TD heuristiques

Tournée de véhicules : méthode des « savings ».

A partir d'un entrepôt 0, une entreprise doit desservir des clients en produit. L'entreprise dispose d'une flotte de véhicules de capacité Q chacun et basés en 0 l'entrepôt. Chaque client c_i (i=1 à n) a demandé une quantité q_i de produit. La distance entre 2 clients c_i et c_j quelconques est d(i,j). Les distances sont supposées symétriques d(i,j)=d(j,i). On connait aussi la distance symétrique entre le dépôt 0 et chaque client c_i notée d(0,i).

L'entreprise doit desservir tous les clients avec ses véhicules (supposés en nombre illimité) en respectant la contrainte de capacité Q de chaque véhicule. Le but est de trouver une desserte qui minimise la distance totale parcourue par les véhicules.

On modélise le problème à l'aide du graphe complet dont les sommets sont les clients et le dépôt. Les sommets sont {0,1,2,…,n}. L'arête qui joint deux sommets i et j est munie du poids d(i,j).

On définit l'écartement de l'arête [i,j] par $e(i,j)=d(0,i)+d(0,j)-d(i,j)$ $1 \leq i<j \leq n$.

1-On suppose 2 tournées T_i et T_j l'une se terminant par le client c_i et l'autre commençant par le client c_j. Montrer que e(i,j) est l'économie faite en fusionnant T_i et T_j en une seule tournée (on remplace [i,0] et[0,j] par[i,j]).

Cette remarque est à la base de l'algorithme de Fletcher dite méthode des « savings ».

Algorithme.

- On classe les arêtes par ordre des écartements décroissants. Soit la L la liste ordonnée des arêtes.
- On construit les tournées T_i contenant chacune un seul client c_i (i=1 à n).
- Tant que L non vide
    - Soit $[i,j] \in L$ l'arête d'écartement maximum. On note T_i (resp T_j) la tournée contenant c_i (resp. c_j).
    - **Si** c_i et c_j ne sont pas dans la même tournée
    - **Si** c_i (resp. c_j) est l'une des extrémités de T_i (resp. T_j)
    - **Si** la fusion de T_i et T_j ne viole pas la capacité Q du véhicule
    - **Alors** fusionner T_i et T_j (on remplace [i,0] et[0,j] par[i,j])
    - retirer [i,j] de L
- fin tant que

2-Appliquer cet algorithme à l'instance suivante de n=5 clients :

Distances (km)	1	2	3	4	5
0	18	7	10	22	20

1		15	7	13	11
2			6	22	6
3				19	4
4					20

Demandes (tonne)	Client 1	Client 2	Client 3	Client 4	Client 5
	6	4	4	6	5

Capacité véhicule Q=14 tonnes

3-Les 2 arêtes en tête de L peuvent être permutées car elles ont même écartement. Refaire tourner l'algorithme en changeant l'ordre des arêtes de tête. Que constate-t-on ?

Note : on trouve une distance totale parcourue de 80 ou 103 km selon l'ordre de la tête de L. 80 est la valeur optimale. L'algorithme de Fletcher est une heuristique gloutonne qui donne de bonnes solutions mais pas forcément l'optimum.

4-Etant donnée une solution S (ensemble de tournées) on note C(S) le coût total des tournées et E(S) la somme des écartements des arêtes [i,j] de S $i \neq 0$ et $j \neq 0$.

Vérifier sur les solutions S trouvées aux questions 2 et 3 que $C(S)+E(S)=2\Sigma_{i=1,\ldots,n}d(0,i)$. Généraliser.

Note : comme $2\Sigma_{i=1,\ldots,n}d(0,i)$ est une constante, on en déduit que chercher S qui minimise C(S) revient à chercher S qui maximise E(S). Ce qui justifie le choix de l'algorithme de prendre les arêtes en commençant par celles de plus grand écartement.

Comparaison de 2 types de voisinage pour le voyageur de commerce (TSP).

On considère le TSP défini sur n=8 villes avec les distances (symétriques) inter-villes suivantes :

distance	1	2	3	4	5	6	7	8
1	x	2	4	4	1	4	4	1
2		x	1	4	3	1	4	4
3			x	1	3	3	4	4
4				x	1	4	4	4
5					x	2	4	4
6						x	1	3
7							x	1
8								x

On étudie 2 types de voisinage : le 2-opt et la permutation de 2 villes.

On part du tour T_0=[123456781].

1-Faire un mouvement 2-opt sur les arêtes 1-2 et 5-6 c'est-à-dire remplacer 1-2 et 5-6 par 1-5 et 2-6. Quel est le tour obtenu et sa valeur ? Quel est le gain par rapport à T_0 ?

2-On note T_1 le tour obtenu précédemment. Montrer que pour obtenir T_1 à partir de T_0, il faut faire 2 permutations successives de villes.

3-Si on considère le voisinage de T_0, $V_{perm}(T_0)$, comme les tours que l'on peut atteindre avec 1 permutation de 2 villes, montrer que T_0 est un minimum local strict i.e. que $\forall T \neq T_0 \in V_{perm}(T_0)$ coût(T)>coût(T_0). Qu'en est-il si on considère le voisinage construit avec les mouvements 2-opt ?

Coloration des sommets d'un graphe.

Un graphe (non orienté) G=(X,E). La coloration des sommets de G consiste à associer une couleur à chaque sommet de sorte que 2 sommets adjacents (reliés par une arête) aient 2 couleurs différentes. On veut colorier G avec un minimum de couleurs.

On considère l'algorithme A (glouton) suivant :

- On numérote les sommets $X=\{s_1,s_2,\ldots,s_n\}$.
- Pour i=1 à n
    - Affecter à s_i la plus petite couleur non encore affectée à ses voisins

1-Appliquer cet algorithme A aux graphes suivants :

- G=({s1,s2,s3,s4},{[s1,s2],[s2,s3],[s3,s4]})
- G=({s1,s2,s3,s4},{[s1,s3],[s3,s4],[s4,s2]})

2-On considère un graphe biparti complet fait de 2 paquets de sommets X1 et X2 : G=(X1$\cup$X2 , E). Il n'y a pas d'arêtes entre les sommets d'un même paquet X_i, par contre, il y a toutes les arêtes possibles entre X1 et X2.

Appliquer cet algorithme A aux graphes bipartis complets formés avec les paquets X1 et X2 suivants :

- X1={s1,s2,s3,s4} et X2={s5,s6,s7,s8}
- X1={s1,s3,s5,s7} et X2={s2,s4,s6,s8}

3-Dans le cas général, montrer que l'algorithme A colorie G avec un nombre de couleurs inférieur ou égal à $\max_{i=1,\ldots,n} \min \ \{d_i + 1, i\}$ où d_i est le degré du sommet s_i.

-Vérifier ce résultat sur les cas précédents.

-Avec ce résultat, déduire que l'on peut toujours colorier un graphe avec un nombre de couleurs inférieur ou égal à $d_{max}(G)+1$ où $d_{max}(G)$ est le degré maximum dans G.

4-On a des épreuves sportives a,b,c,…,h à organiser dans des gymnases. Un gymnase contient au plus une épreuve au même instant.

Les heures des épreuves sont indiquées ici :

épreuves	a	b	c	d	e	f	g	h
début	9H	10H	11H	12H	13H30	14H30	15H	8H
fin	11H30	12H30	14H	13H	15H30	16H	17H	10H30

On veut trouver le nombre minimum de gymnases pour organiser toutes les épreuves.

-Tracer le graphe des incompatibilités entre les épreuves : sommets=épreuves et une arête entre deux épreuves si elles ne peuvent avoir lieu dans le même gymnase.

Le problème revient maintenant à colorier le graphe des incompatibilités avec un minimum de couleurs.

-Numéroter les sommets (les épreuves) par ordre de dates de début croissantes. Appliquer l'algorithme A. Montrer que la coloration obtenue est minimale. En déduire le nombre minimum de gymnases.

-Le graphe que l'on vient de construire est un graphe d'intervalles. Montrer que pour un graphe d'intervalles, si on numérote les sommets (les intervalles) par ordre croissant des extrémités gauches des intervalles, l'algorithme A donne toujours le nombre minimum de couleurs.

Coloration des arêtes d'un graphe.

Soit un graphe non orienté G=(X,E). On veut colorier les arêtes de G de sorte que 2 arêtes adjacentes à un même sommet aient 2 couleurs distinctes. On veut un minimum de couleurs.

Le line-graphe L(G) d'un graphe G est le graphe dont les sommets sont les arêtes de G et il existe une arête entre 2 sommets de L(G) (2 arêtes de G) si et seulement si les 2 arêtes de G ont un sommet en commun.

1-Soit $G_{exemple}$=({s1,s2,s3,s4,s5} , {[s1,s2],[s2,s3],[s3,s4],[s1,s5],[s5,s3]}). Colorier les arêtes de G. Construire L($G_{exemple}$). Colorier les sommets de L($G_{exemple}$).

2-Montrer que colorier les arêtes de G est équivalent à colorier les sommets de L(G).

3-On note h le degré max dans G. Peut-on colorier les arêtes de G avec moins de h couleurs ?

-Montrer que le degré max de L(G) est inférieur ou égal à 2h-2.

-Si on colorie L(G) avec l'algorithme A vu plus haut, en déduire que l'on utilisera au plus 2h-1 couleurs.

-A partir de ce résultat, montrer que A est un algorithme 2-approché pour colorier les arêtes de G c'est-à-dire qu'il donne un nombre de couleurs pour les arêtes de G ne dépassant jamais le double du nombre de couleurs minimum.