RAMASSAGE DE DECHETS
Projet algorithmique des graphes

Benjamin DAYRES
Mohamed Taha SANDI
Samuel LANDEAU
Thomas COUTAYE

DFD
DFD

Table des matières

0.1 Introduction

0.1.1 Présentation du projet

Le projet à réaliser consiste à concevoir et programmer des algorithmes de ramassage pour un robot chargé de collecter K déchets répartis alétoirement sur une grille carrée de côté N. Le robot en question possède une vitesse de déplacement constante unitaire et peut changer de direction en tournant sur lui même, avec un temps de changement de direction dépendant de l'angle de rotation. Après avoir ramassé tous les déchets, le robot devra retourner à sa position initiale. Le monde pourra éventuellement contenir des obstacles ayant une forme rectangulaire. Dans ce cas là, le robot n'aura pas la possibilité de les traverser et devra chercher un chemin alternatif sans obstacles pour atteindre sa destination.

0.1.2 Objectif du projet

Le but du projet est de répondre aux trois questions suivantes :

- Comment pourra-t-on modéliser algorithmiquement les deux mondes où évoluera notre robot ?
- Quel algorithme nous permettra de resoudre de manière optimale notre problème ?
- Une fois que nous avons modélisé les deux mondes, comment pouvons-nous rendre l'utilisation de nos implémentations facile et accessible à d'autres personnes ?

0.2 Modélisation des deux mondes

Pour modéliser le monde sans obstacles, on utilisera un graphe complet dont le nombre de vertices est égal aux nombre de déchets présents sur la grille plus notre position initiale. Ajoutons à cela que chaque arête de notre graphe portera un poids qui n'est autre que la distance euclidienne entre ses deux extremités.

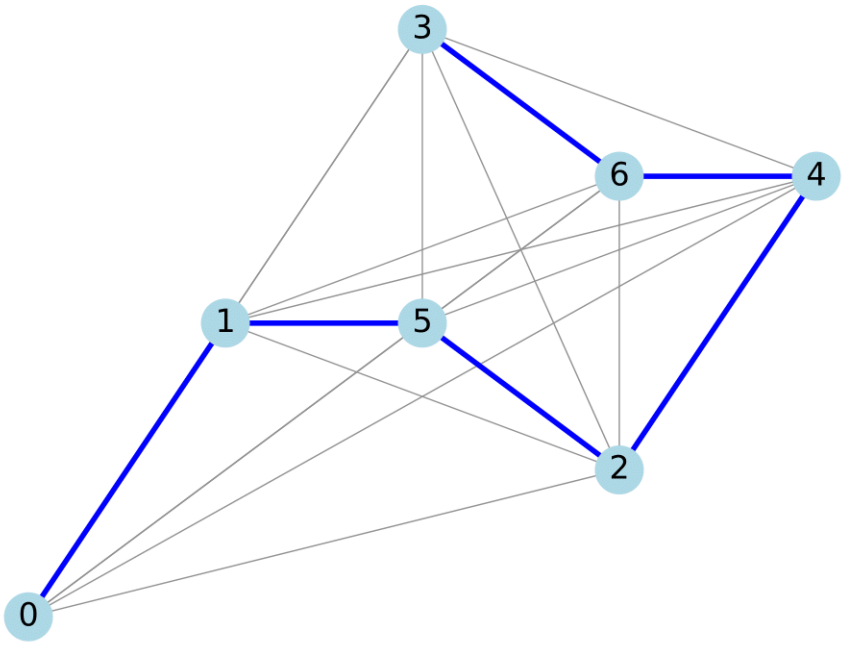

Figure 1: Exemple de graphe représentant un monde sans obstacle

0.3 Traduction du problème en graphe et représentation de ceux-ci

0.4 Nos algorithmes solutions

Afin de résoudre ce problème de manière optimal nous nous sommes posé la question suivante : Comment trouver le chemin de rammassage optimal ?

Une solution évidente est de calculer tous les chemins possibles et de choisir le plus cour. Si les n dechets sont représenté par une suites sans répétition de nombre entre 1 et n - 1 et la position de départ par le sommet 0, on peut conclure qu'il faut calculer la longueur du chemin parcourus pour toute permutation de l'ensemble [1 ; n].

Ainsi nous définissons les fonctions auxiliaire suivantes :

[H] Un tableau contenant toutes les permutations de [1 ; n] allPermutations(n)

[H] Le temps que le robot prendra pour parcourir la permutation time = 0 currentVector = [0;1] i from 0 to longueur(permutation) - 1 time = time + poid(i, i+1) newVector = [pos[i][0] - pos[i+1][0] ; pos[i][1] - pos[i+1][1]] time = C * angle(currentVector, newVector) currentVector = newVector return time permLength()

[H] Le meilleur chemin pour rammasser les déchets permutations = allPermutations(n) bestPath = [] minLength = ∞ i from 0 to longueur(permutations) permLength(permutations[i]) $<$ minLength minLength = permLength(permutations[i]) bestPath = permutations[i] return bestPath bruteForce()

Cependant, on constate rapidement que cette approche à ses limites. En effet, le nombre de permutation pour un ensemble de taille n est $!n$. La complexité en temps et en mémoire de notre alogrithme est donc de $O(n)$.

Bien qu'il soit possible de réduire la complexité en espace en fonction de comment l'implémentation est fait dans un language données, en utilisant par exemple, un itérateur en Python, la complexité en temps

elle reste inchangé.

Afin de trouver un autre algorithme, nous nous sommes demandé s'il n'existait pas déjà un problème documenté auquel nous pourrions ramener le notre.

Et, en effet, il est possible de ramener ce problème du rammassage de dechets au problème connu du marchand de commerce.

Il existe de nombeurs algorithm permettant de résoudre celui-ci, et nous nous sommes interressé à l'algorithme de Christofides.

L'algorithme de Christofides est une donne une approximation de la solution du marchand de commerce.

Nous avons décidé d'utiliser une variante de celui-ci. En effet, l'algorithme de Christofides consiste à chercher l'arbre couvrant de poid minimum d'un graphe, puis d'effectuer un traitement sur celui-ci.

Cependant, nous allons simplement effectuer un parcours en profondeur de celui-ci, et ajouter les sommets dans l'ordre de ce parcours à notre tourné.

Cependant, la solution obtenue n'est pas très satisfaisante et il est possible de très simplment améliorer ce résultat.

En effet, dans le parcours obtenue, certaines arrêtes peuvent être croisé, ce qui n'est pas optimal. Ainsi, en décroisant celles-ci, on obtient un bien meilleur résultat.