1 Formulation MTZ pour le voyageur de commerce

La formulation la plus facile à entrer pour le voyageur de commerce est celle proposée par Miller, Tucker et Zemlin, communément appelée MTZ.

On note $N = \{1, \dots, n\}$ l'ensemble des n villes à visiter. La distance entre deux villes i et j est notée c_{ij}.

$$
\begin{aligned}
\min \sum_{i \in N} \sum_{j \in N, j \neq i} c_{ij} x_{ij} & \\
\sum_{j \in N, j \neq i} x_{ij} = 1, & \quad \forall i \in N \\
\sum_{j \in N, j \neq i} x_{ji} = 1, & \quad \forall i \in N \\
u_1 = 1, & \\
u_i - u_j + (n-1) x_{ij} \leq n - 2, & \quad \forall i \neq 1, \forall j \neq 1 \\
x_{ij} \in \mathbb{N}, & \quad \forall i \in N, j \in N \\
u_i \in \mathbb{N}, & \quad \forall i \in N
\end{aligned}
$$

Question 1. Ecrivez la formulation MTZ pour le TSP.

2 Formulation flot pour le voyageur de commerce

On s'intéresse à une formulation flot pour le voyageur de commerce qui a été proposée par Picard et Queyranne [1]. Cette formulation a l'intérêt d'être de taille polynomiale, mais elle est plus faible que la formulation utilisant les sous-tours.

On rappelle le problème de voyageur de commerce. On dispose de n villes, et on connaît c_{ij} la distance entre chaque paire de villes i et j. On cherche un circuit passant une fois et une seule par chaque ville tout en minimisant la distance parcourue. Sans perte de généralité, on considére qu'on débute le tour par le sommet 1.

La formulation est basée sur un modèle de flot. On crée n^2 sommets. Chaque sommet (i, k) correspond à la ville i visitée en position k. On parle de graphe par niveaux : on dit que tous les sommets (i, k) de même k sont dans le même niveau.

En pratique, on ne crée pas explicitement les nœuds, ils apparaissent uniquement dans les contraintes de conservation du flot. Les variables correspondent aux arcs. On notera x_{ijk} la variable binaire qui prend la valeur 1 si on utilise l'arc allant du sommet (i, k) au sommet $(j, k+1)$ dans le graphe, 0 sinon. Emprunter cet arc signifie aller de i vers j en position k.

Par exemple, une solution valide pour la figure 2 est le chemin $(1,1), (3,2), (2,3), (4,4), (1,5)$ qui correspond dans le problème initial à visiter les villes dans l'ordre $(1, 3, 2, 4, 1)$. Dans cet exemple, on a $x_{131} = x_{322} = x_{243} = x_{414} = 1$ et les autres variables à 0.

Les contraintes sont les suivantes :

— on doit partir du sommet 1 (= on doit emprunter exactement un arc sortant de $(1,1)$ et aucun arc sortant de $(i,1)$ pour $i > 1$) ;
— on doit respecter en chaque sommet central (pour k allant de 2 à n) les contraintes de conservation de flot ;
— on n'a pas le droit de revenir au sommet 1 avant le dernier niveau du graphe ;
— on doit passer exactement une fois sur chaque sommet du graphe initial.

1. Picard, Jean-Claude and Queyranne, Maurice, The Time-Dependent Traveling Salesman Problem and Its Application to the Tardiness Problem in One-Machine Scheduling, Operations Research 26 (1), pp. 86–110, 1978

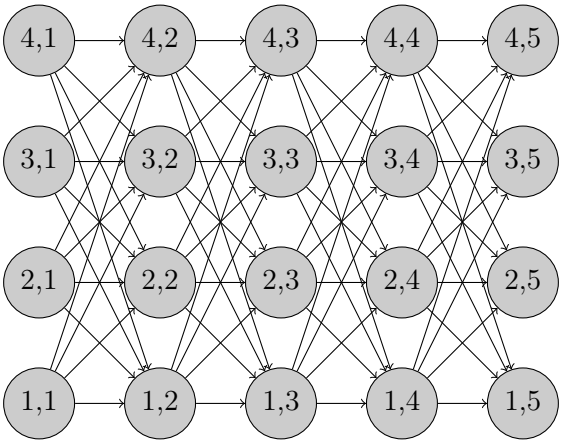

FIGURE 1 – Exemple de réseau pour 4 villes.

L'objectif est de minimiser $\sum_i \sum_j \sum_k c_{i,j} x_{i,j,k}$.

On remarquera que beaucoup d'arcs sont inutiles. Dans la première formulation, on peut les créer même s'ils ne sont pas utilisés dans la formulation. On pourra néanmoins ajouter la contrainte disant que chaque arc allant de (i, k) à $(i, k+1)$ est interdit (car il correspond à une boucle sur le sommet i).

2.1 Ecriture de la formulation

Question 2. Implémentez la version simple de la formulation.

Vous pouvez tester votre formulation sur les instances de la TSPlib (sur moodle) données dans un format simplifié.

2.2 Améliorations du programme

Comme indiqué précédemment, il y a beaucoup de sommets et d'arcs inutiles dans le graphe présenté.

Si on prend en compte qu'on doit partir de 1 et revenir en 1 à la fin, qu'on n'a pas le droit de revenir en 1 avant le dernier niveau, et qu'on n'emprunte jamais un arc allant de (i, k) à $(i, k+1)$, on obtient le graphe simplifié suivant.

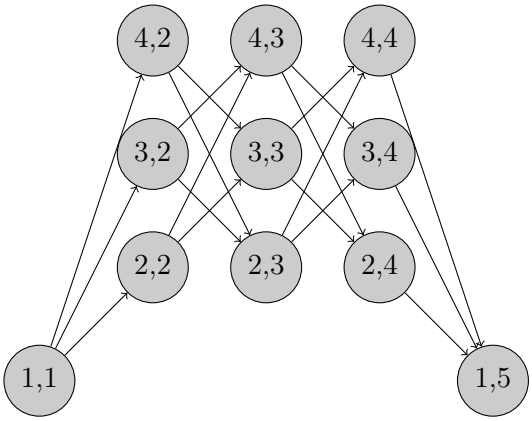

FIGURE 2 – Exemple de réseau simplifié pour 4 villes.

Question 3. Implémentez la version de la formulation correspondante, dans laquelle les variables inutiles ne sont pas utilisées.

2.3 User cuts

On peut améliorer la formulation en ajoutant des inégalités valides qui empêchent les sous-tours de longueur deux $(..., i, j, i, ...)$. Ces inégalités peuvent s'écrire de la manière suivante.

$$x_{i,j,k} \leq \sum_{\ell=1,\ldots,n,\ell\neq i} x_{j,\ell,k+1}, \quad i = 1,\ldots,n, \quad j = 1,\ldots,n, \quad k = 2,\ldots,n-2$$

Elle signifie que si on emprunte l'arc allant de (i,k) à $(j,k+1)$ alors le flot correspondant sortant de $(j,k+1)$ doit s'écouler sans passer par l'arc allant vers $(i,k+2)$.

Question 4. Améliorez la formulation ajoutant les inégalités valides proposées qui seront ajoutées uniquement lorsque c'est nécessaire par le biais d'un callback.

2.4 Comparaison des deux versions

Question 5. En mettant une limite de temps de une minute, comparez les performances des formulations avec et sans inégalités valides en termes de meilleure solution trouvée, meilleure borne, et gap sur les instances de la TSPlib (disponible sur moodle).

3 Formulation sous-tours

On rappelle la formulation du problème de voyageur de commerce, dans laquelle $\mathcal{S}$ est l'ensemble des sous-tours, c_{ij} le coût pour aller de i à j, et pour chaque paire (i,j), la variable binaire x_{ij} égale à 1 si on emprunte l'arc (i,j), 0 sinon.

$$\begin{aligned}
\min & \sum_i \sum_j c_{ij} x_{ij} \\
\text{s.t.} & \sum_i x_{ij} = 1, \quad j = 1,\ldots,n \\
& \sum_j x_{ij} = 1, \quad i = 1,\ldots,n \\
& \sum_{(i,j)\in S} x_{ij} \leq |S| - 1, \quad S \in \mathcal{S} \\
& x_{ij} \in \{0,1\} \quad i = 1,\ldots,n, j = 1,\ldots,n
\end{aligned}$$

Question 6. Ecrivez le modèle (sans la contrainte de sous-tours).
Question 7. Ecrivez la fonction pour détecter une contrainte violée de sous-tour (si la solution est entière).
Question 8. Utilisez cette fonction dans un callback pour obtenir une solution optimale pour le problème.
Question 9. Testez votre programme sur les instances fournies.
Question 10. Implémentez le callback permettant d'éliminer les sous-tours dans des solutions fractionnaires.