DELCOURT Louis
KAMILOGLU Emre

Rapport projet d'Algorithmique des graphes

---

1 – Construction du Graphe et de la Matrice des poids

Afin de pouvoir manipuler les 6 algorithmes demandés nous avons créé une fonction create_graph_alea qui prend en paramètre un entier et renvoie un graphe avec n points ((x,y) position).

Pour définir les poids entre les sommets nous avons défini une fonction matrice_poids qui prend en paramètre un graphe. Cette fonction utilise une fonction poids_arrete qui renvoie la distance entre deux points passés en paramètre. C'est une matrice des distances euclidiennes du graphe.

2 – Six algorithmes d'approximation

2.1 Ppvoisin

Nous avons implanté l'algorithme du plus proche voisin à l'aide de la fonction Ppvoisin.
Cette fonction prend en paramètre un graphe et sa matrice des poids ainsi qu'un sommet de départ.
A Partir de ce sommet, on cherche le sommet le plus proche, en prenant soin de bien noter les sommets déjà parcourus. On applique ce principe jusqu'à ce que tous les sommets soient parcourus.
A prendre en considération qu'il s'agit d'une approche basique et que le résultat n'est pas toujours optimal.

2.2 OptimisePpvoisin

Étant donné qu'il est possible d'améliorer le cycle obtenu par l'algorithme Ppvoisin (un algorithme glouton) nous avons donc défini une fonction OptimisePpvoisin qui prend en paramètre un cycle hamiltonien (celui obtenu via l'algorithme Ppvoisin). Pour chaque arête, on vérifie si elle se croise avec

une autre. Si c’est le cas, on vérifie que le décroisement est avantageux (donc que la longueur décroisée est bien plus courte que la longueur croisée). Si c’est le cas, on décroise les arêtes, sinon on ne fait rien. Un problème qui a lieu, c’est que le décroisement se fait également dans les cas désavantageux.
Le temps d’exécution est également assez élevé et peut prendre du temps pour les graphes avec beaucoup de sommets (sachant que le résultat n’est pas optimal).

2.3 Apminimum

Afin d’implanter l’algorithme glouton qui consiste à choisir, à chaque étape, l’arête de poids minimum qui ne referme pas prématurément le cycle tant que celui-ci ne passe pas par tous les sommets, nous avons crée une fonction Apminimum qui prend en paramètre un graph et un sommet de ce graphe. L’algorithme renvoie le cycle qui choisit les arêtes de poids minimum à partir
du sommet jusqu'à parcourir tout le graphe.

2.4 PvcPrim

Une autre approximation pour le PVC utilise l’algorithme de Prim. Nous avons donc défini une fonction Pvcprim qui prend en paramètres un graphe et un sommet . Le cycle hamiltonien du graphe qui visite les sommets de l’arbre couvrant de poids minimum construit par l’algorithme de Prim, dans l’ordre préfixe.

2.5 Esdemisomme

L’heuristique de la demi-somme va énumérer toutes les solutions du PVC pour trouver le chemin le plus optimisé, malgré un temps moyen d’éxecution plus, on est sûr de trouver la meilleure solution possible.

3 - Affichage

Afin d’effectuer une étude statistique cohérente nous avons recours à une fonction d’affichage qui prend en paramètres un graphe, sa matrice de poids et n le nombre de sommets de ce graphe.

On affichage le temps d’exécution des algorithmes au-dessus de leur graphe ainsi que la distance de chaque algorithme (c’est-à-dire le poids du chaque cycle hamiltonien) et un rapport de distance entre plus proche Voisin et son optimisation.
Une étude statistique sur les algorithmes est également affichée sur une base de 100 essais.