Compte rendu du TP

Filière F3 : systèmes d'information et aide à la décision

Evaluation et optimisation des systèmes

Problème de routage multiobjectif

Réalisé par : MAKOUAR ALI - KOUROU YOUNES

Professeur : Mr. KERIVIN HERVE

Campus des Cézeaux. 1 rue de la Chébarde. TSA 60125. 63178 Aubiére CEDEX

Table des matières

1 Introduction

Le succès opérationnel des réseaux dépend en grande partie de la manière dont le trafic est acheminé efficacement. Le routage approprié est crucial pour optimiser l'utilisation des ressources et garantir une qualité de service optimale. Ce projet vise à résoudre le problème complexe de routage dans le contexte spécifique d'un opérateur réseau. L'opérateur envisage de louer une partie de son infrastructure pour héberger des réseaux privés virtuels, ce qui nécessite une gestion efficace du trafic.

2 Le Problème de Routage

2.1 Contexte

Dans le contexte de ce projet, le réseau de l'opérateur est représenté sous forme de graphe orienté, où chaque nœud correspond à un point d'acheminement potentiel, et chaque arc représente une connexion entre ces points. Chaque arc est caractérisé par une capacité maximale et un coût de routage par unité de trafic.

2.2 Objectifs du Projet

L'objectif fondamental de ce projet est de définir les chemins de routage optimaux permettant de satisfaire différentes demandes de trafic. Cette optimisation doit tenir compte de trois fonctions objectif distinctes :

Coût de Routage : Minimiser le coût total de routage pour allouer les ressources de manière économique.

Utilisation Maximum d'un Arc : Minimiser l'utilisation maximale de chaque arc pour une ingénierie de trafic efficace.

Utilisation Moyenne d'un Arc : Assurer une utilisation moyenne minima équilibrée des arcs pour améliorer la qualité de service.

Ces objectifs concurrents nécessiteront une approche méthodique et l'application de modèles linéaires pour parvenir à des solutions optimales ou efficaces, conformément aux besoins opérationnels de l'opérateur réseau.

3 Modèles Linéaires (Suite)

3.1 Objectif 1 : Coût de Routage

Le premier objectif du projet vise à minimiser le coût total de routage. Pour atteindre cet objectif, nous formulons une fonction objectif linéaire qui prend en compte la somme pondérée des coûts de routage de chaque arc, en tenant compte des volumes de trafic associés.

$$\min \sum_{i \in S} \sum_{j \in S \setminus i} \sum_{a \in A} x_{ij}^{a} \cdot t_{ij} \cdot w_a \tag{1}$$

Cette fonction cherche à allouer les flux de trafic de manière à minimiser le coût total du réseau, en prenant en considération les poids spécifiques associés à chaque arc.

3.2 Objectif 2 : Utilisation Maximum d'un Arc

Le deuxième objectif est de minimiser l'utilisation maximale de chaque arc, favorisant ainsi une ingénierie de trafic plus efficiente. La fonction objectif correspondante est formulée comme suit :

$$\min \max \sum_{i \in S} \sum_{j \in S \setminus i} x_{ij}^{a} \cdot t_{ij} \tag{2}$$

Cette fonction cherche à répartir équitablement la charge de trafic sur tous les arcs, favorisant une utilisation équilibrée de l'ensemble du réseau.

3.3 Objectif 3 : Utilisation Moyenne d'un Arc

Le troisième objectif est d'optimiser la qualité de service en minimisant la maximisation de l'utilisation des arcs. La fonction objectif correspondante est formulée comme suit :

$$\min \sum_{i \in S} \sum_{j \in S \setminus i} \sum_{a \in A} x_{ij}^{a} \cdot t_{ij} \cdot \frac{1}{|A|} \tag{3}$$

Cette fonction cherche à minimiser l'utilisation maximale d'un arc, contribuant ainsi à une utilisation plus équilibrée des ressources du réseau.

3.4 Contraintes

Pour assurer la cohérence et la faisabilité du modèle, plusieurs contraintes doivent être respectées :

$$\textbf{Contrainte (1)} : \sum_{a \in \Gamma_{\text{out}}(i)} x_{ij}^{a} = 1 \quad \forall i \in S, j \in S \setminus \{i\} \tag{4}$$

Cette contrainte garantit que la somme des proportions de trafic sortant d'un nœud est égale à 1.

$$\textbf{Contrainte (2)} : \sum_{a \in \Gamma_{\text{in}}(j)} x_{ij}^{a} = 1 \quad \forall i \in S, j \in S \setminus \{i\} \tag{5}$$

Cette contrainte assure que la somme des proportions de trafic entrant d'un nœud est égale à 1.

$$\textbf{Contrainte (3)} : \sum_{a \in \Gamma_{\text{in}}(v)} x_{ij}^{a} = \sum_{a \in \Gamma_{\text{out}}(v)} x_{ij}^{a} \quad \forall i \in S, j \in S \setminus \{i, j\} \tag{6}$$

Cette contrainte équilibre le flux entrant et sortant de chaque nœud intermédiaire.

$$\textbf{Contrainte (4)} : \sum_{i \in S} \sum_{j \in S \setminus \{i\}} x_{ij}^{a} \cdot t_{ij} \leq c_a \quad \forall a \in A \tag{7}$$

Cette contrainte impose que la somme des flux à travers chaque arc ne dépasse pas sa capacité maximale.

3.5 Variables et Notations

x_{ij}^a représente la quantité de trafic routé de i à j via l'arc a. (8)

t_{ij} représente le volume de trafic entre le nœud i et le nœud j. (9)

w_a représente le poids associé à l'arc a. (10)

c_a représente la capacité maximale de l'arc a. (4)

$\Gamma_{\text{in}}(v)$ représente l'ensemble des arcs entrants au nœud v. (11)

$\Gamma_{\text{out}}(v)$ représente l'ensemble des arcs sortants du nœud v. (12)

(13)

4 Génération d'Instances

La génération d'instances de graphes revêt une importance capitale dans l'évaluation des performances des algorithmes de routage. Trois types de graphes distincts ont été considérés, chacun offrant des caractéristiques uniques pour simuler divers scénarios de réseaux. Chaque graphe est généré aléatoirement tout en respectant des propriétés spécifiques.

4.1 Génération d'un Graphe Complet Bidirectionnel

Nous avons créé un graphe complet bidirectionnel où chaque nœud est connecté à tous les autres nœuds du graphe, établissant ainsi des liaisons bidirectionnelles entre chaque paire de nœuds.

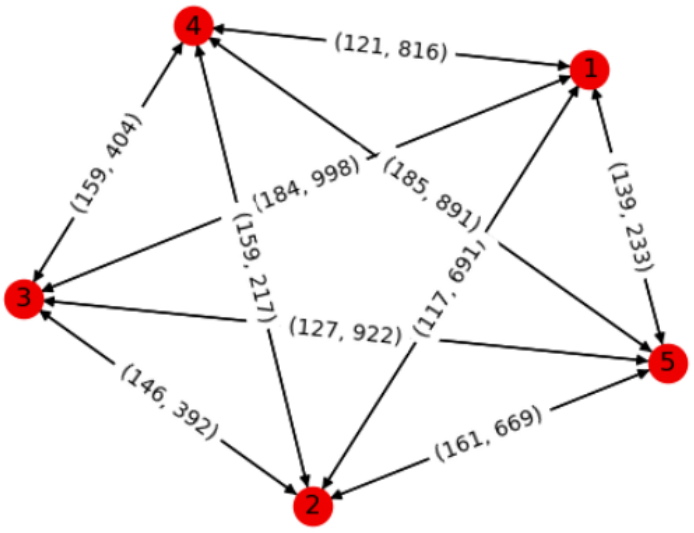

FIGURE 1 – Graphe complet bidirectionnel

Afin de simplifier notre test, nous avons attribué la même capacité et le même coût dans les deux sens entre chaque paire de nœuds. Cette décision permet de rendre les liaisons entre les nœuds réciproques de manière symétrique, offrant ainsi une représentation équilibrée des relations dans le graphe. Cette configuration est pertinente pour modéliser des réseaux denses avec une connectivité globale, où chaque nœud est directement lié à tous les autres.

4.2 Génération d'un Graphe Complet Mixte

Le deuxième type de graphe que nous avons généré est le graphe complet mixte, une variante du graphe complet qui intègre à la fois des arcs unidirectionnels et bidirectionnels. Cette combinaison de directions de liaisons offre une représentation plus nuancée pour modéliser des réseaux où certaines connexions sont asymétriques tandis que d'autres sont symétriques.

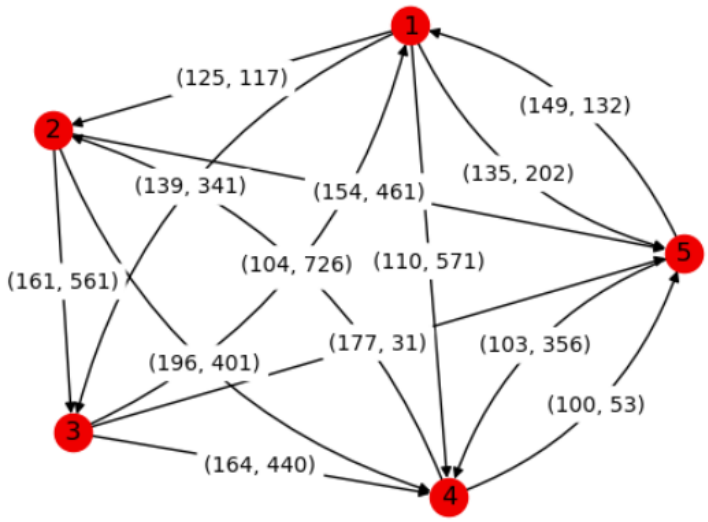

FIGURE 2 – Graphe complet mixte

La fonction de génération attribue des arcs avec des poids distincts pour chaque direction, que ce soit unidirectionnelle ou bidirectionnelle. Cette variabilité dans les caractéristiques des arcs contribue à une représentation plus réaliste des réseaux complexes, où les relations entre les nœuds peuvent être diverses et refléter des scénarios du monde réel.

4.3 Génération d'un Graphe Cyclique Unidirectionnel

Le troisième type de graphe que nous avons créé est le graphe cyclique unidirectionnel, caractérisé par des arcs orientés dans une seule direction. Chaque nœud est connecté uniquement à son successeur, créant ainsi une séquence linéaire de liaisons.

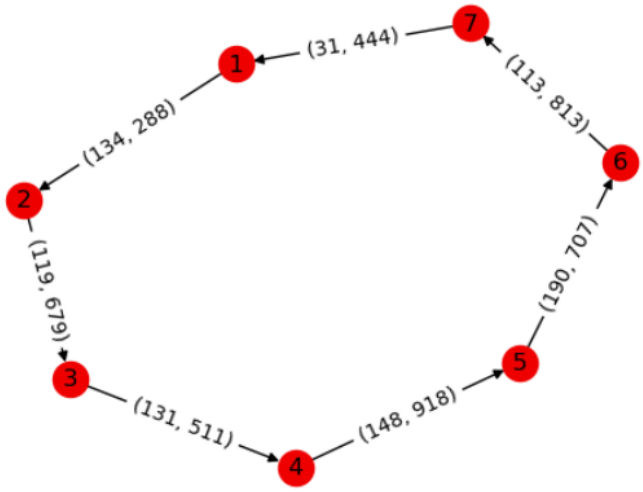

FIGURE 3 – Graphe cyclique unidirectionnel

Cependant, il est important de noter que dans des graphes cycliques unidirectionnels, où chaque nœud est connecté à son successeur immédiat, l'optimisation peut être limitée par la structure linéaire du graphe. Les contraintes de direction unique peuvent restreindre les possibilités d'optimisation par rapport à des graphes plus complexes,car il n'y a qu'un seul trajet possible pour aller d'un point à un autre. Ainsi, la décision d'optimiser un graphe cyclique unidirectionnel doit être guidée par les objectifs spécifiques du système modélisé.

4.4 Génération d'un Graphe Cyclique Bidirectionnel

Le deuxième type de graphe que nous avons créé est le graphe cyclique bidirectionnel. Ces graphes se caractérisent par une structure circulaire où chaque nœud est relié à la fois à son successeur et à son prédécesseur par des arcs bidirectionnels. Cette configuration permet de modéliser des réseaux avec des boucles potentielles, offrant ainsi une connectivité circulaire.

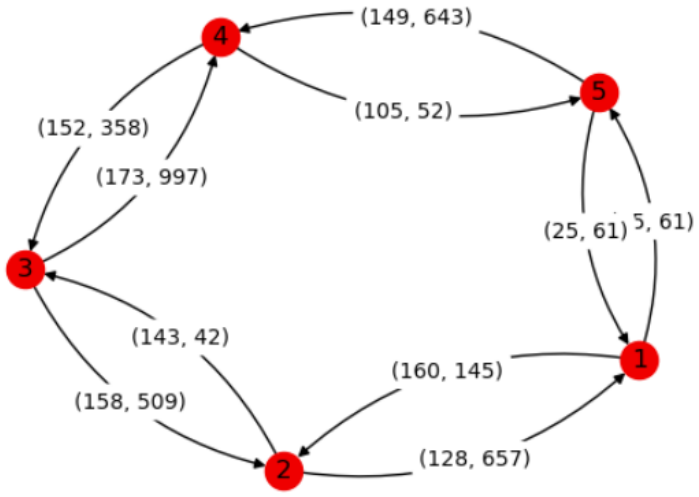

FIGURE 4 – Graphe cyclique bidirectionnel

La génération de ce graphe s'effectue en établissant des arcs bidirectionnels entre chaque nœud et ses voisins suivant et précédent. Cette structure particulière est particulièrement utile pour simuler des situations où des données doivent circuler de manière récurrente dans le réseau, créant ainsi des boucles d'information entre les nœuds. L'aspect bidirectionnel permet une flexibilité dans les connexions, offrant des possibilités de circulation des données dans les deux sens le long du cycle, ce qui peut être avantageux dans des contextes où la rétroaction ou la redondance sont essentielles.

4.5 Comparaison entre le graphe Cyclique Bidirectionnel et Unidirectionnel

Par exemple, dans l'exemple suivant, pour aller du nœud 4 au nœud 3, dans le graphe cyclique unidirectionnel, nous avons un seul trajet possible. Par conséquent, dans ce cas, aucune optimisation n'est possible en raison de la structure linéaire du graphe, limitant les choix de chemins. En revanche, dans le graphe cyclique bidirectionnel, deux trajets sont possibles pour aller du nœud 4 au nœud 3. Cette caractéristique offre une plus grande flexibilité dans le choix des chemins, ce qui peut être avantageux dans des contextes où la redondance ou la diversité des itinéraires est essentielle pour l'optimisation du réseau.

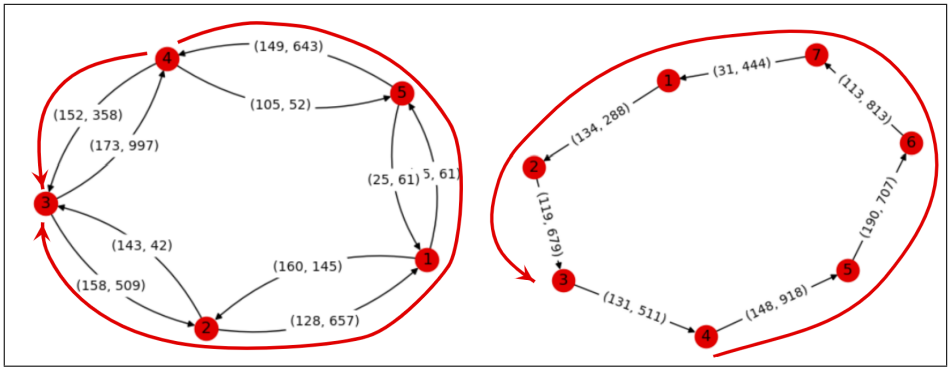

FIGURE 5 – Comparaison entre le graphe Cyclique Bidirectionnel et Unidirectionnel

5 Résolution des Programmes Linéaires

Pour résoudre les problèmes linéaires, l'approche consiste à mettre en œuvre et résoudre ces problèmes en utilisant Python avec l'API Gurobi.

Gurobi utilise des algorithmes sophistiqués tels que le Simplex pour les problèmes linéaires, le Barémoïque pour les problèmes quadratiques, et le Branch-and-Bound pour les problèmes mixtes entiers.

Pour résoudre les problème linéaire sous-contraintes avec Gurobi, on suit plusieurs étapes résumés dans l'algorithm ci-dessous :

Data: Données du problème linéaire
Result: Solution optimale

Étape 1 : Initialiser le modèle
model ← `Model()` ;

Étape 2 : Ajouter les variables de décision
for *chaque variable dans le problème* **do**
  x ← `addVar()` ;
end

Étape 3 : Définir la fonction objective
model.`setObjective()` ;

Étape 4 : Ajouter les contraintes
for *chaque contrainte dans le problème* **do**
  model.`addConstr()` ;
end

Étape 5 : Optimiser le modèle
model.`optimize()` ;

Étape 6 : Récupérer la solution optimale
solution ← *model*.`getVars()` ;

La solution consiste en l'ensemble des valeurs des variables qui minimisent (ou maximisent) la fonction objectif.

Dans notre problème, nous avons trois fonctions objectifs à résoudre : (1), (2) et (3). Soumisent aux même quatres contraintes : (4), (5), (6), (7).

Tout d'abord, nous allons résoudre chaque fonction objectif de manière indépendante, en appliquant les mêmes quatre contraintes à chacune. Il sera intéressant d'observer les résultats variés pour chaque type de graphe et de les comparer afin de parvenir à une conclusion pertinente.

Par la suite, nous résoudrons le problème en utilisant toutes les fonctions objectif (problème linéaire multiobjectif) avec trois méthodes qu'on verra dans la partie 7.

6 Expérimentations

Dans cette section, nous utiliserons le solveur Gurobi pour résoudre des problèmes linéaires mono-objectif, en testant les résultats pour différents types de graphes.

6.1 Graphe Complet Bidirectionnel

Nous débutons nos expérimentations en considérant un graphe complet bidirectionnel. Ce type de graphe est caractérisé par des arêtes entre chaque paire de

nœuds, permettant des déplacements dans les deux directions. Nous analysons comment le modèle réagit à la résolution de problèmes de routage sur ce type de topologie en utilisant différentes fonctions objectives.

1. **Minimisation du Coût de Routage :** Les résultats obtenus en minimisant le coût de routage sont illustrés dans la Figure.

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: 12th Gen Intel(R) Core(TM) i7-1250U, instruction set [SSE2|AVX|AVX2]
Thread count: 10 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 68 rows, 240 columns and 408 nonzeros
Model fingerprint: 0x71a7ee87
Coefficient statistics:
  Matrix range     [1e+00, 4e+01]
  Objective range  [5e+01, 4e+04]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+02]
Presolve removed 20 rows and 120 columns
Presolve time: 0.00s
Presolved: 48 rows, 120 columns, 270 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    9.7662000e+04   0.000000e+00   0.000000e+00      0s
       0    9.7662000e+04   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Optimal objective  9.766200000e+04
----------------------------------------------------
====================================================
----------------------------------------------------
****************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 97662.0
****************************************************
```

FIGURE 6 – Résultats pour la minimisation du coût de routage.

2. **Minimisation de l'Utilisation Maximale d'un Arc :** Les résultats obtenus en minimisant l'utilisation maximale d'un arc sont présentés dans la Figure .

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: 12th Gen Intel(R) Core(TM) i7-1250U, instruction set [SSE2|AVX|AVX2]
Thread count: 10 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 68 rows, 240 columns and 408 nonzeros
Model fingerprint: 0x53bf8189
Coefficient statistics:
  Matrix range     [1e+00, 4e+01]
  Objective range  [2e-01, 2e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+02]
Presolve removed 0 rows and 60 columns
Presolve time: 0.01s
Presolved: 68 rows, 180 columns, 360 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.3600000e+01   0.000000e+00   0.000000e+00      0s
       0    2.3600000e+01   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Optimal objective  2.360000000e+01
------------------------------------------------
================================================
------------------------------------------------
************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 23.599999999999994
************************************************
```

FIGURE 7 – Résultats pour la minimisation de l'utilisation maximale d'un arc.

3. **Minimisation de l'Utilisation Moyenne d'un Arc :** Les résultats pour la minimisation de l'utilisation moyenne d'un arc sont montrés dans la Figure.

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: 12th Gen Intel(R) Core(TM) i7-1250U, instruction set [SSE2|AVX|AVX2]
Thread count: 10 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 69 rows, 241 columns and 649 nonzeros
Model fingerprint: 0x33c56361
Coefficient statistics:
  Matrix range     [1e+00, 4e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+02]
Presolve removed 1 rows and 61 columns
Presolve time: 0.01s
Presolved: 68 rows, 180 columns, 360 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    4.7200000e+02   0.000000e+00   0.000000e+00      0s
       0    4.7200000e+02   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Optimal objective  4.720000000e+02
------------------------------------------------
================================================
------------------------------------------------
************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 472.0
************************************************
```

FIGURE 8 – Résultats pour la minimisation de l'utilisation moyenne d'un arc.

6.2 Graphe Complet mixte

Nous poursuivons nos expérimentations en examinant un graphe complet mixte. Dans ce type de graphe, les arêtes peuvent être soit bidirectionnelles, soit unidirectionnelles, introduisant ainsi une diversité supplémentaire dans la topologie du réseau.

1. **Minimisation du Coût de Routage :** Les résultats obtenus en minimisant le coût de routage sont illustrés dans la Figure.

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: 12th Gen Intel(R) Core(TM) i7-1250U, instruction set [SSE2|AVX|AVX2]
Thread count: 10 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 235 rows, 1650 columns and 1980 nonzeros
Model fingerprint: 0xcafab9f7
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [3e+01, 5e+04]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+03]
Presolve removed 134 rows and 1339 columns
Presolve time: 0.02s
Presolved: 101 rows, 311 columns, 656 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    3.0597300e+05   1.550000e+01   0.000000e+00      0s
      10    3.6478300e+05   0.000000e+00   0.000000e+00      0s

Solved in 10 iterations and 0.03 seconds (0.00 work units)
Optimal objective  3.647830000e+05
------------------------------------------------
================================================
------------------------------------------------
************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 364783.0
************************************************
```

FIGURE 9 – Résultats pour la minimisation du coût de routage.

2. **Minimisation de l'Utilisation Maximale d'un Arc :** Les résultats obtenus en minimisant l'utilisation maximale d'un arc sont présentés dans la Figure .

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: 12th Gen Intel(R) Core(TM) i7-1250U, instruction set [SSE2|AVX|AVX2]
Thread count: 10 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 235 rows, 1650 columns and 1980 nonzeros
Model fingerprint: 0xe6c77005
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [2e-02, 9e-01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+03]
Presolve removed 49 rows and 1009 columns
Presolve time: 0.02s
Presolved: 186 rows, 641 columns, 1170 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.6436364e+01   3.000000e+00   0.000000e+00      0s
       3    2.6436364e+01   0.000000e+00   0.000000e+00      0s

Solved in 3 iterations and 0.02 seconds (0.00 work units)
Optimal objective  2.643636364e+01
--------------------------------------------------
==================================================
--------------------------------------------------
**************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 26.436363636363645
**************************************************
```

FIGURE 10 – Résultats pour la minimisation de l'utilisation maximale d'un arc.

3. **Minimisation de l'Utilisation Moyenne d'un Arc :** Les résultats pour la minimisation de l'utilisation moyenne d'un arc sont montrés dans la Figure.

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: 12th Gen Intel(R) Core(TM) i7-1250U, instruction set [SSE2|AVX|AVX2]
Thread count: 10 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 236 rows, 1651 columns and 3631 nonzeros
Model fingerprint: 0x0bef5703
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+03]
Presolve removed 50 rows and 1010 columns
Presolve time: 0.02s
Presolved: 186 rows, 641 columns, 1170 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.4540000e+03   3.000000e+00   0.000000e+00      0s
       3    1.4540000e+03   0.000000e+00   0.000000e+00      0s

Solved in 3 iterations and 0.03 seconds (0.00 work units)
Optimal objective  1.454000000e+03
--------------------------------------------------
==================================================
--------------------------------------------------
**************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 1454.0
**************************************************
```

FIGURE 11 – Résultats pour la minimisation de l'utilisation moyenne d'un arc.

6.3 Graphe Cyclique Bidirectionnel

Nous poursuivons nos expérimentations en explorant un graphe cyclique bidirectionnel. Dans ce type de graphe, les nœuds sont disposés en cycle, et chaque nœud est connecté à ses deux voisins dans les deux directions, formant ainsi une boucle.

1. **Minimisation du Coût de Routage :** Les résultats obtenus en minimisant le coût de routage sont illustrés dans la Figure.

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: 12th Gen Intel(R) Core(TM) i7-1250U, instruction set [SSE2|AVX|AVX2]
Thread count: 10 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 58 rows, 120 columns and 204 nonzeros
Model fingerprint: 0xdf908780
Coefficient statistics:
  Matrix range     [1e+00, 4e+01]
  Objective range  [2e+02, 4e+04]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+03]
Presolve removed 39 rows and 88 columns
Presolve time: 0.01s
Presolved: 19 rows, 32 columns, 74 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    7.0600429e+04   1.857143e+01   0.000000e+00      0s
Extra simplex iterations after uncrush: 1
       7    1.3090659e+05   0.000000e+00   0.000000e+00      0s

Solved in 7 iterations and 0.02 seconds (0.00 work units)
Optimal objective  1.309065862e+05
------------------------------------------------
================================================
------------------------------------------------
************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 130906.58620689655
************************************************
```

FIGURE 12 – Résultats pour la minimisation du coût de routage.

2. **Minimisation de l'Utilisation Maximale d'un Arc :** Les résultats obtenus en minimisant l'utilisation maximale d'un arc sont présentés dans la Figure .

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: 12th Gen Intel(R) Core(TM) i7-1250U, instruction set [SSE2|AVX|AVX2]
Thread count: 10 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 58 rows, 120 columns and 204 nonzeros
Model fingerprint: 0x8c6b42ce
Coefficient statistics:
  Matrix range     [1e+00, 4e+01]
  Objective range  [2e-01, 5e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+03]
Presolve removed 39 rows and 88 columns
Presolve time: 0.01s
Presolved: 19 rows, 32 columns, 72 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    5.2400000e+01   0.000000e+00   0.000000e+00      0s
       0    5.2400000e+01   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Optimal objective  5.240000000e+01
------------------------------------------------
================================================
------------------------------------------------
************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 52.40000000000002
************************************************
```

FIGURE 13 – Résultats pour la minimisation de l'utilisation maximale d'un arc.

3. **Minimisation de l'Utilisation Moyenne d'un Arc :** Les résultats pour la minimisation de l'utilisation moyenne d'un arc sont montrés dans la Figure.

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: 12th Gen Intel(R) Core(TM) i7-1250U, instruction set [SSE2|AVX|AVX2]
Thread count: 10 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 59 rows, 121 columns and 325 nonzeros
Model fingerprint: 0x4afd2395
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+03]
Presolve removed 40 rows and 89 columns
Presolve time: 0.00s
Presolved: 19 rows, 32 columns, 72 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    5.2400000e+02   0.000000e+00   0.000000e+00      0s
       0    5.2400000e+02   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Optimal objective  5.240000000e+02
------------------------------------------------
================================================
------------------------------------------------
************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 524.0
************************************************
```

FIGURE 14 – Résultats pour la minimisation de l'utilisation moyenne d'un arc.

6.4 Graphe Cyclique Unidirectionnel

Les expérimentations sur le graphe cyclique unidirectionnel ont révélé une particularité intéressante. Malheureusement, toutes les fonctions objectifs considérées (minimisation du coût de routage, minimisation de l'utilisation maximale d'un arc et minimisation de l'utilisation moyenne d'un arc) ont généré des solutions infaisables pour le problème de routage.

```
************************************************
Le problème est infaisable.
************************************************
```

FIGURE 15 – Résultats pour toutes les fonctions objectives.

Une solution infaisable indique que le modèle n'a pas réussi à trouver une configuration valide pour le routage du trafic sur le graphe unidirectionnel selon les contraintes spécifiées. Cette observation suggère que la topologie particulière du graphe unidirectionnel peut présenter des défis uniques ou des contraintes qui rendent difficile la résolution du problème de routage tel qu'il est formulé.

7 Obtention de Solutions Efficaces

7.1 Optimisation Lexicographique

7.1.1 Explication de la méthode

La méthode d'optimisation lexicographique, également appelée préemptive, résout le problème linéaire mulobjetctif en le transformant en une séquence de problèmes d'optimisation linéaire mono-objectif. Chaque objectif est optimisé séquentiellement, en garantissant que la valeur optimale du premier objectif est atteinte avant de passer au suivant.

Schema de résolution pour $p = 3$ (le cas de notre porblème)
- Soit $w_1^* = \min\{c_1^T x : x \in X\}$
- Soit $w_2^* = \min\{c_2^T x : x \in X, c_1^T x \leq w_1^*\}$
- Toute solution optimale de $\min\{c_3^T x : x \in X, c_1^T x \leq w_1^*, c_2^T x \leq w_2^*\}$ est efficace pour le problème linéaire multi-objectif initial.

7.1.2 Implémentation

Pour mettre en œuvre cette méthode, nous avons d'abord résolu la première fonction objectif sous contraintes pour obtenir la valeur optimale w_1.

Ensuite, nous avons déterminé la valeur optimale w2 pour la deuxième fonc-

tion objectif en lui ajoutant la contrainte :

$$\sum_{i \in S} \sum_{j \in S \setminus i} \sum_{a \in A} x_{ij} \cdot t_{ij} \cdot w_a \leq w_1$$

Finalement, nous avons résolu la troisième fonction objectif en lui ajoutant deux contraintes :

$$\sum_{i \in S} \sum_{j \in S \setminus i} \sum_{a \in A} x_{ij} \cdot t_{ij} \cdot w_a \leq w_1$$

et

$$\max \sum_{i \in S} \sum_{j \in S \setminus i} x_{ij} \cdot t_{ij} \leq w_2$$

Pour déterminer w3 qui est la solution efficace du problème linéaire multiobjectif.

Les trois fonctions utilisées pour implémenter la méthode lexicographique sont ; resoudre_prob_lexicographique_step1, resoudre_prob_lexicographique_step2, et resoudre_prob_lexicographique. Ils opèrent de manière séquentielle. La première résout la première fonction objectif sélectionnée sous contraintes pour récupérer w1. La deuxième utilise cette solution pour déterminer w1, ajouter la nouvelle contrainte et retourner w2 en cas de succès.
Enfin, la dernière fonction permet d'obtenir une solution générale pour le problème multiobjectif en appelant successivement les deux premières fonctions.

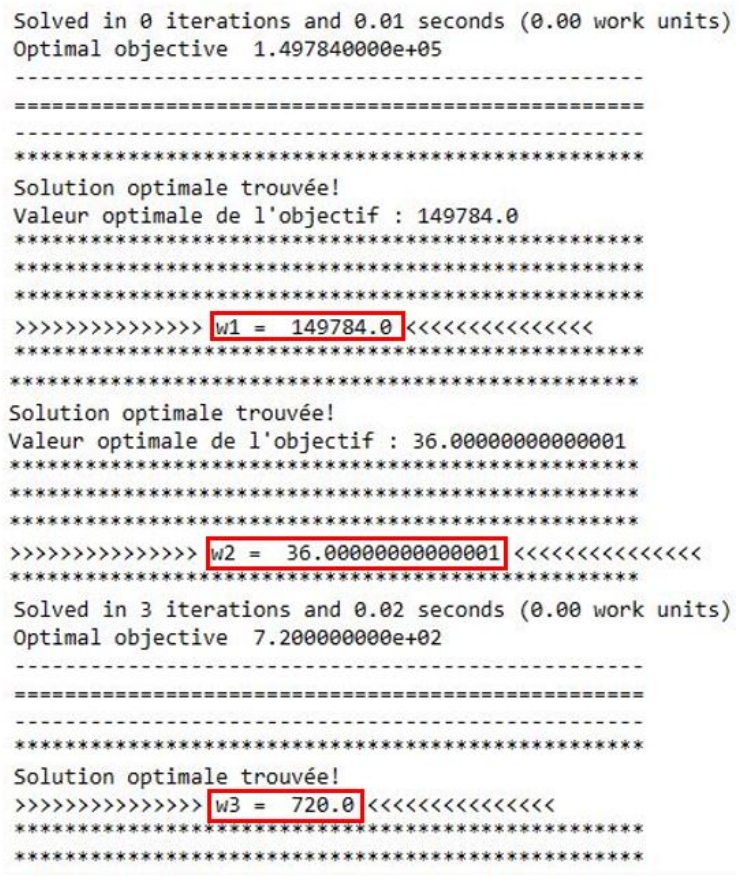

FIGURE 16 – Solutions par la méthode lexicographique

7.2 ε-Contraintes

7.2.1 Explication de la méthode

La méthode des ε-contraintes est une technique d'optimisation multicritère qui convertit un problème multicritère en plusieurs problèmes mono-objectifs. Chaque critère est traité individuellement en tant qu'objectif principal, tandis que les autres critères sont intégrés comme contraintes avec des seuils spécifiques (ε). L'approche génère un ensemble de solutions candidates, explorant ainsi le compromis entre les critères. En ajustant les valeurs des ε, on obtient une série de solutions représentant différentes compromis entre les objectifs.

Soit le problème linéaire multiobjectif suivant :

$$\min_{x \in X} Cx \tag{14}$$

où X :

$$X = \{x \in R^n : Ax = b, x \geq 0\}$$

La méthode par ε-contraintes ramène le problème (1) à un problème d'optimisation linéaire mono-objectif :

$$\min\{c_j^T x : x \in X, c_k^T x \leq \varepsilon_k \text{ pour } k \in \{1, \ldots, p\} \backslash \{j\}\} \quad (2)$$

avec $j \in \{1, \ldots, p\}$ et $\varepsilon_k \in R$ pour $k \in \{1, \ldots, p\} \backslash \{j\}$.

Cela revient à sélectionner une fonction objectif à optimiser, tandis que les autres sont incorporés sous forme de contraintes.

7.2.2 Implémentation

La première considération cruciale concerne le choix de la fonction objectif à résoudre. Dans notre approche, nous avons exploré trois cas distincts : sélectionner successivement la première, la deuxième, puis la troisième fonction. L'objectif est de comparer les résultats de chaque cas afin de tirer des conclusions éclairées pour le choix de la meilleure fonction objectif.

Parallèlement, le choix des valeurs d'epsilon est essentiel. Nous avons opté pour des valeurs dans l'ordre de grandeur des résultats obtenus lors de la résolution des différents problèmes considérés comme des cas mono-objectifs distincts, où chaque fonction objectif a été résolue sur les contraintes.

— **Choix 1** : trouver une solution à la première fonction objectif, tandis que les autres sont intégrées comme des contraintes. Pour $\varepsilon 2 = 100000$, $\varepsilon 2 = 5000$ on a :

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 71 rows, 241 columns and 890 nonzeros
Model fingerprint: 0xf65d0716
Coefficient statistics:
  Matrix range     [1e+00, 5e+04]
  Objective range  [1e-01, 2e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+07]
Presolve removed 1 rows and 24 columns
Presolve time: 0.01s
Presolved: 70 rows, 217 columns, 829 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    3.5999000e+01   9.000012e+01   0.000000e+00      0s
       1    7.5600000e+02   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.01 seconds (0.00 work units)
Optimal objective  7.560000000e+02
------------------------------------------------
================================================
------------------------------------------------
************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 756.0
************************************************
************************************************

756.0
```

FIGURE 17 – Solutions par ε-contraintes pour le choix 1

— **Choix 2** : trouver une solution à la deuxième fonction objectif, tandis que les autres sont intégrées comme des contraintes. Pour $\varepsilon 2 = 100000$, $\varepsilon 2 = 5000$ on a :

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 71 rows, 241 columns and 890 nonzeros
Model fingerprint: 0xf65d0716
Coefficient statistics:
  Matrix range     [1e+00, 5e+04]
  Objective range  [1e-01, 2e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+07]
Presolve removed 1 rows and 24 columns
Presolve time: 0.01s
Presolved: 70 rows, 217 columns, 829 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    3.5999000e+01   9.000012e+01   0.000000e+00      0s
       1    7.5600000e+02   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.01 seconds (0.00 work units)
Optimal objective  7.560000000e+02
--------------------------------------------------
==================================================
--------------------------------------------------
**************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 756.0
**************************************************
**************************************************

756.0
```

FIGURE 18 – Solutions par ε-contraintes pour le choix 2

— **Choix 3** : trouver une solution à la troisième fonction objectif, tandis que les autres sont intégrées comme des contraintes. Pour $\varepsilon 2 = 100000$, $\varepsilon 2 = 5000$ on a :

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 71 rows, 241 columns and 1129 nonzeros
Model fingerprint: 0x09407b6e
Coefficient statistics:
  Matrix range     [1e-01, 5e+04]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+07]
Presolve removed 1 rows and 25 columns
Presolve time: 0.00s
Presolved: 70 rows, 216 columns, 828 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    7.2000000e+02   0.000000e+00   0.000000e+00      0s
       0    7.2000000e+02   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Optimal objective  7.200000000e+02
--------------------------------------------------
==================================================
--------------------------------------------------
**************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 720.0
**************************************************
**************************************************

720.0
```

FIGURE 19 – Solutions par ε-contraintes pour le choix 3

7.3 Sommes Pondérées

7.3.1 Explication de la méthode

La méthode des sommes pondérées est une approche qui transforme le problème multicritère (1) en un problème d'optimisation linéaire mono-objectif de la forme :

$$\min_{x \in X} \sum_{k=1}^{p} \lambda_k c_k^T x \quad (3)$$

où C est la matrice des objectifs et λ est un vecteur de poids.

La méthode est facile à implémenter, permettant une recherche de solution réalisable avec $\lambda = 0$ et ignorant les cas où $\lambda \leq 0$ en raison de la définition de la non-dominance. Les solutions obtenues dépendent fortement des poids λ_k spécifiés avant l'optimisation. Bien qu'elle offre un meilleur équilibre des objectifs par rapport à la méthode préemptive, elle ne capture pas des préférences complexes.

7.3.2 Implémentation

Après avoir transformé le problème d'optimisation linéaire multiobjectif initial, nous cherchons à résoudre à l'aide de cette méthode le problème suivant :

$$\min \sum_{i \in S} \sum_{j \in S \setminus \{i\}} \sum_{a \in A} \lambda_1 x_{ij}^a t_{ij}^a w_a + \lambda_2 x_{ij}^a t_{ij}^a \frac{1}{|A|} + \lambda_3 \max\{x_{ij} t_{ij}^a\}$$

Dans le contexte de notre problème, une comparaison pertinente réside dans l'analyse des solutions obtenues lorsque les poids des objectifs sont équivalents (c'est-à-dire, $\lambda_1 = \lambda_2 = \lambda_3 = 1/3$) et dans les cas où les λ_k diffèrent.

Pour résoudre le prolème linéaire on utilisera l'algorithme suivant :

— **Cas** $\lambda_1 = \lambda_2 = \lambda_3 = 1/3$: le problème de routage a été résolu avec succès, aboutissant à **une valeur optimale de l'objectif de 246349**. Les coefficients équilibrés dans la fonction objective, attribués de manière uniforme (1/3) à chaque composante, ont permis d'obtenir une solution optimale. Cette dernière représente la meilleure combinaison de chemins de routage, optimisant la fonction objective tout en prenant en compte équitablement les différents critères pondérés.

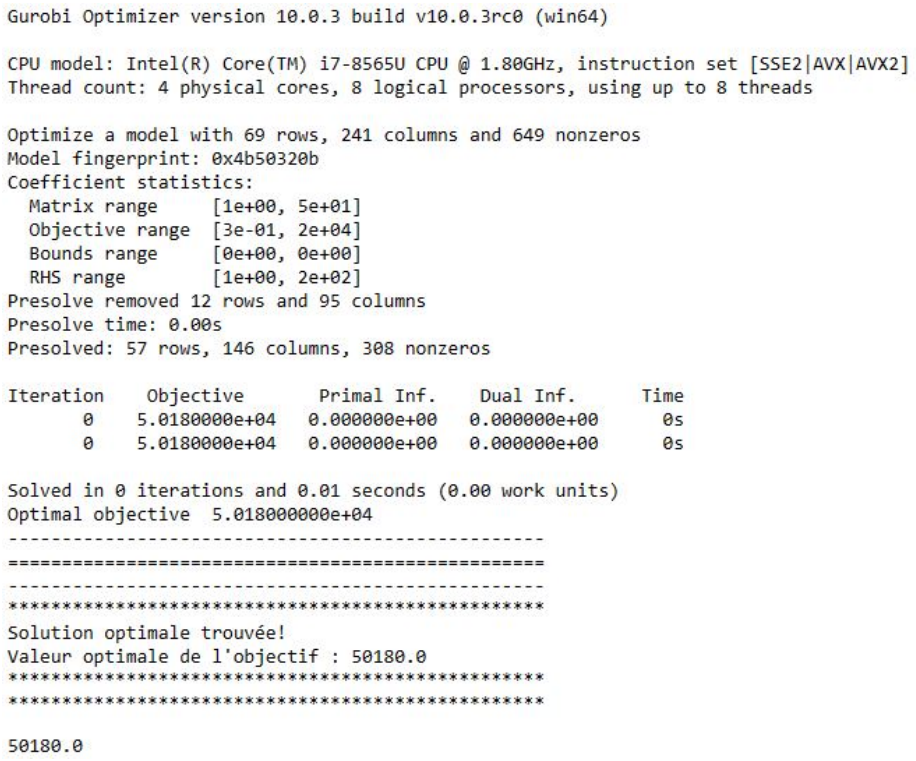

FIGURE 20 – Solutions avec la méthode de somme pondérée pour $\lambda_1 = \lambda_2 = \lambda_3 = 1/3$

— **Cas** $\lambda_1 = 0.1; \lambda_2 = 0.2; \lambda_3 = 0.7$: le problème a été résolu avec succès, atteignant **une valeur optimale de l'objectif de 74678.2**. Dans cette itération, les pondérations de la fonction objective étaient modifiées, accordant une plus grande importance (0.7) à la variable auxiliaire maxvar. Cette variation dans les poids indique une priorité accrue à la maximisation de maxvar, suggérant une allocation plus significative de certaines ressources ou mettant en évidence une caractéristique spécifique du problème que l'optimisation visait à accentuer.

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 69 rows, 241 columns and 649 nonzeros
Model fingerprint: 0xfe2429ad
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [7e-01, 5e+03]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+02]
Presolve removed 12 rows and 95 columns
Presolve time: 0.01s
Presolved: 57 rows, 146 columns, 308 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.5489600e+04   0.000000e+00   0.000000e+00      0s
       0    1.5489600e+04   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Optimal objective  1.548960000e+04
--------------------------------------------------
==================================================
--------------------------------------------------
**************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 15489.6
**************************************************
**************************************************

15489.6
```

FIGURE 21 – Solutions avec la méthode de somme pondérée pour $\lambda_1 = 0.1; \lambda_2 = 0.2; \lambda_3 = 0.7$

— **Cas** $\lambda_1 = 0.2; \lambda_2 = 0.7; \lambda_3 = 0.1$: le problème a été résolu avec succès, atteignant **une valeur optimale de l'objectif de 147746**. Dans cette itération, les pondérations de la fonction objective étaient ajustées, attribuant davantage de poids (0.7) à la deuxième composante de la fonction objective. Cette modification suggère que l'optimisation a accordé une priorité plus élevée à cette composante par rapport aux autres, ce qui peut refléter une importance accrue accordée à une caractéristique particulière du problème.

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 69 rows, 241 columns and 649 nonzeros
Model fingerprint: 0xc7558100
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [7e-01, 9e+03]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+02]
Presolve removed 12 rows and 95 columns
Presolve time: 0.00s
Presolved: 57 rows, 146 columns, 308 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    3.0464400e+04   0.000000e+00   0.000000e+00      0s
       0    3.0464400e+04   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Optimal objective  3.046440000e+04
--------------------------------------------------
==================================================
--------------------------------------------------
**************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 30464.399999999998
**************************************************
**************************************************

30464.399999999998
```

FIGURE 22 – Solutions avec la méthode de somme pondérée pour $\lambda_1 = 0.2; \lambda_2 = 0.7; \lambda_3 = 0.1$

— **Cas** $\lambda_1 = 0.7; \lambda_2 = 0.1; \lambda_3 = 0.2$: le problème a été résolu avec succès, atteignant **une valeur optimale de l'objectif de 516623**. Dans cette itération, les pondérations de la fonction objective étaient ajustées, attribuant davantage de poids (0.7) à la première composante de la fonction objective. Cette modification suggère que l'optimisation a accordé une priorité plus élevée à cette composante par rapport aux autres, indiquant une importance accrue pour une caractéristique spécifique du problème associée à cette composante.

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 69 rows, 241 columns and 649 nonzeros
Model fingerprint: 0x528b8039
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [2e-01, 3e+04]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+02]
Presolve removed 12 rows and 95 columns
Presolve time: 0.01s
Presolved: 57 rows, 146 columns, 308 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.0499640e+05   0.000000e+00   0.000000e+00      0s
       0    1.0499640e+05   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Optimal objective  1.049964000e+05
--------------------------------------------------
==================================================
--------------------------------------------------
**************************************************
Solution optimale trouvée!
Valeur optimale de l'objectif : 104996.4
**************************************************
**************************************************

104996.4
```

FIGURE 23 – Solutions avec la méthode de somme pondérée pour $\lambda_1 = 0.7; \lambda_2 = 0.1; \lambda_3 = 0.2$

Ces différents cas montrent comment la résolution du problème de routage évolue en ajustant les pondérations des composantes de la fonction objective.

La détermination des pondérations s'opère en fonction des objectifs particuliers du réseau privé, en établissant une hiérarchie d'importance pour chaque objectif de minimisation des paramètres associés au réseau. Cela englobe notamment la minimisation des coûts ainsi que l'optimisation de l'utilisation des chemins pour chaque type de trafic.

8 Conclusion

En conclusion, notre projet a exploré de manière approfondie le problème complexe du routage dans des réseaux virtuels privés, en considérant diverses topologies de graphes et en cherchant à optimiser des objectifs parfois contradictoires. Les modèles linéaires que nous avons formulés visaient à minimiser le coût de routage, maximiser l'utilisation d'un arc, et optimiser l'utilisation

moyenne d'un arc.

Les expérimentations sur différents types de graphes ont généré des résultats significatifs, mettant en lumière l'influence cruciale de la topologie du graphe sur les solutions optimales. Cependant, ces investigations n'ont pas été sans défis, notamment avec des solutions infaisables identifiées pour le graphe cyclique unidirectionnel.

Dans notre quête de solutions efficaces, nous avons exploré diverses approches, telles que l'optimisation lexicographique, les ε-contraintes, et les sommes pondérées. Chacune de ces méthodes offre des perspectives uniques pour trouver des compromis judicieux entre des objectifs parfois antagonistes.

En définitive, ce projet a considérablement élargi notre compréhension de la résolution de problèmes de routage dans des réseaux complexes. Les résultats obtenus pointent vers des avenues prometteuses pour des recherches futures, notamment l'optimisation fine des paramètres du modèle et l'exploration de méthodes heuristiques mieux adaptées à des topologies de graphes spécifiques.