Projet — Flot de coût minimum

Pierre Pinet, Nicolas Baert

janvier 2023

1 Choix de l'algorithme

Nous avons choisi d'implémenter l'algorithme des plus courts chemins successifs (pccs).

On fait un pré traitement des graphes pour leurs ajouter un sommet super source et un sommet super puit et les arcs pour les connecter au graphe.

En conséquence dans le pccs on choisit toujours la super source et le super puit comme sommets en excès.

On utilise Dijkstra (version naïve) pour calculer les plus courts chemins dans le graphe résiduel car on sait que les coûts initiaux et les coût réduits sont positifs.

2 Usage du programme

Le code est répartit entre deux fichiers sources, graphe.cpp qui contient les définitions des nos classes (Graph et Arc) et de leurs méthodes, minCost.cpp qui contient la fonction main et les fonctions nécessaires au pccs.

L'exécutable compilé fait les tests sur les fichiers d'instances, il doit être mis dans le même répertoire que le dossier "instances".

Un test fait un pccs sur une instance, affiche la valeur de la solution, si elle est valide et le temps pris pour l'obtenir.

3 Faiblesses du code

Notre code fait la supposition que le graphe initial (avant la symétrisassion) n'est pas un multigraphe, c'est le cas des instances sur lesquelles nous avons fait les tests. Si un graphe est un multigraphe avant la symétrisassion alors il peut y avoir plusieurs arcs parallèles avec une capacité résiduelle positive, il faut alors choisir sur lequel on souhaite pousser du flot lorsque on doit pousser du flot entre les deux sommets. Notre méthode increaseFlow ne fait pas ça, elle pousse le flot sur le premier arc avec une capacité résiduelle positive (qui est toujours unique dans le cas de nos instances).

Pour stocker les arcs parallèles entre deux sommets on a opté pour un vecteur, ce dernier ne dépasse jamais deux éléments en taille sur nos instances mais il faut quand même utiliser une boucle pour le parcourir, cela alourdi fortement le code et laisse penser qu'il existe une meilleure technique.

À cause des arcs parallèles il faut retrouver par quels arcs Dijkstra a trouvé les plus courts chemins lorsque on veut pousser du flot le long d'un chemin donné par le vecteur des prédécesseurs, nous aurions du davantage modifier Dijkstra pour qu'il retourne aussi un vecteur d'arêtes.

4 Résultats

Nous avons testé notre algorithme sur toutes les instances fournies et tout les résultats (coûts et temps) sont conformes aux résultats attendus.