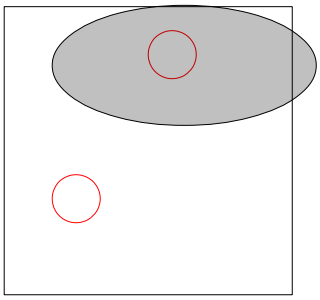
BERNARD Alexis
DEHOVE Simon
GREJON Thomas

1BG6
Trinôme 7



PROJET INFORMATIQUE

I. Première analyse.

Le but du projet est de programmer un outil pour calculer le chemin ayant le plus faible coût (ce qui correspond ici au temps de parcours).

La carte informatique est la donnée à traiter par notre programme. Elle est composée de différentes zones dans lesquelles la vitesse de déplacement est uniforme.

Pour calculer l'itinéraire au moindre coût nous utiliserons l'Algorithme de Dijkstra. Cet algorithme calcule le coût minimum entre l'origine et les points du réseau (il trouve le chemin au moindre coût pour relier des points via des arcs du réseau connus).

La recherche, par l'intermédiaire de l'algorithme, du plus court chemin nécessite la « transposition » de la carte en un réseau de nœuds et l'attribution de coût aux arcs reliant ces différents nœuds.

A ce niveau, deux possibilités se dégagent :

- définition de la carte sous la forme d'une matrice ou d'une grille, en utilisant un système de quadrillage avec des points (éléments de matrice ou intersection de droites) de coordonnées déterminées.
  Cette approche permet de déterminer aisément le chemin de proche en proche puisqu'il se fait par des déplacements « élémentaires » verticaux, horizontaux ou diagonaux (soit huit déplacements possibles pour chaque point). On obtient alors un réseau où chaque point est relié par des arcs de longueur 1 ou √2 à huit voisins et où le coût des arcs vaut le produit de cette longueur par la vitesse attribuée à la zone située entre ces deux points.
  Cette méthode comporte cependant un certain nombre de difficultés notamment la suivante : il est possible que pour relier deux points non contigus le déplacements ne puisse se faire en utilisant uniquement des déplacements élémentaires de proche en proche.

- définition de la carte par des zones dans lesquelles la vitesse est uniforme. Dans ce cas, le déplacement le plus court dans une zone se fait en ligne droite (à partir du moment où la zone n'est pas de forme convexe ce qui est toujours le cas si on prend le soin de scinder toutes les zones initialement convexes en deux ou plusieurs zones concaves).
  Par conséquent, si on prend un point parent (une origine) dans une zone, ses voisins sont tous les points des frontières de la zone.
  Le problème de cette méthode est qu'il y a une infinité de points appartenant aux frontières. Il est impossible de traiter l'infinité de ces points. Ainsi, la résolution de notre « problème » sera une approximation dont la précision dépendra de la « résolution » choisie. Il faudra donc au préalable définir la résolution du réseau comme le nombre de points appartenant aux frontières à inclure dans le réseau.
  De cette résolution dépendra la qualité de l'approximation mais aussi la vitesse d'exécution du programme.

II. Deuxième analyse.

A. Fabrication de la carte.

On construit une carte formée par des zones distinctes. Celles-ci sont délimitées par des segments.

- On place tous les points définissant les segments sur un graphique.

Fonction (coordonnées des points) => affichage des points dans la carte (sur le graphique)

- On relie les points formant les frontières sur le graphique.

Fonction (A,B) => [AB]

- On définit des zones « à l'intérieur » des segments tracés.

Fonction ([AB,BC,CA]) => zone 1 (triangle ABC)

- Pour chaque zone, on donne une vitesse de parcours.

Fonction (zone 1) => V1 = …

Problèmes :

- Peut-on se déplacer sur les frontières (on pense que oui à la vitesse la plus rapide des deux champs conjoints).
- La route sera-t-elle définie comme une suite de segments où l'on se déplace à grande vitesse, ou comme une zone classique ?
- On ne traitera pas le cas des zones non convexes car dans ce type de zone certaines frontières seront « cachées ».

Pas de déplacement direct possible entre A et la zone hachurée.

B. Transformation de la carte en réseau.

Le but de cette étape est de transformer la carte (infinité de points appartenant à des segments) en un réseau de points.
On utilise « une grille » qui par l'intersection avec les segments donne des points.

Pour chaque segment frontière, on cherche le coefficient directeur a :

- Si $-1 \leq a \leq 1$, on considère l'intersection entre le segment et les droites verticales de la grille.
  Fonction ([AB]) => a
- Si $a \in \mathbf{R} \setminus \{-1,1\}$, on considère l'intersection entre le segment et les droites horizontales.

⇨ Avant cette étape, on peut régler le nombre de points du segment et donc du réseau en changeant la résolution de la grille (influence sur le temps mais aussi sur la qualité de l'approximation).

C. Définition des coûts de déplacements

Le but de cette opération est de déterminer le coût des déplacements entre différents points.

- Calcul de la distance entre deux points.

Utilisation d'une fonction des variables A et B deux points qui à partir des coordonnées donne la longueur du segment [AB].

- Calcul du coût entre deux nœuds A_i et B_i du réseau appartenant à une même zone.

Détermination de la zone à laquelle appartiennent les points A_i et B_i puis utilisation d'une fonction des variables A_i, B_i et v_i (vitesse de déplacement dans la zone i) => cette fonction donnera le temps de parcours entre les deux nœuds ($t = \frac{d}{v} = \frac{A_i B_i}{v_i}$).

D. Utilisation de l'Algorithme de Dijkstra.

A partir du réseau et des coûts, on applique l'algorithme de Dijkstra.

E. Stockage des données.

Dans toutes les étapes de notre projet, nous avons stocké les données de la manière suivante :
Les matrices points qui stockent tous les points par couples de coordonnées (C, Ptszone, PtsZone_bis)
Les matrices définissant les zones qui regroupent les points appartenant à une zone par ligne. Ces points portent alors un nom (ou un indice). L'indice de ce point fait référence à sa position dans la matrice point. (New,

III. Réalisation du programme.

A. Première étape : réalisation de la carte.

B. Deuxième étape : transformation en réseau.

C. Troisième étape :

D. Quatrième étape : « raccordement de toutes les fonctions.