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 une 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 $v_0 - v_1 - \cdots - v_{n-1}$ 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, montrez 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, étant donnée une tournée `P` en entrée, renvoie le gain du premier flip réalisable (et réalise le flip dans `P` en sortie) ou bien 0 s'il n'y en a pas. (Aide : attention aux indices possibles pour i et j.)*

Question 4. *En déduire la fonction `tsp_flip(V,n,P)` qui calcule une tournée `P` (et renvoie sa longueur) obtenue en appliquant autant que possible les flips réalisables. Vous pourrez partir de la tournée de votre choix, par exemple, la tournée triviale définie par `P[i]=i`.*

Question 5. *Est-ce que vos fonctions terminent toujours ? Discuter de leurs complexités. Est-il possible d'obtenir un gain après un flip sans qu'il n'y ait initialement de croisement ? Si l'on applique un flip, peut-on obtenir plus de croisements qu'avant le flip ? Est-il possible d'avoir une séquence de flips où la même paire d'arête est flipée plusieurs fois ?*

Question 6. *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 choisi. 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, et mettez tout dans le même répertoire que la dernière fois :

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

Vous aurez à éditer `tsp_heuristic.c`, à décommenter quelques lignes dans `tsp_main.c`, et toujours compiler avec `make tsp_main`, et lancer l'exécution avec `./tsp_main`. Pour n'exécuter que la partie `tsp_heuristic.c`, vous pouvez déactiver les parties précédentes (brute-force et programmation dynamique) en ajoutant une typo aux définitions `#ifdef TSP_..._H` correspondantes dans le `tsp_main.c`. Si vos fonctions bouclent, pensez à ajouter un `(... && running)` dans tout test suspect, vous permettant de quitter plus simplement le programme.

Il faudra de plus :

1. Complétez `reverse()`, `first_flip()` et `tsp_flip()` du fichier `tsp_heuristic.c`.
2. Faites varier n grâce à la ligne de commande. Jouez avec `generatePoints()`, `generateCircles()`, `generateGrid()` de façon à mettre en défaut l'heuristique. (NB : Pour une grille avec au moins une des dimensions paires, la solution optimale ne peut pas contenir de segment en diagonal.) Vous pouvez zoomer et déplacer les points avec la souris, tester les touches `'o'`, `'r'` et `'t'` (voir `tools.h`). Observez qu'il peut y avoir des flips alors qu'il n'y a pas de croisements.
3. Ajoutez l'heuristique `tsp_greedy()`. Comparez l'heuristique `tsp_flip()` seule avec `tsp_greedy()` suivie de `tsp_flip()`. Pour cela modifiez `tsp_main.c`