Optimisation Combinatoire - Travaux Dirigés n° 1

— *Maîtrise d'Informatique* —

Arbres recouvrant de poids minimal

Lors de la phase de conception d'un circuit électrique, on a souvent besoin de relier entre elles des broches de composants électriquement équivalents. Pour interconnecter un ensemble de n broches, on peut utiliser $n-1$ câbles, chacun reliant deux broches, le choix d'une interconnexion utilisant une longueur de câble minmale est souhaitable.

On peut modéliser ce problème de cablage à l'aide d'un graphe non-orienté connexe $G = (S, A)$, où S représente l'ensemble des broches et A l'ensemble des interconnexions possibles entre les paires de broches et pour chaque arête $(u, v) \in A$, on définit un poids $w(u, v)$ qui spécifie le coût (la longueur de câble nécessaire) pour connecter u et v. On cherche alors à déterminer un sous-ensemble acyclique T des arêtes A du graphe qui connecte tous les sommets dans leur ensemble et dont le poids total

$$w(T) = \sum_{(u,v)\in T} w(u, v)$$

est minimal. Comme T est acyclique et connecte tous les sommets, c'est un arbre, appelé *arbre couvrant*. Une solution au problème défini précedemment porte le nom d'*arbre recouvrant de poids minimal*.

La suite présente deux algorithmes permettant déterminer un arbre couvrant de poids minimal : l'algorithme de Kruskal et l'algorithme de Prim. Ces deux algorithme illustrent une heuristique d'optimisation appelée stratégie "gloutonne". A chaque étape de l'algorithme, une option doit être choisie. La stratégie gloutonne effectue le choix qui semble le meilleur à l'instant où il est fait, sans revenir sur ce choix ultérieurement. Une telle stratégie n'aboutit généralement pas à trouver des solutions globalement optimales. Toutefois, dans le cas de la recherche d'un arbre couvrant de poids optimal certaines stratégies gloutonnes engendrent effectivement un arbre couvrant de poids minimal.

Algorithme de Kruskal

Pour déteminer un arbre couvrant de poids minimum d'un graphe connexe à n sommets, on sélectionne $n-1$ arêtes (le graphe ayant n sommets) en choisissant successivement une arête de poids minimal qui ne forme pas de cycle avec celles qui ont déjà été choisie.

L'algorithme de Kruskal se déroule en deux phases :

1. trier les arêtes par ordre de poids croissants

2. tant que l'on n'a pas selectionner $n-1$ arêtes,
   - examiner dans l'ordre obtenu après le tri, la première arête non examinée :
   si elle forme un cycle avec les arêtes précédemment selectionnées la rejeter,
   sinon la garder.

Algorithme de Prim

Le principe de l'algorithme de Prim est le suivant:
- on sélectionne $n-1$ arêtes du graphe de telle sorte que

1. à chaque étape, on étend un arbre couvrant une partie des sommets du graphe, en atteignant un sommet supplémentaire
2. on choisit pour cela l'arête de poids minimal parmi celles qui joignent l'ensemble des sommets déjà couverts à l'ensemble des sommets non encore couverts.

▶ **Exercice 1** Une compagnie souhaite acheminer du gaz à partir de Geiser Town en direction de 9 autres villes, elle cherche pour cela à déterminer le plan du circuit de distribution le plus court possible permettant à toutes les villes d'être alimentées en gaz. Les distances séparant les villes sont indiquées dans le tableau suivant

		2	3	4	5	6	7	8	9	10
Geyser Town	1	8	5	9	12	14	12	16	17	22
Lakeside	2		9	15	17	8	11	18	14	22
Mountainview	3			7	9	11	7	12	12	17
Riverview	4				3	17	10	7	15	18
Twin Streams	5					8	10	6	15	15
Snowdania	6						9	14	8	16
Skier's Paradise	7							8	6	11
Park City	8								11	11
Rolling Downs	9									10
Tumbling Basin	10									

On donnera plusieurs solutions.

▶ **Exercice 2** Trouver un arbre couvrant de poids minimal pour le graphe à 11 sommets dont le poids des arêtes est donné dans la table suivante

	2	3	4	5	6	7	8	9	10	11
1	3	4	9	8	7	6	5	4	20	15
2		16	19	20	4	12	14	17	8	3
3			13	1	2	5	9	14	15	20
4				16	4	9	7	6	11	12
5					13	3	14	5	9	19
6						6	7	9	3	12
7							4	5	19	6
8								7	9	19
9									10	15
10										20

Le problème du voyageur de commerce

Etant donné un graphe non orienté complet $G = (S, A)$ dont chaque arête a un coût positif ou nul $w(u, v)$, on cherche à déterminer un *cycle hamiltonien* (c'est-à-dire un cycle qui passe une fois et une seule par chaque sommet du graphe) de coût minimal.

Dans de nombreuses situations pratiques, il est moins coûteux d'aller directement d'un sommet u à un sommet v que de passer par un sommet intermédiaire t. Ce cas de figure se formalise en disant que la fonction de coût w vérifie *l'inégalité triangulaire*, autrement dit : pour trois sommets quelconques $u, v, t \in S$

$$w(u, v) \leq w(u, t) + w(t, v).$$

Le problème du voyageur de commerce étant NP-complet, on est mené à rechercher des algorithmes donnant une solution approchée à ce problème.

▶ **Exercice 3** On peut en utilisant l'algorithme de Prim obtenir une solution approchée au problème du voyageur de commerce de la manière suivante

1. choisir un sommet qui servira de "racine"
2. construire un arbre couvrant minimal T à partir de la racine à l'aide de l'algorithme de Prim
3. établir la liste L des sommets visités lors d'un parcours préfixe de T
4. retourner le cycle hamiltonien qui parcourt les sommets dans l'ordre correspondant à la liste L

- En reprenant les données de l'Exercice 1, chercher une approximation du plus court parcours cyclique des différentes villes à partir de Geyser Town.
- Montrer que l'algorithme qui vient d'être décrit donne un algorithme d'approximation ayant une borne égale à 2 pour le problème du voyageur de commerce si la fonction de coût du graphe vérifie l'inégalité triangulaire.

▶ **Exercice 4** L'*heuristique du point le plus proche* permet de trouver une solution approchée avec une borne 2 au problème du voyageur de commerce. La méthode est la suivante :

1. on part d'un cycle trivial constitué d'un unique sommet
2. à chaque étape, on identitifie un sommet u n'appartenant pas au cycle et dont la distance à un des sommets du cycle est minimale. Soit v le sommet réalisant ce minimum,on étend alors le cycle en y incluant u juste après le sommet v.
3. on itère cette opération jusqu'à ce que tous les sommets se trouvent dans le cycle

En reprenant les données de l'Exercice 1, chercher une approximation du plus court parcours cyclique des différentes villes à partir de Geyser Town.