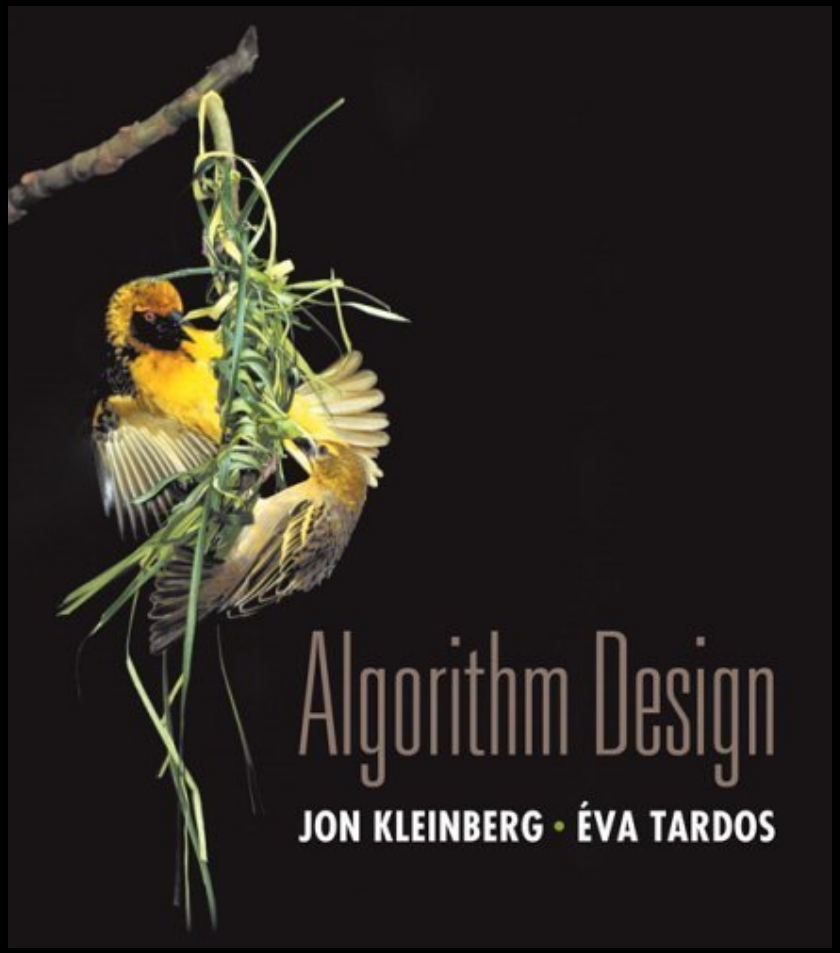


Chapitre 4

Algorithmes gloutons

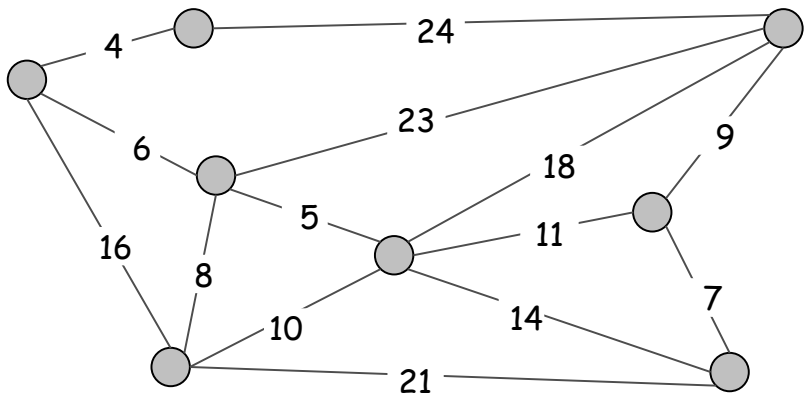
(suite et fin)

Arbre couvrant de poids minimal

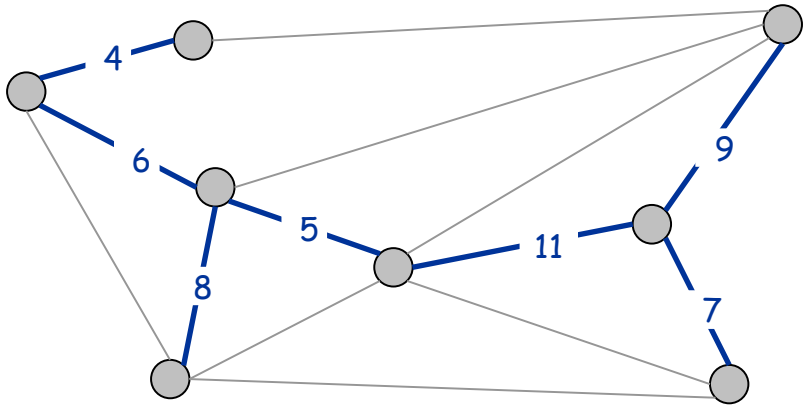
Minimum Spanning Tree

Arbre couvrant (de poids) minimal (MST)

Minimum spanning tree. Etant donné un graphe connexe G = (V, E) avec des arêtes valuées par des réels c_e (poids), un arbre couvrant minimal (MST) est un graphe partiel $T \subseteq E$ tel que T est un arbre dont l'ensemble d'arêtes est V (i.e. T est un arbre couvrant) et dont le poids total est minimal parmi les arbres couvrants.



G = (V, E)



T, $\Sigma_{e \in T} c_e = 50$

- Théorème de Cayley. Il y a n^{n-2} arbres couvrants de K_n (graphe complet à n sommets).

il est donc déraisonnable de recherche un MST par "force brute"

Applications

MST est un problème fondamental aux applications multiples.

- Network design.
  - téléphone, rés. électriques, rés. hydrauliques, TV (cable), rés. informatiques
- Algorithmes d'approximation pour des problèmes NP-difficiles.
  - voyageur de commerce, arbre de Steiner
- Applications indirectes.
  - détection de risques d'embouteillage (max bottleneck paths)
  - codes LDPC pour la correction d'erreurs
  - enregistrement d'images (entropie de Renyi)
  - apprentissage de traits marquants pour la reconnaissance de visage en temps réel
  - stockage de données dans le séquençage des amino-acides d'une protéine
  - modélisation de l'interaction de particules dans une turbulence
  - protocoles d'autoconfiguration pour ponts Ethernet
- Clustering.

Algorithmes gloutons

Algorithme de Kruskal. On initialise T = ϕ. On considère les arêtes selon l'ordre croissant des poids. On ajoute l'arête e à T si elle n'introduit pas de cycle.

Algorithme "Reverse-Delete" . On initialise T = E. On considère les arêtes dans l'ordre décroissant des poids. On supprime e de T à moins que cela déconnecte T.

Algorithme de Prim. On initialise T = {s} et on fait croître T de façon gloutonne à partir de s. À chaque étape, on ajoute l'arête de poids minimal parmi celles qui ont exactement une extrémité dans T.

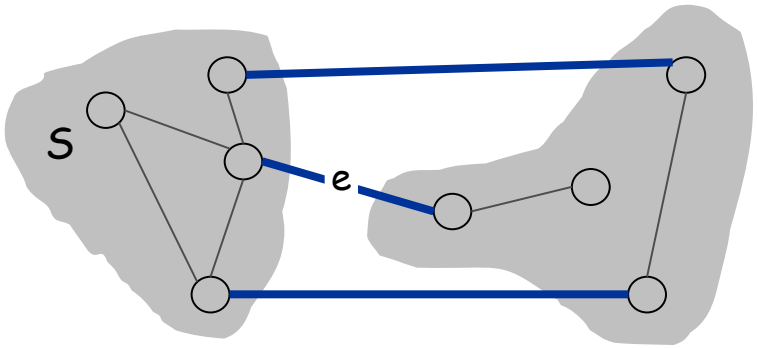
Remarque. Ces trois algorithmes produisent un MST.

Algorithmes gloutons

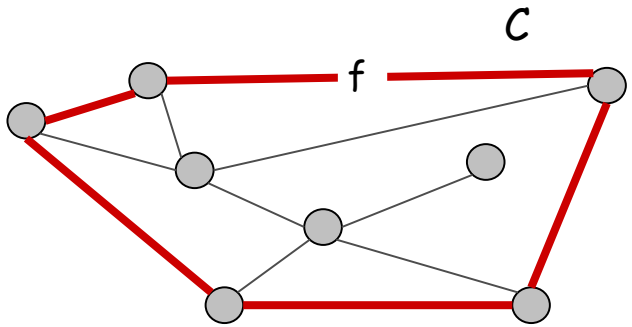
Hypothèse simplificatrice. Les poids c_e sont deux à deux distincts.

Propriété de la coupe. Soit S un sous-ensemble (strict) de V, et e l'arête de plus petit poids parmi celles qui ont exactement une extrémité dans S. Alors le MST contient e. [Pourquoi le MST ?]

Propriété du cycle. Soit C un cycle, et f son arête de plus grand poids. Alors le MST ne contient pas f.



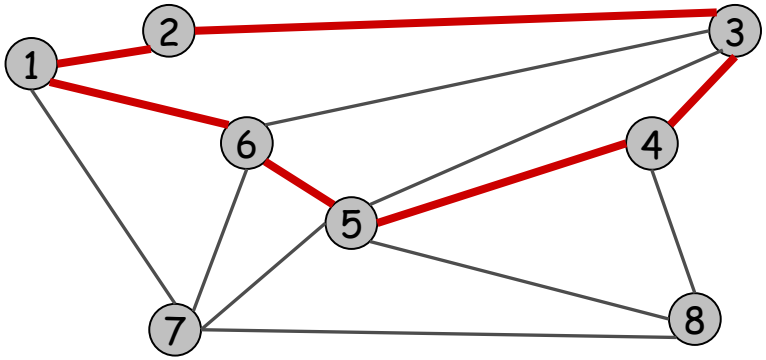
e est dans le MST



f n'est pas dans le MST

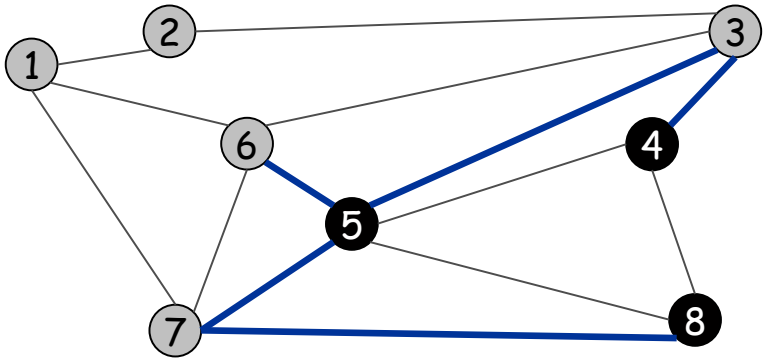
Cycles et Coupes

Cycle (simple). Ensemble d'arêtes de la form a-b, b-c, c-d, …, y-z, z-a.



Cycle C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1

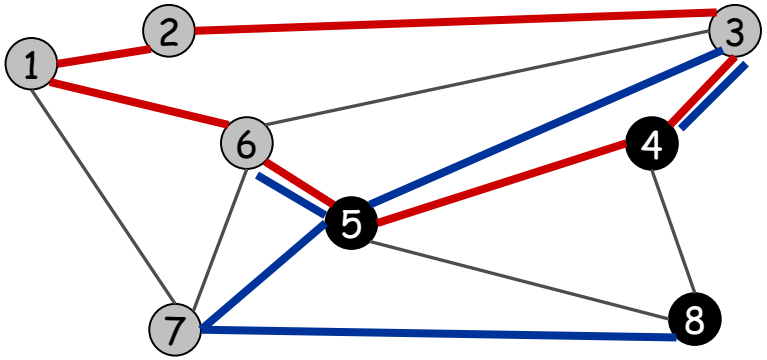
Coupe. Ensemble S de sommets (cut) ou ensemble D des arêtes ayant exactement une extrémité dans S (cutset.)



Coupe S = { 4, 5, 8 }
Cutset D = 5-6, 5-7, 3-4, 3-5, 7-8

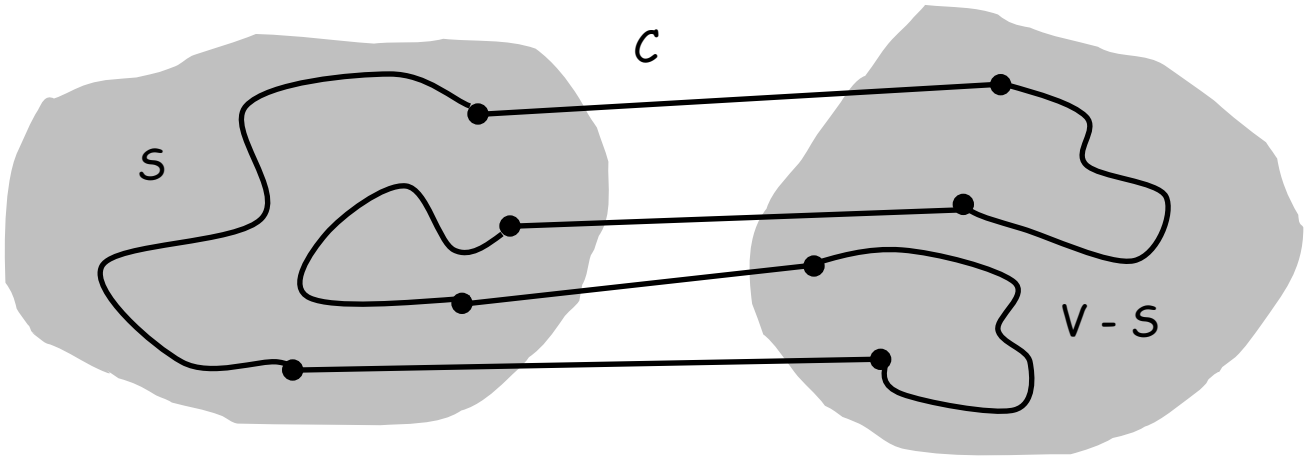
Intersection cycle-coupe

Propriété. Un cycle et une coupe ont un nombre paire d'arêtes communes.



Cycle C = 1-2, 2-3, 3-4, 4-5, 5-6, 6-1
Coupe D = 3-4, 3-5, 5-6, 5-7, 7-8
Intersection = 3-4, 5-6

Preuve. (par le dessin)



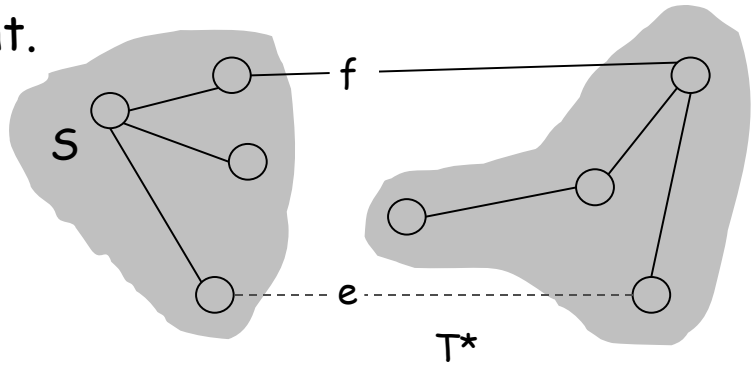
Algorithmes gloutons

Hypothèse simplificatrice. Les poids c_e sont deux à deux distincts.

Propriété de la coupe. Soit S un sous-ensemble (strict) de V, et e l'arête de plus petit poids parmi celles qui ont exactement une extrémité dans S. Alors le MST T* contient e.

Preuve. (argument de l'échange)

- On suppose que e n'appartient pas à T*.
- Ajouter e à T* créer un cycle C dans T*.
- L'arête e est à la fois dans le cycle C et dans la coupe (cutset) D correspondant à S $\Rightarrow$ il existe une autre arête, par exemple f, qui est à la fois dans C et D.
- T' = T* $\cup$ { e } - { f } est un arbre couvrant.
- Comme $c_e < c_f$, poids(T') < poids(T*).
- Contradiction. ▪



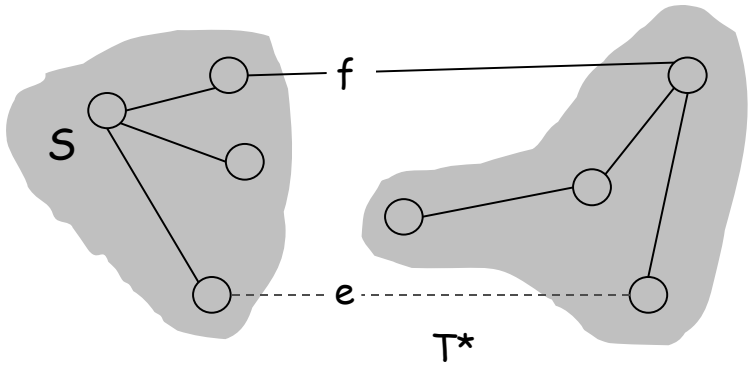
Greedy Algorithms

Hypothèse simplificatrice. Les poids c_e sont deux à deux distincts.

Propriété du cycle. Soit C un cycle, et f son arête de plus grand poids. Alors le MST T* ne contient pas f.

Preuve. (argument de l'échange)

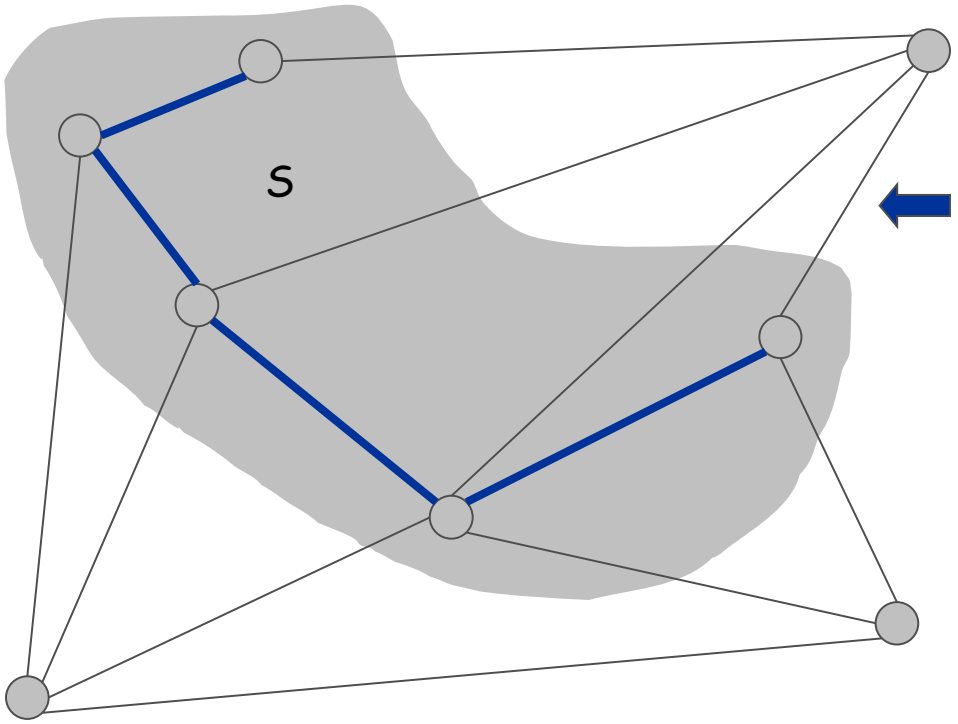
- On suppose que f appartient à T*.
- Ôter f de T* déconnecte S du reste de T*.
- L'arête f est à la fois dans le cycle C et dans la coupe (cutset) D correspondant à S $\Rightarrow$ il existe une autre arête, par exemple e, qui est à la fois dans C et D.
- T' = T* $\cup$ { e } - { f } est un arbre couvrant.
- Comme $c_e < c_f$, poids(T') < poids(T*).
- Contradiction. ▪



Algorithme de Prim : preuve de correction

Algorithm de Prim. [Jarník 1930, Dijkstra 1957, Prim 1959]

- On initialise S = {s}, s un sommet quelconque.
- On applique la propriété de la coupe à S.
- On ajoute à T l'arête de plus petit poids de la coupe (cutset) correspondant à S, et on ajoute un nouveau sommet exploré u à S.



Algorithme de Prim : implémentation:

Implémentation. On utilise une file de priorité *alla* Dijkstra.

- On maintient un ensemble de sommets explorés S.
- Pour chaque sommet v non exploré, on maintient a[v] = poids de l'arête minimale partant de v et arrivant dans S.
- $O(n^2)$ avec un tableau ; $O(m \log n)$ avec un tas binaire.

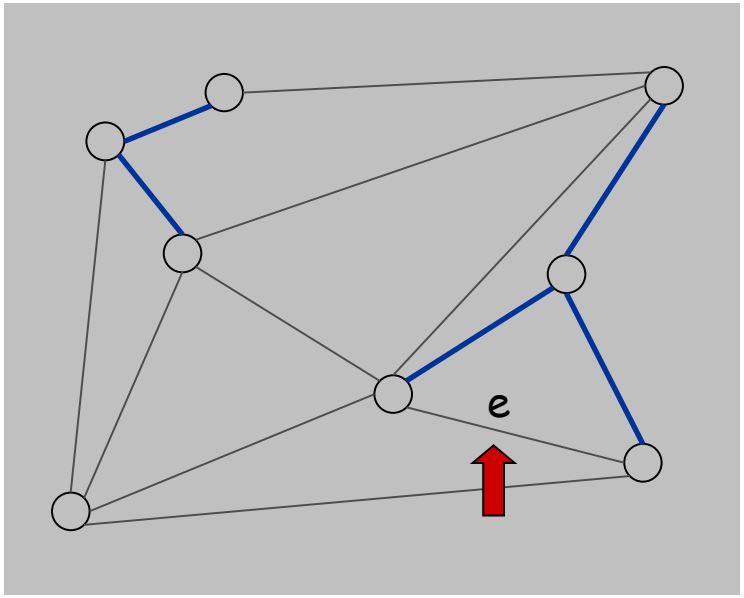
```
Prim(G, c) {
   pour tout v ∈ V faire a[v] ← ∞
   initialiser une file de priorité vide Q
   pour tout v ∈ V insérer v dans Q
   initialiser l'ensemble des sommets explorés S ← ϕ

   tant que (Q n'est pas vide) {
      supprimer l'élément minimal u de Q
      S ← S ∪ { u }
      pour toute (arête e = (u, v) incidente à u)
          si ((v ∉ S) et (c_e < a[v]))
             mettre à jour la priorité a[v] ← c_e
}
```

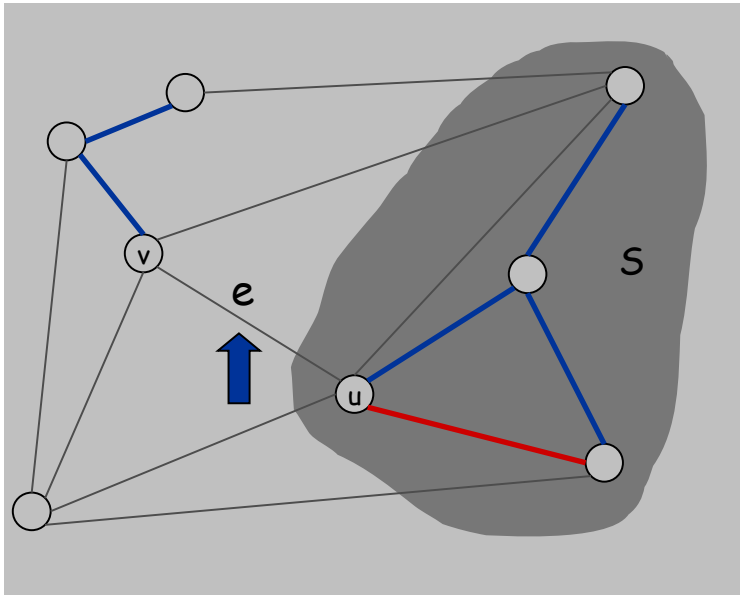
Algorithme de Kruskal : preuve de correction

Algorithme de Kruskal. [Kruskal, 1956]

- On considère les arêtes selon l'ordre croissant des poids.
- Cas 1 : si l'ajout de e à T crée un cycle, on rejette e (application de propriété du cycle).
- Cas 2: sinon, on ajoute e = (u, v) à T (application de la propriété de la coupe à S = composante connexe de u.



Cas 1



Cas 2

Algorithme de Kruskal : implémentation

Implémentation. On utilise la structure de données union-find.

- On construit un ensemble T d'arêtes appartenant au MST.
- On maintient un ensemble pour chaque composante connexe.
- $O(m \log n)$ pour le tri des arêtes et $O(m \, \alpha(m, n))$ pour l' union-find.

$m \le n^2 \Rightarrow \log m$ est $O(\log n)$

essentiellement une constante

```
Kruskal(G, c) {
   trier les arêtes selon leur poids afin c1 ≤ c2 ≤ ... ≤ cm.
   T ← ϕ

   pour tout u ∈ V initialiser un ensemble ne contenant que u

   pour i = 1 à m                    u et v sont-ils connectés ?
      (u,v) = ei                     ↙
      si (u et v sont dans des ensembles différents) {
         T ← T ∪ {ei}
         fusionner les ensembles contenant u et v
      }                              ↖ fusion des composantes
   retourner T
}
```

Lexicographic Tiebreaking

Pour abandonner l'hypothèse que les poids des arêtes sont deux à deux distincts : on "perturbe" légèrement les poids afin de les rendre deux à deux distincts.

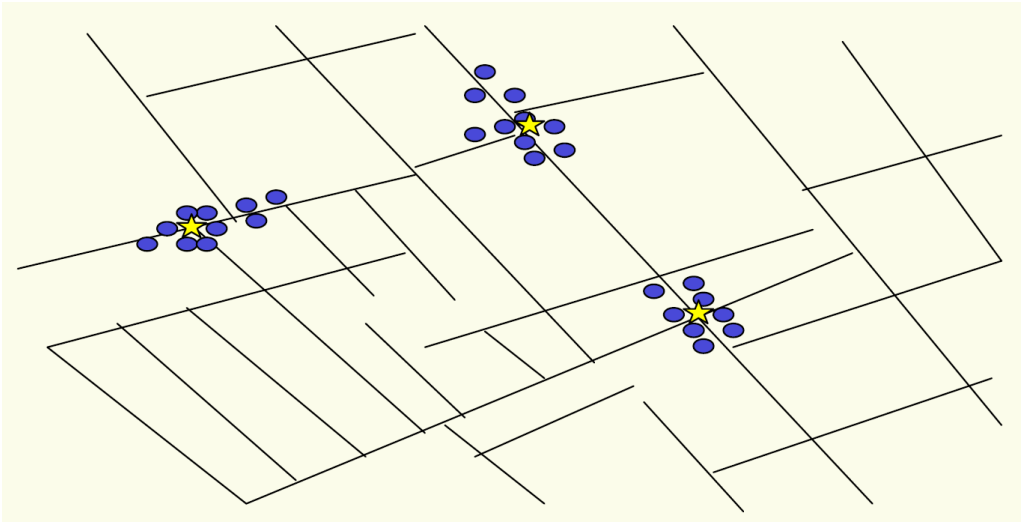
Impact. Kruskal et Prim ne prennent en compte les poids que via des comparaisons entre deux arêtes. Si les perturbations sont assez petites, le MST calculé avec les poids perturbés est le MST calculé avec les poids d'origine.

par ex., si les poids sont entiers,
on perturbe le poids de l'arête e_i par i / n^2

Implémentation. On peut simuler des perturbations aussi petites que possible par lexicographic tiebreaking (en utilisant les indices.)

```
boolean less(i, j) {
   if      (cost(e_i) < cost(e_j)) return true
   else if (cost(e_i) > cost(e_j)) return false
   else if (i < j)                 return true
   else                            return false
}
```

Mise en grappes (Clustering)



Apparition des morts par choléra à Londres dans les années 1850.
Référence : Nina Mishra, HP Labs

Clustering

Clustering. Etant donné un ensemble U de n objets étiquetés par p_1, …, p_n, les classer en groupes cohérents.

photos, documents. micro-organismes

Fonction de distance. Valeur numérique spécifiant le "degré de similitude" entre deux objets.

par ex., le nombre de couples de pixels "en correspondance" dont la différence d'intensité ne dépasse pas un seuil donné

Problème fondamental. Diviser en grappes de sorte que les éléments de grappes distinctes soient éloignés les uns des autres.

- Routage dans les réseaux mobiles *ad hoc*.
- Identification de patterns dans des gènes.
- Categorisation de documents pour la recherche sur le web.
- Recherche de similarité dans des bases d'images médicales
- Skycat : regrouper par grappes 10^9 objets : étoiles, quasars, galaxies.

Éloignement maximal entre grappes

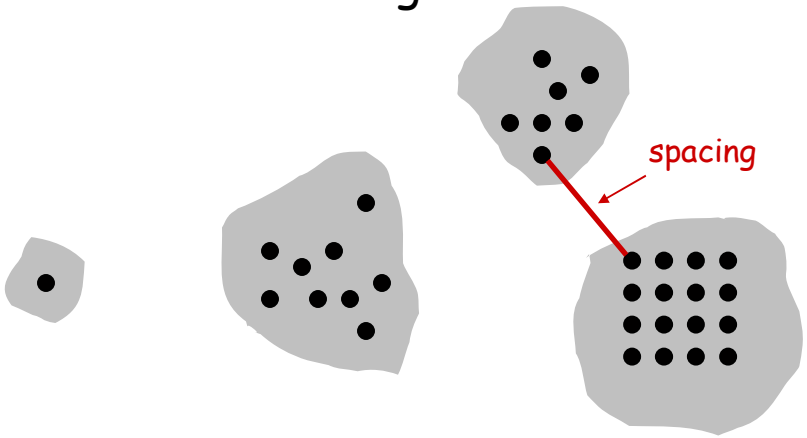
k-clustering. Diviser les objets en k grappes non vides.

Fonction de distance. On suppose qu'elle satisfait des propriétés naturelles.

- $d(p_i, p_j) = 0$ iff $p_i = p_j$ (identité des indiscernables)
- $d(p_i, p_j) \geq 0$ (positivité)
- $d(p_i, p_j) = d(p_j, p_i)$ (symétrie)

Éloignement. Min {d(p,q) | p et q dans des grappes différentes}.

Clustering of maximum spacing. Étant donné un entier k, trouver une mise en k grappes maximisant l'éloignement.



k = 4

Algorithme de clustering glouton

Single-link k-clustering algorithm.

- Former un graphe sur les noeuds U, correspondant à n grappes.
- Parmi les paires d'objets appartenant à des grappes différentes, trouver qui minimise la distance, et ajouter une arête entre les deux objets.
- Répéter l'étape précédente n-k fois, jusqu'à ce qu'il y ait exactement k grappes.

Observation clé. C'est précisément l'algorithme de Kruskal (mais on s'arrête quand on a k composantes connexes).

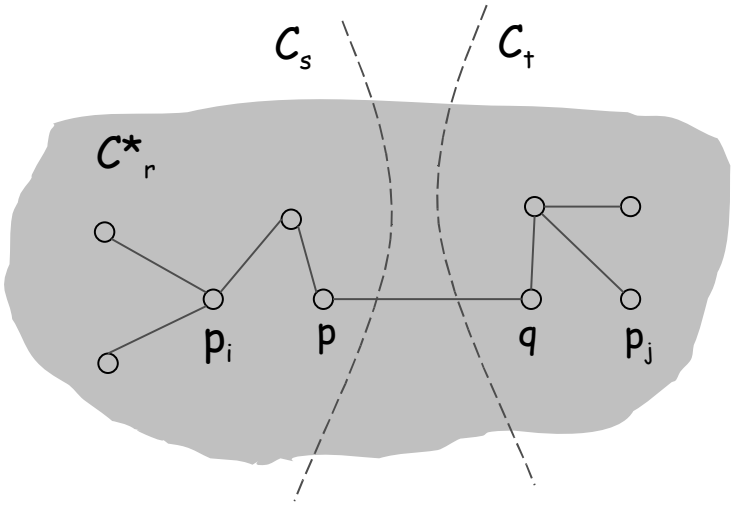
Remarque. Cela revient à trouver un MST puis à supprimer les k-1 arêtes de plus grand poids.

Algorithme de clustering glouton : analyse

Théorème. Soit C^* le clustering $C^*_1, \ldots, C^*_k$ obtenu en supprimant les k-1 arêtes de plus grand poids d'un MST. C^* est un k-clustering maximisant l'éloignement.

Preuve. Soit C un autre k-clustering $C_1, \ldots, C_k$.

- L'éloignement de C^* est le poids d^* de l'arête de $(k-1)^e$ poids.
- Soit p_i, p_j appartenant à la même grappe de C^*, par ex. C^*_r, mais à des grappes différentes de C, par ex. C_s et C_t.
- Soit (p, q) une arête du chemin p_i-p_j dans C^*_r relie deux grappes différentes de C.
- Toutes les arêtes du chemin p_i-p_j ont un poids $\leq d^*$ (car choisies par Kruskal.)
- L'éloignement de C est $\leq d^*$ car p et q sont dans des grappes différentes. ▪



Algorithmes pour le calcul d'un MST : résultats théoriques

Algorithmes déterministes à base de comparaisons.

- $O(m \log n)$ [Jarník, Prim, Dijkstra, Kruskal, Boruvka]
- $O(m \log \log n)$. [Cheriton-Tarjan 1976, Yao 1975]
- $O(m \beta(m, n))$. [Fredman-Tarjan 1987]
- $O(m \log \beta(m, n))$. [Gabow-Galil-Spencer-Tarjan 1986]
- $O(m \alpha(m, n))$. [Chazelle 2000]

Saint Graal. $O(m)$.

Notable.

- $O(m)$ randomisé. [Karger-Klein-Tarjan 1995]
- $O(m)$ vérification. [Dixon-Rauch-Tarjan 1992]

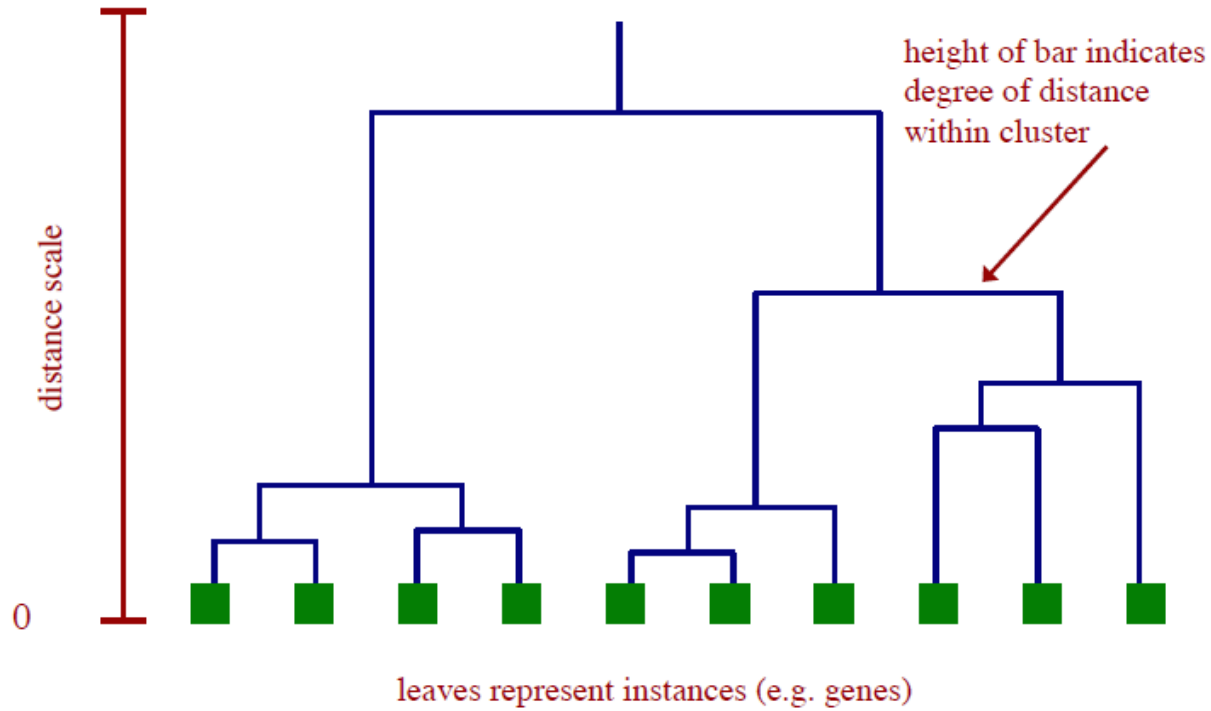
MST euclidien.

- 2-d: $O(n \log n)$.
- k-d: $O(k n^2)$. Prim "dense"

Dendrogramme

Dendrogramme. Outil de visualisation d'une suite hypothétique d'événéments (dans une évolution).

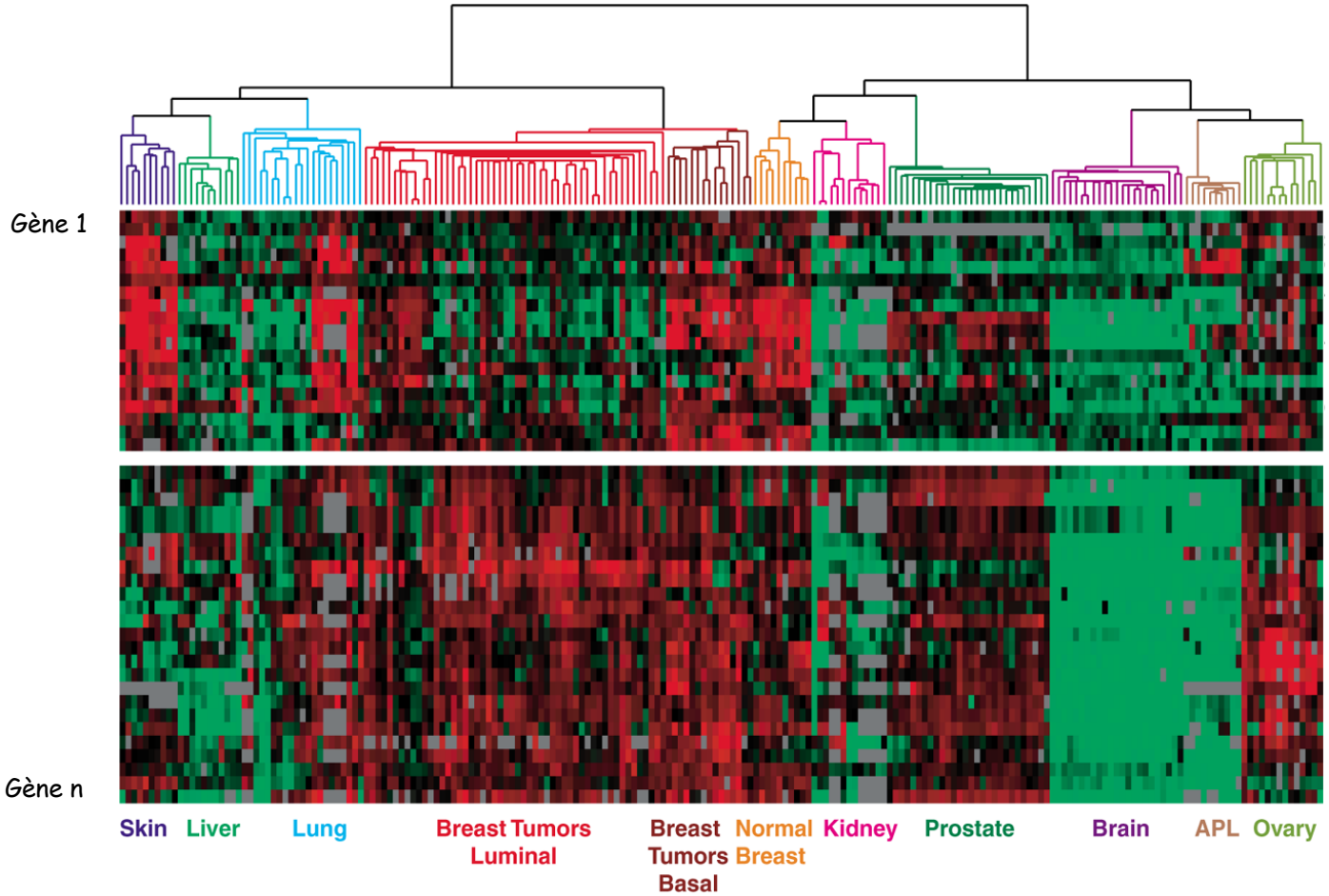
- Feuilles = gènes.
- Noeuds internes = ancêtres hypothétiques.



Référence: http://www.biostat.wisc.edu/bmi576/fall-2003/lecture13.pdf

Dendrogramme de tumeurs cancéreuses

Les tumeurs apparaissant dans des tissus similaires sont regroupées en grappes.



Référence: Botstein & Brown group