in M$ de marque $\pi(i)$ minimale.

soit $u = \alpha(i)$.

CC est augmenté du sommet i , M est décrémenté de sommet i et T est augmenté de l'arête u .

c) On remet à jour les marques $\pi(j)$ et les index $\alpha(j)$ des sommets de M :

- Pour tout $u = (i, j)$ de $\omega(u)$ tel que $j \in M$ faire

$$\pi(j) \leftarrow \omega(u)$$

Si $\omega(u) < \pi(j)$ alors

$$\alpha(j) \leftarrow u$$

d) Si $M \neq \emptyset$ alors retourner en b)

4.6 Arbre de poids maximum

- Soit $G = [S, A]$ un graphe. A chaque arc $u \in A$ correspond un poids $\omega(u)$.
- Soit $G' = [S, A']$ un graphe partiel de G

- Le problème de l'arbre de poids maximum – rechercher un arbre τ^* de G tel que :

$$\omega(\tau^*) = \max_{\tau} \{\omega(\tau)\}$$

Le maximum est pris sur l'ensemble de tous les arbres de G possibles.

Le problème est équivalent au problème de l'arbre de poids maximum.

Les algorithmes précédents sont encore applicables avec les changements suivants :

- $\min \rightarrow \max$
- $\max \rightarrow \min$
- ordre croissant $\rightarrow$ ordre décroissant
- ordre décroissant $\rightarrow$ ordre croissant