GALMICHE Mélody & BIDET Jérémy

2 Juin 2014

Projet Caterer

Documentation pour développeur

UNIVERSITE PARIS-EST MARNE-LA-VALLEE

Licence Informatique 3ème année

Structures de données utilisées

Pour la liste des arcs et sommets, l'interface List a été utilisé avec comme implémentation l'ArrayList.

Pour résoudre le problème une copie du graphe (et de ses arcs et sommets) est effectuée avant toutes opérations.

Des interface Set et Map ont été utilisés pour respectivement assurer l'ajout d'arcs et sommets sans doublons dans les listes (Set) et colorier le graphe pendant le parcours et la recherche du cycle (Map).

La classe statique Struct dans la classe Simplex a été implémentée pour simuler la suppression d'arcs dans la solution pendant la recherche du cycle, sans altérer donc la solution.

Choix des algorithmes

Recherche de la solution initiale :

Etape 0 : choix d'un sommet w tel que pour tout sommet s étant une source un arc a_{sw} existe, pour tout sommet p étant un puit un arc a_{wp} existe. Nous avons opté pour le choix arbitraire suivant : le sommet w sera le sommet du milieu du graphe.

Etape 1 : On ajoute tous les arcs mentionnés précédemment s'ils n'existent pas, en changeant leur coût : 0 pour ceux qui existaient déjà, et 1 pour les nouveaux qu'on appellera FakeArc dans le programme.

Etape 2 : On applique alors l'algorithme du simplex (expliqué ci-après) tant qu'il existe un arc e meilleur.

Vérification de la solution initiale :

Il s'agit d'un parcours simple de la liste des arcs, le graphe n'est pas parcouru, donc réduction de la compléxité. On recherche l'existence d'un FakeArc après application du simplex. Et si, existence, la valeur de son coût. On en conclu alors la fermeture du programme avec un message d'explication, ou la poursuite du programme.

Recherche d'une solution optimale :

Application du simplex tant qu'il existe un arc e (arc qui n'est pas dans la solution initiale) meilleur que l'arc de la solution touchant un sommet de e.

Simplex :

Etape 0 : on commence par séparer les arcs de la solution et les arcs du graphe d'origine qui ne sont pas dans la solution.

Etape 1 : on cherche le meilleur arc e qui n'est pas dans la solution. On le compare avec les arcs sortants du sommet où e entre. Si l'on rencontre une différence positive entre le coût de e et le coût

de l'arc sortant, on sauvegarde cette différence afin de la comparer avec la différence d'un autre arc e et d'un de ses arcs entrant. Et ainsi de suite, jusqu'à obtenir le meilleur.

Etape 2 : on ajoute cet arc à la solution. Cela crée un cycle dans la solution.

Etape 3 : recherche de l'arc f de la solution (avec f différent de e) le moins bon.

Etape 3.1 : cherche les sommets qui compose ce cycle.

Etape 3.1.1 : on crée une structure contenant pour chaque sommet le nombre d'arcs connectés à lui (sortant et entrant).

Etape 3.1.2 : on parcours le graphe en profondeur récursivement, pour chaque feuille (sommet ne contenant qu'un seul arc entrant ou sortant), on décrément de 1 le nombre d'arcs du sommet touchant cet unique arc. Ainsi on simule la suppression de cet arc dans le graphe, sans le supprimer réellement.

Etape 3.1.3 : a la fin du parcours la modification du nombre d'arc pour chaque sommet et remontée récursivement, ainsi à la fin de la fonction, il ne reste que des sommets avec au plus 2 arcs entrant ou sortant (au plus 2 car il n'y a qu'un seul cycle par itération du simplex). Les sommets avec 2 arcs font partie du cycle, les autres non.

Etape 3.2 : on « tri » le cycle ainsi obtenue par ses arcs. Avec pour premier arc, l'arc suivant e dans le sens de e, et pour dernier arc : e

Etape 3.3 : on cherche alors dans ce cycle l'arc f dans le sens opposé à e, qui aura le coût le plus fort.

Etape 3.4 : on met à jour les flots de chaque arcs, puis on supprime cet arc de la solution.

Etape 4 : on relance le simplex tant qu'on trouve au moins un arc e respectant les précédentes conditions.

Classes utilisées

- Arc : un arc est assigné à un graphe, il possède deux sommets (les extrémités), un coût et un flot.
- FakeArc : étend de Arc, mêmes attributs, mêmes méthodes, il s'agit d'un arc artificiel.
- Graph : un graphe possède un id, un nom (nom du fichier souvent), une liste de sommets et une liste d'arcs
- Vertex : un vertex (sommet) est assigné à un graphe. Il possède un vecteur de poids, un id (unique), un coût (lors de la résolution du problème), et une liste d'arcs qui entrent ou sortent de ce sommet.
- FilePattern : Interface définissant la méthode de parsing et les expressions régulière qui valide et parse le fichier
- GraphParser : implémente la méthode de parsing, et les sous-méthodes nécessaires à l'extraction des données.
- Simplex : définit l'algorithme de résolution du problème de réseau de transport.

- GraphException : définie un nouveau type d'exceptions, liés au graphe.
- Main : Lance le programme et la résolution du problème.

Complexités

Les tests précis de complexité n'ont pas été exécutée, mais elle est estimée approximativement en $O(n^2)$ avec n le nombre de sommets dans le graphe.