PROBLÈME DE ROUTAGE INCERTAIN MULTIOBJECTIF

L'évaluation du travail se fera sur la base d'un rapport au format pdf qui devra être déposé sur la page ENT du cours au plus tard le **12 janvier 2025**. *Il devra comprendre :*

- *une page de garde (avec nom et prénom de l'étudiant),*
- *une introduction et une conclusion,*
- *une description détaillée du travail réalisé,*
- *des résultats expérimentaux (avec une discussion),*
- *des commentaires (problèmes rencontrés, ce qu'il reste à faire, etc.),*
- *le listing commenté de vos programmes devra égalemenent être fourni.*

C'est un travail que vous devez réaliser en binôme (voire monôme). Si vous échangez vos travaux ou idées avec d'autres étudiants, assurez-vous que ces derniers les mentionnent comme étant les votres. Dans le cas contraire, votre note (et celle des autres étudiants impliqués) sera divisée par le nombre d'occurrences non-référencées de ces travaux ou idées, voire sera mise à zéro. Votre travail, et par conséquent votre note, doit refléter votre compréhension du cours et votre investissement dans le projet. Il est donc impératif que ces règles méthodologiques soient respectées :

- *mentionner toutes les sources utilisées,*
- *citer le nom de l'auteur quand une citation est utilisée et la mettre entre guillemets,*
- *citer le nom de l'auteur quand une idée vous est suggérée,*
- *ne pas faire de "copié-collé", mais rédiger vos propres phrases.*

Afin d'éviter toute plagiat, le logiciel anti-plagiat Magister, par Compilation.net, mis à disposition par l'Université Clermont Auvergne, sera utilisé. Il est important de savoir que ce logiciel permet de détecter le plagiat sur Internet et sur les documents qu'il a déjà analysés (par exemple, ceux de vos camarades ou ceux analysés dans d'autres cours).

1 Le problème de routage

Considérons un opérateur de réseau désirant louer une partie de son réseau pour le déploiement de réseaux privés virtuels. Le réseau de l'opérateur est représenté par un graphe orienté $G = (V, A)$ où V est l'ensemble des noeuds et A l'ensemble des arcs. Pour chaque arc $a \in A$, une capacité $c_a \in \mathbb{R}_+$ est installée et un coût de routage $w_a \in \mathbb{R}_+$ par unité de trafic est connu.

Un réseau privé virtuel est caractérisé par un ensemble $S \subseteq V$ de noeuds entre lesquels l'opérateur devra acheminer du trafic représenté par un vecteur $\mathbf{t} \in \mathbb{R}_+^{|S|(|S|-1)}$. La composante t_{ij} de $\mathbf{t}$ spécifie le volume de trafic que l'opérateur doit acheminer du noeud $i \in S$ vers le noeud $j \in S \setminus \{i\}$ via un ou plusieurs chemins de G.

L'opérateur a besoin de considérer des fonctions objectif antagonistes, d'une part celles liées à l'ingénierie de trafic (i.e., allocation des ressources) et d'autre part celles liées à la qualité de son service de réseau privé virtuel. Pour cela, il considère trois fonctions objectif :

1. le coût de routage (objectif lié à l'ingénierie de trafic),
2. l'utilisation maximum d'un arc (objectif lié à l'ingénierie de trafic) et
3. l'utilisation moyenne d'un arc (objectif lié à qualité de service).

L'objectif de ce projet est de déterminer le(s) chemin(s) de routage pour toutes les demandes de trafic spécifiées dans $\mathbf{t}$ (plus précisément les proportions de trafic passant par chacun de ces chemins) qui correspondent à des solutions optimales ou efficaces pour les trois fonctions objectif mentionnées ci-dessus.

2 Modèles linéaires

Pour chacune des trois fonctions objectif, donner un programme linéaire permettant de trouver les chemins de routage optimaux. Pour modéliser les chemins de routage, vous pourrez considérer les variables $x_a^{i,j}$ représentant la proportion du trafic associé à la composante t_{ij} de $\mathbf{t}$ passant par l'arc $a \in A$.

3 Génération d'instances

Afin de pouvoir tester et analyser les performances de vos algorithmes/programmes, écrire un programme permettant de générer

- des graphes appartenant à des classes données (e.g., complet, cycle, aléatoire, etc),
- les capacités associées aux arcs des graphes,
- les coûts de routage unitaire associés aux arcs des graphes,
- des sous-ensembles S de noeuds et les vecteurs de demandes de trafic $\mathbf{t}$ associés.

Vous pourrez utiliser NetworkX (`https://networkx.org/`) pour la génération des graphes. L'utilisation de NetworkX vous permettra d'utiliser ultérieurement, si besoin, des algorithmes dans les graphes (e.g., plus court chemin, flow maximum, etc.) sans avoir besoin de les implémenter. Chaque instance ainsi générée devra être stockée dans un fichier csv.

4 Résolution des programmes linéaires

Dans cette section, nous allons écrire un programme permettant de résoudre les programmes linéaires introduits à la section 2. Ce programme doit donc permettre de

- lire une instance du problème de routage créée à la section 3,
- créer le programme linéaire (mono-objectif) associé pour une des trois fonctions objectif qui sera passée en paramètre,
- résoudre le programme linéaire et
- afficher les solution et valeur optimales.

Pour ce faire, le programme devra s'interfacer avec le logiciel d'optimisation Gurobi 10.0.2 via son API Python. Pour une prise en main de Gurobi 10.0.2 et de son API Python,vous pourrez

1. vous référer aux tutoriels relatifs à la prise en main de l'API Python de Gurobi
   `https://support.gurobi.com/hc/en-us/articles/17278438215313`
   et
   `https://support.gurobi.com/hc/en-us/articles/17307437899025`,
2. saisir dans Visual Code Studio, comprendre, et excécuter le code Python accessible dans "Example Details" du deuxième tutoriel.

5 Expérimentations

Tester le programme de la section 4 pour

- des cycles,
- des graphes complets et
- des graphes aléatoires.

6 Obtention de solutions efficaces

Implémenter les méthodes

- d'optimisation lexicographique,
- d'ϵ-contraintes et
- de sommes ponderées

pour obtenir des solutions efficaces pour le problème de routage avec les trois fonctions objectif mentionnées dans la section 1. Reprendre les expérimentations de la section 5 pour trouver des chemins de routage efficaces.

7 Le problème de routage uncertain

Considérons de nouveau le problème de routage introduit dans la première section. Nous allons maintenant considérer la situation où les vecteurs $\mathbf{t}$ de demandes de trafic et les vecteurs $\mathbf{w}$ de coûts unitaires de routage ne sont pas explicitement connus. La seule information que l'opérateur possède est que les vecteurs $\mathbf{t}$ et $\mathbf{w}$ appartiennent à des ensembles d'incertitude $\mathcal{U}_T$ et $\mathcal{U}_W$, respectivement. En raison de contraintes opérationnelles, l'opérateur désire que les chemins de routage soient indépendants des valeurs futures que prendra le vecteur de demandes de trafic, tant que ce dernier appartient à $\mathcal{U}_T$.

Reprendre les modèles linéaires de la section 2 afin qu'ils prennent en compte l'incertitude, supposée par lignes, pour les ensembles d'incertitude

- $\mathcal{U}_T$ est un polytope,
- $\mathcal{U}_W = \{\mathbf{w} \in \mathbb{R}^{|A|} : \boldsymbol{\ell} \leq \mathbf{w} \leq \mathbf{u}\}$ où l_a et u_a sont des bornes inférieure et supérieure pour le coût de routage unitaire sur l'arc $a \in A$, respectivement.

Vous donnerez très précisément la contrepartie robuste sous sa forme la plus compacte possible.

8 Génération d'ensembles d'incertitude

Écrire un programme permettant de générer des ensembles d'incertitude pour les instances générées dans la section 3. Pour chaque instance, vous devrez obligatoirement générer des ensembles $\mathcal{U}_T$

- de type 'modèle hose' ou
- de type 'polytope de routage', c'est-à-dire où la matrice des coefficients correspond à une matrice de routage et le vecteur des membres de droite des contraintes correspond au volume de trafic maximal mesuré sur chaque lien.

Pour chaque instance, vous génèrerez également des ensembles $\mathcal{U}_W$ comme définis ci-dessus.

9 Solution de la contrepartie robuste

Modifier les progammes écrits dans les sections 4 et 6 afin qu'ils permettent de traiter la contrepartie robuste lorsque $\mathcal{U}_T$ est soit de type 'modèle hose' soit de type 'polytope de routage', et $\mathcal{U}_W$ est défini comme indiqué dans la section 7.

Tester vos programmes pour toutes les instances générées dans la section 8. Comparer les solutions obtenues avec celles que vous obtiendriez avec un vecteur $\mathbf{t^m} \in \mathbb{R}^{|S|(|S|-1)}$ de demandes de trafic où t^m_{ij} correspond à la plus grande valeur de t_{ij} parmi tous les vecteurs $\mathbf{t}$ de $\mathcal{U}_T$.