Université de Rouen
L3 Informatique
Makhloufi Khalil
Deltour Maxence
Diar Dia Modou

Semestre 1 , 2023
Projet d'Algorithmique des graphes

Quelques algorithmes d'approximation pour le problème du voyageur de commerce

Introduction

En informatique, le problème du voyageur de commerce, ou problème du commis voyageur, est un problème qui consiste à déterminer, étant donné une liste de villes et les distances entre toutes les paires de villes, le plus court-circuit qui passe par chaque ville une et une seule fois.
Le travail proposé par ce projet est d'implémenter en Python quatre de ces algorithmes d'approximation et de faire une étude statistique des résultats obtenus.
Ce problème étant un problème NP-complet, ces algorithmes ne renvoient pas forcéement un résultat optimal, encore une fois ce sont des fonctions d'approximation.

Choix de programmation

Le projet se décompose globalement en trois parties: la phase d'implémentation des fonctions correspondant aux algorithmes demandés, la réalisation de plusieurs fonctions outils dont nous nous servons dans ces dernieres fonctions et l'affichage.

L'algorithme du plus proche voisin

la fonction Ppvoisin : Elle prend en entrée un sommet s et renvoie la liste L correspondant au cycle hamiltonien obtenu à partir de s. Elle procède de la manière suivante:

1. Crée tout d'abord une liste contenant s,

2. Transforme le graphe en liste d'arêtes et le trie,

3. Effectue une boucle qui récupère, pour chaque dernier sommet enregistré, sont successeurs en fonction de l'arête à laquelle il est rattaché.

4. Retourne la liste des sommets L lorsque celle-ci à la même longueur que le nombre de sommets dans le graphe.

   Cette fonction utilise les fonctions outils takeValue et aretes_related qui respectivement renvoie la valeur du poids dans le triplet (s1,s2,p) et renvoie la liste des arêtes associée à un sommet.

Défaire les croisements

La fonction OptimisePpvoisin est une optimisation de la première fonction. Elle prend en entrée un cycle L, fourni par l'algorithme du plus proche voisin, et décroise, si le décroisement est avantageux, tous les couples d'arêtes envisageables, jusqu'à ce qu'il n'y ait plus aucun couple d'arêtes croisé non avantageux. Elle procède de la façon qui suit :

1. Compare le poids du cycle L à son cycle inverse.

2. Renvoie le cycle ayant le poids minimal.

Elle utilise les fonctions auxiliaires sommePoidsCycle et decroiserListe qui respectivement calcule la somme des poids du cycle

Utiliser l'arbre couvrant de poids minimum

Cet algorithme d'approximation pour le PVC utilise l'algorithme de Prim. Il construit un arbre couvrant (ACM) de poids minimum de G, en utilisant l'algorithme de Prim dans sa version la plus efficace, celle qui utilise un tas pour la gestion des sommets qui ne sont pas encore dans l'ACM courant. La fonction Pvcprim implemente cet algorithme. Elle prend en entrée un sommet s du graphe et donne en sortie le cycle hamiltonien du graphe qui visite les sommets de l'arbre couvrant de poids minimum. couvrant
Une fois l'arbre couvrant renvoyé nous faisons appel à parcous_prefix afin de faire un parcours préfix sur cet arbre, qui représentera le cycle hamiltonien

L'heuristique de la demi-somme

Cette méthode consiste à énumérer toutes les solutions du PVC, de sorte 'à éliminer, au cours de l'algorithme, les solutions partielles qui ont le moins de chances de mener à une solution optimale. Il s'agit de faire un parcours de l'arborescence de toutes les possibilités. Cette méthode utilise l'heuristique de la demi-somme qui permet de calculer une borne avec laquelle exclure les solutions potentielles moins intéressantes. Nous parcourons alors l'arbre en cherchant une meilleure borne à explorer , tout en remontant les niveaux si la nouvelle borne ne garantis pas un cycle hamiltonien de poids minimum.

Temps d'éxécution raisonnable

1. **Ppvoisin :** à partir de 50 sommets, l'éxécution commence à prendre pas mal de temps (Plus des 3s).
2. **Apminimum et PvcPrim:** Algorithmes très rapides, ne semblent pas rencontrer de soucis quant au nombre de sommets.
3. **Esdemisomme :** Algorithme très lent, le temps devient déraisonnable à partir de 15 sommets.

Affichage

Concernant l'affichage, nous nous servons de la fonction éponyme, cette dernière affiche le graphe, ainsi que des barres pur une étude statistique, comme demandé dans l'énoncé, en fonction du temps d'execution.

Difficultés rencontrées

1. Difficultés lors de l'implémentationde Esdemisomme ,l'algorithme de branch and bound n'a pas été facile à écrire.
2. optimisations de Ppvoisin
3. Obtention d'un cycle hamiltonien avec Apminimum.