# Approche pour la planification et décision

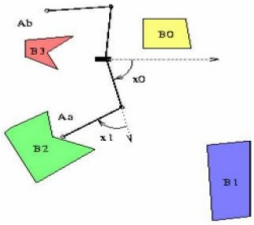
MENASRI Riad

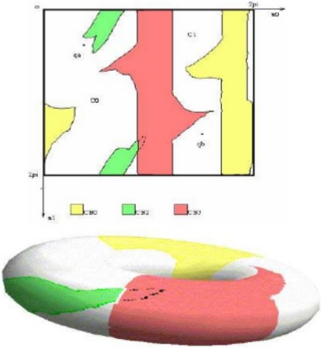
01-2015

menasri.riad@gmail.com

# Problème

**But** : Explorer $C_{libre}$ afin de trouver un chemin sans collision connectant $q_s$ et $q_g$



# Complétude

Algorithme Complet : Trouve la solution si elle existe, renvoie échec sinon

Or, complexité exponentielle du problème de planification en fonction de la dimension de l'espace de configurations (n) → Des algorithmes complet résolvant une instance du pb ont été proposé pour n=2, 3 ou 4

*[Schwartz & Sharir 81, Canny 87]* proposent deux algorithmes complets résolvant le cas général du problème de planification (exponentiel en n) mais jamais implémenté

Algorithme complet → intéressant de point de vue théorique
→ en pratique, difficile de les mettre en oeuvre

# Complétude

Que peut on faire ?

1. Laisser de coté la notion de complétude et choisir des heuristiques ⇒ marche bien dans certains cas mais aucune garantie sur la convergence et sur les performances

2. Opter pour une complétude plus faible
   - *Complétude à une résolution donnée* : une discrétisation systématique de C, complétude garantit pour un niveau de résolution donnée (n petit)
   - *Complétude probabiliste* : la probabilité de trouver une solution tend vers 1 quand le temps de calcul temps vers l'infini (mais si une solution n'est pas trouvée en temps fini, que conclure) ??

L'approche est-elle complète ?

| | | |
|---|---|---|
| 1. | Les méthodes exactes | Complète |
| 2. | Les techniques approximatives | Complète à une résolution |
| 3. | Les approches probabilistes | Complète en probabilité |
| 4. | Les heuristiques | Incomplète |

Est ce que l'approche calcule explicitement l'espace des configurations ?
Est ce que l'approche essaie de capturer la topologie de C ?

Approche à requête unique (Single query) → dépend de $q_s$ et $q_g$
Approche à requêtes multiple (Multiple query) → ne dépend pas de $q_s$ et $q_g$

# Classification des approches

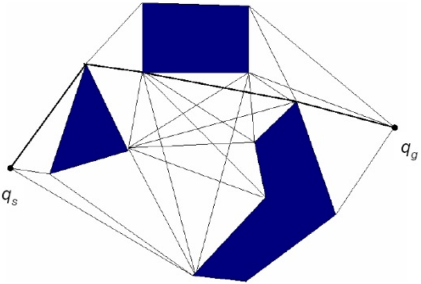
1. <u>Méthodes construisant un graphe capturant la connectivité de C</u>
   - Tend à capturer la topologie de l'espace de configuration dans une structure de graphe
   - Le calcul de l'espace de configuration est indépendant des configurations initiale et finale du robot (multiple query)
2. <u>Méthode construisant incrémentalement un arbre</u>
   - Ne cherche pas à capturer la connectivité de l'espace de configuration mais à résoudre une requête de planification
   - L'arbre construit dépend fortement des configurations initiale/finale du robot

# Approches conduisant à la construction d'un graphe

- Graphe de Visibilité *[Nilsson 69]*
- Rétraction
  - Diagramme de Voronoï *[Dunlaing & Yap 82]*
- Décomposition cellulaire
  - Exacte
  - Approximative
- Réseaux probabilistes (et leurs variantes)

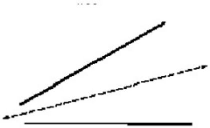
# Graphe de visibilité

Construit un réseau (1D) capturant la topologie de $C_{semi\text{-}libre}$
Planification : (1) connecté $q_s$ et $q_g$ au graphe, (2) phase de recherche
Fournit le chemin le plus court dans le cas 2D (n'est plus vrai en 3D)



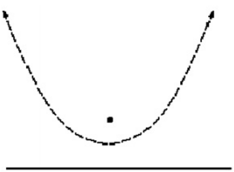
# Diagramme de Voronoi

- Construit une rétraction (surjection) de l'espace libre à partir de points équidistants des obstacles
- La connectivité de l'espace initial doit être préservée
- Construit un réseau 1D dans $C_{libre}$ composé de : segments +arcs paraboliques



(a)

(b)

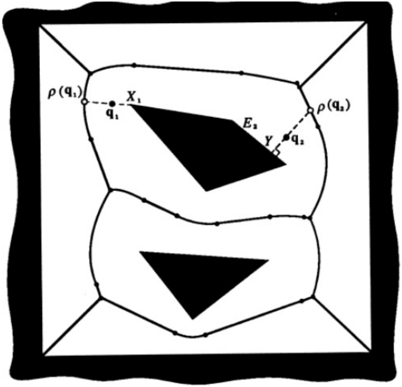


(c)

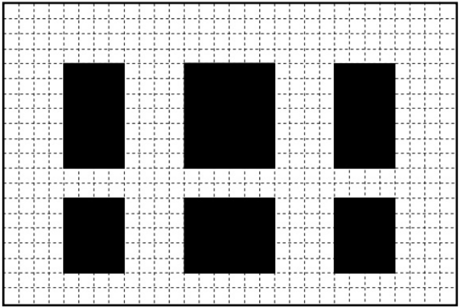
# Diagramme de Voronoi

Pour des obstacles polygonaux ($X_i$ , $E_i$) :

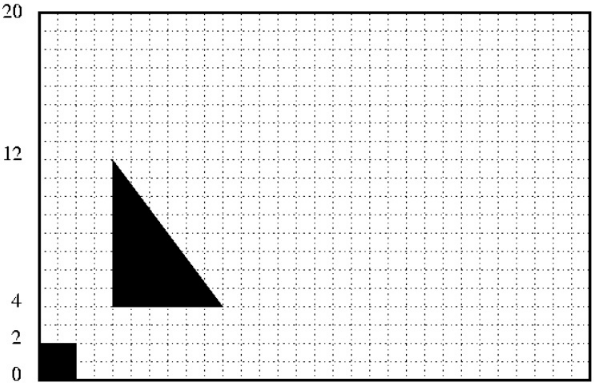
- (Xi , Xj) et (Ei , Ej) ⇒ segments
- (Xi , Ej) ⇒ arcs de paraboles



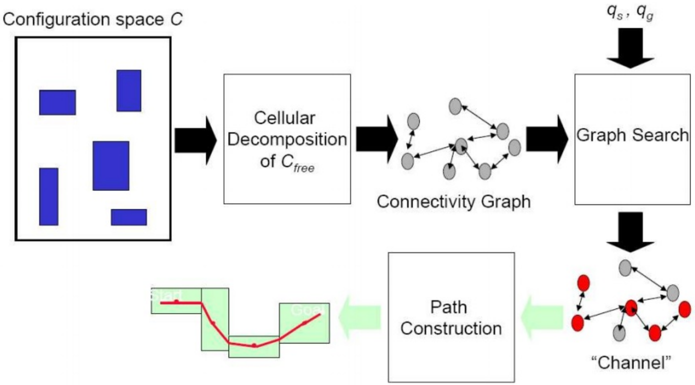
# Exercice



# Exercice

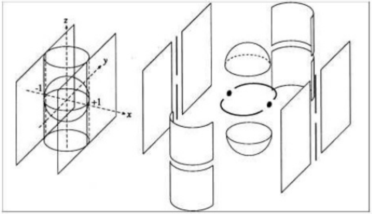


# Décomposition cellulaire

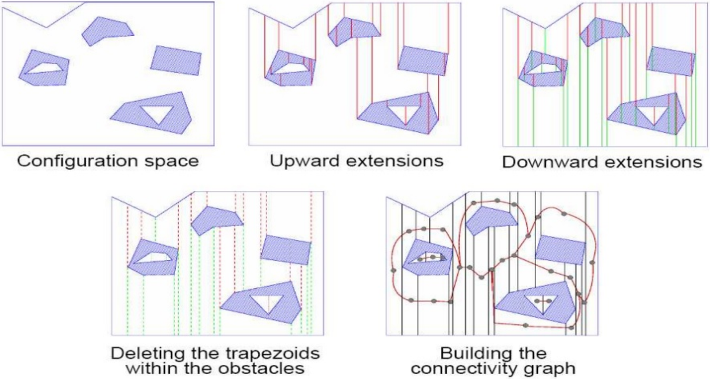


# Décomposition cellulaire exacte

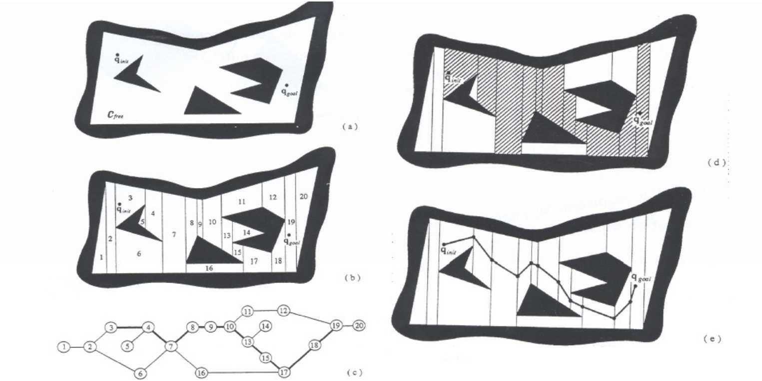
- Caractéristiques :
  - Complète : $\cup$ cellules = $C_{libre}$
  - Nombre réduit de cellules
  - Complexité croissante de la décomposition et de la construction du graphe



# Configurations trapézoïdale

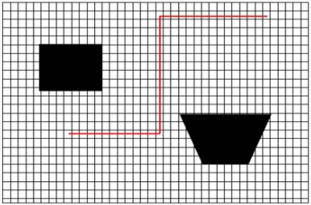


# Décomposition cellulaire exacte

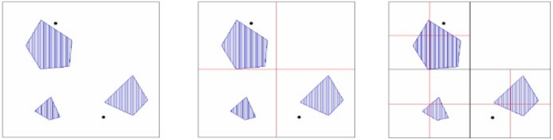


# Décomposition cellulaire approximative

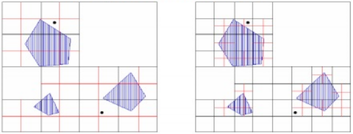
- Caractéristiques :
  - Complétude à une résolution donnée: ∪ cellules ⊂ $C_{libre}$
  - Forme des cellules fixées
  - Grand nombre de cellules
  - Complexité réduite de décomposition et de construction du graphe
- Exemple : décomposition rectangulaire



# Quadtree

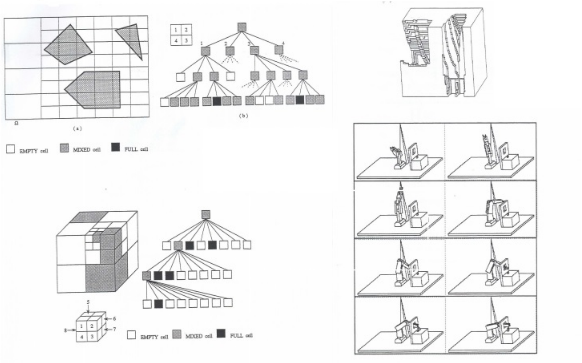


Décomposition hiérarchique



utiliser des cellules de taille variable en fonction de la complexité locale de l'environnement

# Octree


(a)
(b)
EMPTY cell
MIXED cell
FULL cell
EMPTY cell
MIXED cell
FULL cell

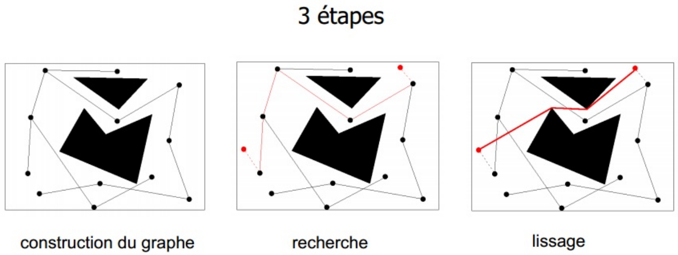
# Réseaux probabilistes

- Idée :
  Pourquoi une décomposition exacte ou approximative de l'espace de configuration → Il suffit de capturer sa connectivité dans un graphe

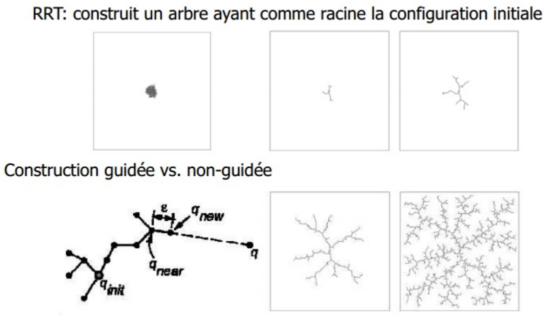
- Principe :
  1. Générer aléatoirement des configurations
  2. Garder celle dans $C_{libre}$
  3. Les connecter par des chemins

⇒ Carte routière (Roadmap) : réseau 1D → approximation de la connectivité de $C_{libre}$

# Principe des réseaux probabilistes

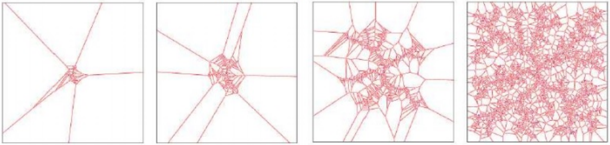


# Rapidly-exploring random trees (RRT)

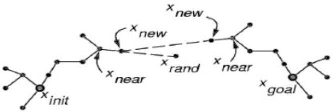


# Rapidly-exploring random trees (RRT)

Interprétation de RRT par des digrammes de Voronoï



RRT bidirectionnel



# Exercice