COMPLEX
Cours 7 - Algorithmes probabilistes de graphe

Damien Vergnaud

Sorbonne Université – CNRS

Table des matières

Voisins les plus proches

Voisins les plus proches

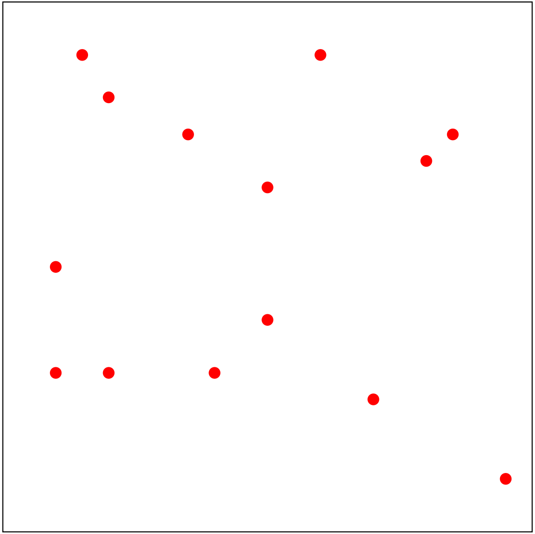
- Entrée : $n \geq 2$ un entier et $(P_1, \ldots, P_n)$ n points de $\mathbb{R}^2$
- Sortie : $(i, j) \in \{1, \ldots, n\}$ avec $i \neq j$ tel que $\mathrm{d}(P_i, P_j)$ minimale

- 1976 – M. O. Rabin
- Algorithme probabiliste en temps $O(n)$!

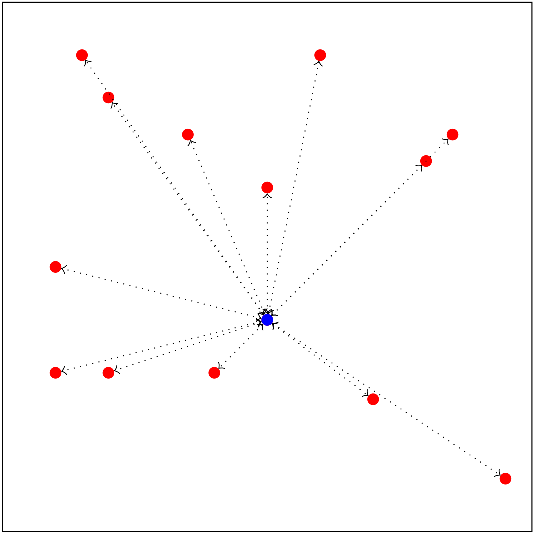
https://rjlipton.wordpress.com/2009/03/01/rabin-flips-a-coin/

- Nous allons voir une variante plus simple à analyser
- 1995 – S. Khuller, Y. Matias

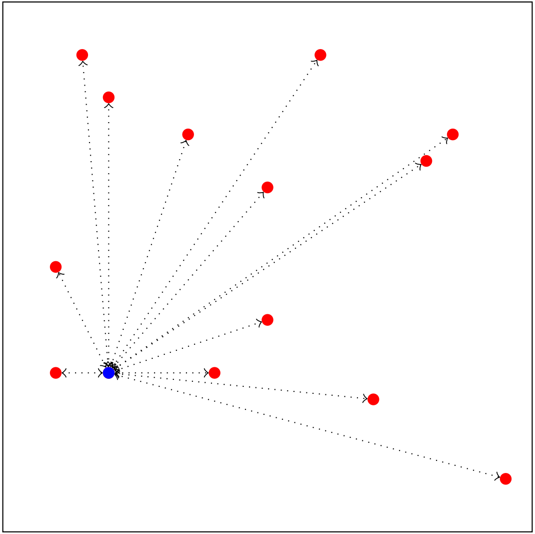
Voisins les plus proches - Algorithme déterministe



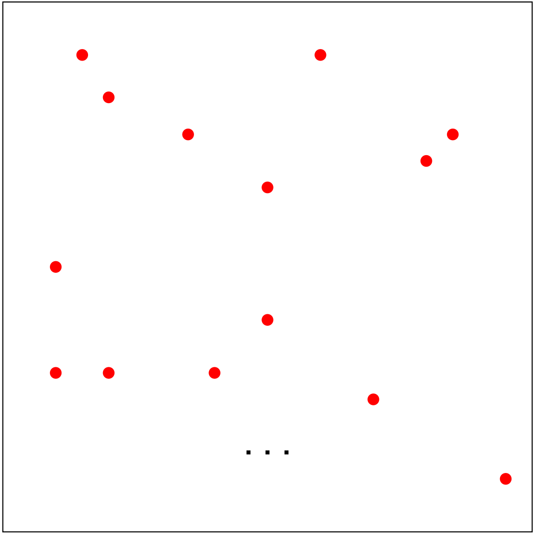
Voisins les plus proches - Algorithme déterministe



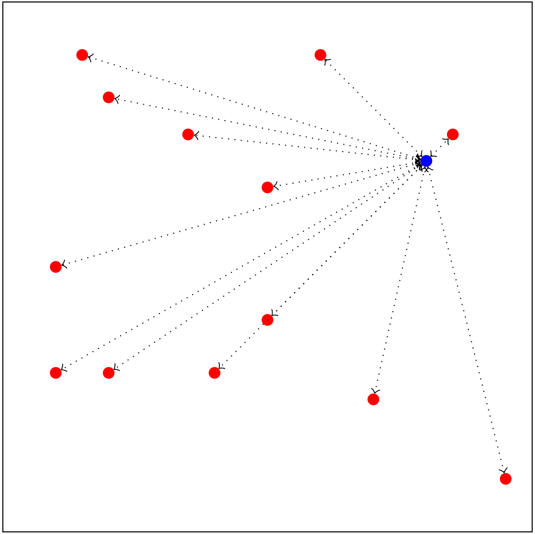
Voisins les plus proches - Algorithme déterministe



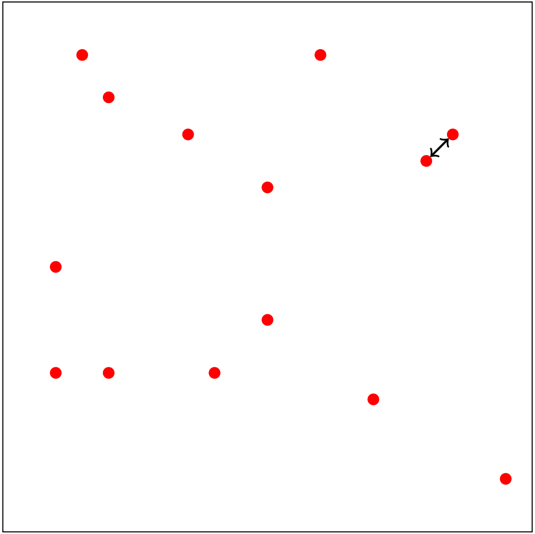
Voisins les plus proches - Algorithme déterministe



Voisins les plus proches - Algorithme déterministe



Voisins les plus proches - Algorithme déterministe



Voisins les plus proches - Algorithme déterministe

Entrée: $n \in \mathbb{N}$, $P_1, \ldots, P_n \in \mathbb{R}^2$
Sortie: $(i^\star, j^\star) \in \{1, \ldots, n\}$ avec $i \neq j$ tel que $d(P_i, P_j)$ minimale

$i^\star \leftarrow 0$; $j^\star \leftarrow 0$, $d \leftarrow +\infty$
pour i de 1 à $n-1$ **faire**
  pour j de $i+1$ à n **faire**
    si $d(P_i, P_j) < d$ **alors**
      $(i^\star, j^\star) \leftarrow (i, j)$
      $d \leftarrow d(P_i, P_j)$
    fin si
  fin pour
fin pour
retourner $(i^\star, j^\star)$

Complexité :

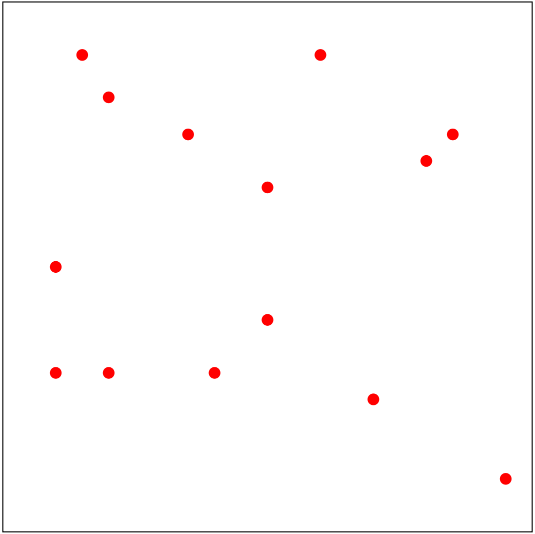
Voisins les plus proches - Algorithme déterministe

Entrée: $n \in \mathbb{N}$, $P_1, \ldots, P_n \in \mathbb{R}^2$
Sortie: $(i^\star, j^\star) \in \{1, \ldots, n\}$ avec $i \neq j$ tel que $d(P_i, P_j)$ minimale

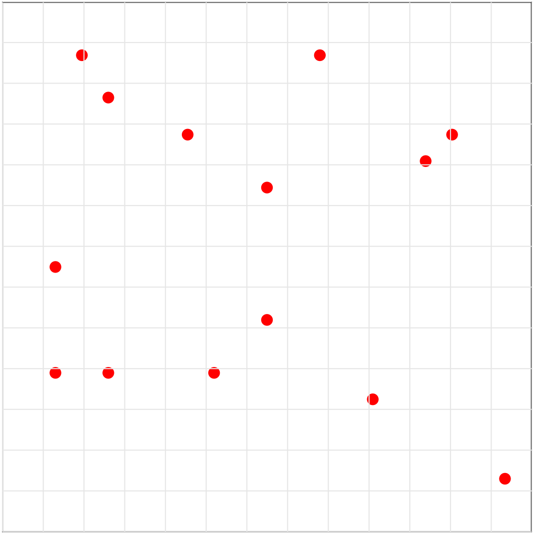
$i^\star \leftarrow 0$; $j^\star \leftarrow 0$, $d \leftarrow +\infty$
pour i de 1 à $n-1$ **faire**
  pour j de $i+1$ à n **faire**
    si $d(P_i, P_j) < d$ **alors**
      $(i^\star, j^\star) \leftarrow (i, j)$
      $d \leftarrow d(P_i, P_j)$
    fin si
  fin pour
fin pour
retourner $(i^\star, j^\star)$

Complexité : $O(n^2)$

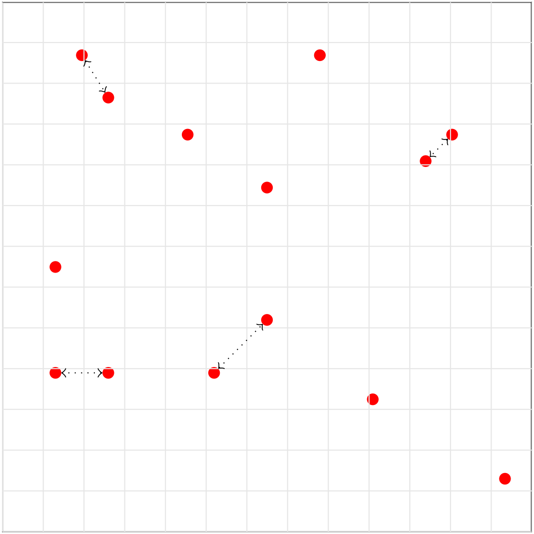
Voisins les plus proches - Idée de Rabin



Voisins les plus proches - Idée de Rabin

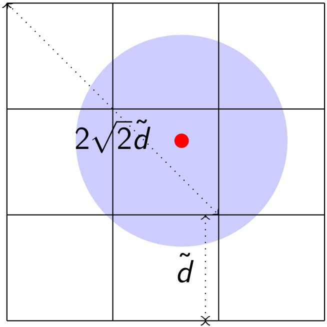


Voisins les plus proches - Idée de Rabin



Voisins les plus proches - Idée de Rabin

- calculer une approximation $\tilde{d}$ de la distance minimale $d^\star$
- construire un maillage de taille $\tilde{d}$



- Tout point a distance $\leq \tilde{d}$ est dans le « *voisinage* »
- Tout point a distance $> 2\sqrt{2}\tilde{d}$ est hors du son « *voisinage* »

Voisins les plus proches - Idée de Rabin

- si $\tilde{d} \leq d^\star \leq 3\tilde{d}$, il suffit de :
  - associer à chaque point P_i de S son voisinage

$$P_i = (x_i, y_i) \rightsquigarrow (m, n) \text{ tel que } \begin{cases} m\tilde{d} \leq x_i < (m+1)\tilde{d} \\ n\tilde{d} \leq y_i < (n+1)\tilde{d} \end{cases}$$

  - calculer les distances de chaque point de ce voisinage à P_i
    - **combien ?**
    - **comment ?**
  - retourner le couple $(i^\star, j^\star)$ pour lequel la distance $d(P_{i^\star}, P_{j^\star})$ est minimale
- Comment trouver $\tilde{d}$?

Voisins les plus proches - Idée de Rabin

- si $\tilde{d} \leq d^\star \leq 3\tilde{d}$, il suffit de :
  - associer à chaque point P_i de S son voisinage

$$P_i = (x_i, y_i) \rightsquigarrow (m, n) \text{ tel que } \begin{cases} m\tilde{d} \leq x_i < (m+1)\tilde{d} \\ n\tilde{d} \leq y_i < (n+1)\tilde{d} \end{cases}$$

  - calculer les distances de chaque point de ce voisinage à P_i
    - **combien ?**
    - **comment ?**
  - retourner le couple $(i^\star, j^\star)$ pour lequel la distance $d(P_{i^\star}, P_{j^\star})$ est minimale
- Comment trouver $\tilde{d}$?

Voisins les plus proches - Idée de Rabin

- si $\tilde{d} \leq d^\star \leq 3\tilde{d}$, il suffit de :
  - associer à chaque point P_i de S son voisinage

$$P_i = (x_i, y_i) \rightsquigarrow (m, n) \text{ tel que } \begin{cases} m\tilde{d} \leq x_i < (m+1)\tilde{d} \\ n\tilde{d} \leq y_i < (n+1)\tilde{d} \end{cases}$$

  - calculer les distances de chaque point de ce voisinage à P_i
    - **combien ?**
    - **comment ?**
  - retourner le couple $(i^\star, j^\star)$ pour lequel la distance $d(P_{i^\star}, P_{j^\star})$ est minimale
- Comment trouver $\tilde{d}$?

Voisins les plus proches - Idée de Rabin

- si $\tilde{d} \leq d^\star \leq 3\tilde{d}$, il suffit de :
  - associer à chaque point P_i de S son voisinage

$$P_i = (x_i, y_i) \rightsquigarrow (m, n) \text{ tel que } \begin{cases} m\tilde{d} \leq x_i < (m+1)\tilde{d} \\ n\tilde{d} \leq y_i < (n+1)\tilde{d} \end{cases}$$

  - calculer les distances de chaque point de ce voisinage à P_i
    - **combien ?**
    - **comment ?**
  - retourner le couple $(i^\star, j^\star)$ pour lequel la distance $d(P_{i^\star}, P_{j^\star})$ est minimale
- Comment trouver $\tilde{d}$?

Voisins les plus proches - Idée de Rabin

- si $\tilde{d} \leq d^\star \leq 3\tilde{d}$, il suffit de :
  - associer à chaque point P_i de S son voisinage

$$P_i = (x_i, y_i) \rightsquigarrow (m, n) \text{ tel que } \begin{cases} m\tilde{d} \leq x_i < (m+1)\tilde{d} \\ n\tilde{d} \leq y_i < (n+1)\tilde{d} \end{cases}$$

  - calculer les distances de chaque point de ce voisinage à P_i
    - **combien ?**
    - **comment ?**
  - retourner le couple $(i^\star, j^\star)$ pour lequel la distance $d(P_{i^\star}, P_{j^\star})$ est minimale
- Comment trouver $\tilde{d}$?

Voisins les plus proches - Idée de Rabin

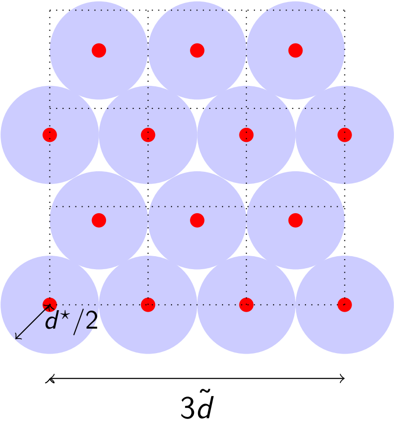
- si $\tilde{d} \leq d^\star \leq 3\tilde{d}$, il suffit de :
  - associer à chaque point P_i de S son voisinage

$$P_i = (x_i, y_i) \rightsquigarrow (m, n) \text{ tel que } \begin{cases} m\tilde{d} \leq x_i < (m+1)\tilde{d} \\ n\tilde{d} \leq y_i < (n+1)\tilde{d} \end{cases}$$

  - calculer les distances de chaque point de ce voisinage à P_i
    - **combien ?**
    - **comment ?**
  - retourner le couple $(i^\star, j^\star)$ pour lequel la distance $d(P_{i^\star}, P_{j^\star})$ est minimale
- Comment trouver $\tilde{d}$?

Voisinages

- **combien ?** $3\tilde{d} \geq d^\star \geq \tilde{d}$



≤ 14 dans un voisinage

Voisinages

- **comment ?** $\rightsquigarrow$ construction d'un « dictionnaire »

$$P = (x, y) \rightsquigarrow (m, n) \text{ tel que } \begin{cases} m\tilde{d} \leq x < (m+1)\tilde{d} \\ n\tilde{d} \leq y < (n+1)\tilde{d} \end{cases}$$

  - Ajout de $[(m, n), P]$ au « dictionnaire »
  - Obtenir la liste des points dans la « *cellule* » (m, n)

- Avec hachage (probabiliste) $\rightsquigarrow$ temps $O(1)$
- Avec arbre équilibré (déterministe) $\rightsquigarrow$ temps $O(\log n)$

Voisinages

- **comment ?** $\rightsquigarrow$ construction d'un « dictionnaire »

$$P = (x, y) \rightsquigarrow (m, n) \text{ tel que } \begin{cases} m\tilde{d} \leq x < (m+1)\tilde{d} \\ n\tilde{d} \leq y < (n+1)\tilde{d} \end{cases}$$

  - Ajout de $[(m, n), P]$ au « dictionnaire »
  - Obtenir la liste des points dans la « *cellule* » (m, n)

- Avec hachage (probabiliste) $\rightsquigarrow$ temps $O(1)$
- Avec arbre équilibré (déterministe) $\rightsquigarrow$ temps $O(\log n)$

Algorithme avec valeur approchée (1/2)

Entrée: $n \in \mathbb{N}$, $P_1, \dots, P_n \in \mathbb{R}^2$; $\tilde{d}$ tel que $\tilde{d}/3 \leq d^\star \leq \tilde{d}$
Sortie: $(i^\star, j^\star) \in \{1, \dots, n\}$ avec $i \neq j$ tel que $d(P_i, P_j) = d^\star$

$i^\star \leftarrow 0$; $j^\star \leftarrow 0$, $d \leftarrow +\infty$
$\Upsilon \leftarrow \emptyset$ $\quad\triangleright$ Dictionnaire
pour $P = (x, y)$ dans S **faire**
  $(m, n) \leftarrow \textsc{Cellule}(x, y, \tilde{d})$
  $\textsc{Ajout}([(m, n), P], \Upsilon)$
fin pour
... $\quad\triangleright$ Dictionnaire construit

Algorithme avec valeur approchée (2/2)

Entrée: $n \in \mathbb{N}$, $P_1, \ldots, P_n \in \mathbb{R}^2$; $\tilde{d}$ tel que $\tilde{d}/3 \leq d^\star \leq \tilde{d}$
Sortie: $(i^\star, j^\star) \in \{1, \ldots, n\}$ avec $i \neq j$ tel que $d(P_i, P_j) = d^\star$

...
pour $P_i = (x_i, y_i)$ dans S **faire**
  $(m, n) \leftarrow \textsc{Cellule}(x_i, y_i, \tilde{d})$
  pour c dans $\textsc{Voisinage}((m, n), \tilde{d})$ **faire**
    pour P_j dans $\Upsilon(c)$ **faire**
      si $d(P_i, P_j) < d$ **alors**
        $(i^\star, j^\star) \leftarrow (i, j)$
        $d \leftarrow d(P_i, P_j)$
      fin si
    fin pour
  fin pour
fin pour
retourner $(i^\star, j^\star)$

Algorithme avec valeur approchée – Analyse

- Si $\tilde{d}/3 \leq d^\star \leq \tilde{d}$,
  - Le maillage n'est **pas trop fin**
    $\rightsquigarrow$ La distance minimale est toujours trouvée
  - Le maillage n'est **pas trop grossier**
    $\rightsquigarrow$ Le nombre de points dans chaque cellule est constant

Complexité totale : $O(n)$ ou $O(n \log n)$

Probabilité d'erreur : 0

Algorithme avec valeur approchée – Analyse

- Si $\tilde{d}/3 \leq d^{\star} \leq \tilde{d}$,
  - Le maillage n'est **pas trop fin**
    $\rightsquigarrow$ La distance minimale est toujours trouvée
  - Le maillage n'est **pas trop grossier**
    $\rightsquigarrow$ Le nombre de points dans chaque cellule est constant

Complexité totale : $O(n)$ ou $O(n \log n)$

Probabilité d'erreur : 0

Algorithme avec valeur approchée – Analyse

- Si $\tilde{d}/3 \leq d^\star \leq \tilde{d}$,
  - Le maillage n'est **pas trop fin**
    $\rightsquigarrow$ La distance minimale est toujours trouvée
  - Le maillage n'est **pas trop grossier**
    $\rightsquigarrow$ Le nombre de points dans chaque cellule est constant

Complexité totale : $O(n)$ ou $O(n \log n)$

Probabilité d'erreur : 0

Approcher la distance minimale

- Au hasard !

- on choisit un point au hasard
- on calcule la distance minimale de ce point à tous les autres $\rightsquigarrow \tilde{d}$
- **Est-ce que** $\tilde{d}/3 \leq d^\star \leq \tilde{d}$ **?**

- On regarde ...

Approcher la distance minimale

- Au hasard !

- on choisit un point au hasard
- on calcule la distance minimale de ce point à tous les autres $\rightsquigarrow \tilde{d}$
- **Est-ce que** $\tilde{d}/3 \leq d^{\star} \leq \tilde{d}$ **?**

- On regarde ...

Approcher la distance minimale

- Au hasard !

- on choisit un point au hasard
- on calcule la distance minimale de ce point à tous les autres $\rightsquigarrow \tilde{d}$
- **Est-ce que** $\tilde{d}/3 \leq d^\star \leq \tilde{d}$ **?**

- On regarde ...

Approcher la distance minimale

- Au hasard !

- on choisit un point au hasard
- on calcule la distance minimale de ce point à tous les autres $\rightsquigarrow \tilde{d}$
- **Est-ce que** $\tilde{d}/3 \leq d^{\star} \leq \tilde{d}$ **?**

- On regarde ...

Approcher la distance minimale

- On construit une suite décroissante d'ensembles :

$$S = S_0 \supseteq S_1 \supseteq S_2 \supseteq \cdots \supseteq S_i \supseteq \ldots$$

- Pour chaque entier i,
  - on choisit un point au hasard dans S_i
  - on calcule la distance minimale de ce point à tous les autres dans $S_i \rightsquigarrow d_i$ (avec $d_i \geq d^\star$)
  - On construit le maillage associé à la distance $d_i/3$
  - On conserve tous les points dont les voisinages ne sont pas vides $\rightsquigarrow S_{i+1}$
    - $S_{i+1} = \emptyset \rightsquigarrow d^\star < d_i/3$ et on applique l'algorithme précédent
    - $S_{i+1} \neq \emptyset \rightsquigarrow d^\star \geq d_i/3$ et on continue

Approcher la distance minimale

- On construit une suite décroissante d'ensembles :

$$S = S_0 \supseteq S_1 \supseteq S_2 \supseteq \cdots \supseteq S_i \supseteq \dots$$

- Pour chaque entier i,
  - on choisit un point au hasard dans S_i
  - on calcule la distance minimale de ce point à tous les autres dans $S_i \rightsquigarrow d_i$ (avec $d_i \geq d^\star$)
  - On construit le maillage associé à la distance $d_i/3$
  - On conserve tous les points dont les voisinages ne sont pas vides $\rightsquigarrow S_{i+1}$
    - $S_{i+1} = \emptyset \rightsquigarrow d^\star < d_i/3$ et on applique l'algorithme précédent
    - $S_{i+1} \neq \emptyset \rightsquigarrow d^\star \geq d_i/3$ et on continue

Approcher la distance minimale

- On construit une suite décroissante d'ensembles :

$$S = S_0 \supseteq S_1 \supseteq S_2 \supseteq \cdots \supseteq S_i \supseteq \ldots$$

- Pour chaque entier i,
  - on choisit un point au hasard dans S_i
  - on calcule la distance minimale de ce point à tous les autres dans $S_i \rightsquigarrow d_i$ (avec $d_i \geq d^\star$)
  - On construit le maillage associé à la distance $d_i/3$
  - On conserve tous les points dont les voisinages ne sont pas vides $\rightsquigarrow S_{i+1}$
    - $S_{i+1} = \emptyset \rightsquigarrow d^\star < d_i/3$ et on applique l'algorithme précédent
    - $S_{i+1} \neq \emptyset \rightsquigarrow d^\star \geq d_i/3$ et on continue

Approcher la distance minimale

- On construit une suite décroissante d'ensembles :

$$S = S_0 \supseteq S_1 \supseteq S_2 \supseteq \cdots \supseteq S_i \supseteq \ldots$$

- Pour chaque entier i,
  - on choisit un point au hasard dans S_i
  - on calcule la distance minimale de ce point à tous les autres dans $S_i \rightsquigarrow d_i$ (avec $d_i \geq d^\star$)
  - On construit le maillage associé à la distance $d_i/3$
  - On conserve tous les points dont les voisinages ne sont pas vides $\rightsquigarrow S_{i+1}$
    - $S_{i+1} = \emptyset \rightsquigarrow d^\star < d_i/3$ et on applique l'algorithme précédent
    - $S_{i+1} \neq \emptyset \rightsquigarrow d^\star \geq d_i/3$ et on continue

Approcher la distance minimale

- On construit une suite décroissante d'ensembles :

$$S = S_0 \supseteq S_1 \supseteq S_2 \supseteq \cdots \supseteq S_i \supseteq \ldots$$

- Pour chaque entier i,
  - on choisit un point au hasard dans S_i
  - on calcule la distance minimale de ce point à tous les autres dans $S_i \rightsquigarrow d_i$ (avec $d_i \geq d^\star$)
  - On construit le maillage associé à la distance $d_i/3$
  - On conserve tous les points dont les voisinages ne sont pas vides $\rightsquigarrow S_{i+1}$
    - $S_{i+1} = \emptyset \rightsquigarrow d^\star < d_i/3$ et on applique l'algorithme précédent
    - $S_{i+1} \neq \emptyset \rightsquigarrow d^\star \geq d_i/3$ et on continue

Algorithme probabiliste (1/3)

Entrée: $n \in \mathbb{N}$, $P_1, \dots, P_n \in \mathbb{R}^2$
Sortie: $(i^\star, j^\star) \in \{1, \dots, n\}$ avec $i \neq j$ tel que $d(P_i, P_j) = d^\star$

$S \leftarrow \{P_1, \dots, P_n\}$; $\tilde{d} \leftarrow +\infty$
tant que VRAI **faire**
  $P_i \xleftarrow{\text{⚂⚃}} S$
  pour P_j dans S avec $P_i \neq P_j$ **faire**
    si $d(P_i, P_j) < \tilde{d}$ **alors**
      $\tilde{d} \leftarrow d(P_i, P_j)$
    fin si
  fin pour
  ...
fin tant que

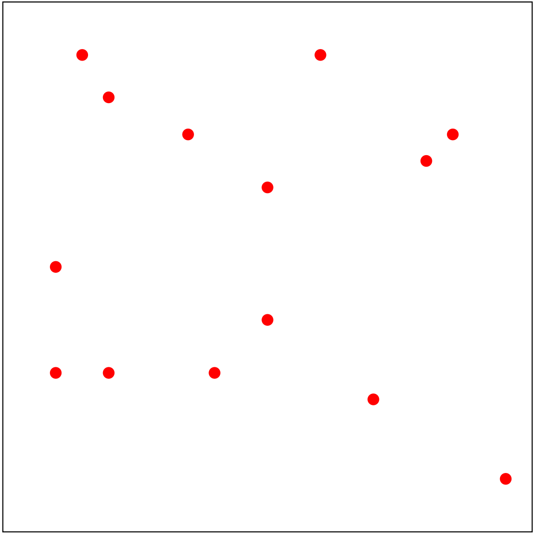
Algorithme probabiliste (2/3)

tant que VRAI **faire**
  ...
  $\Upsilon \leftarrow \emptyset$ ▷ Dictionnaire
  pour $P = (x, y)$ dans S **faire**
    $(m, n) \leftarrow \textsc{Cellule}(x, y, \tilde{d}/3)$
    $\textsc{Ajout}([(m, n), P], \Upsilon)$
  fin pour
  $T \leftarrow \emptyset$
  pour $P = (x, y)$ dans S **faire**
    $(m, n) \leftarrow \textsc{Cellule}(x, y, \tilde{d}/3)$
    pour c dans $\textsc{Voisinage}((m, n), \tilde{d}/3)$ **faire**
      si $\Upsilon(c) \neq \emptyset$ **alors**
        $T \leftarrow T \cup \{P\}$
      fin si
    fin pour
  fin pour
  ...

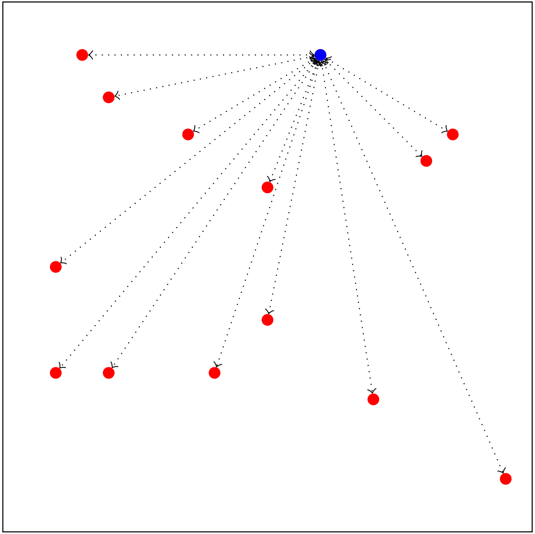
Algorithme probabiliste (3/3)

tant que VRAI **faire**
  ...
  si $T \neq \emptyset$ **alors**
    $S \leftarrow T$
  sinon
    Algorithme_avec_approx$(\{P_1, \ldots, P_n\}, \tilde{d})$
  fin si
fin tant que

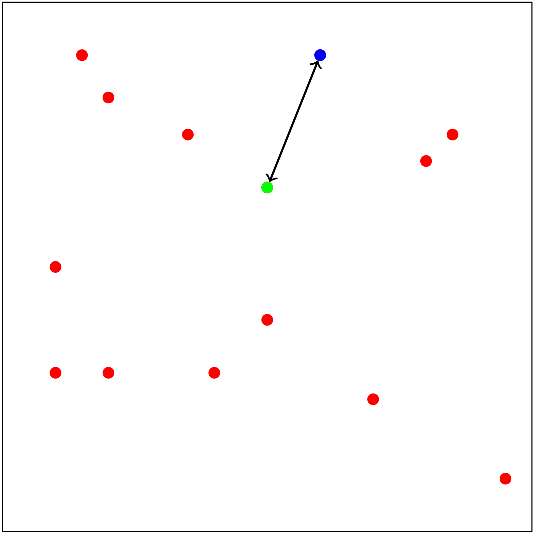
Voisins les plus proches



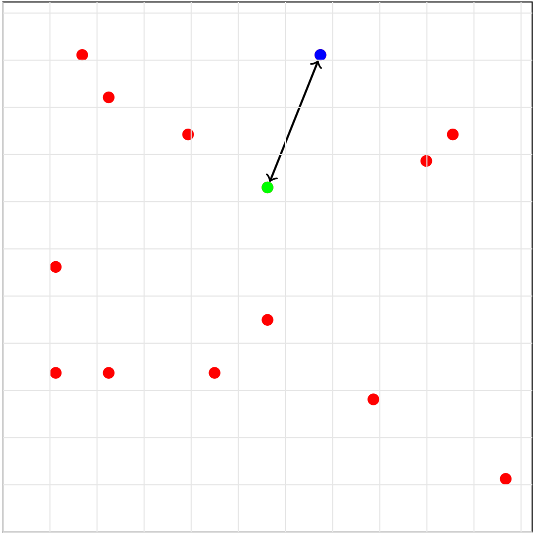
Voisins les plus proches



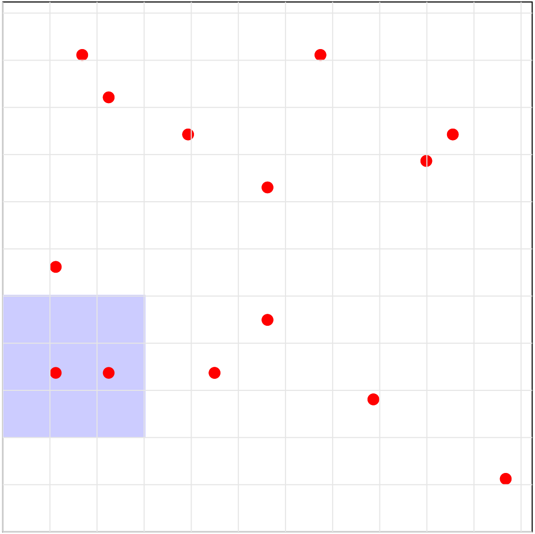
Voisins les plus proches



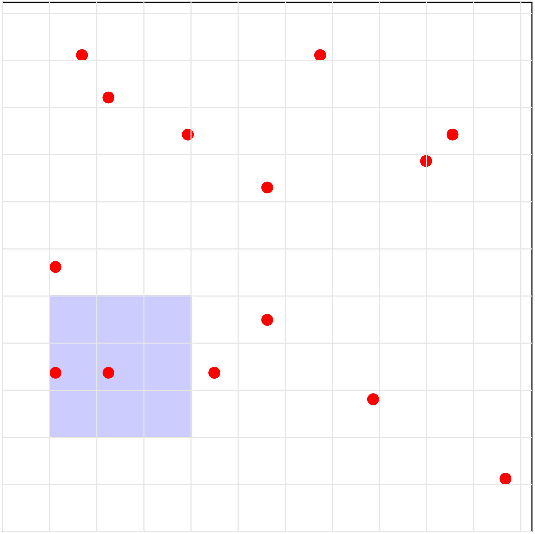
Voisins les plus proches



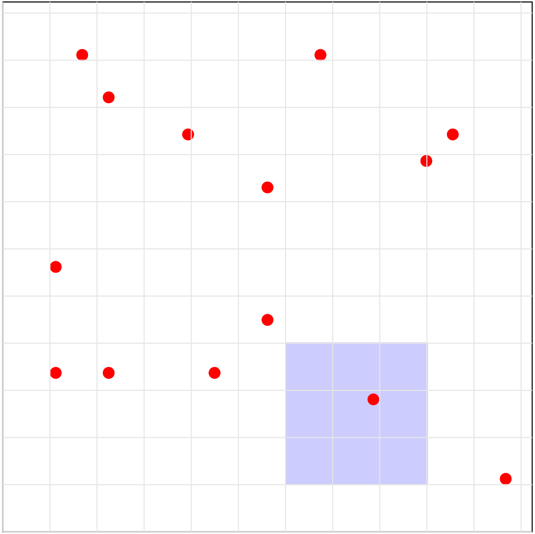
Voisins les plus proches



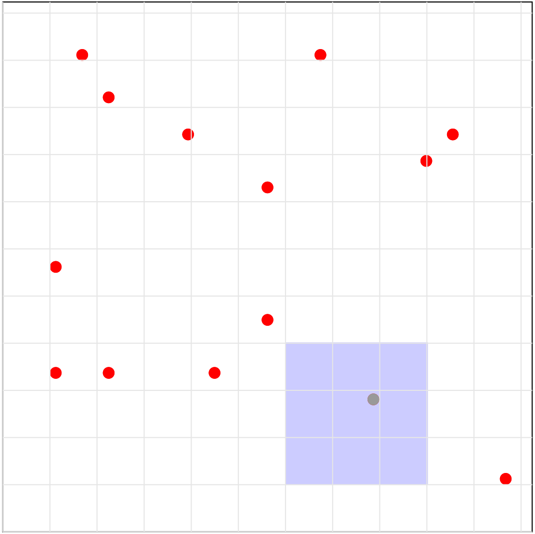
Voisins les plus proches



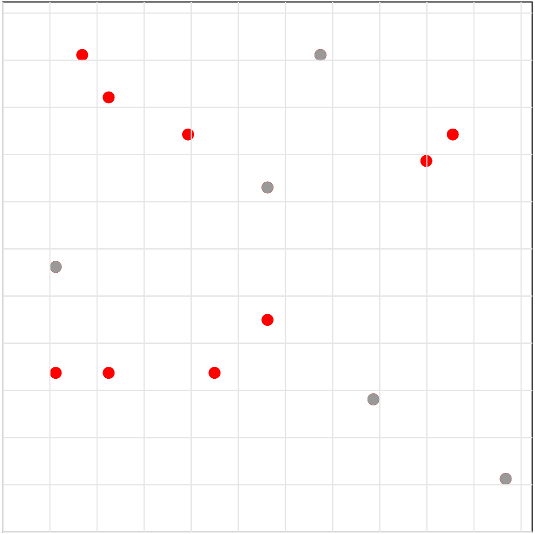
Voisins les plus proches



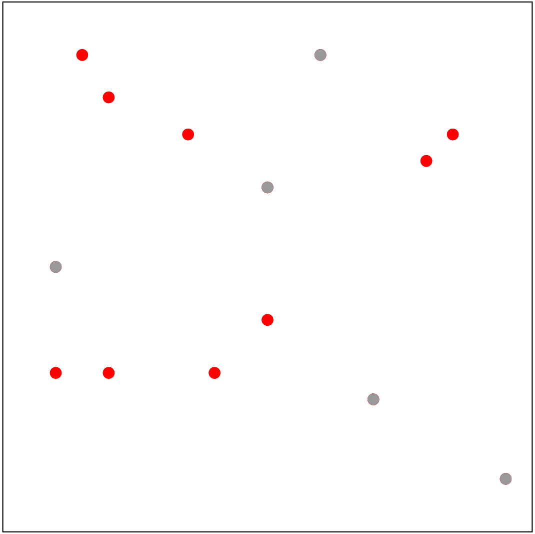
Voisins les plus proches



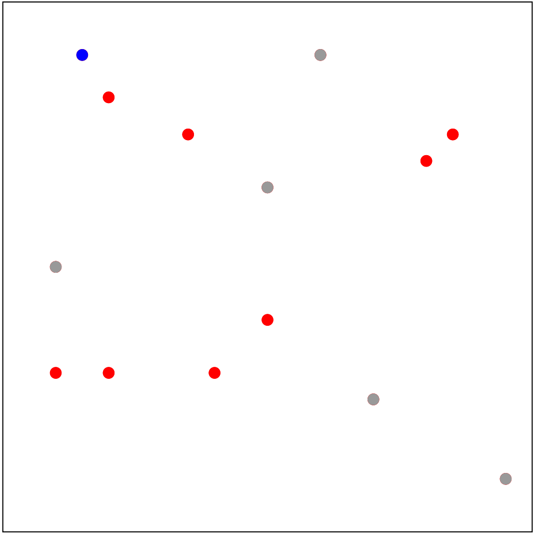
Voisins les plus proches



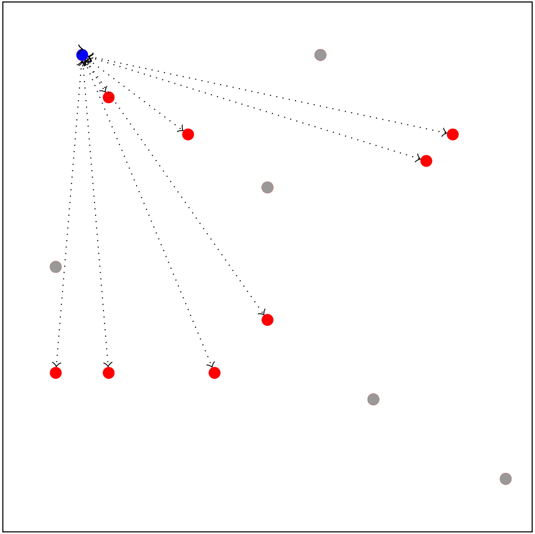
Voisins les plus proches



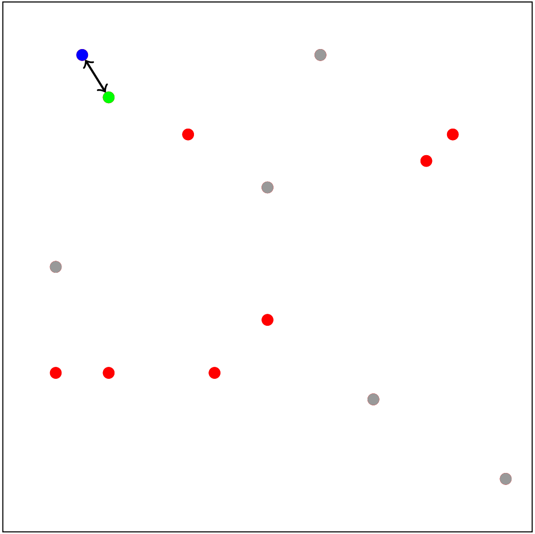
Voisins les plus proches



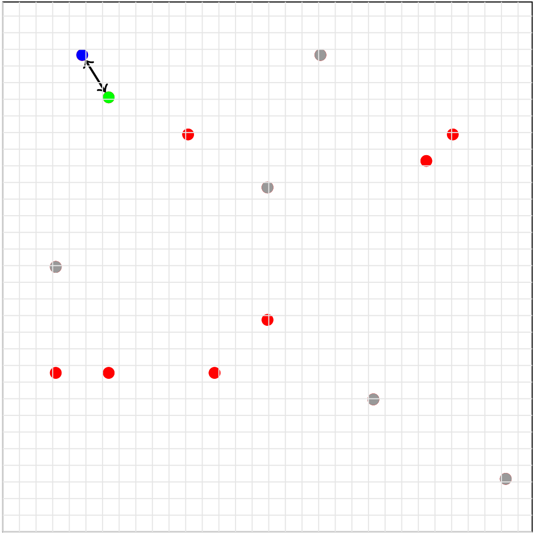
Voisins les plus proches



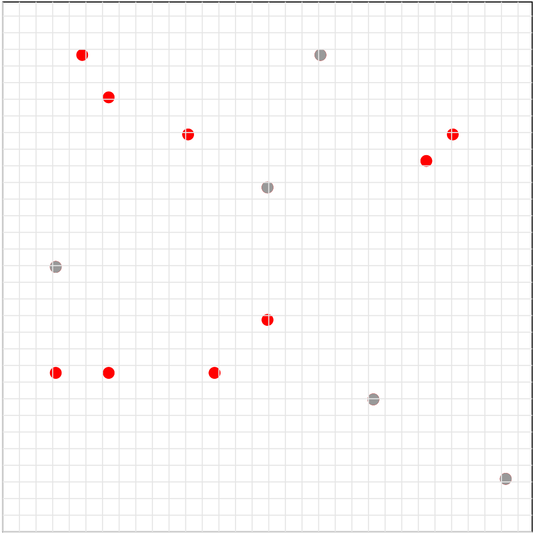
Voisins les plus proches



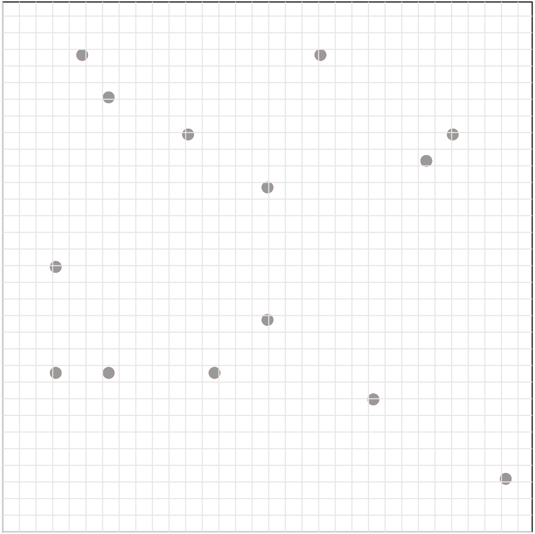
Voisins les plus proches



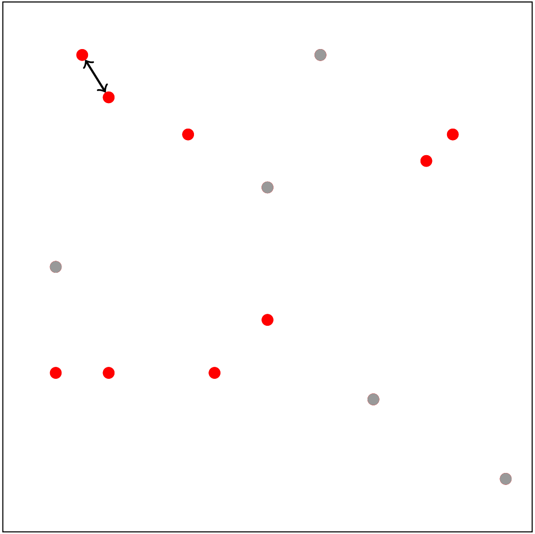
Voisins les plus proches



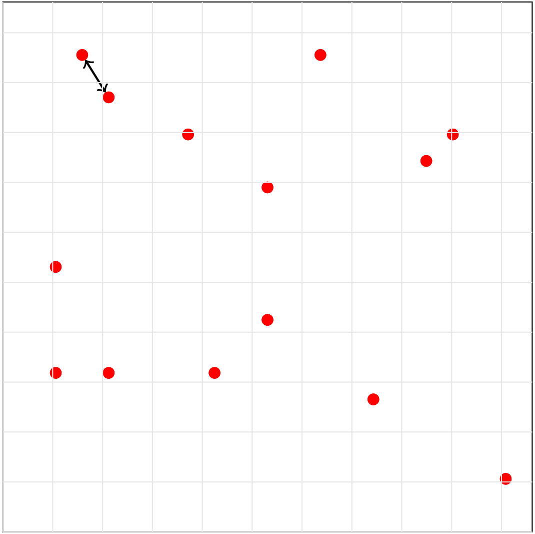
Voisins les plus proches



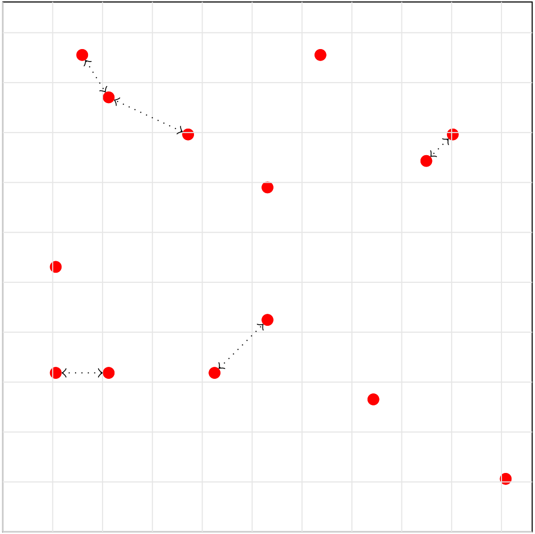
Voisins les plus proches



Voisins les plus proches



Voisins les plus proches



Analyse (1/2)

- Algorithme de type Las Vegas
- À chaque itération de la boucle **tant que**

$$O(\#S_i) \text{ opérations du dictionnaire}$$

- Notons

$$\delta(P_k) = \min_{P_j \neq P_k} d(P_k, P_j)$$

- Si P est le point tiré aléatoirement dans S_i, P_k est conservé dans S_{i+1} seulement si $\delta(P_k) \leq \delta(P)$
- En moyenne,

$$\mathbb{E}(\#S_{i+1}) = \mathbb{E}(\#S_i)/2$$

Analyse (2/2)

- $\#S_0 = \#S = n$ et par récurrence $\mathbb{E}(S_i) = \dfrac{n}{2^i}$
- Par linéarité de l'espérance, la complexité de l'algorithme est :

$$
\begin{aligned}
& O\left(\sum_{i=0}^{\infty} \#S_i\right) \text{ opérations du dictionnaire} \\
= & O\left(\sum_{i=0}^{\infty} \frac{n}{2^i}\right) \text{ opérations du dictionnaire} \\
= & O(2n) \text{ opérations du dictionnaire}
\end{aligned}
$$

$\square$

Complexité totale : $O(n)$ ou $O(n \log n)$

Probabilité d'erreur : 0

Analyse (2/2)

- $\#S_0 = \#S = n$ et par récurrence $\mathbb{E}(S_i) = \dfrac{n}{2^i}$
- Par linéarité de l'espérance, la complexité de l'algorithme est :

$$
\begin{aligned}
& O\left(\sum_{i=0}^{\infty} \#S_i\right) \text{ opérations du dictionnaire} \\
=& O\left(\sum_{i=0}^{\infty} \frac{n}{2^i}\right) \text{ opérations du dictionnaire} \\
=& O(2n) \text{ opérations du dictionnaire}
\end{aligned}
$$

$\square$

Complexité totale : $O(n)$ ou $O(n \log n)$

Probabilité d'erreur : 0

Analyse (2/2)

- $\#S_0 = \#S = n$ et par récurrence $\mathbb{E}(S_i) = \dfrac{n}{2^i}$
- Par linéarité de l'espérance, la complexité de l'algorithme est :

$$
\begin{aligned}
& O\left(\sum_{i=0}^{\infty} \#S_i\right) \text{ opérations du dictionnaire} \\
= & O\left(\sum_{i=0}^{\infty} \frac{n}{2^i}\right) \text{ opérations du dictionnaire} \\
= & O(2n) \text{ opérations du dictionnaire}
\end{aligned}
$$

$\square$

Complexité totale : $O(n)$ ou $O(n \log n)$

Probabilité d'erreur : 0

Table des matières

Coupe minimale

- $G = (V, E)$ **graphe non-orienté**
- Une **coupe** de G est une partition de V en deux sous-ensembles non-vides $S \neq \emptyset$ et $(V \setminus S) \neq \emptyset$
- Le **cardinal d'une coupe** $(S, V \setminus S)$ est le nombre d'arêtes de E ayant une extrémité dans S et l'autre dans $V \setminus S$
- Une coupe minimale de G est une coupe de cardinal minimal

Coupe minimale

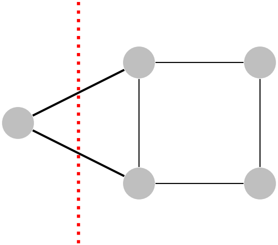
- $G = (V, E)$ **graphe non-orienté**
- Une **coupe** de G est une partition de V en deux sous-ensembles non-vides $S \neq \emptyset$ et $(V \setminus S) \neq \emptyset$
- Le **cardinal d'une coupe** $(S, V \setminus S)$ est le nombre d'arêtes de E ayant une extrémité dans S et l'autre dans $V \setminus S$
- Une coupe minimale de G est une coupe de cardinal minimal

Coupe minimale

- $G = (V, E)$ **graphe non-orienté**
- Une **coupe** de G est une partition de V en deux sous-ensembles non-vides $S \neq \emptyset$ et $(V \setminus S) \neq \emptyset$
- Le **cardinal d'une coupe** $(S, V \setminus S)$ est le nombre d'arêtes de E ayant une extrémité dans S et l'autre dans $V \setminus S$
- Une coupe minimale de G est une coupe de cardinal minimal

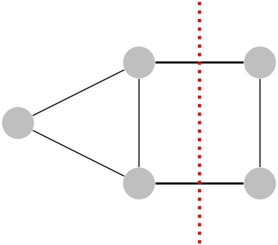
Coupe minimale

- $G = (V, E)$ **graphe non-orienté**
- Une **coupe** de G est une partition de V en deux sous-ensembles non-vides $S \neq \emptyset$ et $(V \setminus S) \neq \emptyset$
- Le **cardinal d'une coupe** $(S, V \setminus S)$ est le nombre d'arêtes de E ayant une extrémité dans S et l'autre dans $V \setminus S$
- Une coupe minimale de G est une coupe de cardinal minimal



Coupe minimale

- $G = (V, E)$ **graphe non-orienté**
- Une **coupe** de G est une partition de V en deux sous-ensembles non-vides $S \neq \emptyset$ et $(V \setminus S) \neq \emptyset$
- Le **cardinal d'une coupe** $(S, V \setminus S)$ est le nombre d'arêtes de E ayant une extrémité dans S et l'autre dans $V \setminus S$
- Une coupe minimale de G est une coupe de cardinal minimal



Coupe minimale

COUPE MINIMALE

- ENTRÉE : $G = (V, E)$ graphe non-orienté
- SORTIE : $S \subset V$ avec $S \neq \emptyset$ tel que $(S, V \setminus S)$ coupe minimale

- Théorème flot-max/coupe-min $\rightsquigarrow$ résoudre le problème de flot maximum associé.

- Complexité pour un problème de flot ($n = \#V$ et $m = \#E$)
  - Algorithme d'Edmonds-Karp $\rightsquigarrow O(nm^2)$
  - Algorithme de Dinic $\rightsquigarrow O(n^2 \cdot m)$
  - Algorithme de Goldberg-Tarjan $\rightsquigarrow O(n^3)$ ou $O(n^2 \cdot \sqrt{m})$
- Complexité pour la coupe minimum $\rightsquigarrow O(n^5)$ ou $O(n^4)$

- Algorithme probabiliste ?
  $\rightsquigarrow$ 1993 – D. R. Karger

Coupe minimale

Coupe minimale

- Entrée : $G = (V, E)$ graphe non-orienté
- Sortie : $S \subset V$ avec $S \neq \emptyset$ tel que $(S, V \setminus S)$ coupe minimale

- Théorème flot-max/coupe-min $\rightsquigarrow$ résoudre le problème de flot maximum associé.

- Complexité pour un problème de flot ($n = \#V$ et $m = \#E$)
  - Algorithme d'Edmonds-Karp $\rightsquigarrow O(nm^2)$
  - Algorithme de Dinic $\rightsquigarrow O(n^2 \cdot m)$
  - Algorithme de Goldberg-Tarjan $\rightsquigarrow O(n^3)$ ou $O(n^2 \cdot \sqrt{m})$
- Complexité pour la coupe minimum $\rightsquigarrow O(n^5)$ ou $O(n^4)$

- Algorithme probabiliste ?
  $\rightsquigarrow$ 1993 – D. R. Karger

Multi-graphes et contraction

- $G = (V, E)$ **multi-graphe non-orienté**
  - V ensemble de sommets
  - E multi-ensemble d'arêtes
    (c.-à-d. une arête peut apparaître plusieurs fois)

- Pour $e = \{u, v\} \in E$, la contraction de l'arête e produit un nouveau multi-graphe $G/e = (V', E')$ avec
  - $V' = (V \setminus \{u, v\}) \cup \{uv\}$
  - E' est défini par

$$\begin{aligned} E' = \quad & \{e \in E | u \notin e \wedge v \notin e\} \\ & \cup \{\{w, uv\} | \{w, u\} \in E, w \neq v\} \\ & \cup \{\{w, uv\} | \{w, v\} \in E, w \neq u\} \end{aligned}$$

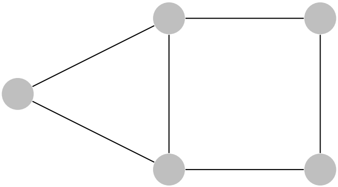
Multi-graphes et contraction

- $G = (V, E)$ **multi-graphe non-orienté**
  - V ensemble de sommets
  - E multi-ensemble d'arêtes
    (c.-à-d. une arête peut apparaître plusieurs fois)

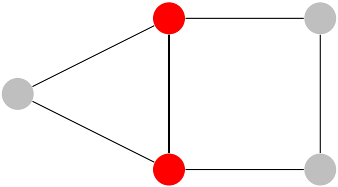
- Pour $e = \{u, v\} \in E$, la contraction de l'arête e produit un nouveau multi-graphe $G/e = (V', E')$ avec
  - $V' = (V \setminus \{u, v\}) \cup \{uv\}$
  - E' est défini par

$$
\begin{aligned}
E' = \quad & \{e \in E | u \notin e \wedge v \notin e\} \\
& \cup \{\{w, uv\} | \{w, u\} \in E, w \neq v\} \\
& \cup \{\{w, uv\} | \{w, v\} \in E, w \neq u\}
\end{aligned}
$$

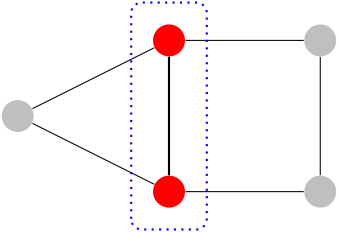
Multi-graphes et contraction



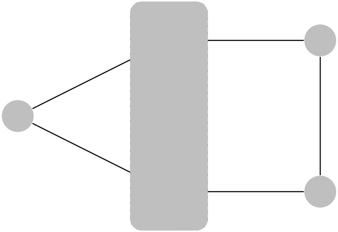
Multi-graphes et contraction



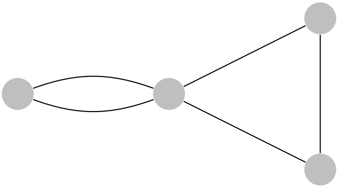
Multi-graphes et contraction



Multi-graphes et contraction



Multi-graphes et contraction



Coupes minimales et nombre d'arêtes

Lemme

Soit $G = (V, E)$ un multi-graphe non-orienté. Si sa coupe minimale est de cardinal k, alors $m \geq (n \cdot k)/2$ (avec $m = \#E$ et $n = \#V$).

Démonstration.

- Tout sommet a au moins k arêtes incidentes — Pourquoi ?

$$\forall v \in V, \quad \deg(v) \geq k$$

- Le nombre d'arêtes est égal à : — Pourquoi ?

$$m = \frac{1}{2} \sum_{v \in V} \deg(v)$$

Coupes minimales et nombre d'arêtes

Lemme

Soit $G = (V, E)$ un multi-graphe non-orienté. Si sa coupe minimale est de cardinal k, alors $m \geq (n \cdot k)/2$ (avec $m = \#E$ et $n = \#V$).

Démonstration.

- Tout sommet a au moins k arêtes incidentes — Pourquoi ?

$$\forall v \in V, \quad \deg(v) \geq k$$

- Le nombre d'arêtes est égal à : — Pourquoi ?

$$m = \frac{1}{2} \sum_{v \in V} \deg(v)$$

Coupes minimales et nombre d'arêtes

Lemme

Soit $G = (V, E)$ un multi-graphe non-orienté. Si sa coupe minimale est de cardinal k, alors $m \geq (n \cdot k)/2$ (avec $m = \#E$ et $n = \#V$).

Démonstration.

- Tout sommet a au moins k arêtes incidentes — Pourquoi ?

$$\forall v \in V, \quad \deg(v) \geq k$$

- Le nombre d'arêtes est égal à : — Pourquoi ?

$$m = \frac{1}{2} \sum_{v \in V} \deg(v)$$

□

Algorithme de Karger

Entrée: $G = (V, E)$ graphe non-orienté
Sortie: $S \subset V$ avec $S \neq \emptyset$ tel que $(S, V \setminus S)$ coupe minimale

tant que $\#V > 2$ **faire**
  $e \xleftarrow{\text{⚂⚃}} E$
  $G \leftarrow G/e$
fin tant que
$\{v_1, v_2\} \leftarrow V$ ▷ Il reste deux sommets dans V
retourner $S = \{$ sommets qui « apparaissent » dans v_1 $\}$

Algorithme de Karger – Complexité

- L'algorithme effectue exactement

$$n - 2 \text{ répétitions de la boucle } \textbf{tant que}$$

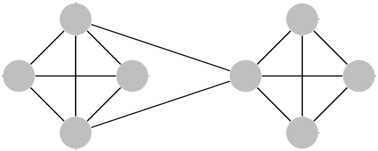
- Chaque opération de contraction peut être réalisé en

$$O(n) \text{ opérations}$$

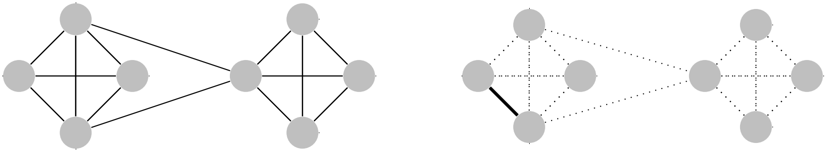
si le graphe est stocké par matrice d'adjacence

Complexité : $O(n^2)$ opérations sur la matrice d'adjacence

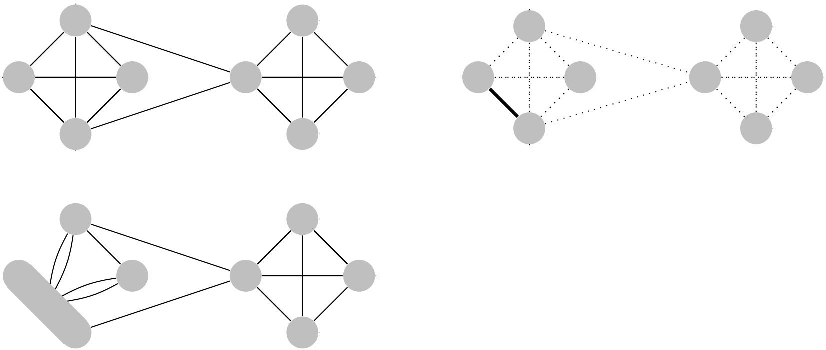
Algorithme de Karger – Exemple 1 (1/3)



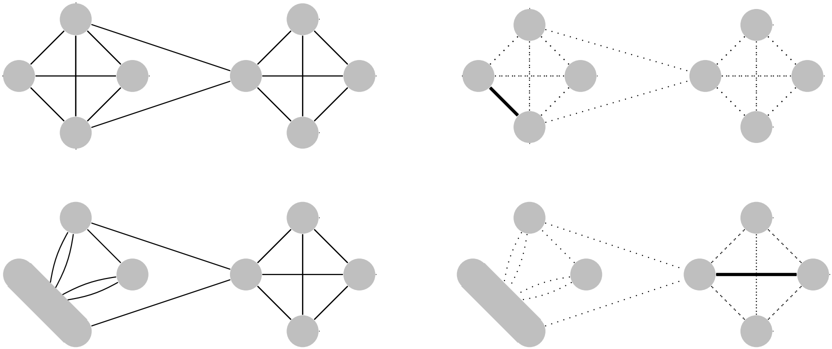
Algorithme de Karger – Exemple 1 (1/3)



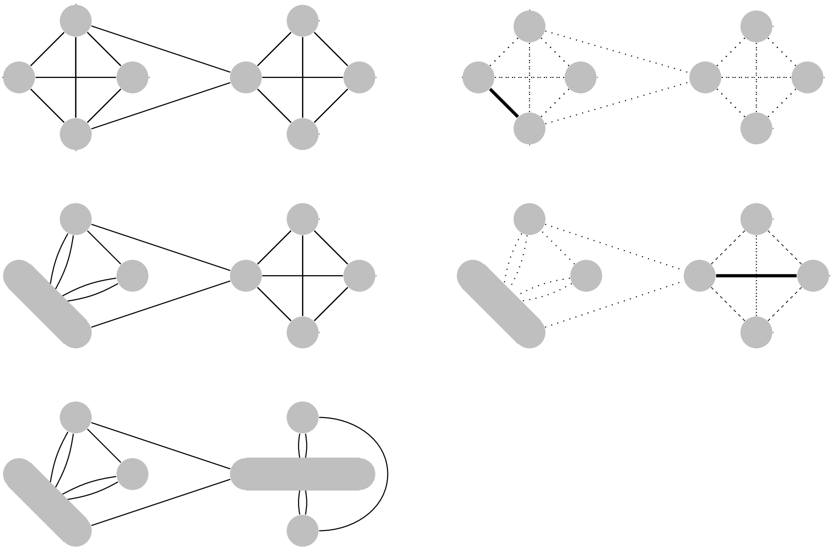
Algorithme de Karger – Exemple 1 (1/3)



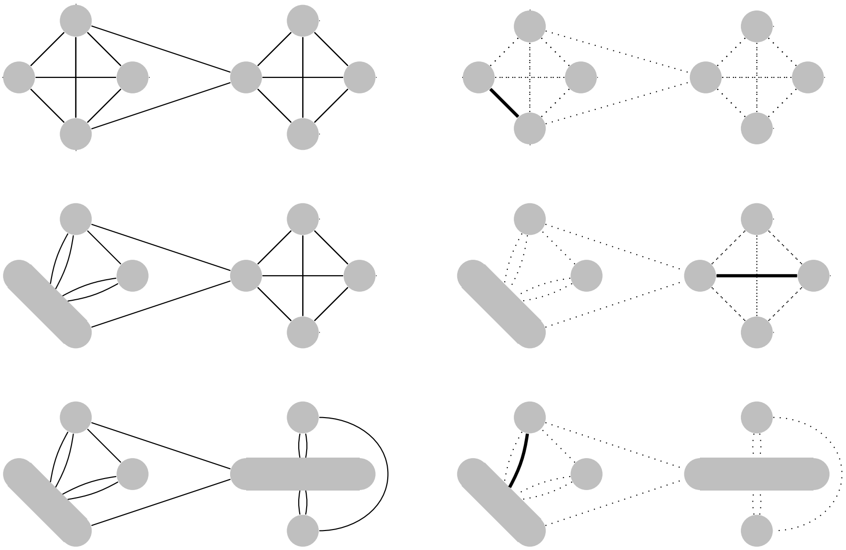
Algorithme de Karger – Exemple 1 (1/3)



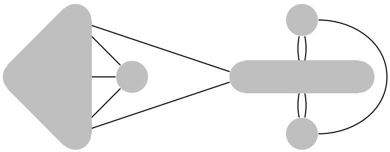
Algorithme de Karger – Exemple 1 (1/3)



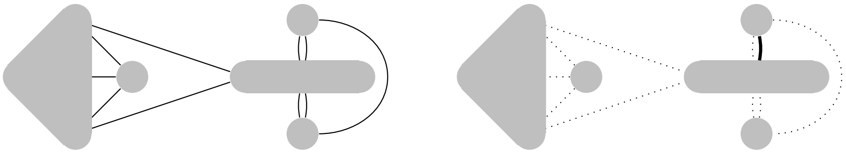
Algorithme de Karger – Exemple 1 (1/3)



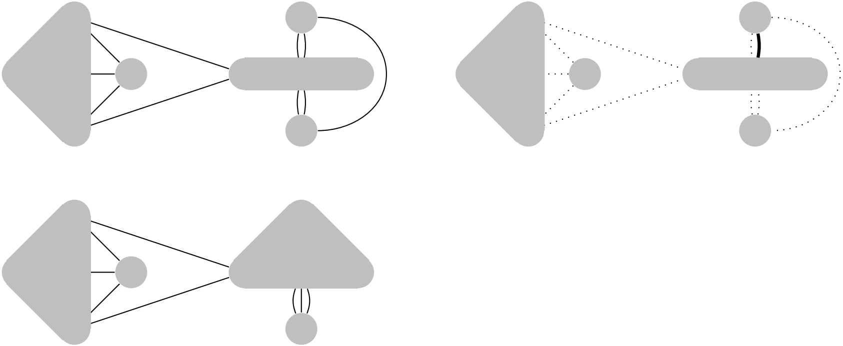
Algorithme de Karger – Exemple 1 (2/3)



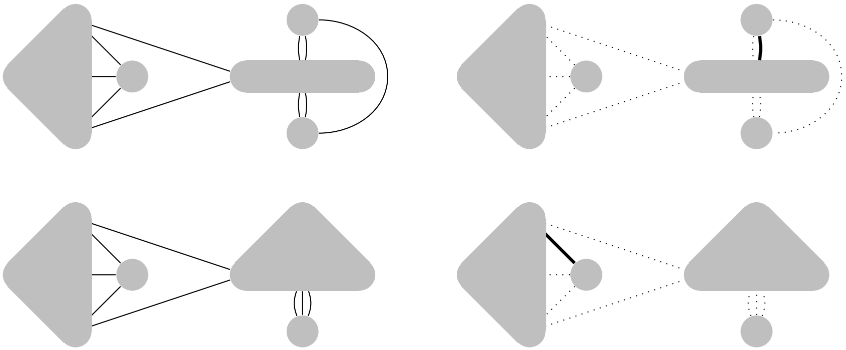
Algorithme de Karger – Exemple 1 (2/3)



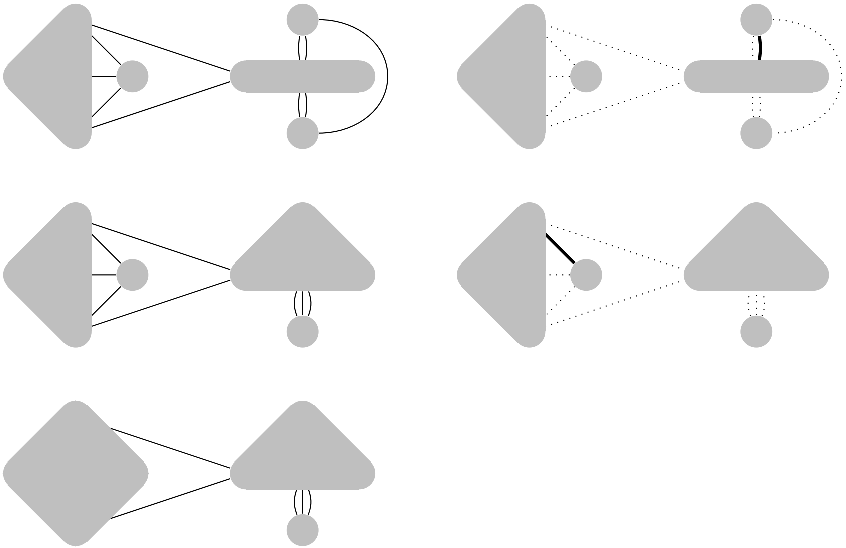
Algorithme de Karger – Exemple 1 (2/3)



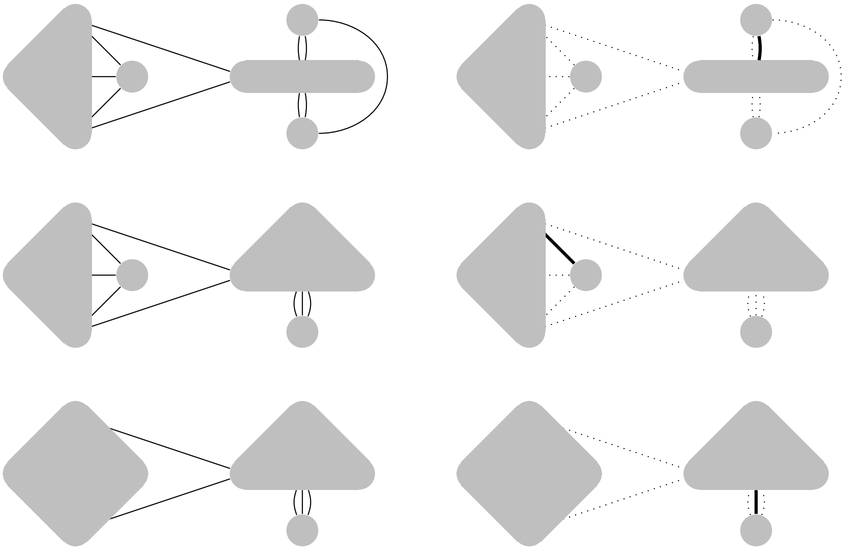
Algorithme de Karger – Exemple 1 (2/3)



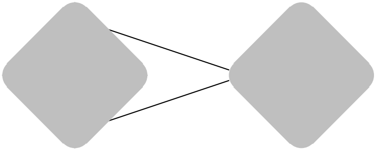
Algorithme de Karger – Exemple 1 (2/3)



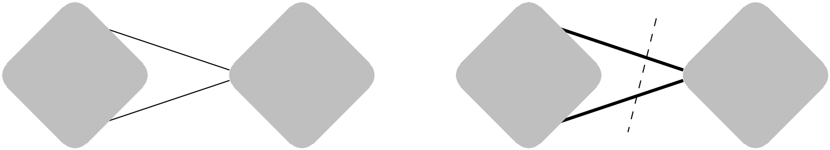
Algorithme de Karger – Exemple 1 (2/3)



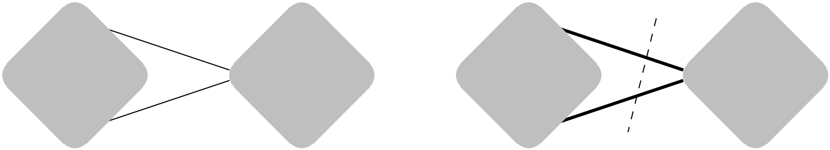
Algorithme de Karger – Exemple 1 (3/3)



Algorithme de Karger – Exemple 1 (3/3)

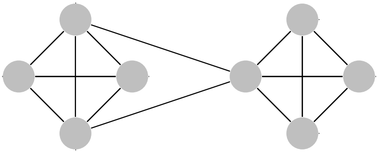


Algorithme de Karger – Exemple 1 (3/3)

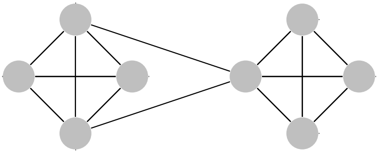


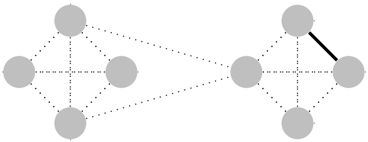
- L'exécution précédente produit une coupe de cardinal 2
- Il s'agit bien d'une coupe minimale !

Algorithme de Karger – Exemple 2 (1/3)

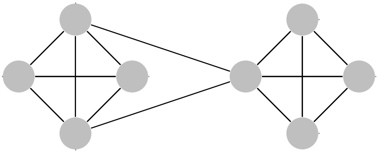


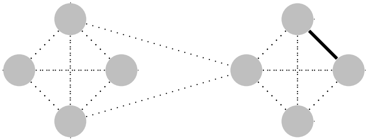
Algorithme de Karger – Exemple 2 (1/3)

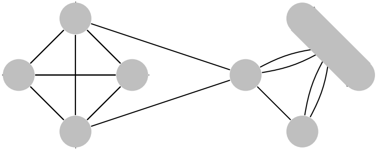




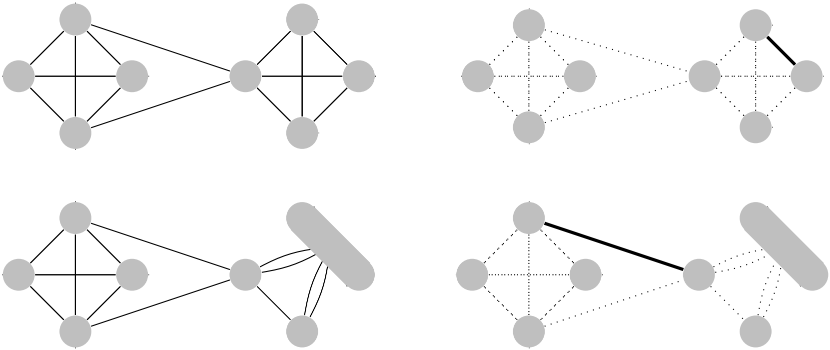
Algorithme de Karger – Exemple 2 (1/3)



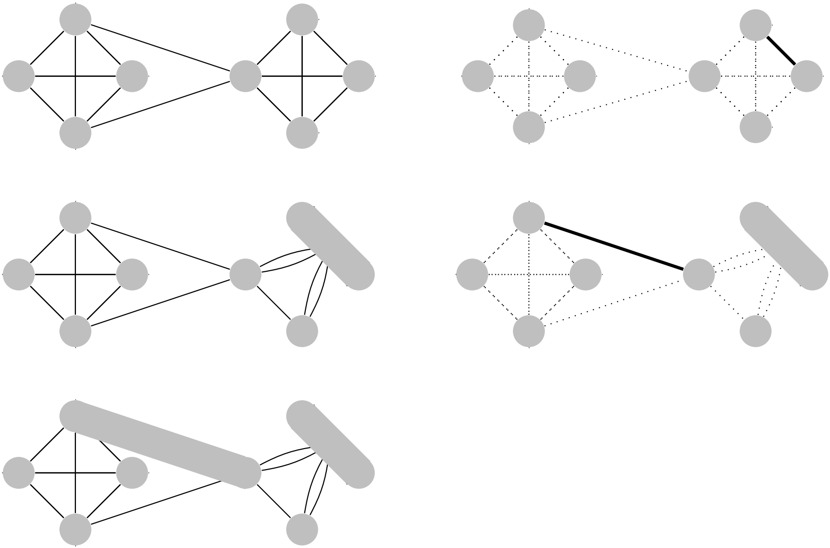




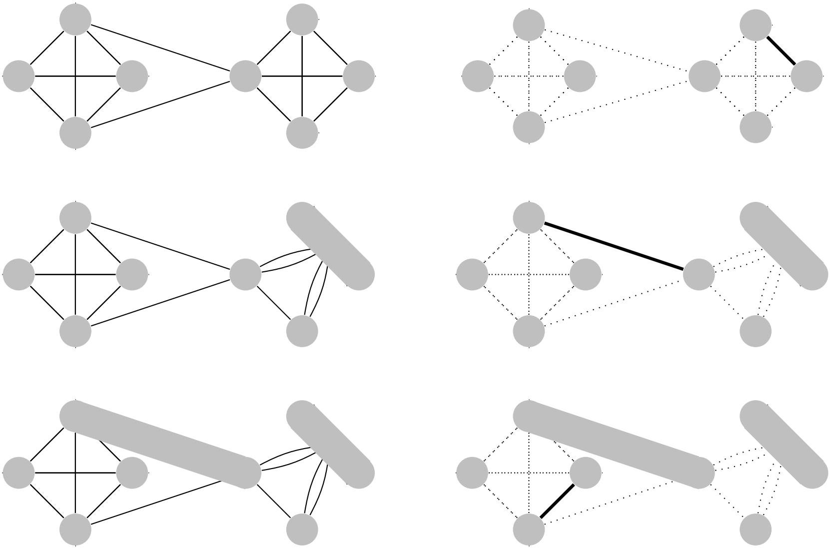
Algorithme de Karger – Exemple 2 (1/3)



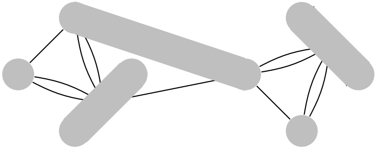
Algorithme de Karger – Exemple 2 (1/3)



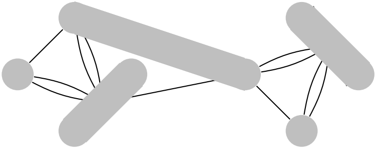
Algorithme de Karger – Exemple 2 (1/3)

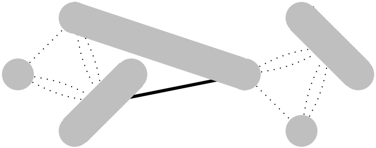


Algorithme de Karger – Exemple 2 (2/3)

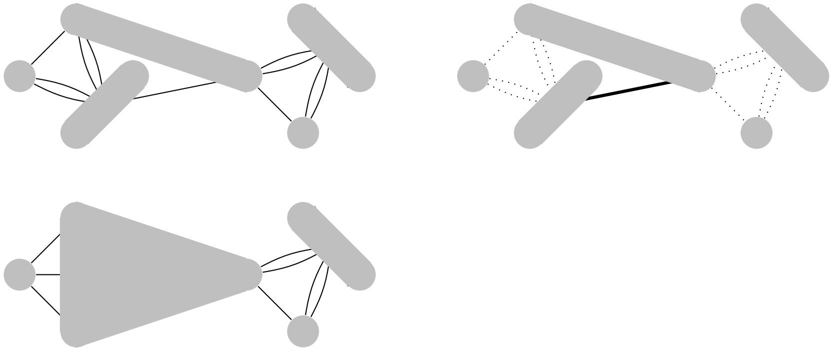


Algorithme de Karger – Exemple 2 (2/3)

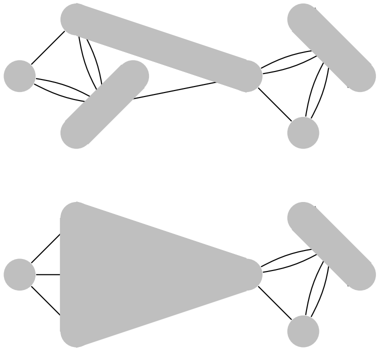




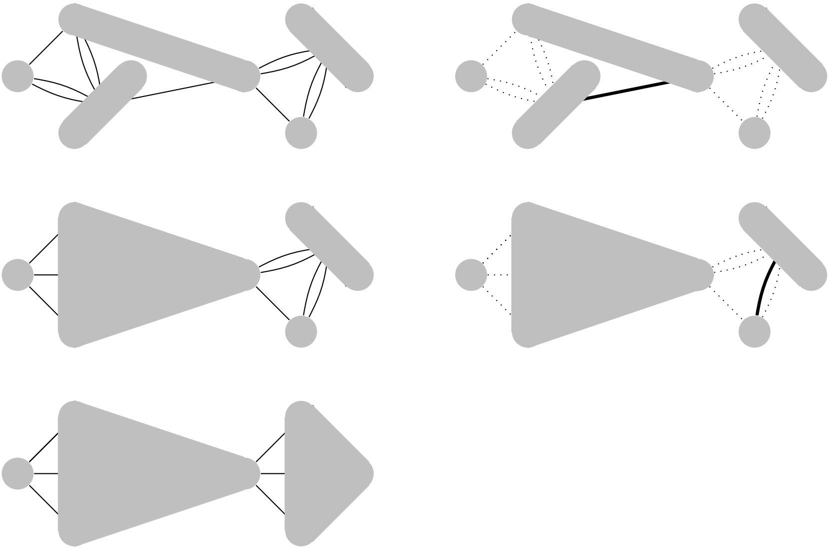
Algorithme de Karger – Exemple 2 (2/3)



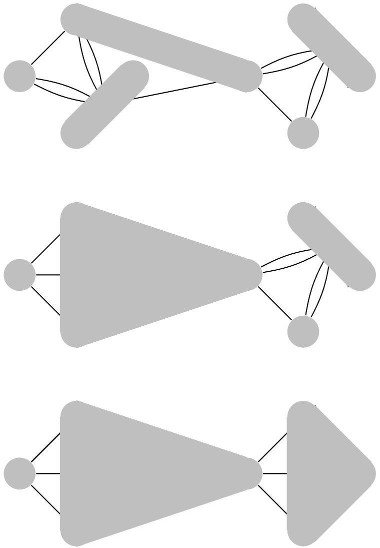
Algorithme de Karger – Exemple 2 (2/3)



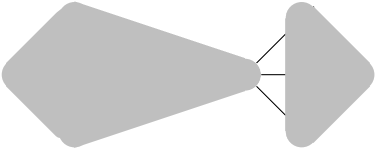
Algorithme de Karger – Exemple 2 (2/3)



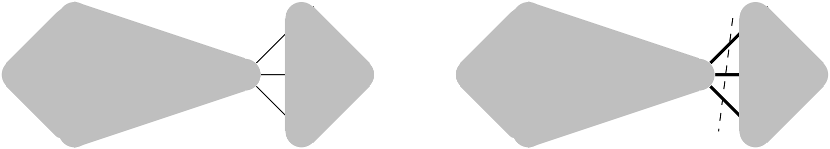
Algorithme de Karger – Exemple 2 (2/3)



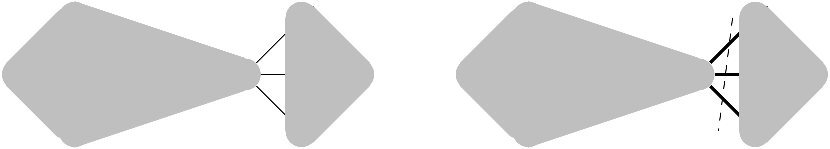
Algorithme de Karger – Exemple 2 (3/3)



Algorithme de Karger – Exemple 2 (3/3)



Algorithme de Karger – Exemple 2 (3/3)



- L'exécution précédente produit une coupe de cardinal 3
- L'algorithme de Karger peut donc se tromper . . .
- Algorithme de type Monte-Carlo !

Algorithme de Karger – Analyse (1/6)

- Soit $(S, V \setminus S)$ une coupe minimale de G
- Si l'algorithme de Karger **ne tire pas d'arête** dans la coupe minimale $(S, V \setminus S)$, il retourne bien $(S, V \setminus S)$
- Puisque la coupe est « petite », on peut espérer que ça arrive « souvent »
- Notons k le cardinal de la coupe $(S, V \setminus S)$
  Notons $\mathcal{E}$ l'événement correspondant à l'algorithme retourne $(S, V \setminus S)$
- Pour que $\mathcal{E}$ se produise, il faut que l'algorithme tire successivement $n-2$ arêtes qui ne sont pas dans la coupe

Algorithme de Karger – Analyse (2/6)

- Notons $\mathcal{E}_i$ correspondant à l'algorithme ne tire pas une arête de la coupe à l'étape k (pour $k \in \{1, \ldots, n-2\}$)

- Nous avons

$$\mathcal{E} = \bigcap_{i=1}^{n-2} \mathcal{E}_i \text{ et } \mathbb{P}(\mathcal{E}) = \mathbb{P}\left(\bigcap_{i=1}^{n-2} \mathcal{E}_i\right)$$

- Les événements $\mathcal{E}_i$ ne sont pas indépendants
- Il faut utiliser les probabilités conditionnelles

$$\mathbb{P}\left(\bigcap_{i=1}^{n-2} \mathcal{E}_i\right) = \mathbb{P}\left(\mathcal{E}_{n-2} \middle| \bigcap_{i=1}^{n-3} \mathcal{E}_i\right) \cdot \mathbb{P}\left(\mathcal{E}_{n-3} \middle| \bigcap_{i=1}^{n-4} \mathcal{E}_i\right) \cdots$$
$$\cdots \mathbb{P}\left(\mathcal{E}_3 | \mathcal{E}_1 \cap \mathcal{E}_2\right) \cdot \mathbb{P}\left(\mathcal{E}_2 | \mathcal{E}_1\right) \cdot \mathbb{P}\left(\mathcal{E}_1\right)$$

Algorithme de Karger – Analyse (3/6)

- $\overline{\mathcal{E}_1}$: la première arête choisie est dans la coupe

$$\mathbb{P}(\overline{\mathcal{E}_1}) = \frac{k}{m}$$

- La coupe minimale du graphe est de cardinal k donc tout sommet a au moins k arêtes adjacentes

  Nous avons

$$m \geq k \cdot n/2$$

- Donc

$$\mathbb{P}(\overline{\mathcal{E}_1}) = \frac{k}{m} \leq \frac{k}{nk/2} = \frac{2}{n}$$

  et

$$\mathbb{P}(\mathcal{E}_1) = 1 - \mathbb{P}(\overline{\mathcal{E}_1}) \geq 1 - \frac{2}{n} = \frac{n-2}{n}$$

Algorithme de Karger – Analyse (3/6)

- $\overline{\mathcal{E}_1}$: la première arête choisie est dans la coupe

$$\mathbb{P}(\overline{\mathcal{E}_1}) = \frac{k}{m}$$

- La coupe minimale du graphe est de cardinal k donc tout sommet a au moins k arêtes adjacentes
  Nous avons

$$m \geq k \cdot n/2$$

- Donc

$$\mathbb{P}(\overline{\mathcal{E}_1}) = \frac{k}{m} \leq \frac{k}{nk/2} = \frac{2}{n}$$

et

$$\mathbb{P}(\mathcal{E}_1) = 1 - \mathbb{P}(\overline{\mathcal{E}_1}) \geq 1 - \frac{2}{n} = \frac{n-2}{n}$$

Algorithme de Karger – Analyse (3/6)

- $\overline{\mathcal{E}_1}$: la première arête choisie est dans la coupe

$$\mathbb{P}(\overline{\mathcal{E}_1}) = \frac{k}{m}$$

- La coupe minimale du graphe est de cardinal k donc tout sommet a au moins k arêtes adjacentes
  Nous avons

$$m \geq k \cdot n/2$$

- Donc

$$\mathbb{P}(\overline{\mathcal{E}_1}) = \frac{k}{m} \leq \frac{k}{nk/2} = \frac{2}{n}$$

  et

$$\mathbb{P}(\mathcal{E}_1) = 1 - \mathbb{P}(\overline{\mathcal{E}_1}) \geq 1 - \frac{2}{n} = \frac{n-2}{n}$$

Algorithme de Karger – Analyse (4/6)

- $\overline{\mathcal{E}_j}$: la j-ième arête choisie est dans la coupe
  $\bigcap_{i=1}^{j-1} \mathcal{E}_i$: les $j-1$ arêtes choisies avant ne l'étaient pas
- Au début de l'étape j, il reste $n-j+1$ sommets
- Si $\bigcap_{i=1}^{j-1} \mathcal{E}_i$, la coupe minimale du multi-graphe restant est de cardinal k et tout sommet a au moins k arêtes
  Le nombre m_j d'arêtes restant au début de l'étape j vérifie
  $$m_j \geq k \cdot (n-j+1)/2$$
- Nous avons
  $$\mathbb{P}\left(\overline{\mathcal{E}_j} \middle| \bigcap_{i=1}^{j-1} \mathcal{E}_i\right) = \frac{k}{m_j} \leq \frac{k}{k(n-j+1)/2} = \frac{2}{n-j+1}$$

Algorithme de Karger – Analyse (4/6)

- $\overline{\mathcal{E}_j}$: la j-ième arête choisie est dans la coupe
  $\bigcap_{i=1}^{j-1} \mathcal{E}_i$: les $j-1$ arêtes choisies avant ne l'étaient pas
- Au début de l'étape j, il reste $n-j+1$ sommets
- Si $\bigcap_{i=1}^{j-1} \mathcal{E}_i$, la coupe minimale du multi-graphe restant est de cardinal k et tout sommet a au moins k arêtes
  Le nombre m_j d'arêtes restant au début de l'étape j vérifie
  $$m_j \geq k \cdot (n-j+1)/2$$
- Nous avons
  $$\mathbb{P}\left(\overline{\mathcal{E}_j} \,\middle|\, \bigcap_{i=1}^{j-1} \mathcal{E}_i\right) = \frac{k}{m_j} \leq \frac{k}{k(n-j+1)/2} = \frac{2}{n-j+1}$$

Algorithme de Karger – Analyse (5/6)

- Nous obtenons donc

$$\mathbb{P}\left(\mathcal{E}_j \middle| \bigcap_{i=1}^{j-1} \mathcal{E}_i\right) = 1 - \mathbb{P}\left(\overline{\mathcal{E}_j} \middle| \bigcap_{i=1}^{j-1} \mathcal{E}_i\right) \geq 1 - \frac{2}{n-j+1} = \frac{n-j-1}{n-j+1}$$

- Finalement

$$\begin{aligned}\mathbb{P}\left(\bigcap_{i=1}^{n-2} \mathcal{E}_i\right) &= \mathbb{P}\left(\mathcal{E}_{n-2} \middle| \bigcap_{i=1}^{n-3} \mathcal{E}_i\right) \cdot \mathbb{P}\left(\mathcal{E}_{n-3} \middle| \bigcap_{i=1}^{n-4} \mathcal{E}_i\right) \cdots \\ &\qquad \cdots \mathbb{P}\left(\mathcal{E}_3 | \mathcal{E}_1 \cap \mathcal{E}_2\right) \cdot \mathbb{P}\left(\mathcal{E}_2 | \mathcal{E}_1\right) \cdot \mathbb{P}\left(\mathcal{E}_1\right) \\ &\geq \frac{n-(n-2)-1}{n-(n-2)+1} \cdot \frac{n-(n-3)-1}{n-(n-3)+1} \cdots \frac{n-2}{n} \\ &= \frac{1}{3} \cdot \frac{2}{4} \cdots \frac{n-2}{n} = \frac{(n-2)!}{n!/2} = \frac{2}{n(n-1)}\end{aligned}$$

Algorithme de Karger – Analyse (6/6)

- La probabilité de succès de l'algorithme est donc $\geq 2/n(n-1)$
- Cette probabilité semble faible mais ...
  - combien y-a-t'il de coupes possibles ?
  - si l'on tire un coupe au hasard quelle est la probabilité qu'elle soit minimale ?

Algorithme de Karger – Analyse (6/6)

- La probabilité de succès de l'algorithme est donc $\geq 2/n(n-1)$
- Cette probabilité semble faible mais . . .
  - combien y-a-t'il de coupes possibles ?
  - si l'on tire un coupe au hasard quelle est la probabilité qu'elle soit minimale ?

Amplifier la probabilité de succès

- Il suffit de répéter l'algorithme de Karger plusieurs fois et de retourner la coupe la plus petite obtenue

- Si on répète T fois, cet algorithme échoue si les T exécutions de l'algorithme de Karger échouent ce qui arrive avec probabilité

$$\leq \left(1 - \frac{2}{n(n-1)}\right)^T$$

Amplifier la probabilité de succès

- Il suffit de répéter l'algorithme de Karger plusieurs fois et de retourner la coupe la plus petite obtenue
- Si on répète T fois, cet algorithme échoue si les T exécutions de l'algorithme de Karger échouent ce qui arrive avec probabilité

$$\leq \left(1 - \frac{2}{n(n-1)}\right)^T$$

Une inégalité à connaître

$$\frac{1}{4} < \left(1 - \frac{1}{x}\right)^x \leq \frac{1}{e}$$

Donc

$$\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)/2} \leq \frac{1}{e}$$

et

$$\left(1 - \frac{2}{n(n-1)}\right)^{\ln(n)\cdot n(n-1)/2} \leq \left(\frac{1}{e}\right)^{\ln(n)} = \frac{1}{n}$$

Amplifier la probabilité de succès

- En répétant l'algorithme de Karger

$$\ln(n) \cdot n(n-1)/2 = O(n^2 \log n) \text{ fois },$$

la probabilité de ne pas retourner une coupe mimimale est $\leq 1/n$

Complexité totale : $O(n^2 \log n \cdot n^2) = O(n^4 \log n)$

Probabilité d'erreur : $\leq 1/n$

Algorithme de Karger-Stein

- La probabilité de choisir une arête de la coupe minimale augmente quand il reste peu d'arêtes
- Ne pas recommencer du début à chaque fois !
- **Cas de base :** (probabilité de succès doublée !)
  - on applique l'algorithme jusqu'à ce qu'il reste $\simeq n/\sqrt{2}$ sommets
  - on applique deux fois l'algorithme à partir de ce point
  - on garde la meilleure solution
- On applique cette idée **récursivement** (« diviser pour régner »)
  $\rightsquigarrow$ 1996 – D. R. Karger et C. Stein

Complexité totale : $O(n^2 \log^3 n)$

Probabilité d'erreur : $\leq 1/n$

Algorithme de Karger-Stein

- La probabilité de choisir une arête de la coupe minimale augmente quand il reste peu d'arêtes
- Ne pas recommencer du début à chaque fois !
- **Cas de base :** (probabilité de succès doublée !)
  - on applique l'algorithme jusqu'à ce qu'il reste $\simeq n/\sqrt{2}$ sommets
  - on applique deux fois l'algorithme à partir de ce point
  - on garde la meilleure solution
- On applique cette idée **récursivement** (« diviser pour régner ») $\rightsquigarrow$ 1996 – D. R. Karger et C. Stein

Complexité totale : $O(n^2 \log^3 n)$

Probabilité d'erreur : $\leq 1/n$

Algorithme de Karger-Stein

- La probabilité de choisir une arête de la coupe minimale augmente quand il reste peu d'arêtes
- Ne pas recommencer du début à chaque fois !
- **Cas de base :** (probabilité de succès doublée !)
  - on applique l'algorithme jusqu'à ce qu'il reste $\simeq n/\sqrt{2}$ sommets
  - on applique deux fois l'algorithme à partir de ce point
  - on garde la meilleure solution
- On applique cette idée **récursivement** (« diviser pour régner »)
  $\rightsquigarrow$ 1996 – D. R. Karger et C. Stein

Complexité totale : $O(n^2 \log^3 n)$

Probabilité d'erreur : $\leq 1/n$

Algorithme de Karger-Stein

- La probabilité de choisir une arête de la coupe minimale augmente quand il reste peu d'arêtes
- Ne pas recommencer du début à chaque fois !
- **Cas de base :** (probabilité de succès doublée !)
  - on applique l'algorithme jusqu'à ce qu'il reste $\simeq n/\sqrt{2}$ sommets
  - on applique deux fois l'algorithme à partir de ce point
  - on garde la meilleure solution
- On applique cette idée **récursivement** (« diviser pour régner ») $\rightsquigarrow$ 1996 – D. R. Karger et C. Stein

Complexité totale : $O(n^2 \log^3 n)$

Probabilité d'erreur : $\leq 1/n$

Algorithme de Karger-Stein

- La probabilité de choisir une arête de la coupe minimale augmente quand il reste peu d'arêtes
- Ne pas recommencer du début à chaque fois !
- **Cas de base :** (probabilité de succès doublée !)
  - on applique l'algorithme jusqu'à ce qu'il reste $\simeq n/\sqrt{2}$ sommets
  - on applique deux fois l'algorithme à partir de ce point
  - on garde la meilleure solution
- On applique cette idée **récursivement** (« diviser pour régner »)
  $\rightsquigarrow$ 1996 – D. R. Karger et C. Stein

Complexité totale : $O(n^2 \log^3 n)$

Probabilité d'erreur : $\leq 1/n$