TECHNIQUES ALGORITHMIQUES ET PROGRAMMATION

Voyageur de commerce – Heuristiques

1 Flips et algorithme glouton

Étant donnée une tournée déjà construite, on considère l'heuristique consistant à supprimer deux arêtes indépendantes et à reconnecter les deux morceaux de tournée de façon à former un nouvelle tournée de longueur plus petite. On parle alors du "flip" des arêtes en question. Des arêtes indépendantes sont des arêtes qui n'ont pas d'extrémités en commun, donc qui ne se suivent pas dans la tournée. Le premier flip d'une tournée est un flip d'arêtes $v_i - v_{i+1}$ et $v_j - v_{j+1}$ où $i < j$ et où i est le plus petit possible.

Question 1. *Précisez le principe du flip des arêtes $v_i - v_{i+1}$ et $v_j - v_{j+1}$ avec $i < j$.*

Question 2. *Dans le cas où les arêtes indépendantes $v_i - v_{i+1}$ et $v_j - v_{j+1}$ se croisent sans être alignées, montrer que leur flip permet un gain strictement positif sur la longueur de la tournée. (Aide : considérer le point d'intersection et les quatre segments.)*

On suppose que vous avez déjà écrit la fonction `reverse(T,p,q)` qui renverse la partie du tableau d'entiers `T[p]...T[q]` (bornes comprises) si `p<q`.

Question 3. *Écrivez la fonction `first_flip(V,n,P)` qui renvoie le gain du premier flip réalisable (et réalise le flip) ou bien 0 s'il n'y en a pas. En déduire la fonction `tsp_flip(V,n,P)` qui calcule la tournée `P` (et renvoie sa valeur) en appliquant autant que possible les flips réalisables.*

Question 4. *Est-ce que vos fonctions terminent toujours ? Discuter de leur complexité.*

Question 5. *Écrivez la fonction `tsp_greedy(V,n,P)` donnant une tournée obtenue en choisissant à chaque fois le point libre le plus proche du dernier point atteint. Quelle est sa complexité en temps ?*

2 En TP

Téléchargez (*Enregistrer la cible du lien sous ...*) les fichiers correspondant au TP à partir de la page de l'UE disponible ci-après :

http://dept-info.labri.fr/~gavoille/UE-TAP/

Vous aurez à éditer `tsp_approx.c`, à décommenter quelques lignes dans `tsp_main.c`, et toujours compiler avec `make -B tsp`, et lancer l'exécution avec `./tsp`. Il faudra de plus :

1. Écrire `reverse()`, ajouter `first_flip()` et `tsp_flip()`.
2. Faites varier n grâce à la ligne de commande, jouez avec `generatePoints()` et `generateCircles()` de façon à mettre en défaut les heuristiques. Vous pouvez zoomer et déplacer les points avec la souris ! Observez qu'il peut y avoir des flips alors qu'il n'y a pas de croisements.
3. Ajouter l'heuristique `tsp_greedy()`. Comparer `tsp_flip()` seule avec `tsp_greedy()` suivit de `tsp_flip()`.