Feuille d'exercices #5 — Planification - Solutions

IFT 3345

Question 1. ***Champs de potentiels artificiels (potential fields)***

Un robot se déplace dans $\mathbb{R}^2$. Sa position actuelle est

$$x = (1, 1)$$

L'objectif est situé en

$$x_{\text{goal}} = (3, 1)$$

et un obstacle est situé en

$$x_{\text{obs}} = (1, 3).$$

On définit les potentiels suivants :

— Potentiel attractif :

$$U_{\text{att}}(x) = \frac{1}{2}\|x - x_{\text{goal}}\|^2$$

— Potentiel répulsif (pour chaque obstacle) :

$$U_{\text{rep}}(x) = \begin{cases} \frac{1}{2}\left(\frac{1}{d(x)} - \frac{1}{d_0}\right)^2 & \text{si } d(x) \leq d_0 \\ 0 & \text{sinon} \end{cases}$$

où $d(x) = \|x - x_{\text{obs}}\|$ et $d_0 = 3$.

(a) Calculer la force attractive :

$$F_{\text{att}} = -\nabla U_{\text{att}}(x).$$

(b) Calculer la force répulsive :

$$F_{\text{rep}} = -\nabla U_{\text{rep}}(x).$$

(c) Calculer la force totale :

$$F = F_{\text{att}} + F_{\text{rep}}.$$

(d) Dans quelle direction générale le robot va-t-il se déplacer ?

Solution 1.

(a)

$$U_{\text{att}}(x) = \frac{1}{2}\|x - x_{\text{goal}}\|^2$$

$$\nabla U_{\text{att}}(x) = x - x_{\text{goal}}$$

$$F_{\text{att}} = -\nabla U_{\text{att}}(x) = x_{\text{goal}} - x$$

$$F_{\text{att}} = (3, 1) - (1, 1) = (2, 0)$$

(b) On a :

$$d(x) = \|x - x_{\text{obs}}\| = \|(1,1) - (1,3)\| = 2$$

Comme $d(x) = 2 < d_0 = 3$, le potentiel est actif.

$$U_{\text{rep}}(x) = \frac{1}{2}\left(\frac{1}{d(x)} - \frac{1}{d_0}\right)^2$$

On utilise la règle de chaîne :

$$F_{\text{rep}} = -\nabla U_{\text{rep}}(x) = \left(\frac{1}{d(x)} - \frac{1}{d_0}\right)\frac{1}{d(x)^2}\frac{x - x_{\text{obs}}}{d(x)}$$

Calcul des termes :

$$\frac{1}{d(x)} = \frac{1}{2}, \quad \frac{1}{d_0} = \frac{1}{3}$$

$$\frac{1}{d(x)} - \frac{1}{d_0} = \frac{1}{6}$$

$$\frac{1}{d(x)^2} = \frac{1}{4}, \quad \frac{1}{d(x)} = \frac{1}{2}$$

$$x - x_{\text{obs}} = (1,1) - (1,3) = (0,-2)$$

Assemblage :

$$F_{\text{rep}} = \frac{1}{6} \cdot \frac{1}{4} \cdot \frac{1}{2} \cdot (0,-2)$$

$$F_{\text{rep}} = \frac{1}{48}(0,-2) = \left(0, -\frac{1}{24}\right)$$

(c)

$$F = F_{\text{att}} + F_{\text{rep}}$$

$$F = (2,0) + \left(0, -\frac{1}{24}\right) = \left(2, -\frac{1}{24}\right)$$

(d) — Le robot se déplace principalement vers la droite (vers l'objectif)
— Il est légèrement repoussé vers le bas (loin de l'obstacle)

Question 2. ***Rapidly-Exploring Random Trees (RRT)***

On considère un robot se déplaçant dans un environnement 2D. La position initiale est :

$$x_{\text{start}} = (0, 0)$$

Un obstacle rectangulaire occupe la région :

$$2 \leq x \leq 4, \quad 1 \leq y \leq 3$$

La taille de pas est fixée à $\delta = 1$ (Avancer depuis ce nœud en direction du point échantillonné d'une distance $\delta = 1$). On vous donne la séquence de points échantillonnés suivante :

$$x_1 = (3, 0), \quad x_2 = (3, 3), \quad x_3 = (5, 2), \quad x_4 = (6, 0)$$

(a) À partir de x_{start}, construire l'arbre RRT étape par étape.

(b) Pour chaque point x_i, indiquer :
- le nœud le plus proche,
- le nouveau nœud ajouté (si applicable),
- si l'arête est rejetée à cause d'une collision.

(c) Dessiner l'arbre final.

Solution 2.

— Initialisation L'arbre contient initialement :

$$\mathcal{T} = \{(0, 0)\}$$

— Itération 1 : $x_1 = (3, 0)$
- Nœud le plus proche : $(0, 0)$
- Direction : $(3, 0) - (0, 0) = (3, 0)$
- Mouvement de longueur $\delta = 1$:
$$x_{\text{new}} = (1, 0)$$
- Collision : aucune (en dehors de l'obstacle)

On ajoute :
$$(1, 0)$$

— Itération 2 : $x_2 = (3, 3)$
- Nœuds existants : $(0, 0), (1, 0)$
- Distances :
$$d((0, 0), (3, 3)) = \sqrt{18}, \quad d((1, 0), (3, 3)) = \sqrt{13}$$
- Nœud le plus proche : $(1, 0)$
- Direction : $(3, 3) - (1, 0) = (2, 3)$
- Normalisation (approx.) :
$$\frac{(2, 3)}{\sqrt{13}} \approx (0.55, 0.83)$$

— Nouveau point :

$$x_{\text{new}} \approx (1.55, 0.83)$$

— Collision : aucune (le point reste sous l'obstacle)

On ajoute :

$$(1.55, 0.83)$$

— Itération 3 : $x_3 = (5, 2)$
  — Nœuds existants : $(0, 0), (1, 0), (1.55, 0.83)$
  — Le nœud le plus proche est $(1.55, 0.83)$
  — Direction vers $(5, 2)$
  — Le segment entre $(1.55, 0.83)$ et le nouveau point traverse la zone :

$$2 \leq x \leq 4, \quad 1 \leq y \leq 3$$

  — Donc il y a collision

  Aucun nœud n'est ajouté.

— Itération 4 : $x_4 = (6, 0)$
  — Nœuds existants : $(0, 0), (1, 0), (1.55, 0.83)$
  — Distances :

$$d((0,0),(6,0)) = 6, \quad d((1,0),(6,0)) = 5,$$
$$d((1.55,0.83),(6,0)) = \sqrt{4.45^2 + 0.83^2} = \sqrt{20.49} \approx 4.53$$

  — Nœud le plus proche : $(1.55, 0.83)$
  — Direction : $(6, 0) - (1.55, 0.83) = (4.45, -0.83)$
  — Normalisation (approx.) :

$$\frac{(4.45, -0.83)}{\sqrt{20.49}} \approx (0.98, -0.18)$$

  — Nouveau point :

$$x_{\text{new}} \approx (1.55, 0.83) + (0.98, -0.18) = (2.54, 0.65)$$

  — Collision : aucune (le segment reste sous $y = 1$, donc en dessous de l'obstacle)

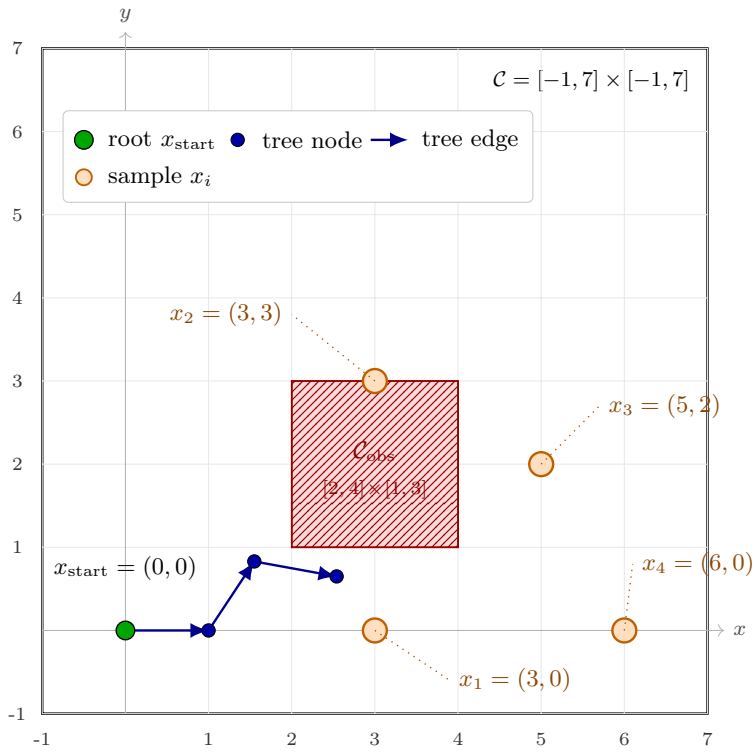
  On ajoute :

$$(2.54, 0.65)$$

Arbre final Les nœuds de l'arbre sont :

$$(0, 0), \quad (1, 0), \quad (1.55, 0.83), \quad (2.54, 0.65)$$

Les arêtes sont :
— $(0, 0) \to (1, 0)$
— $(1, 0) \to (1.55, 0.83)$
— $(1.55, 0.83) \to (2.54, 0.65)$


y
x
$\mathcal{C} = [-1, 7] \times [-1, 7]$
root x_{start}
tree node
tree edge
sample x_i
$x_2 = (3, 3)$
$x_3 = (5, 2)$
$\mathcal{C}_{\text{obs}}$
$[2, 4] \times [1, 3]$
$x_{\text{start}} = (0, 0)$
$x_4 = (6, 0)$
$x_1 = (3, 0)$
7
6
5
4
3
2
1
-1
-1
1
2
3
4
5
6
7

Question 3. ***Planification avec Probabilistic Roadmaps (PRM)***

On considère un robot se déplaçant dans un environnement 2D avec des obstacles. Les positions de départ et d'arrivée sont x_{start} et x_{goal}.

Écrire un algorithme en pseudo-code

`function PRM(x_start, x_goal, N, d)`

pour calculer un chemin de x_{start} à x_{goal} en utilisant la méthode PRM.
Votre algorithme doit :

- Échantillonner des configurations dans l'espace libre
- Construire un graphe de roadmap $G = (V, E)$ (vous pouvez ajouter des sommets et des arêtes au graphe avec les fonctions `G.add_node(v)` et `G.add_edge(v1, v2)`).
- Connecter chaque nœud à ses voisins dans un rayon d
- Ajouter x_{start} et x_{goal} au graphe
- Calculer le chemin final

On suppose que les fonctions suivantes sont disponibles :

- `SampleFree()` : retourne une configuration aléatoire dans l'espace libre
- `CollisionFree(x1, x2)` : retourne `true` si le segment entre x_1 et x_2 est sans collision
- `Near(x, G.V, d)` : retourne l'ensemble des nœuds de V à une distance inférieure à d de x
- `ShortestPath(G, start, goal)` : retourne le plus court chemin dans le graphe G

Solution 3.

```
function PRM(x_start, x_goal, N, d):

    #  Échantillonnage
    G.V = {}
    G.E = {}
    for i = 1 to N:
        x = SampleFree()
        G.add_node(x)


    # Connexion des voisins (rayon d)
    for each x in G.V:
        for each x_near in Near(x, G.V, d):
            if CollisionFree(x, x_near):
                G.add_edge(x, x_near)

    # Ajouter start et goal
    G.add_node(x_start)
    G.add_node(x_goal)

    for each x_near in Near(x_start, G.V, d):
        if CollisionFree(x_start, x_near):
           G.add_edge(x_start, x_near)
```

```
for each x_near in Near(x_goal, G.V, d):
    if CollisionFree(x_goal, x_near):
       G.add_edge(x_goal, x_near)


# Recherche du plus court chemin
path = ShortestPath(G, q_start, q_goal)

return path
```

Question 4.

Wavefront Planner

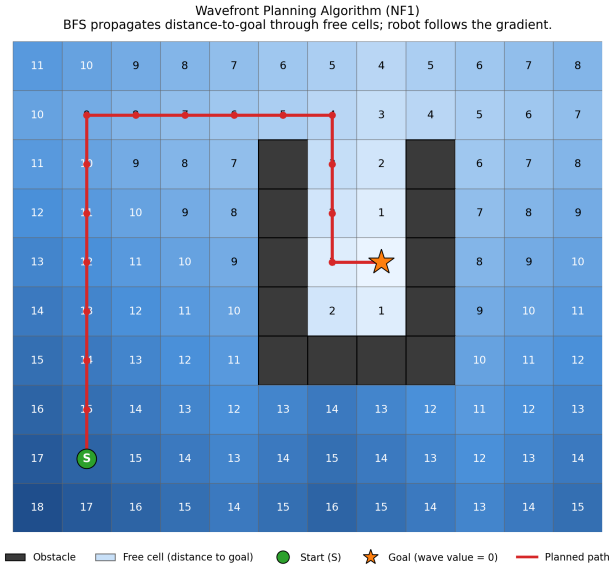


FIGURE 1 – Le **wavefront planner**

On considère un robot évoluant dans une grille 2D discrète. Chaque cellule est soit libre, soit occupée par un obstacle.
Le **wavefront planner** fonctionne comme suit :

- On initialise la cellule objectif avec la valeur 0.
- On propage ensuite des valeurs entières dans la grille :
  - chaque cellule libre voisine reçoit une valeur égale à la valeur de la cellule courante + 1
- On répète jusqu'à ce que toutes les cellules accessibles aient été visitées.
- Pour planifier un chemin depuis x_{start}, on suit à chaque étape le voisin ayant la plus petite valeur.

(a) Ce planificateur est-il **complet** ? Justifier.
(b) Ce planificateur est-il **optimal** ? Justifier.
(c) Ce planificateur est-il **réactif** ou **délibératif** ? Justifier.
(d) Formuler le problème sous forme d'un **processus de décision markovien (MDP)** :
  - définir l'ensemble des états
  - définir l'ensemble des actions
  - définir la fonction de transition
  - définir la fonction de coût (ou récompense)

Solution 4.

(a) Oui, le wavefront planner est **complet** (jusqu'à la discrétisation de la grille).
En effet :
  - la propagation visite toutes les cellules accessibles depuis l'objectif

— si un chemin existe entre x_{start} et x_{goal}, alors x_{start} recevra une valeur
— sinon, aucune valeur n'est assignée

Donc l'algorithme trouve un chemin si et seulement si un chemin existe.

(b) Oui, le wavefront planner est **optimal** (jusqu'à la discrétisation de la grille et la discrétisation de l'espace d'action).
En effet :
— la propagation correspond à une recherche en largeur (*breadth-first search*)
— chaque cellule reçoit la distance minimale (en nombre de pas) jusqu'à l'objectif
— suivre le gradient décroissant donne donc un chemin de longueur minimale

(c) Le wavefront planner est **délibératif**.
— il construit une représentation globale (carte de distances)
— la planification est effectuée avant l'exécution
— le robot suit ensuite un plan pré-calculé

(d) Formulation comme MDP
États :

$$\mathcal{S} = \{\text{cellules libres de la grille}\}$$

Actions :

$$\mathcal{A} = \{\text{déplacements vers les voisins (haut, bas, gauche, droite)}\}$$

Transitions :

$$P(s' \mid s, a) = \begin{cases} 1 & \text{si } s' \text{ est le voisin atteint en appliquant } a \\ 0 & \text{sinon} \end{cases}$$

Coût :

$$c(s, a) = 1 \quad \text{pour chaque déplacement}$$

— le wavefront planner calcule la fonction de coût-to-go (fonction de valeur)
— il correspond à une résolution de type programmation dynamique