Variations de A*

- *Beam search*
  - on met une limite sur le contenu de *open* et *closed*
  - recommandé lorsque pas assez d'espace mémoire

- Iterative deepening
  - on met une limite sur la profondeur
  - on lance A* jusqu'à la limite de profondeur spécifiée
  - si pas de solution on augmente la profondeur et on recommence A*
  - ainsi de suite jusqu'à trouver une solution

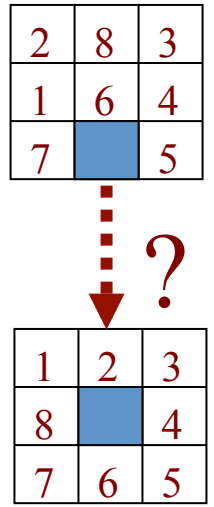
Variations de A*

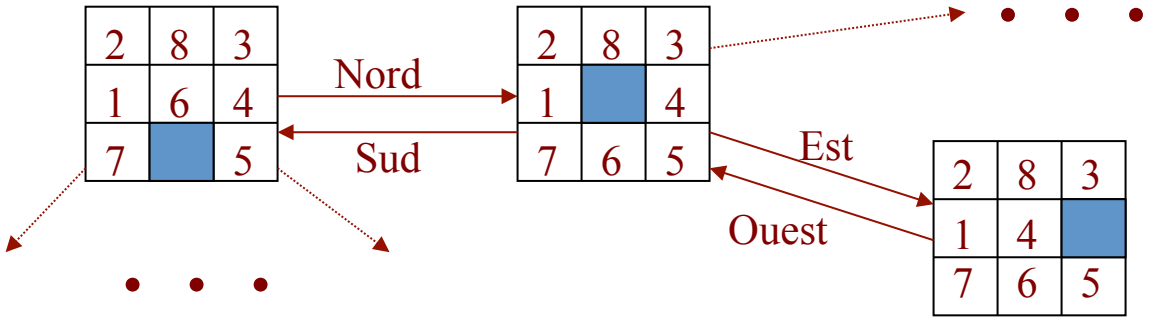
- *Recursive best-first search* (RBFS) et *simplified memory-bounded* A* (SMA*)
  - variantes de A* qui utilisent moins de mémoire mais peuvent être plus lentes

- D* (inventé par Stenz et ses collègues).
  - A* dynamique, où le coût des arrêtes peut changer durant l'exécution
    - » évite de refaire certains calculs lorsqu'il est appelé plusieurs fois pour atteindre le même but, suite à des changements de l'environnement

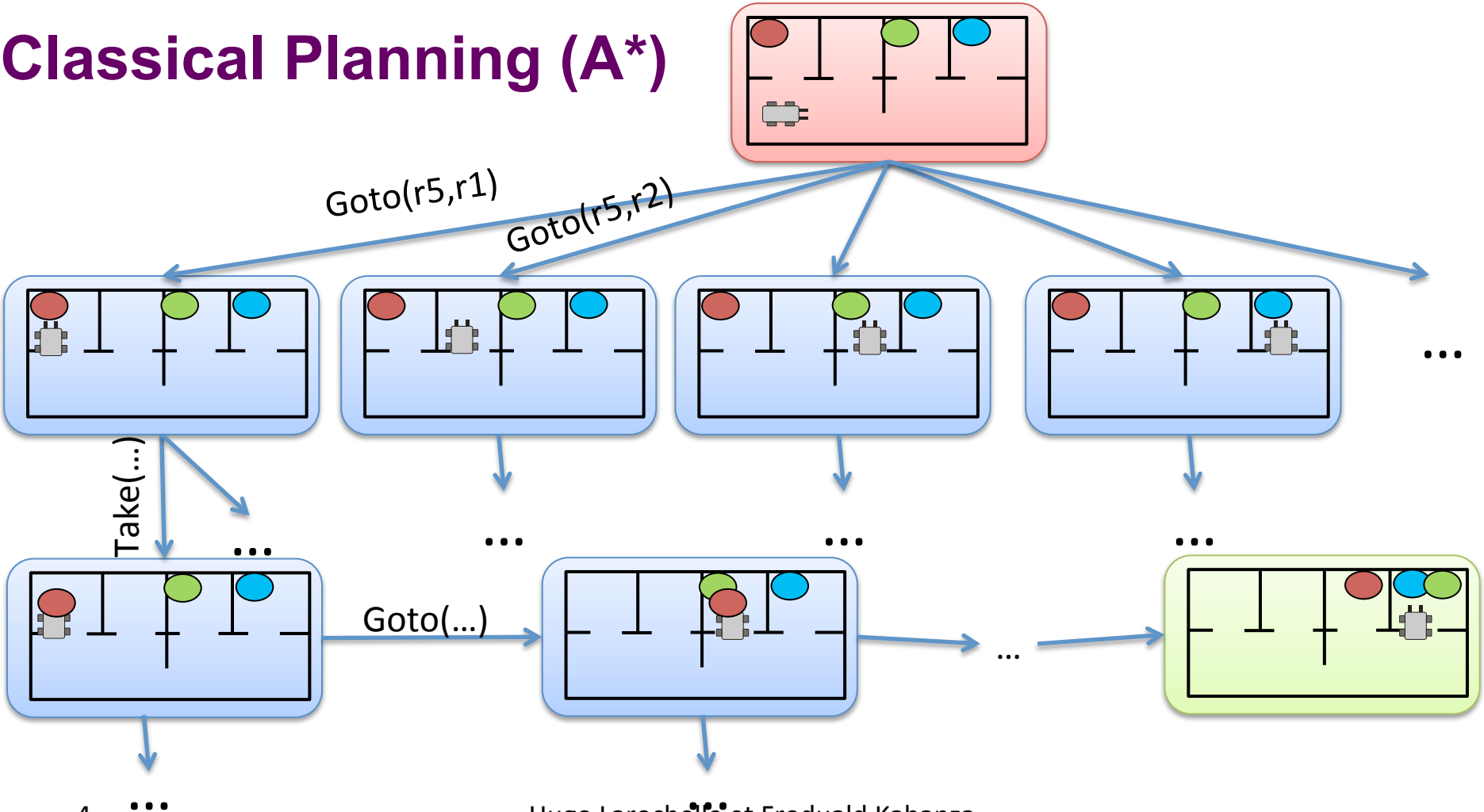
Application : jeu d'énigme

- 8-puzzle
  - *État* : configuration légale du jeu
  - *État initial* : configuration initiale
  - *État final (but)* : configuration gagnante
  - *Transitions*

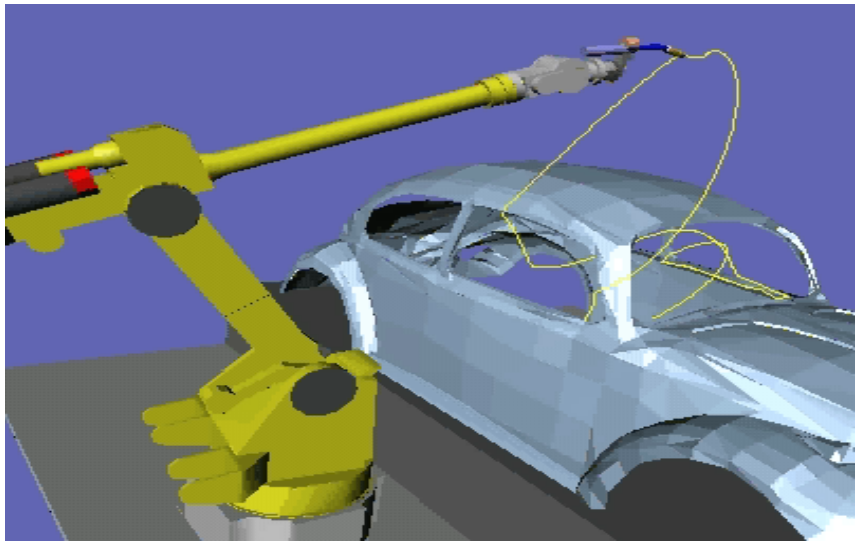


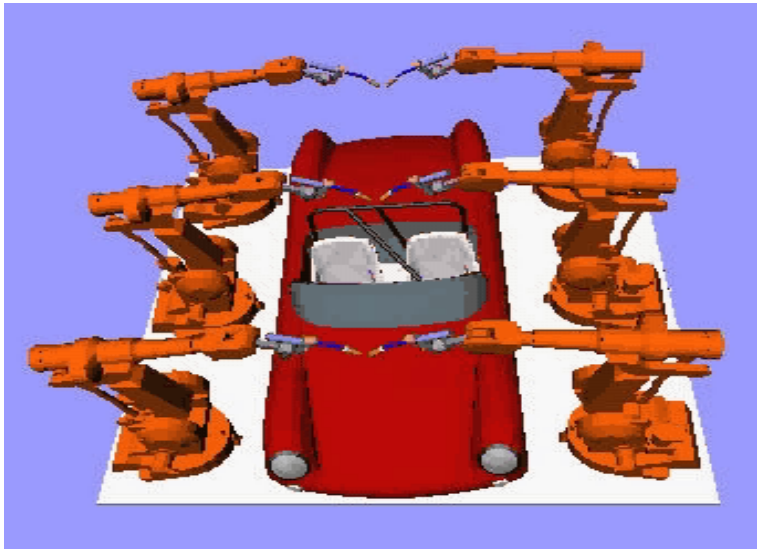


Classical Planning (A*)



Application : industrie automobile





Démos du Motion Planning Kit (Jean-Claude Latombe)

Application : jeux vidéos et cinéma

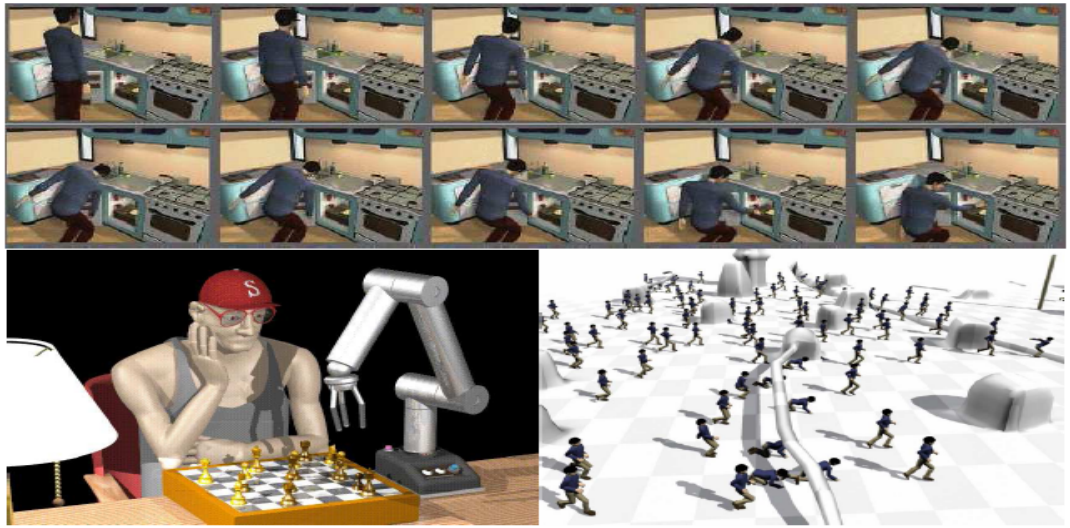


Figure 1.8: Across the top, a motion computed by a planning algorithm, for a digital actor to reach into a refrigerator [499]. In the lower left, a digital actor plays chess with a virtual robot [545]. In the lower right, a planning algorithm computes the motions of 100 digital actors moving across terrain with obstacles [592].

[Steven LaValle. *Planning Algorithms*]

Énoncé du problème

- Calculer une trajectoire géométrique d'un solide articulé sans collision avec des obstacles statiques.

Entrée :

- Géométrie du robot et des obstacles
- Cinétique du robot (degrés de liberté)
- Configurations initiale et finale

Planificateur de trajectoires

Sortie :

- Une séquence continue de configurations rapprochées, sans collision, joignant la configuration initiale à la configuration finale

Cadre générale de résolution du problème

Problème continu

(espace de configuration + contraintes)

↓

Discrétisation

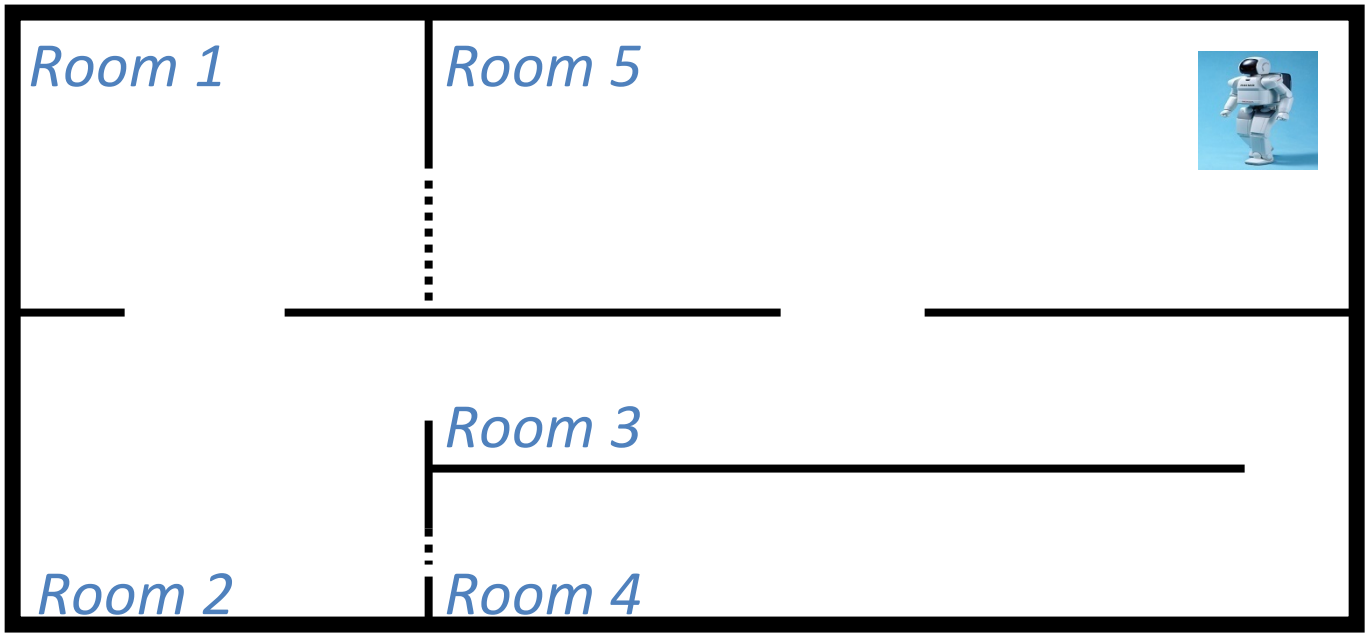
(décomposition, échantillonage)

↓

Recherche heuristique dans un graphe

(A* ou similaire)

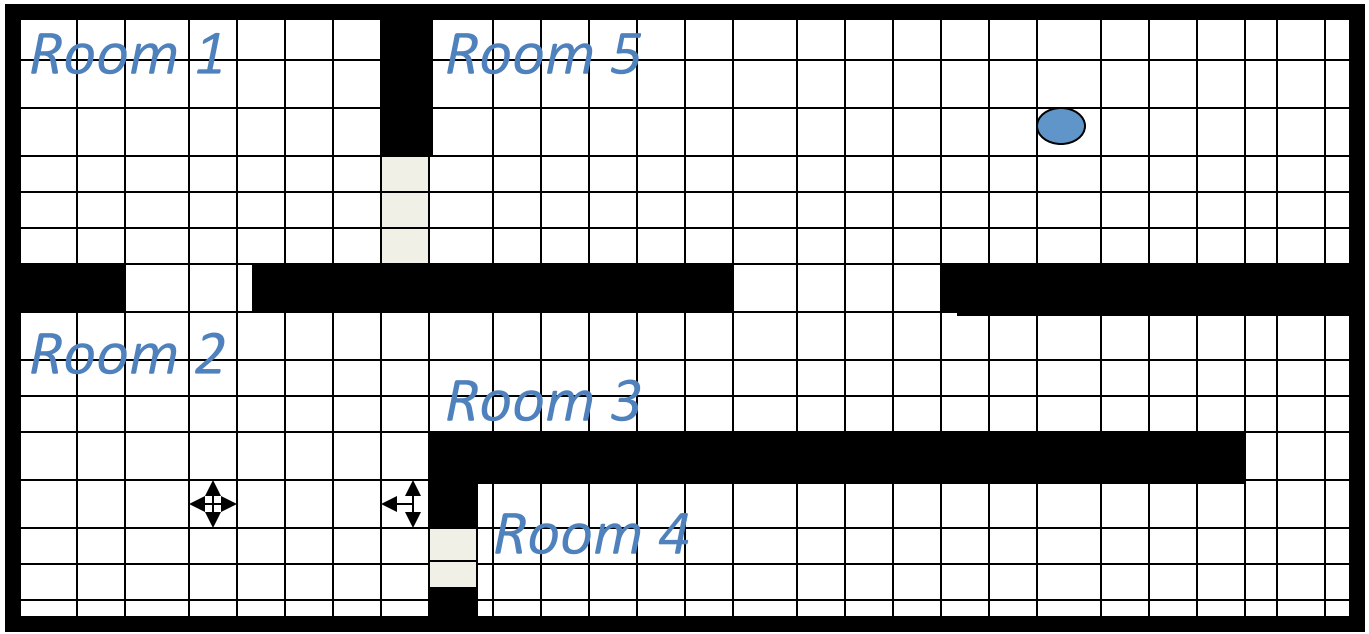
Approche combinatoire par décomposition en cellules



Décomposer la carte en grille (*occupancy grid*) :
4-connected (illustré ici) ou 8-connected.

Noeud : case occupée par le robot + orientation du robot

Transitions :
- Turn left
- Turn right
- Go straight ahead



***Heuristiques* :**
- Distance euclidienne, durée du voyage
- Consommation d'énergie ou coût du billet
- Degré de danger (chemin près des escaliers, des ennemis).

Go east =
(Turn right) +
Go straight ahead