TP de Mathématiques Discrètes - 1
Sujet : Le problème du voyageur de commerce

Définition du problème

Le problème du voyageur de commerce est un problème classique d'optimisation : il s'agit de trouver la plus courte tournée permettant de visiter n villes et de revenir au point de départ en ne visitant chaque ville qu'une seule fois.

Ce problème se trouve généralement formulé dans le langage des graphes : on considère un graphe $G = (S, A)$ où S, les sommets du graphe, représentent les villes, et A, les arêtes, représentent les routes. Un cycle C à k sommets est alors un ensemble ordonné de k sommets $(s_0, s_1, \ldots, s_k)$ tels que $s_k = s_0$ et $(s_i, s_{i+1}) \in A$ pour $i = 0, \ldots, k-1$. La longueur d'un tel cycle est la somme des longueurs des arêtes qui le composent. Elle est noté $L(C)$. Un cycle qui relie tous les sommets une et une seule fois est appelé cycle hamiltonien. Dans la suite, on notera $n = |S|$ le nombre de sommets du graphe.

Le but du problème est de déterminer le cycle hamiltonien de plus courte longueur. Nous travaillerons avec certaines hypothèses :

- Nous supposerons que la distance utilisée pour fournir la longueur des arêtes est euclidienne, et qu'elle vérifie donc l'inégalité triangulaire :

$$d(a, c) \leq d(a, b) + d(b, c) \; \forall a, b, c \in S^3$$

- Nous supposerons de plus que le graphe n'est pas orienté : $\forall a, b \in S^2$, les arcs (a, b) et (b, a) existent et ont la même valuation $d(a, b)$.

- Enfin, nous supposerons que le graphe est complet, c'est-à-dire que chaque sommet est relié à tous les autres.

Justifions brièvement que le plus court cycle que nous recherchons existe : l'hypothèse de la complétude du graphe permet d'affirmer que des cycles hamiltoniens existent. En effet $(s_0, \ldots, s_n, s_0)$ en est bien un. L'ensemble des cycles hamiltonien $\mathcal{C}$ est donc non vide, et il est fini. On peut donc considérer l'élément de longueur minimale.

Visualisation

Nous disposons d'une applet java qui permet d'afficher un parcours sous forme graphique. Le parcours affiché doit commencer par la ville 0. Le fichier DisplayTsp.html décrit le cycle à afficher et la classe DisplayTsp.class contient l'applet java qui permet l'affichage du cycle dans un navigateur web.

A rendre

Nous attendons de vous un petit rapport avec les différentes réponses aux questions. Pour celles nécessitant l'écriture d'une fonction, vous devrez insérer le code **commenté** de cette dernière (uniquement les fonctions principales) dans le rapport. A priori, il faut compter une à deux pages maximum pour les questions théoriques et une page max par fonction (5 sur la totalité du TP). Au total, tout doit tenir en 7 pages.

1 Résolution exacte

Une approche très naïve du problème consiste à déterminer la longueur de tous les cycles hamiltoniens, et à prendre le plus court. On se heurte cependant à une difficulté de taille : leur nombre. Dénombrons-les : se donner un tel cycle, c'est se donner un ordre de parcours des n villes. Il y en a $\frac{(n-1)!}{2}$, car si on considère un parcours donné, il revient au même de commencer à partir du premier sommet, où à partir du 2ème, ..., ou à partir du n-ième. De plus, il revient au même de parcourir le cycle dans un sens ou dans l'autre (d'où la division par deux).

Nous souhaitons pouvoir résoudre un problème à 250 villes. Il y a donc $\frac{249!}{2}$ cycles à tester. Sachant que $\frac{249!}{2} = 10^{490}$, il y a bien plus de cycles à tester que d'atomes dans l'univers.

Considérons l'ensemble des permutations de $[\![1, m]\!]$, dont la racine est le premier sommet, et dont la profondeur est de m. Les $(m-1)$ fils de la racine sont les sommets qui seront visités en deuxième position. Chacun d'entre eux a $(m-2)$ fils qui seront visités en troisième position, et ainsi de suite. Cet arbre à $m!$ feuilles, qui représentent des cycles hamiltoniens.

1. Ecrire une fonction PVC_EXACT_NAIF qui résout exactement le problème du voyageur de commerce (pour un nombre de villes modeste) en explorant systématiquement l'arbre des possibilités (en faisant un parcours en profondeur). Sa complexité est donc factorielle.

2. Une astuce permet de réduire notablement le temps d'exécution de cette fonction : certaines arêtes très longues ne figureront jamais dans des cycles courts. L'exploration de branches entières de l'arbre pourrait donc être évitée s'il s'avère qu'elle conduirait à des cycles plus long que le plus court cycle déjà identifié.

   Ecrire une fonction PVC_EXACT_BRANCH_AND_BOUND qui garde en mémoire la longueur du plus court cycle déjà identifié. Lors de l'exploration de l'arbre, si la somme des longueurs des arêtes conduisant du sommet actuel à la racine est plus longue que le plus court cycle connu, l'exploration de cette branche s'arrête : elle conduirait nécessairement à un cycle qui n'est pas le plus court.

3. Mesurer le temps d'exécution de votre fonction sur des problèmes de petite taille (inférieure à 15).

2 Résolution par approximation : méthode des plus proches voisins

En fait, à ce jour aucun algorithme ne fournit la solution du problème du voyageur de commerce en un temps qui serait une fonction polynomiale du nombre de villes n. Nous allons donc considérer des méthodes d'approximation, fournissant un court cycle hamiltonien en un temps raisonnable.

1. Ecrire une fonction PVC_APPROCHE_PPV qui consiste à prendre comme prochaine ville la plus proche parmi celles non encore visitées. Partant d'un sommet, on se rend au plus proche sommet qu'on n'a pas encore visité, et on réitère le processus. Une fois qu'on a visité tous les sommets du graphe, on retourne au premier pour fermer le cycle.

2. Quelle est la complexité de cette fonction ?

3 Résolution par approximation : un algorithme polynomial avec garantie de performance

On a une idée géniale : on se dit que l'on va exploiter le calcul d'un arbre couvrant de longueur minimale sur G pour en dériver un cycle hamiltonien.

1. Implémenter la fonction CALCUL_ACM qui calcule un arbre couvrant de poids minimum sur G.

2. Expliquer comment à partir de l'arbre couvrant de poids minimum de G, on construit un cycle sur G en empruntant chaque arêtes deux fois. Ce cycle passe au moins une fois par chaque sommet. On le note A.

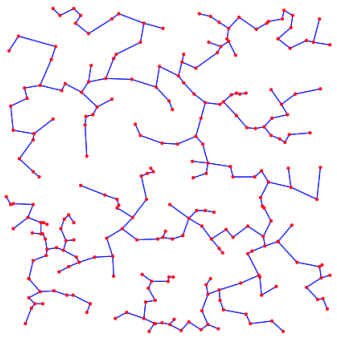

Figure 1: Un arbre couvrant de poids minimal (longueur : 10,40).

3. Proposer une procédure qui construit un cycle hamiltonien H à partir de A. Implémenter cette fonction sous le nom CYCLE_HAMILTONIEN_ACM.

4. Exprimer une relation liant $L(A)$ et $L(H)$

5. Considérons maintenant un cycle hamiltonien de poids minimum, et notons-le Opt. Exprimer une relation liant $L(Opt)$ et $L(A)$.

6. Commenter le résultat que vous obtenez sur les données.

4 Résolution par approximation : optimisation locale

Les solutions fournies par les deux méthodes précédentes sont peu satisfaisantes (on a l'impression qu'à la main on ferait mieux), car elles comportent soit des croisements soit de grandes arêtes. On va se pencher ici sur une méthode systématique et rapide pour améliorer ces solutions.

Tout d'abord, on va munir l'ensemble des cycles hamiltoniens $\mathcal{C}$ d'une distance d_{cycle}. Soit a et b deux cycles,

$$d_{cycle}(a,b) = n - \text{ (le nombre d'arêtes que a et b ont en commun)}$$

1. Montrer que d_{cycle} est une distance.

   L'idée est de partir d'une solution connue (par exemple, fournie par une des deux méthodes précédentes), et d'explorer systématiquement les cycles d'une sphère de rayon 2 autour de cette solution à la recherche d'un cycle qui serait plus court encore. Cette méthode s'appelle l'optimisation $2-Opt$. Notons $\mathcal{C}(a_0,2) = \{c \in \mathcal{C} \mid d_{cycle}(a_0,c) = 2\}$ la sphère de rayon 2 centrée en a_0.

   Partant d'un cycle a_0, les cycles qui sont dans $\mathcal{C}(a_0,2)$ sont exactement les cycles obtenus en "croisant" deux arêtes de a_0 : en effet, si l'on part de a_0 et que l'on modifie deux arêtes, pour que le résultat soit encore un cycle, la seule solution consiste à croiser les deux arêtes (voir figure 2).

2. Déterminer le cardinal de $\mathcal{C}(a_0,2)$.

3. Ecrire une fonction CYCLE_OPT_2 qui calcule le plus court cycle de $\mathcal{C}(a_0,2)$.

4. Une fois que l'on a déterminé le plus court cycle a_1 dans $\mathcal{C}(a_0,2)$, on peut recommencer le processus en considérant la sphère de rayon 2 centrée sur a_1 et ainsi de suite. Ecrire une telle procédure et la tester sur les données.

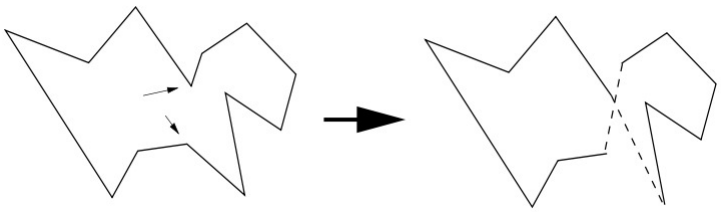

Figure 2: Croisement d'arêtes.

5 Analyse

1. Evaluer de manière expérimentale (temps, longueur du cycle) tous les algos précédents (naif, branch-and-bound, PPV, ACM, Opt2). Il s'agira par exemple de tracer des courbes de temps en faisant varier n, le nombre de villes (Attention, il faudra sans doute enveler la comparaison avec naif à partir d'un certain n_0).

2. Au delà des complexités asymptotiques (qui sont ici comparables), concluez sur l'approche la plus efficace (en temps, en mémoire, en qualité du résultat, ...).

Références

(1) Aupetit, A. Défi des 250 villes (http://labo.algo.free.fr/defi250/defi_des_250_villes.html). 2000.

(2) Charolois,R. Heuristiques pour le problème du voyageur de commerce (http ://eleves.ec-lille.fr/ charoloi/backtojava/projects/x/salesman.fr.html). 2001.

(3) Cormen, Th., Leiserson, Ch., and Rivest, R. Introduction à l'algorithmique, Dunod, 1997.