Géométrie Algorithmique
Données, Modèles, Programmes

2. La puissance de l'aléa : algorithmes randomisés

Jean-Daniel Boissonnat

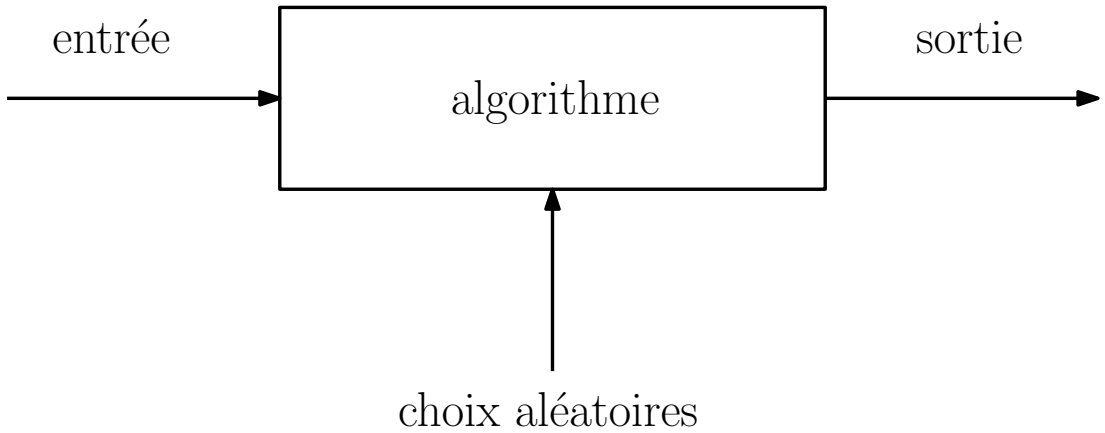
Collège de France
19 avril 2017

Géométrie algorithmique

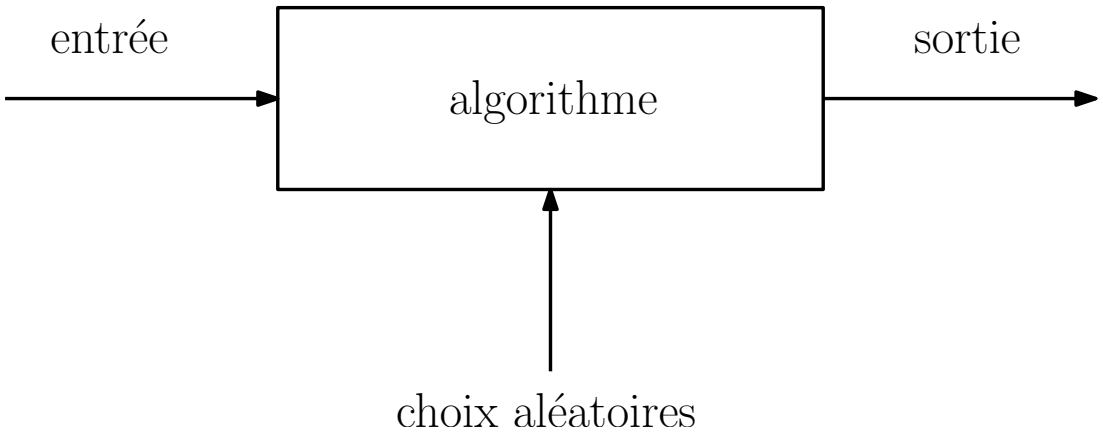
Données, modèles, programmes

1. Modèles géométriques discrets
   F. Cazals : Modèles géométriques pour la prédiction des interactions macro-moléculaires
2. La puissance de l'aléa : algorithmes randomisés
   P. Calka : Probabilités géométriques
3. Le calcul géométrique
   S. Pion : La bibliothèque logicielle CGAL
4. Génération de maillages
   J-M. Mirebeau : Les deux réductions de Voronoï et leur application aux équations aux dérivées partielles
5. Courbes et surfaces
   P. Alliez : Reconstruction de surfaces
6. Espaces de configurations
   A. de Mesmay : Dessin de graphes
7. Structures de données géométriques
   D. Feldman : Core sets
8. Géométrie des données
   F. Chazal : Analyse topologique des données


entrée
algorithme
sortie

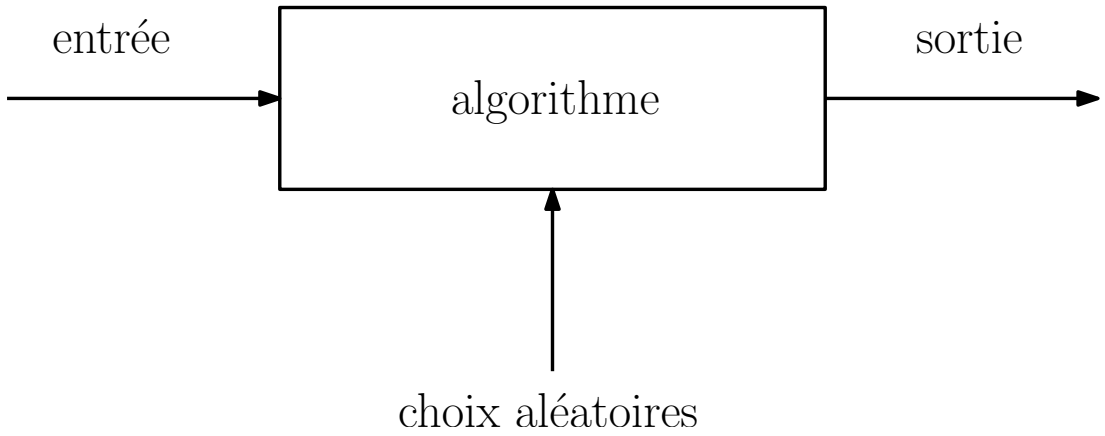


- Modélisation de l'entrée et complexité combinatoire
- Introduction de choix aléatoires pour améliorer la complexité algorithmique



Un algorithme randomisé effectue des choix aléatoires

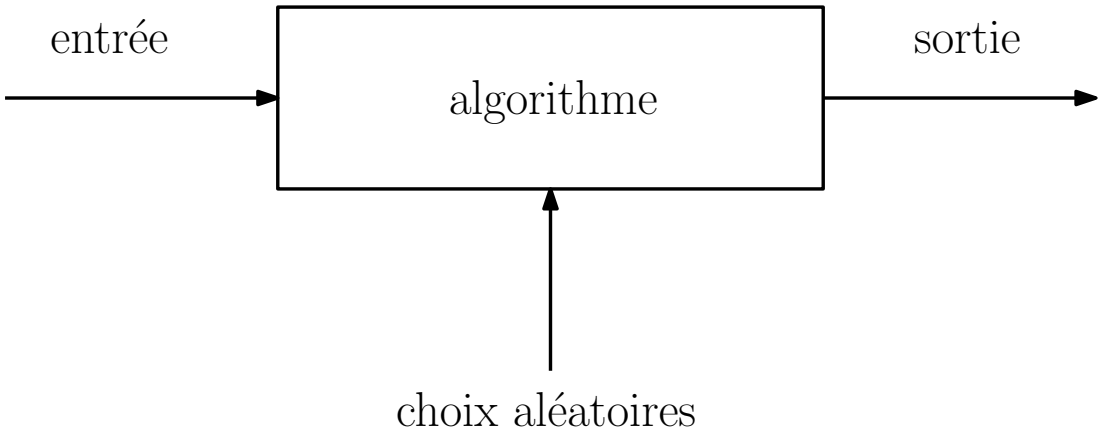
- ordre d'insertion des données d'entrée
- échantillonnage aléatoire
- transformation aléatoire des données (perturbation, projection)



- L'analyse est faite pour l'entrée la pire et en moyenne sur les choix aléatoires
- La sortie peut être toujours exacte (algorithme Las Vegas) ou seulement exacte en probabilité (agorithme Monte Carlo)

Algorithmes randomisés

Motivation

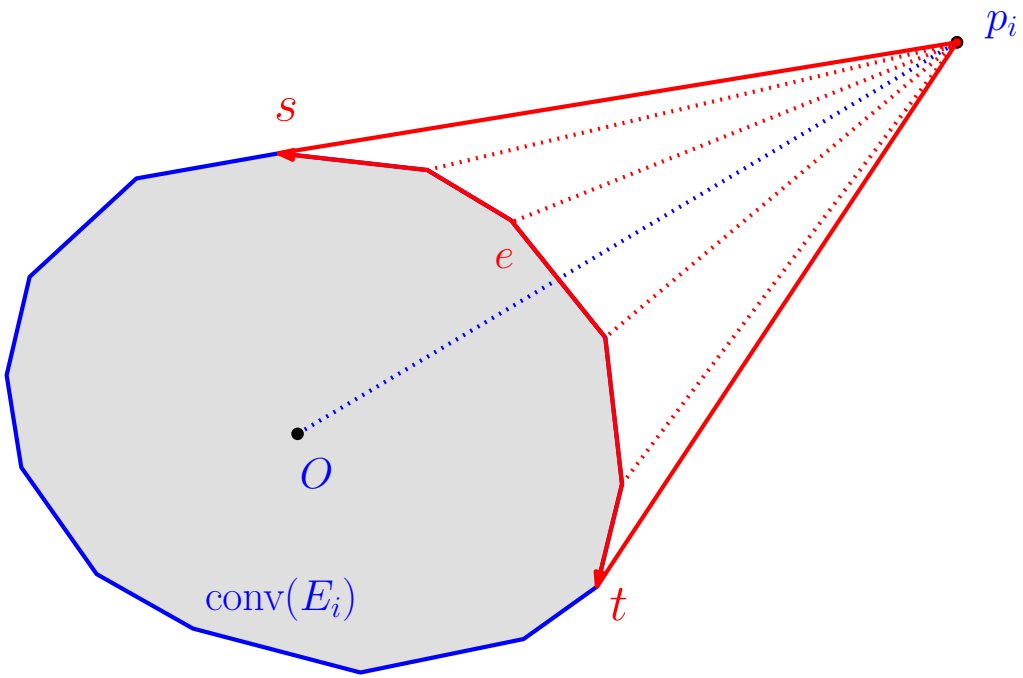


- Simplicité
- Universalité
- Efficacité (théorique et pratique)
- Analyse utile aussi en géométrie combinatoire

1. Algorithmes incrémentaux randomisés
2. Algorithmes en-ligne
3. Analyse combinatoire

Calcul d'une enveloppe convexe

Retour sur l'algorithme incrémental



Analyse dans le cas le pire :

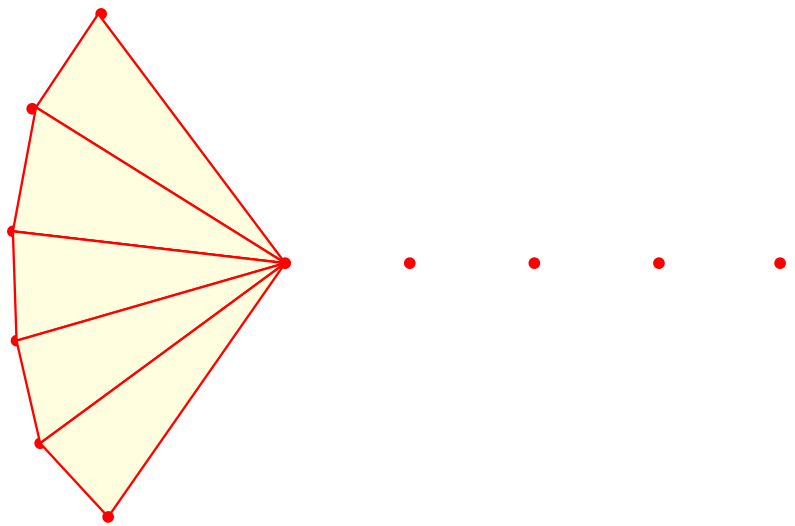
- $d = 3$: $O(n^2)$
- d quelconque : $O(n \log n + n^{\lfloor \frac{d+1}{2} \rfloor})$

(optimal en dimensions paires)

Calcul d'une enveloppe convexe

Retour sur l'algorithme incrémental

- $d = 3$: aucun algorithme incrémental ne peut faire mieux que $O(n^2)$ dans le cas le pire

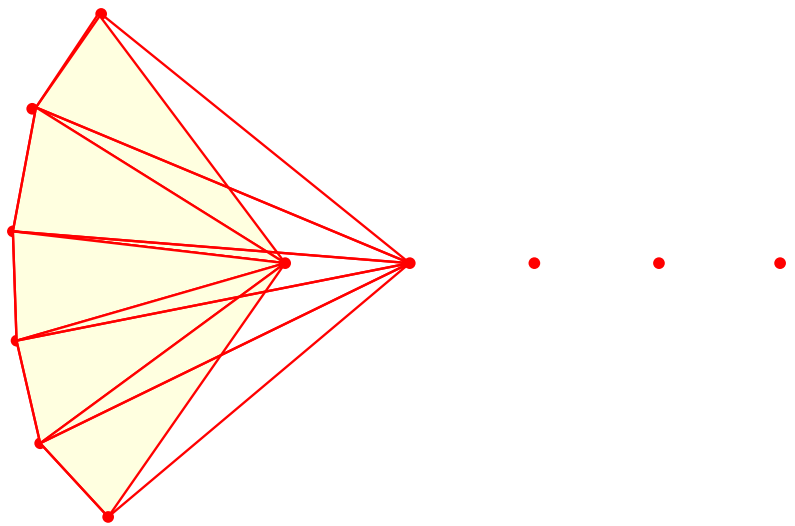


- $d = 3$: la complexité dépend de l'ordre d'insertion

Calcul d'une enveloppe convexe

Retour sur l'algorithme incrémental

- $d = 3$: aucun algorithme incrémental ne peut faire mieux que $O(n^2)$ dans le cas le pire

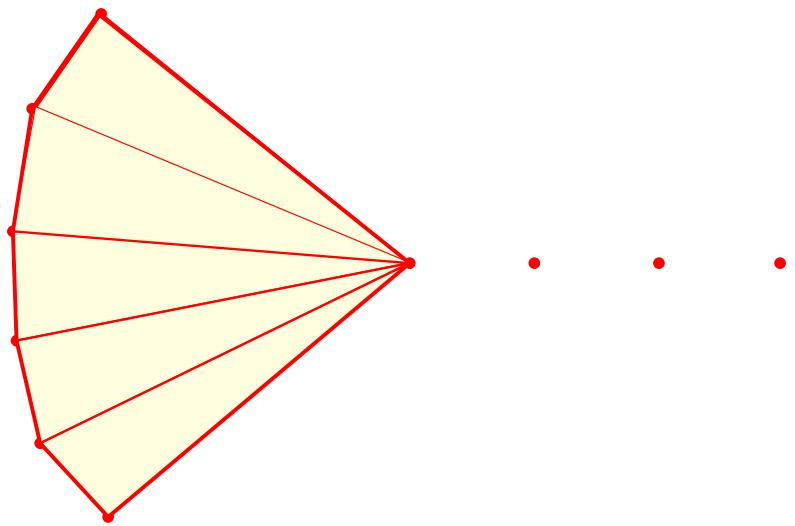


- $d = 3$: la complexité dépend de l'ordre d'insertion

Calcul d'une enveloppe convexe

Retour sur l'algorithme incrémental

- $d = 3$: aucun algorithme incrémental ne peut faire mieux que $O(n^2)$ dans le cas le pire

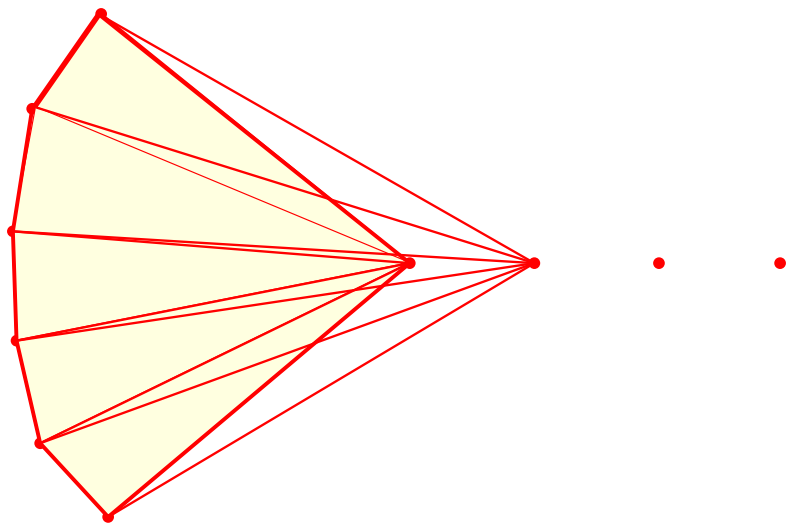


- $d = 3$: la complexité dépend de l'ordre d'insertion

Calcul d'une enveloppe convexe

Retour sur l'algorithme incrémental

- $d = 3$: aucun algorithme incrémental ne peut faire mieux que $O(n^2)$ dans le cas le pire

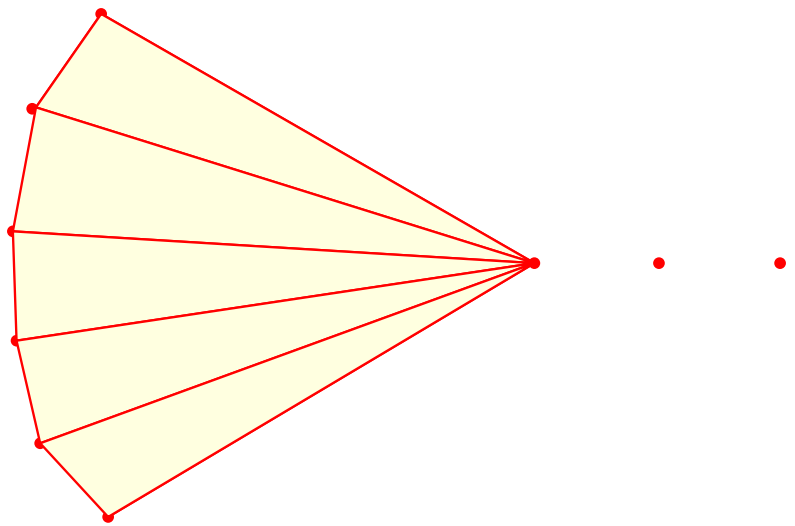


- $d = 3$: la complexité dépend de l'ordre d'insertion

Calcul d'une enveloppe convexe

Retour sur l'algorithme incrémental

- $d = 3$: aucun algorithme incrémental ne peut faire mieux que $O(n^2)$ dans le cas le pire

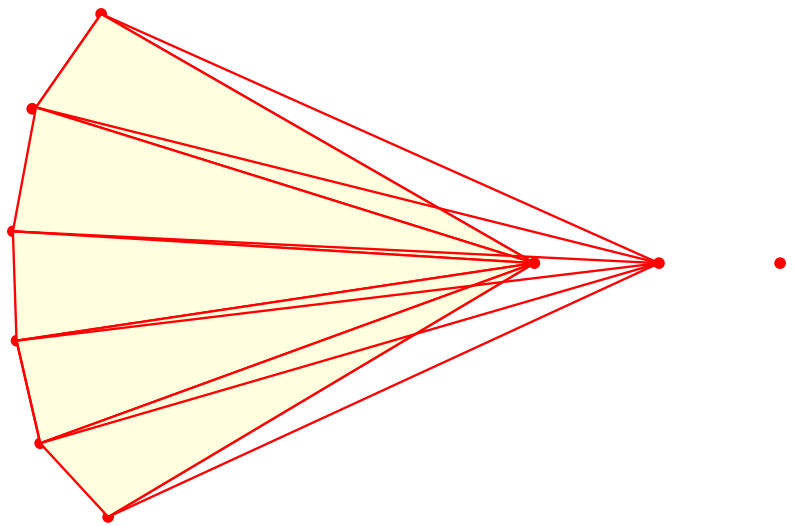


- $d = 3$: la complexité dépend de l'ordre d'insertion

Calcul d'une enveloppe convexe

Retour sur l'algorithme incrémental

- $d = 3$: aucun algorithme incrémental ne peut faire mieux que $O(n^2)$ dans le cas le pire

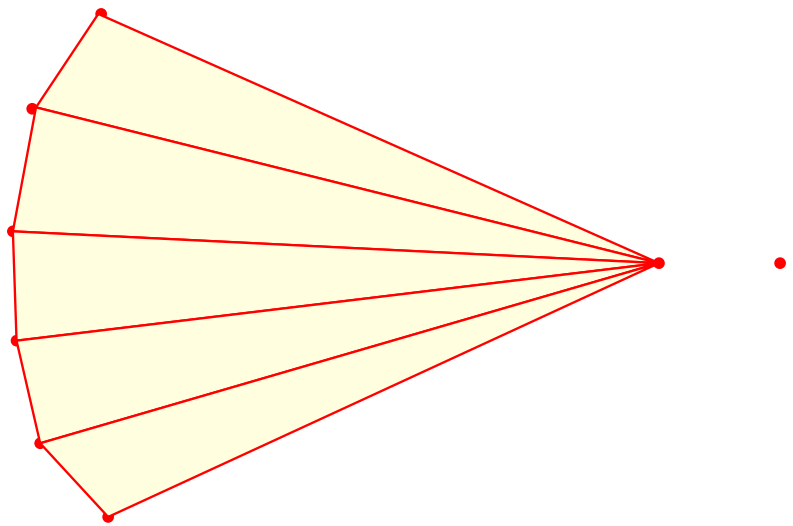


- $d = 3$: la complexité dépend de l'ordre d'insertion

Calcul d'une enveloppe convexe

Retour sur l'algorithme incrémental

- $d = 3$: aucun algorithme incrémental ne peut faire mieux que $O(n^2)$ dans le cas le pire

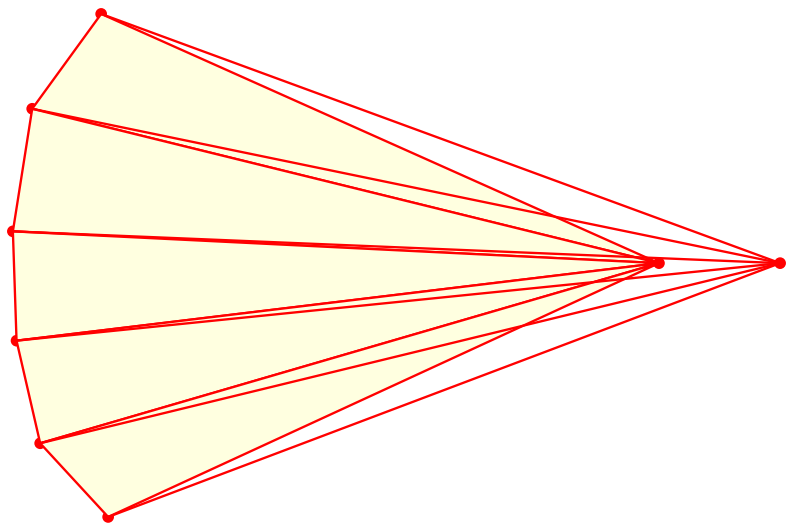


- $d = 3$: la complexité dépend de l'ordre d'insertion

Calcul d'une enveloppe convexe

Retour sur l'algorithme incrémental

- $d = 3$: aucun algorithme incrémental ne peut faire mieux que $O(n^2)$ dans le cas le pire

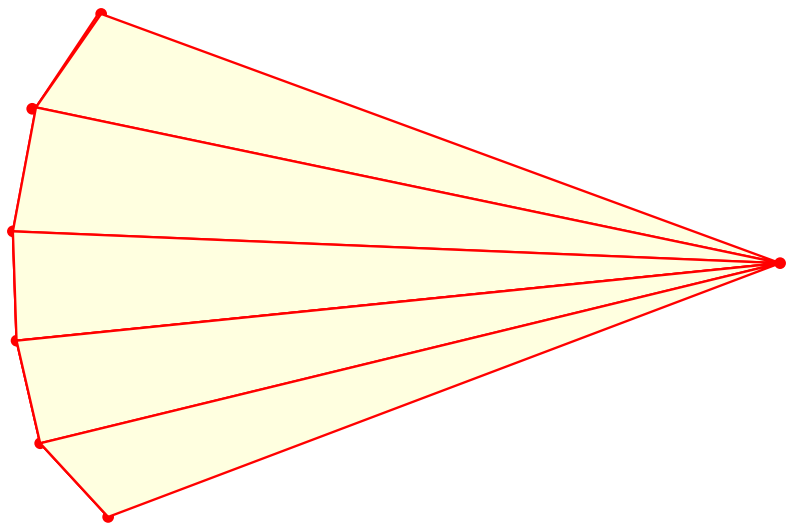


- $d = 3$: la complexité dépend de l'ordre d'insertion

Calcul d'une enveloppe convexe

Retour sur l'algorithme incrémental

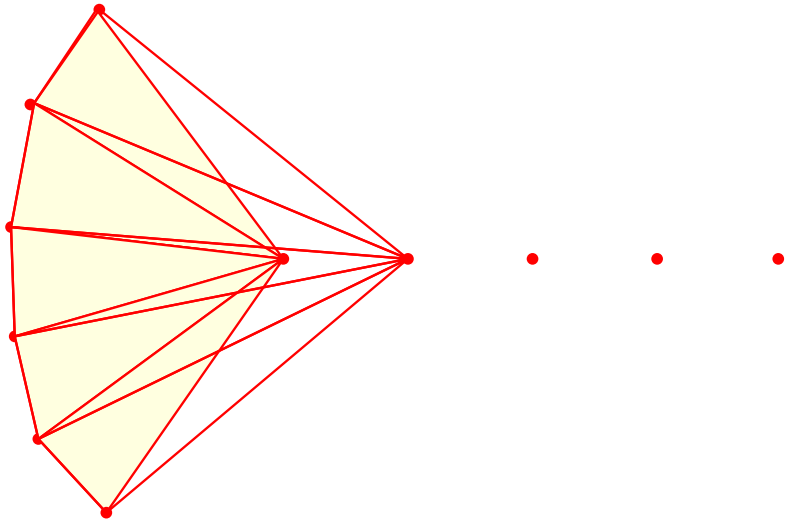
- $d = 3$: aucun algorithme incrémental ne peut faire mieux que $O(n^2)$ dans le cas le pire

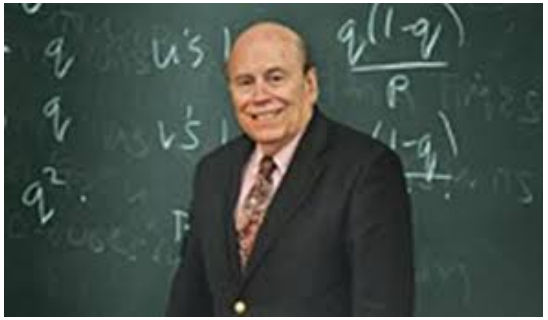


- $d = 3$: la complexité dépend de l'ordre d'insertion

La révolution des algorithmes randomisés

Calcul incrémental d'une enveloppe convexe de n





M. Rabin, K. Clarkson and P. Shor

- l'algorithme randomisé a une complexité moyenne optimale

$$O(n \log n + n^{\lfloor \frac{d}{2} \rfloor})$$

- triangulation de Delaunay 3d : 1 million de points en 8.5s

Analyse randomisée 1

Mises à jour de l'enveloppe convexe et place mémoire

Hyp. : les points de $\mathcal{P}$ sont insérés dans un ordre aléatoire

$\mathcal{P}_i = \{p_1, ..., p_i\}$ est un échantillon aléatoire de taille i

$N(i)$: espérance du nombre de facettes créées à l'étape i

$$
\begin{aligned}
N(i) &= \sum_{f \subset \mathcal{P}, |f|=d} \text{proba}(f \in \text{conv}(\mathcal{P}_i)) \times \frac{d}{i} \\
&= \frac{d}{i} \, \mathsf{E}(|\text{conv}(i, \mathcal{P})|) \quad \text{(espérance sur tous les échantillons aléatoires de } \mathcal{P} \text{ de taille } i) \\
&= O\left(i^{\lfloor \frac{d}{2} \rfloor - 1}\right) \quad \text{(th. de la borne sup.)}
\end{aligned}
$$

Espérance du nombre total de facettes créées

$$N = \sum_{i=1}^{n} N(i) = O\left(n^{\lfloor \frac{d}{2} \rfloor}\right) \quad \text{and} \quad N = O(n) \quad \text{if } d = 2, 3$$

Analyse randomisée 1

Mises à jour de l'enveloppe convexe et place mémoire

Hyp. : les points de $\mathcal{P}$ sont insérés dans un ordre aléatoire

$\mathcal{P}_i = \{p_1, ..., p_i\}$ est un échantillon aléatoire de taille i

$N(i)$: espérance du nombre de facettes créées à l'étape i

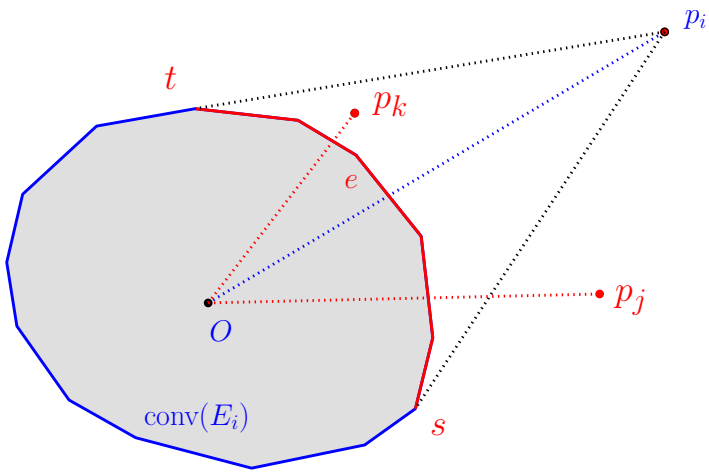
$$
\begin{aligned}
N(i) &= \sum_{f \subset \mathcal{P}, |f|=d} \text{proba}(f \in \text{conv}(\mathcal{P}_i)) \times \frac{d}{i} \\
&= \frac{d}{i} \, \mathsf{E}(|\text{conv}(i, \mathcal{P})|) \quad \text{(espérance sur tous les échantillons aléatoires de } \mathcal{P} \text{ de taille } i) \\
&= O\left(i^{\lfloor \frac{d}{2} \rfloor - 1}\right) \quad \text{(th. de la borne sup.)}
\end{aligned}
$$

Espérance du nombre total de facettes créées

$$
N = \sum_{i=1}^{n} N(i) = O\left(n^{\lfloor \frac{d}{2} \rfloor}\right) \quad \text{and} \quad N = O(n) \quad \text{if } d = 2, 3
$$

L'algorithme incrémental (un peu) révisé

- Localisation rapide
- Recherche des facettes rouges
- Création des nouvelles facettes



Conflit : $p \in \mathcal{P} \setminus \mathcal{P}_{i-1},\ f \in \mathrm{conv}(\mathcal{P}_{i-1}),\ p \dagger f \Leftrightarrow [Op] \cap \mathrm{aff}(f) \neq \emptyset$

Graphe de conflit : $\mathrm{CG}_{i-1} = \{(p,f),\ p \in \mathcal{P} \setminus \mathcal{P}_{i-1} \times f \in \mathrm{conv}(\mathcal{P}_{i-1})\ p \dagger f\}$

Analyse randomisée 2

Mise à jour du graphe de conflit

$\mathcal{P}_i^+ = \mathcal{P}_i \cup \{p_j\}$: un sous-ensemble aléatoire de $i+1$ points de $\mathcal{P}$

$\text{conv}_1(\mathcal{P}_i^+) = \{f \in \mathcal{P}_i^+,\ \ |f| = d,\ \ \exists \text{ unique } q \in \mathcal{P}_i^+ :\ \ f \dagger q\}$

$N(i,j) =$ espérance du nombre de faces de $\text{conv}_1(\mathcal{P}_i^+)$ créées à l'étape i et en conflit avec p_j, $j > i$

$$
\begin{aligned}
N(i,j) &= \sum_{f \in \mathcal{P}, |f|=d} \text{proba}(f \in \text{conv}_1(\mathcal{P}_i^+)) \times \frac{d}{i} \times \frac{1}{i+1} \\
&= \frac{d}{i\,(i+1)}\ \mathsf{E}(|\text{conv}_1(i+1, \mathcal{P})|)
\end{aligned}
$$

espérance sur tous les échantillons aléatoires de $\mathcal{P}$ de taille $i+1$

Borne sur $\mathsf{E}(|\mathrm{conv}_1(r,\mathcal{P})|)$

Analyse arrière

$$R' = R \setminus \{p\}$$

$$f \in \mathrm{conv}(R') \quad \begin{array}{ll} \text{si} & f \in \mathrm{conv}_1(R) \text{ et } p \dagger f \qquad (\text{proba} = \frac{1}{r}) \\ \text{ou} & f \in \mathrm{conv}(R) \text{ et } R' \ni \text{les } d \text{ sommets de } f \qquad (\text{proba} = \frac{r-d}{r}) \end{array}$$

En notant C_0 pour conv et prenant l'espérance

$$\begin{aligned} \mathsf{E}(|C_0(r-1,R)|) &= \frac{1}{r}\,|C_1(R)| + \frac{r-d}{r}\,|C_0(R)| \\ \mathsf{E}(|C_0(r-1,\mathcal{P})|) &= \frac{1}{r}\,\mathsf{E}(|C_1(r,\mathcal{P})) + \frac{r-d}{r}\,\mathsf{E}(|C_0(r,\mathcal{P})|) \\ \mathsf{E}(|C_1(r,\mathcal{P})|) &= d\,\mathsf{E}(|C_0(r,\mathcal{P})|) - r\,(\mathsf{E}(|C_0(r,\mathcal{P})|) - \mathsf{E}(|C_0(r-1,\mathcal{P})|) \\ &\leq d\,\mathsf{E}(|C_0(r,\mathcal{P})|) \end{aligned}$$

Fin de l'analyse randomisée

Mise à jour du graphe de conflit

$N(i,j)$ = espérance du nombre de faces de $\mathrm{conv}_1(\mathcal{P}_i^+)$ créées à l'étape i et en conflit avec p_j, $j > i$

$$\begin{aligned} N(i,j) &= \frac{d}{i\,(i+1)}\,\mathsf{E}(|C_1(i+1,\mathcal{P})| \\ &\leq \frac{d^2}{i\,(i+1)}\,\mathsf{E}(|C_0(i+1,\mathcal{P})|) = O(i^{\lfloor \frac{d}{2} \rfloor - 2}) \end{aligned}$$

Espérance du coût total de mise à jour du graphe de conflit :

$$\begin{aligned} \sum_{i=1}^{n} \sum_{j=i+1}^{n} N(i,j) &= \sum_{i=1}^{n} (n-i)\, O(i^{\lfloor \frac{d}{2} \rfloor - 2}) \\ &= O(n \log n + n^{\lfloor \frac{d}{2} \rfloor}) \end{aligned}$$

Résultat principal et corollaires

- L'enveloppe convexe de n points de $\mathbb{R}^d$ peut être calculée en temps optimal $O(n \log n + n^{\lfloor \frac{d}{2} \rfloor})$ en utilisant un espace mémoire $O(n^{\lfloor \frac{d}{2} \rfloor})$
- Les mêmes bornes s'appliquent au calcul de l'intersection de n demi-espaces de $\mathbb{R}^d$
- et au calcul des diagrammes Voronoï et triangulations de Delaunay de n points de $\mathbb{R}^d$ (remplacer d par $d+1$ dans les bornes)
- L'algorithme est simple et efficace en pratique
- Il peut être dérandomisé (au prix de la simplicité)

[Chazelle 1992]

Constructions incrémentales randomisées

Le cadre général de Clarkson & Shor

$\mathcal{O}$: un ensemble de n objets de complexité constante

Configuration f : définie par $O(1)$ objets

$\mathcal{F}(\mathcal{O}) = \{$ configurations définies sur $\mathcal{O}\}$

Conflit dans $\mathcal{O}$: $o \in \mathcal{O},\ f \in F(\mathcal{O})$: $p \dagger f$

Problème : calculer $\mathcal{F}_0(\mathcal{O}) = \{$ configurations de $\mathcal{F}(\mathcal{O})$ sans conflit dans $\mathcal{O}\}$

Algorithme statique

$\mathcal{O}_i$: ensemble des i premiers objets

GC_i : graphe de conflit qui représente les conflits
entre les objets de $\mathcal{O} \setminus \mathcal{O}_i$ et les configurations de $\mathcal{F}_0(\mathcal{O}_i)$

Algorithme

Pour $i = 1, \ldots, n$, insérer l'objet O_i

1. Accéder aux config. de $\mathcal{F}_0(\mathcal{O}_{i-1})$ en conflit avec O_i en utilisant GC_{i-1}
2. Construire les config. de $\mathcal{F}_0(\mathcal{O}_i)$
   1. Supprimer les config. en conflit
   2. Construire les nouvelles config. définies avec O_i
3. Construire GC_i : établir des liens entre les nouvelles config. et les objets restant à insérer

Conditions d'actualisation

1. on peut décider si O_i et C sont en conflit en temps $O(1)$

2. si O_i est en conflit avec j config., les nouvelles config. de $\mathcal{F}_0(\mathcal{O}_i)$ peuvent être calculées en temps $O(j)$

3. si j' est le nb de conflits entre les nouvelles config. et $\mathcal{O}_i$, les nouveaux conflits peuvent être calculés en temps $O(j + j')$

Théorème

Si les objets sont insérés dans un ordre aléatoire, le coût de l'algorithme est

Place mémoire : $P = O\left(\sum_{i=1}^{n} \frac{1}{i} E(|\mathcal{F}_0(\mathcal{O}_i)|)\right)$

Mise à jour du GC : $T = O\left(\sum_{i=1}^{n} \frac{n-i}{i^2} E(|\mathcal{F}_0(\mathcal{O}_i)|)\right)$

$\longrightarrow$ si $E(|\mathcal{F}_0(\mathcal{O}_i)|) = O(i)$: $P = O(n)$ $T = O(\log n)$

$\longrightarrow$ si $E(|\mathcal{F}_0(\mathcal{O}_i)|) = O(i^\alpha)$ $P = O(n^\alpha)$ $T = O(n^\alpha)$
$\alpha > 1$

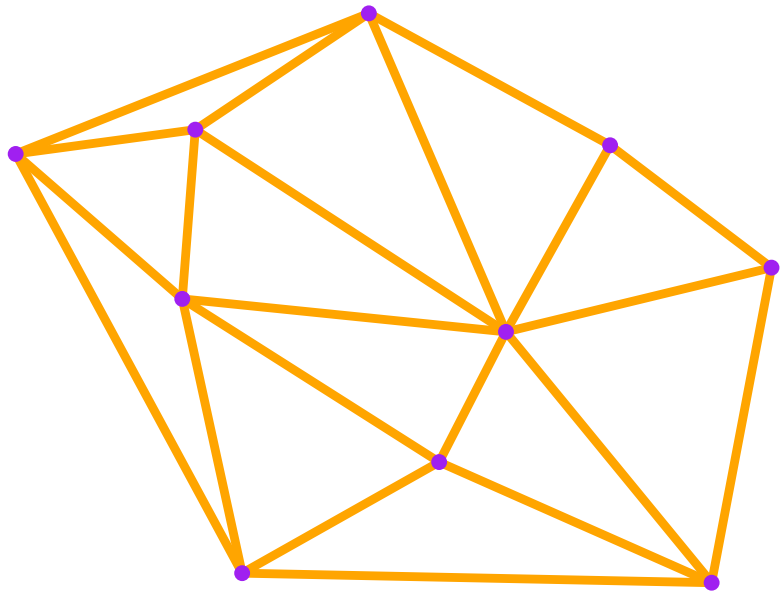
1 Algorithmes incrémentaux randomisés

2 Algorithmes en-ligne

3 Analyse combinatoire

Algorithme en-ligne : arbre de Delaunay

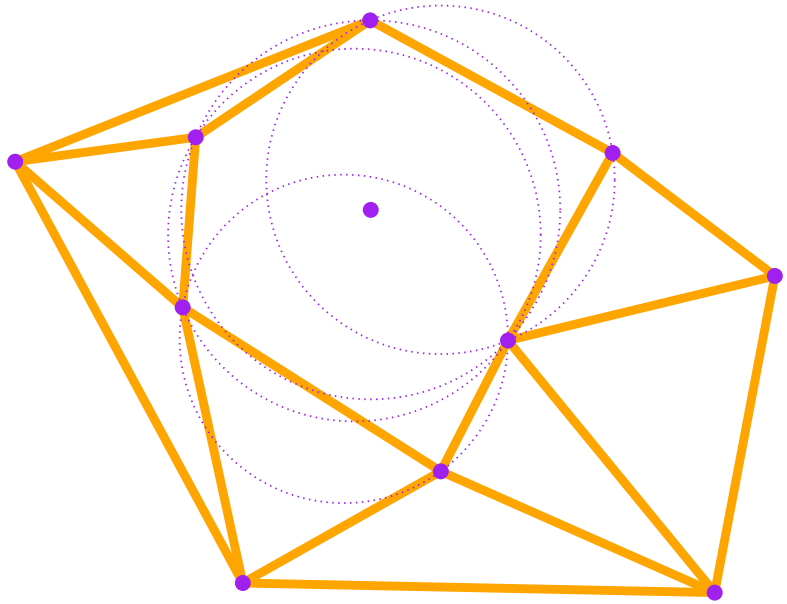
Utiliser l'historique de la construction comme une structure de localisation



- $B(\text{fils}) \subset B(\text{père}) \bigcup B(\text{beau-père})$
- Les simplexes à enlever sont des feuilles de l'arbre de Delaunay

Algorithme en-ligne : arbre de Delaunay

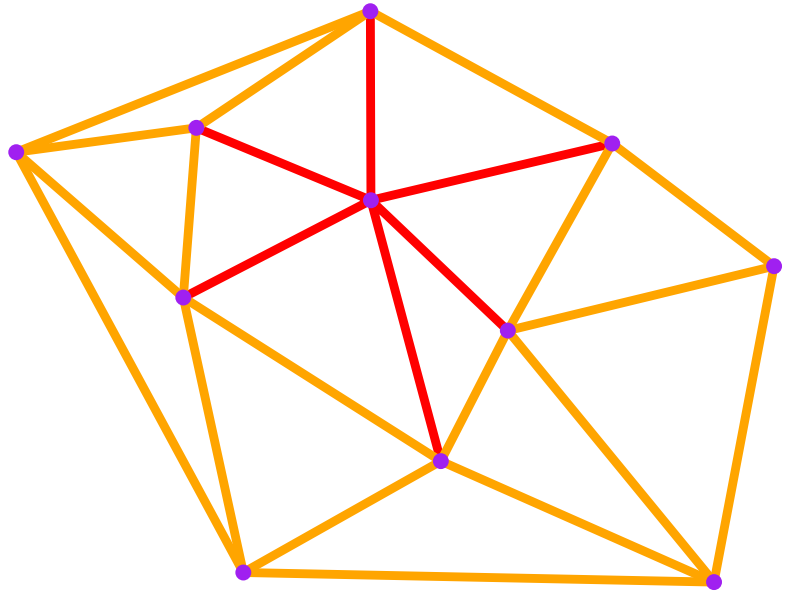
Utiliser l'historique de la construction comme une structure de localisation



- $B(\text{fils}) \subset B(\text{père}) \bigcup B(\text{beau-père})$
- Les simplexes à enlever sont des feuilles de l'arbre de Delaunay

Algorithme en-ligne : arbre de Delaunay

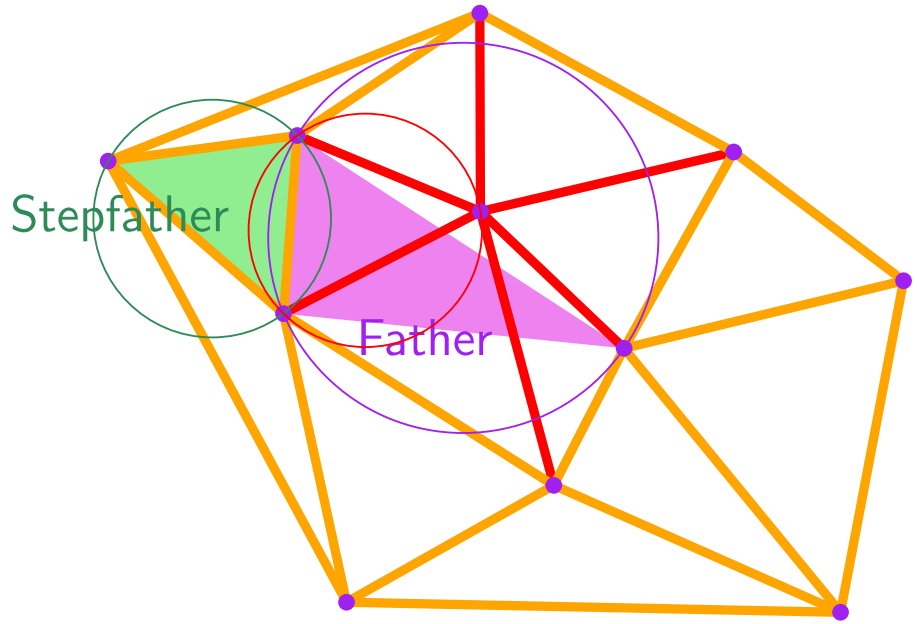
Utiliser l'historique de la construction comme une structure de localisation



- $B(\text{fils}) \subset B(\text{père}) \bigcup B(\text{beau-père})$
- Les simplexes à enlever sont des feuilles de l'arbre de Delaunay

Algorithme en-ligne : arbre de Delaunay

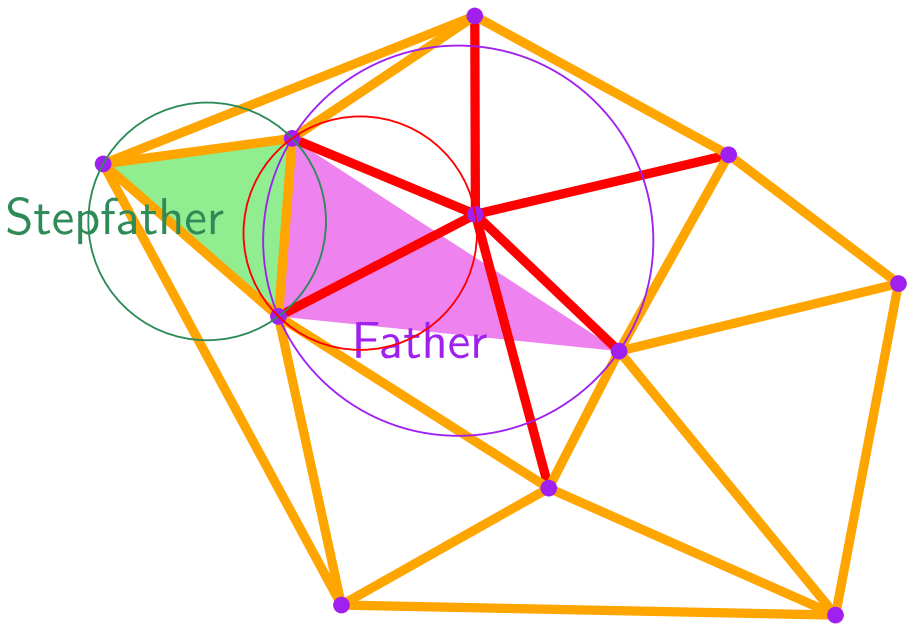
Utiliser l'historique de la construction comme une structure de localisation

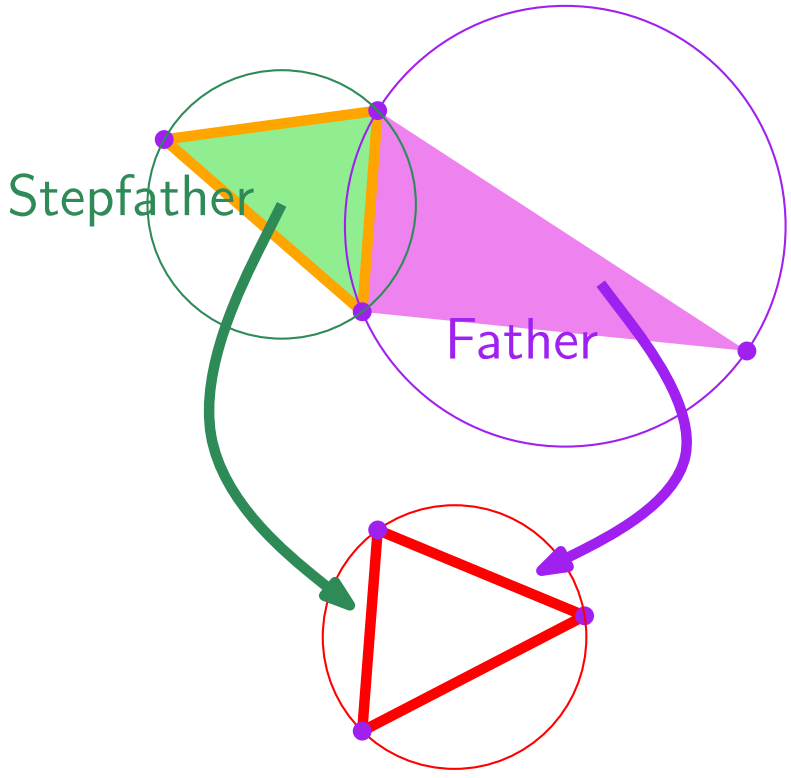


- $B(\text{fils}) \subset B(\text{père}) \bigcup B(\text{beau-père})$
- Les simplexes à enlever sont des feuilles de l'arbre de Delaunay

Algorithme en-ligne : arbre de Delaunay

Utiliser l'historique de la construction comme une structure de localisation

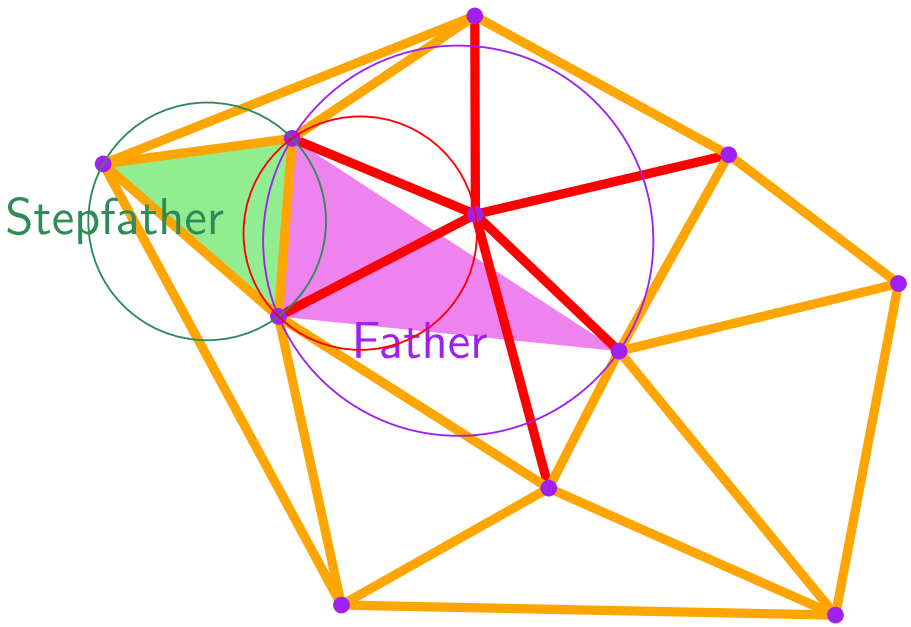


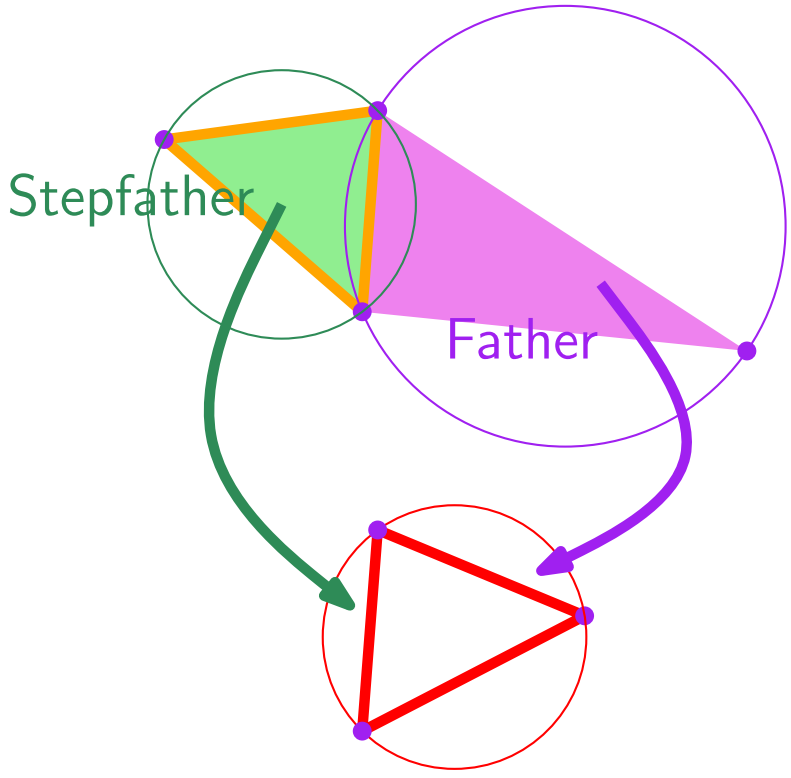


- $B(\text{fils}) \subset B(\text{père}) \bigcup B(\text{beau-père})$
- Les simplexes à enlever sont des feuilles de l'arbre de Delaunay

Algorithme en-ligne : arbre de Delaunay

Utiliser l'historique de la construction comme une structure de localisation

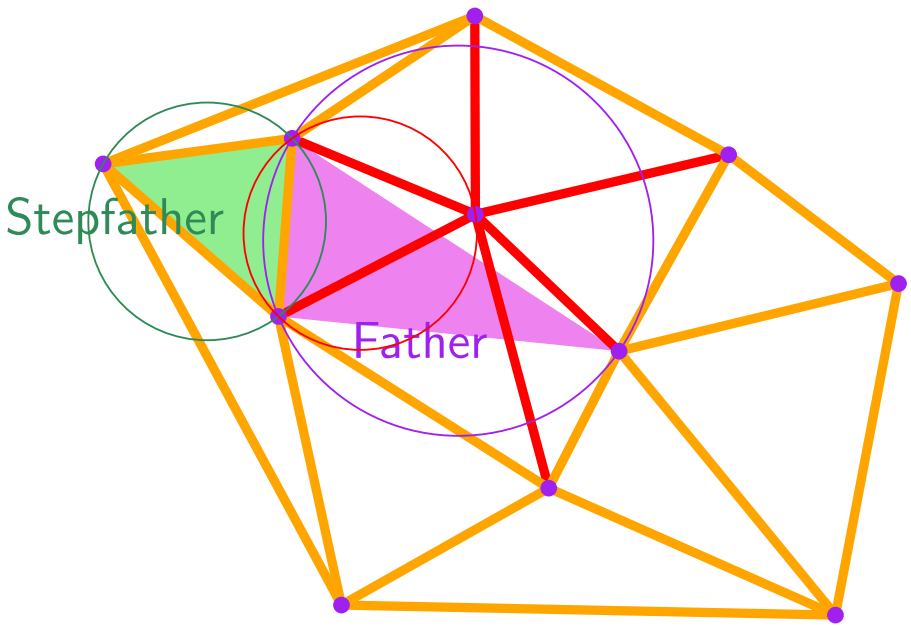


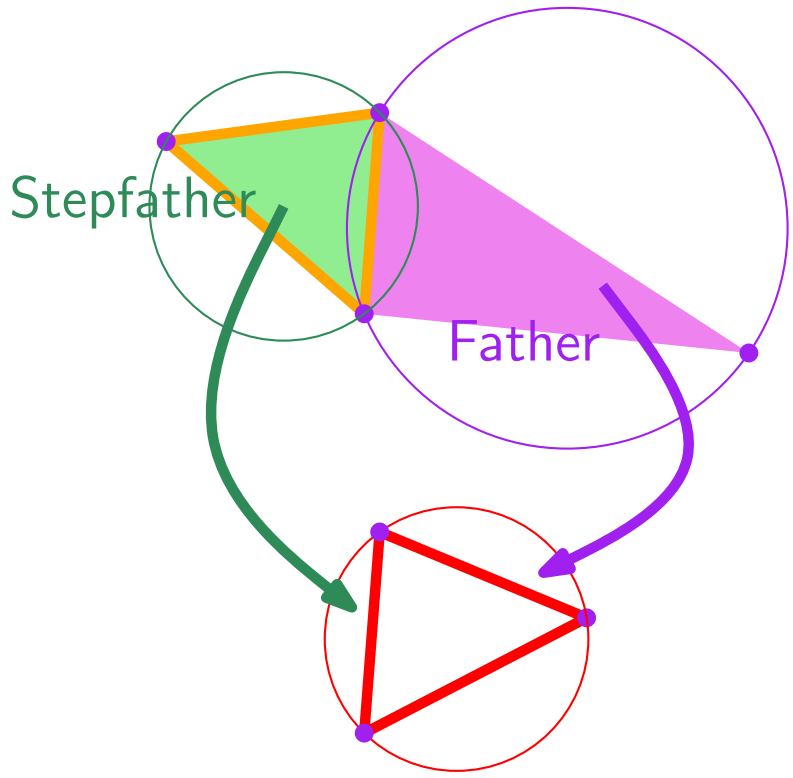


- $B(\text{fils}) \subset B(\text{père}) \bigcup B(\text{beau-père})$
- Les simplexes à enlever sont des feuilles de l'arbre de Delaunay

Algorithme en-ligne : arbre de Delaunay

Utiliser l'historique de la construction comme une structure de localisation





- $B(\text{fils}) \subset B(\text{père}) \bigcup B(\text{beau-père})$
- Les simplexes à enlever sont des feuilles de l'arbre de Delaunay

Algorithmes en ligne

GH_i : GDA d'histoire qui satisfait

P1 : les configurations de $\mathcal{F}_0(\mathcal{O}_i)$ sont des feuilles de GH_i

P2 : si un objet O est en conflit avec la config. attachée au nœud ν, il est en conflit avec un des parents de ν

Algorithme

Pour $i = 1, \ldots, n$, insérer l'objet O_i

1. Chercher les config. de $\mathcal{F}_0(\mathcal{O}_{i-1})$ en conflit avec O_i en parcourant GH_{i-1}
2. Construire GH_i de telle façon que P1 et P2 restent vraies
    1. Créer les nouvelles feuilles de GH_i (nouvelles config.)
    2. Etablir les liens entre les feuilles de GH_{i-1} en conflit et les nouvelles feuilles de GH_i

Conditions d'actualisation

- On peut décider si O_i et une config. sont en conflit en temps $O(1)$
- Si O_i est en conflit avec j config., les nouvelles config. sans conflit sont calculables en temps $O(j)$
- Le nombre de fils d'un nœud $= O(1)$ (cette condition peut être supprimée)

Analyse de l'algorithme en-ligne

$\mathcal{O}_i$: échantillon aléatoire de i objets de $\mathcal{O}$

Place mémoire : $P = O\left(\sum_{i=1}^{n} \frac{1}{i} E(|\mathcal{F}_0(\mathcal{O}_i)|)\right)$

Temps d'insertion : $T = O\left(\sum_{i=1}^{n} \frac{1}{i^2} E(|\mathcal{F}_0(\mathcal{O}_i)|)\right)$

$\longrightarrow$ si $E(|\mathcal{F}_0(\mathcal{O}_i)|) = O(i)$: $P = O(n)$ $T = O(\log n)$

$\longrightarrow$ si $E(|\mathcal{F}_0(\mathcal{O}_i)|) = O(i^\alpha)$ $P = O(n^\alpha)$ $T = O(n^{\alpha-1})$
$\alpha > 1$

Analyse de l'algorithme en-ligne

$\mathcal{O}_i$: échantillon aléatoire de i objets de $\mathcal{O}$

Place mémoire : $P = O\left(\sum_{i=1}^{n} \frac{1}{i} E(|\mathcal{F}_0(\mathcal{O}_i)|)\right)$

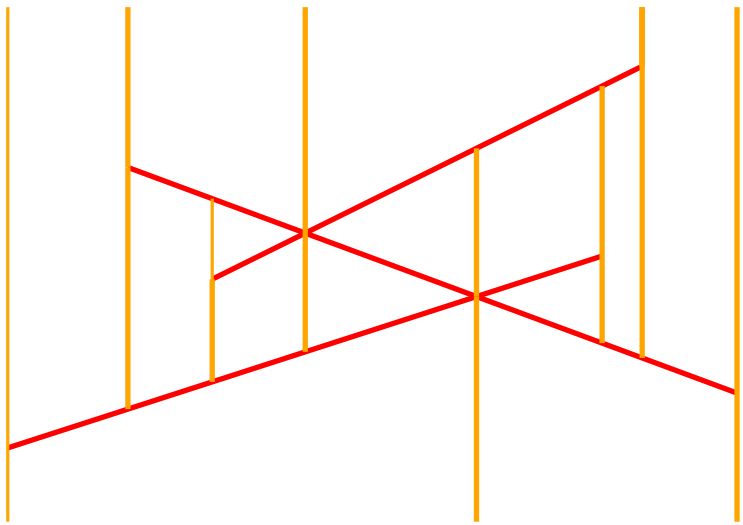
Temps d'insertion : $T = O\left(\sum_{i=1}^{n} \frac{1}{i^2} E(|\mathcal{F}_0(\mathcal{O}_i)|)\right)$

$\longrightarrow$ si $E(|\mathcal{F}_0(\mathcal{O}_i)|) = O(i)$: $P = O(n)$ $T = O(\log n)$

$\longrightarrow$ si $E(|\mathcal{F}_0(\mathcal{O}_i)|) = O(i^\alpha)$ $P = O(n^\alpha)$ $T = O(n^{\alpha-1})$
$\alpha > 1$

Cloisonnement vertical

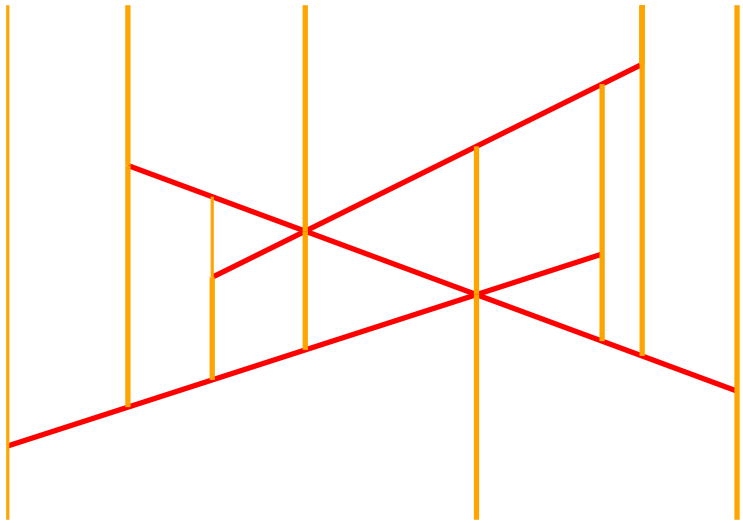
Objets, configurations et conflits



- Un ensemble $\mathcal{O}$ de n segments se coupant en k points
- Une configuration est une des régions (demi-plan, triangle ou trapèze)
- Une configuration T et un segment s sont en conflit si $T \cap s \neq \emptyset$

Cloisonnement vertical

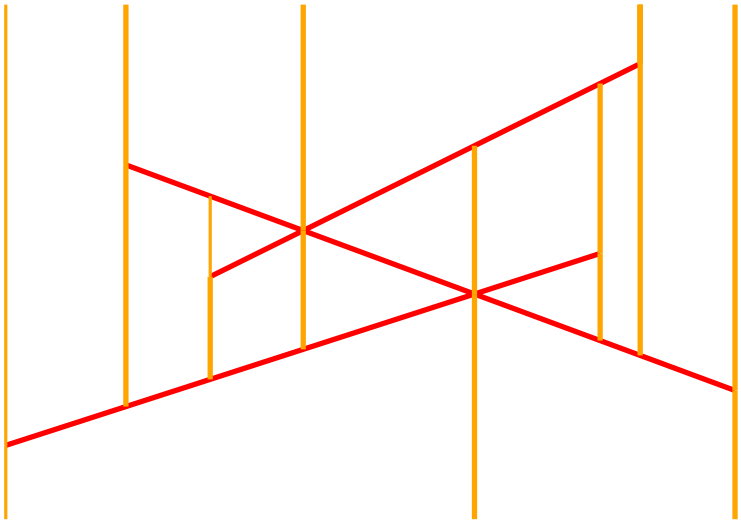
Conditions d'actualisation



- Une configuration est définie par au plus 4 segments
- T est découpé en ≤ 4 nouvelles configurations quand un nouveau segment est inséré
- $\mathcal{F}_0(i, \mathcal{O}) = O(i + k\frac{i^2}{n^2})$

Cloisonnement vertical

Conditions d'actualisation



Théorème

Si on insère les segments dans un ordre aléatoire, on peut

- construire le cloisonnement en temps optimal $O(n \log n + k)$
- se localiser dans une carte planaire en temps $O(\log n)$

Localisation dans une carte planaire

Analyse

Les segments $\mathcal{S}$ de la carte sont insérés dans un ordre aléatoire

Localisation d'un point x quelconque : trouver la cellule du cloisonnement de $\mathcal{S}$ qui contient x

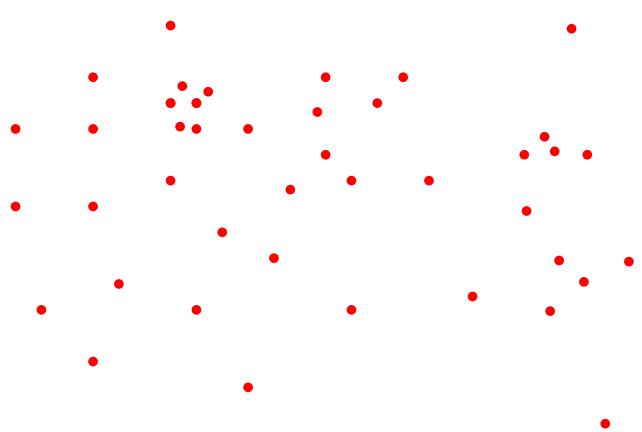
$Q(x) =$ coût de recherche = nb de nœuds du GH en conflit avec x

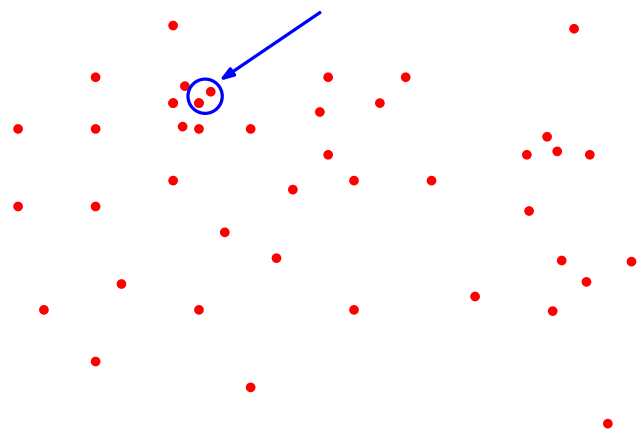
n_i désigne le nœud associé à la cellule qui contient x à l'étape i

$$\begin{aligned} E(Q(x)) &= \sum_{i=1}^{n} \text{proba}(n_i \text{ a été créé à l'étape } i) \\ &= \sum_{i=1}^{n} \frac{4}{i} \\ &= O(\log n) \end{aligned}$$

Recherche des deux plus proches voisins

Problème : Quels sont, parmi n points du plan, les deux points les plus proches ? Peut-on éviter de tester toutes les paires de points ?





Recherche des deux plus proches voisins

Grille et algorithme incrémental

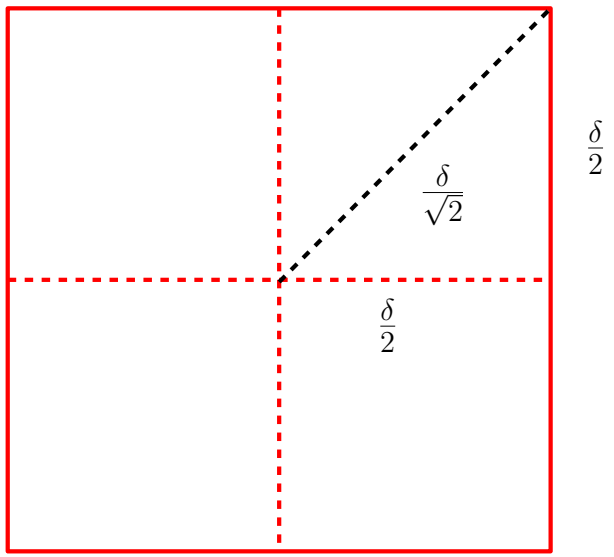
$\mathcal{P}_i =$ ensemble des i premiers points

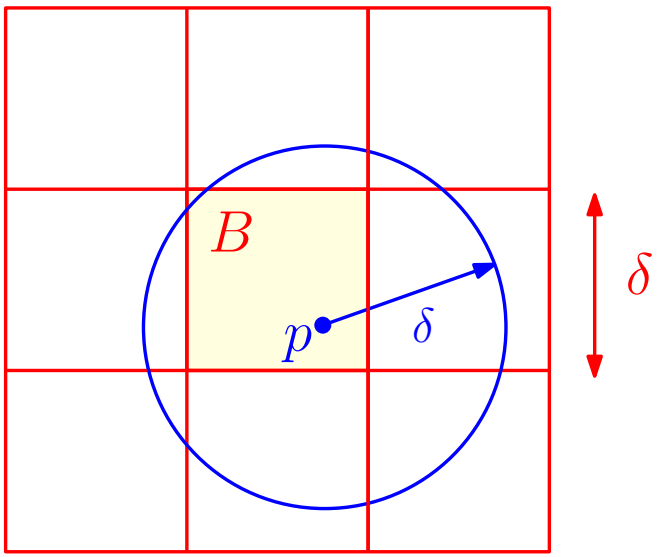
Chaque point de $\mathcal{P}_i$ est stocké dans la cellule qui le contient

1. entrée : $p_1, ..., p_n$
2. $\delta := +\infty$, $G :=$ boîte englobante de $\mathcal{P}$
3. Pour $i = 2, .., n$, insérer p_i dans G et trouver son plus proche voisin v_i dans $\mathcal{P}_i$
   1. si $\|p_i - v_i\| < \delta, \quad ppp := (p_i, v_i)$
      1. $\delta := \|p_i - v_i\|$
      2. construire une nouvelle grille G de côté δ
      3. localiser les points $p_1, ..., p_i$ dans G
4. finpour
5. retourner ppp

Analyse de l'algorithme 1/2

Deux lemmes élémentaires





Lemme 1

Si $\mathcal{P}$ est contenu dans un carré de côté $\delta(\mathcal{P})$, alors $|\mathcal{P}| \leq 4$

Lemme 2

Si $ppp = (p, v)$ et $p \in B$, alors $q \in B$ ou dans une des 8 cellules incidentes à B

Analyse de l'algorithme 2/2

Théorème

On peut calculer les deux plus proches voisins en temps moyen $O(n)$

Démonstration $X_i = 1$ si $\delta(\mathcal{P}_i) \neq \delta(\mathcal{P}_{i-1}$, $X_i = 0$ sinon

$$
\begin{aligned}
T(n) &= 1 + \sum_{i=3}^{n} (1 + X_i \times i) \\
E(T(n)) &= E\left(1 + \sum_{i=3}^{n} (1 + X_i \times i)\right) \\
&\leq n + \sum_{i=2}^{n} E(X_i) \times i \\
&\leq n + \sum_{i=2}^{n} \text{proba}(X_i = 1) \times i \\
&\leq n + \sum_{i=2}^{n} \frac{2}{i} \times i \\
&\leq 3n
\end{aligned}
$$

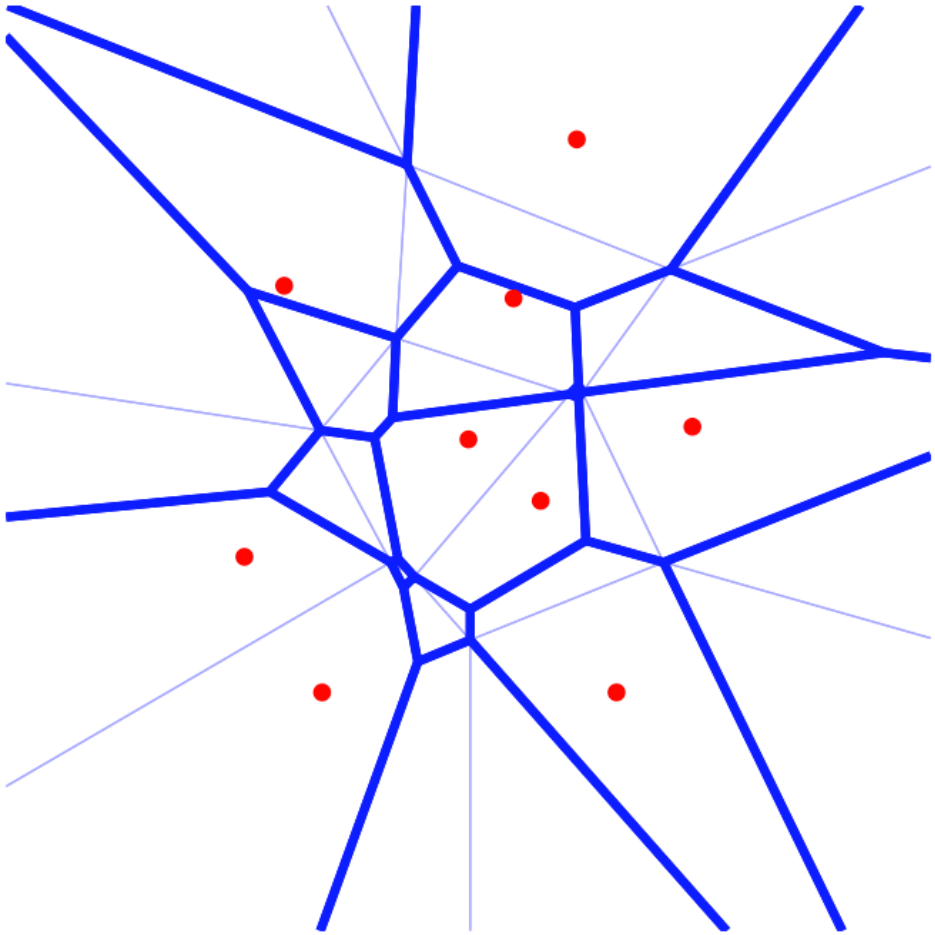
1 Algorithmes incrémentaux randomisés

2 Algorithmes en-ligne

3 Analyse combinatoire

Diagrammes de Voronoï d'ordre k

$\mathcal{P}$ un ensemble fini de points de $\mathbb{R}^d$ (sites)



Chaque cellule de $\mathrm{Vor}_k(\mathcal{P})$ est l'ensemble des points de $\mathbb{R}^d$ qui ont les mêmes k plus proches sites

Complexité combinatoire des diagrammes de Voronoï d'ordre k

Théorème

Le nombre de faces de tous les diagrammes de Voronoï $\mathrm{Vor}_j(P)$ d'ordres $j \leq k$ est

$$O\left(k^{\lceil \frac{d+1}{2} \rceil} \, n^{\lfloor \frac{d+1}{2} \rfloor}\right)$$

Démonstration

- Utiliser le relèvement dans $\mathbb{R}^{d+1}$ (linéarisation)
- Énumérer les sommets de niveaux au plus k dans un arrangement d'hyperplans de $\mathbb{R}^{d+1}$ avec un argument probabiliste
- Vérifier que la borne s'applique aux faces de toutes dimensions (pour d fixé)

Complexité combinatoire des diagrammes de Voronoï d'ordre k

Théorème

Le nombre de faces de tous les diagrammes de Voronoï $\mathrm{Vor}_j(P)$ d'ordres $j \leq k$ est

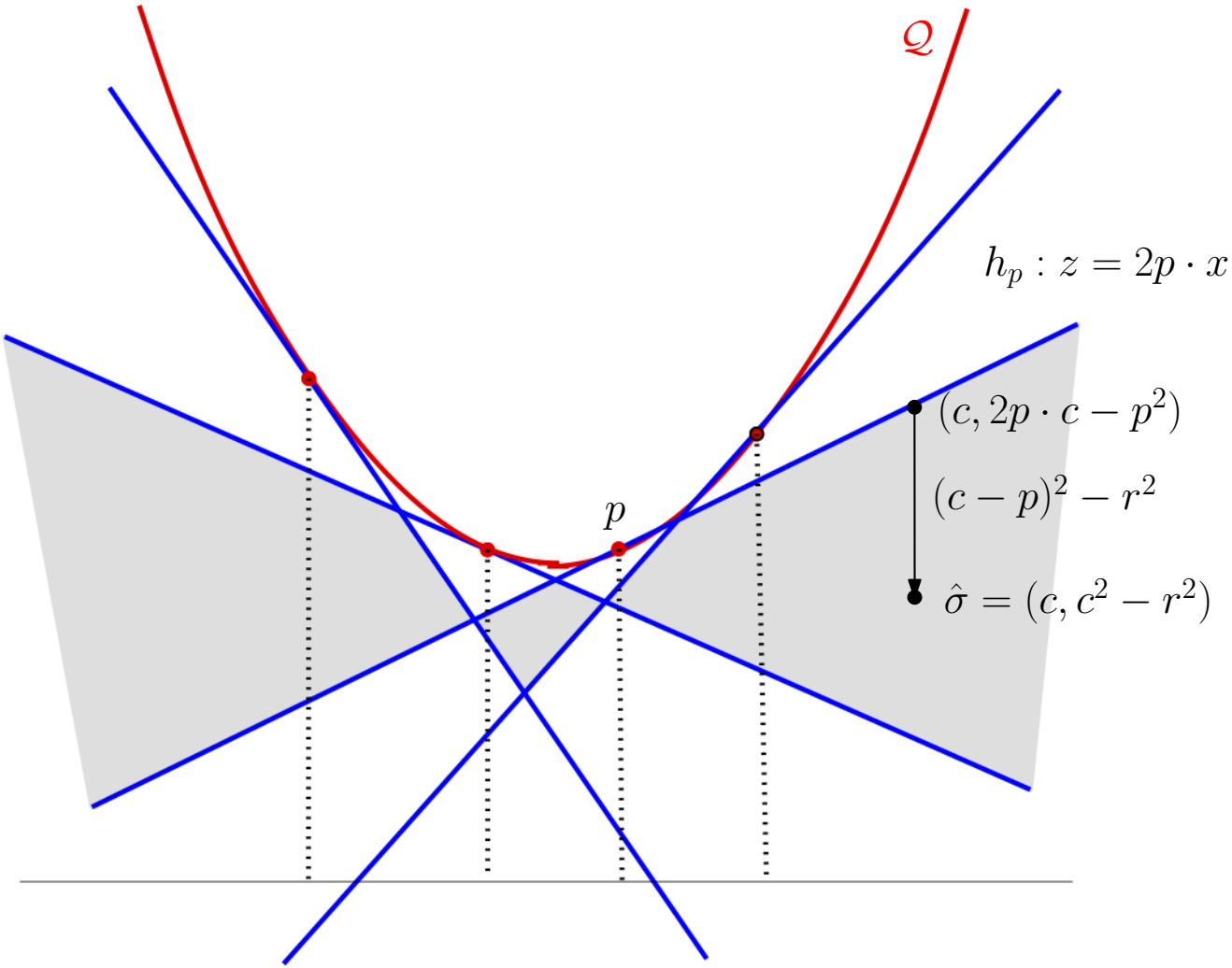
$$O\left(k^{\lceil \frac{d+1}{2} \rceil} \, n^{\lfloor \frac{d+1}{2} \rfloor}\right)$$

Démonstration

- Utiliser le relèvement dans $\mathbb{R}^{d+1}$ (linéarisation)
- Énumérer les sommets de niveaux au plus k dans un arrangement d'hyperplans de $\mathbb{R}^{d+1}$ avec un argument probabiliste
- Vérifier que la borne s'applique aux faces de toutes dimensions (pour d fixé)

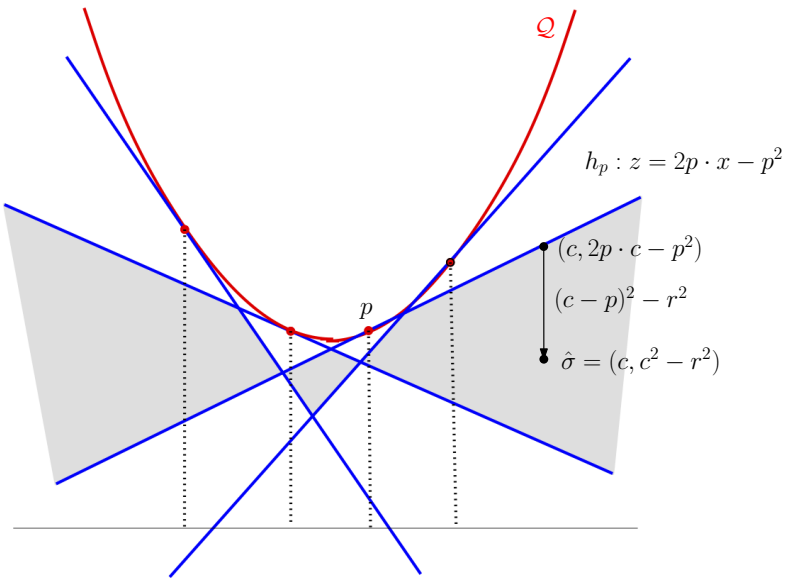
Diagramme de Voronoï d'ordre k et k-niveaux

Linéarisation dans l'espace des sphères



$$c \in \mathrm{Vor}_2(p,q) \;\Leftrightarrow\; \exists\, \sigma(c,r) : \; \hat{\sigma} \in h_p^- \cap h_q^-$$

k-niveaux dans un arrangement d'hyperplans



- $\mathcal{H} =$ arrangement des $h_p,\ p \in \mathcal{P}$
- Les cellules de $\mathrm{Vor}_k(\mathcal{P})$ sont en bijection avec les cellules de $\mathcal{H}$ de niveau k, i.e. avec k hyperplans h_p, $p \in \mathcal{P}$, au dessus
- Le niveau d'un sommet d'une telle cellule de $\mathcal{H}$ varie de k à $\max(0, k-d)$
- En position générale, chaque sommet est incident à d hyperplans

Le théorème de l'échantillon

$\mathcal{O}$ un ensemble de n objets.

$\mathcal{F}(\mathcal{O})$ ensemble des configurations définies par b objets

$\mathcal{F}_j(\mathcal{O})$ ensemble des configurations définies sur $\mathcal{O}$ et en conflit avec j objets

$\mathcal{F}_{\leq k}(\mathcal{O})$ ensemble des configurations définies sur $\mathcal{O}$ en conflit avec $\leq k$ objets de $\mathcal{O}$

$\mathcal{R}(r)$ a random sample of $\mathcal{O}$ of size r

Théorème de l'échantillon [Clarkson & Shor 1992]

Pour $2 \leq k \leq \frac{n}{b+1}$, $\quad |\mathcal{F}_{\leq k}(\mathcal{O})| \leq 4\,(b+1)^b\,k^b\,\mathsf{E}(|\mathcal{F}_0(\mathcal{R}(\lfloor \frac{n}{k} \rfloor))|)$

espérance prise sur tous les échantillons $\mathcal{R} \subset \mathcal{O}$ de taille $\lfloor \frac{n}{k} \rfloor$

Le théorème de l'échantillon

$\mathcal{O}$ un ensemble de n objets.

$\mathcal{F}(\mathcal{O})$ ensemble des configurations définies par b objets

$\mathcal{F}_j(\mathcal{O})$ ensemble des configurations définies sur $\mathcal{O}$ et en conflit avec j objets

$\mathcal{F}_{\leq k}(\mathcal{O})$ ensemble des configurations définies sur $\mathcal{O}$ en conflit avec $\leq k$ objets de $\mathcal{O}$

$\mathcal{R}(r)$ a random sample of $\mathcal{O}$ of size r

Théorème de l'échantillon [Clarkson & Shor 1992]

Pour $2 \leq k \leq \frac{n}{b+1}$, $\quad |\mathcal{F}_{\leq k}(\mathcal{O})| \leq 4\ (b+1)^b\ k^b\ \mathsf{E}(|\mathcal{F}_0(\mathcal{R}(\lfloor \frac{n}{k} \rfloor))|)$

espérance prise sur tous les échantillons $\mathcal{R} \subset \mathcal{O}$ de taille $\lfloor \frac{n}{k} \rfloor$

Démonstration du théorème de l'échantillon

$$\mathsf{E}\left(|\mathcal{F}_0(\mathcal{R}(r)|\right) = \sum_j |\mathcal{F}_j(\mathcal{O})| \frac{\binom{n-b-j}{r-b}}{\binom{n}{r}} \geq |\mathcal{F}_{\leq k}(\mathcal{O})| \frac{\binom{n-b-k}{r-b}}{\binom{n}{r}}$$

un calcul permet de montrer que pour $r = \frac{n}{k}$

$$\frac{\binom{n-b-k}{r-b}}{\binom{n}{r}} \geq \frac{1}{4(b+1)^b k^b}$$

□

Borne sur le nombre de sommets des niveaux $\leq k$

Théorème Le nombre de faces des niveaux $\leq k$ d'un arrangement de n hyperplans de $\mathbb{R}^d$ est $O(k^{\lceil \frac{d}{2} \rceil} n^{\lfloor \frac{d}{2} \rfloor})$

Démonstration

$\mathcal{O}$ l'ensemble des hyperplans

$\mathcal{R}$ un échantillon aléatoire de $\mathcal{H}$

$\mathcal{F}_{\leq k}(\mathcal{O})$: l'ensemble des sommets de $\mathcal{H}(L)$ de niveaux $\leq k$, $k > 1$

Par le théorème de l'échantillon aléatoire et le théorème de la borne supérieure

$$|\mathcal{F}_{\leq k}(\mathcal{O})| \leq 4(d+1)^d \, k^d \, O\left(\lfloor \frac{n}{k} \rfloor^{\lfloor \frac{d}{2} \rfloor}\right)$$

Si les hyperplans sont en position générale (cas le pire), la borne vaut pour les faces de toutes dimensions

Corollaire 1

Diagrammes de Voronoï d'ordres $\leq k$

Complexité combinatoire

Le nombre total de faces de tous les diagrammes de Voronoï d'ordres $\leq k$ de n points de $\mathbb{R}^d$ est

$$O\left(k^{\lceil \frac{d+1}{2} \rceil} n^{\lfloor \frac{d+1}{2} \rfloor}\right)$$

Construction

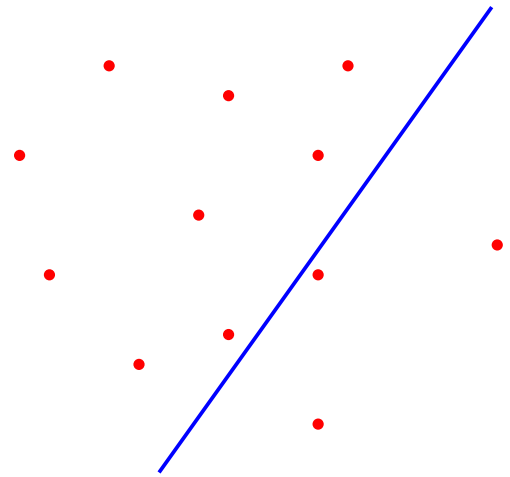
C'est aussi le temps requis pour calculer ces diagrammes si $d \geq 3$.

Pour $d = 2$, le temps est $O(nk^2 \log \frac{n}{k})$

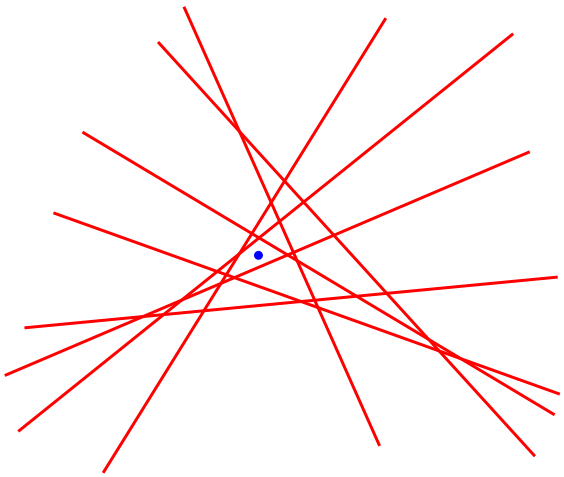
Corollaire 2

k-ensembles

Primal



Dual



Etant donné un ensemble $\mathcal{P}$ de n points de $\mathbb{R}^d$, un k-ensemble est un sous-ensemble de $\mathcal{P}$ de taille k qui peut être séparé des autres points de $\mathcal{P}$ par un hyperplan

Le nombre total de $\leq k$-ensembles de $\mathcal{P}$ est

$$O\left(k^{\lceil \frac{d}{2} \rceil} n^{\lfloor \frac{d}{2} \rfloor}\right)$$

D'autres algorithmes randomisés dans la suite du cours

- La méthode probabiliste et le lemme local de Lovasz
- Projections aléatoires
- La recherche de plus proches voisins en grandes dimensions