université de BORDEAUX

Algorithmes pour l'optimisation en nombres entiers

Qualité de formulation pour le TSP : approche MTZ et CUT

Rapport

February 16, 2026

RAVENDIRANE Gayathiri
Master MAS parcours ROAD,
Année Universitaire : 2025−2026

Table des matières

1 Introduction **3**

2 Description du problème – TSP **3**

3 Formalisation mathématique **3**
3.1 Données . . . 3
3.2 Espace des solutions . . . 3
3.3 Taille d'une instance . . . 3
3.4 Problème de décision . . . 3

4 Formulations étudiées **4**
4.1 Formulation MTZ . . . 4
4.1.1 Données . . . 4
4.1.2 Variables . . . 4
4.1.3 Objectif . . . 4
4.1.4 Contraintes . . . 4
4.1.5 Modèle MTZ complet . . . 5
4.2 Formulation CUT . . . 5
4.2.1 Données . . . 5
4.2.2 Variables . . . 5
4.2.3 Objectif . . . 6
4.2.4 Contraintes . . . 6
4.2.5 Modèle CUT complet . . . 6

5 Méthodologie de résolution et d'expérimentation **6**
5.1 Inégalités valides et coupes . . . 6
5.2 Algorithmes de séparation . . . 6
5.3 Cadre Branch-and-Cut . . . 6
5.4 Détails d'implémentation . . . 7
5.5 Protocole expérimental . . . 7

6 Analyse et interprétation des résultats **7**
6.1 Résultats Obtenus . . . 7
6.2 Performance du modèle MTZ . . . 8
6.3 Performance du modèle CUT . . . 8
6.3.1 CUT (solutions entières) . . . 8
6.3.2 CUT (solutions fractionnaires) . . . 8
6.4 Bilan . . . 8

7 Conclusion **9**

1 Introduction

Le problème du voyageur de commerce (Traveling Salesman Problem, TSP) est un problème fondamental en optimisation combinatoire. Il modélise la situation d'un voyageur de commerce devant visiter un ensemble de villes en minimisant la distance totale parcourue, tout en respectant la contrainte de visiter chaque ville exactement une fois avant de revenir à son point de départ.
Derrière cette description intuitive se cache un problème de grande complexité théorique et algorithmique.
Le TSP constitue ainsi un exemple classique pour étudier la qualité des formulations et l'impact des techniques de génération dynamique de contraintes.

2 Description du problème – TSP

On considère un ensemble de villes entre lesquelles la distance est connue pour chaque paire. L'objectif est de déterminer un circuit hamiltonien de coût minimal, c'est-à-dire une tournée passant une et une seule fois par chaque ville, puis revenant à la ville de départ. Plus formellement, étant donné un graphe orienté complet $G = (N, A)$, où N représente l'ensemble des villes et où chaque arc (i, j) est associé à un coût c_{ij}, le problème consiste à trouver une permutation des sommets minimisant la somme des coûts des arcs empruntés.

3 Formalisation mathématique

3.1 Données

Soit $N = \{1, \ldots, n\}$, l'ensemble des villes, où 1 est la ville de départ du voyageur de commerce. On pose $N^* = N \setminus \{1\} = \{2, \ldots, n\}$. Pour toute paire de sommets $i, j \in N, i \neq j$, on note c_{ij} la distance entre la ville i et la ville j.

3.2 Espace des solutions

Une solution de TSP est une permutation π de l'ensemble N où $\pi(i)$ représente la ville visitée en position i, avec $i \in \{1, \ldots, n\}$, dans le circuit, et $\pi(1) = 1$ afin de fixer la ville de départ.

3.3 Taille d'une instance

Une instance du TSP est définie par un ensemble de n villes et une matrice de distances $(c_{ij})_{i,j \in N}$. Comme il y a $n(n-1)$ distances à spécifier (pour toute paire i et j, $i \neq j$), la taille d'une instance est en $\mathcal{O}(n^2)$.

Remarque : D'ailleurs, ce problème est NP-complet. Par conséquent, le problème d'optimisation associé est NP-difficile.

3.4 Problème de décision

Voici une formulation du problème de décision : *Étant donné une constante $K \in \mathbb{R}^+$, existe-t-il un circuit hamiltonien de coût total inférieur ou égal à K ?*

4 Formulations étudiées

Dans cette section, nous présentons deux formulations classiques du TSP : la formulation Miller–Tucker–Zemlin (MTZ) et la formulation à contraintes de coupe (CUT).

4.1 Formulation MTZ

Une formulation classique visant à éliminer les sous-tours à l'aide de variables de position.

4.1.1 Données

- $N = \{0, \ldots, n-1\}$: l'ensemble des n villes à visiter,
- $N^* = N \setminus \{0\}$,
- c_{ij} : la distance entre deux villes i et j.

4.1.2 Variables

On pose pour tout $i, j \in N$ et $i \neq j$:

$$x_{ij} = \begin{cases} 1 & \text{si on se déplace directement de la ville } i \text{ à la ville } j, \\ 0 & \text{sinon.} \end{cases}$$

$u_i \in \mathbb{N}$ indique la position, comprise entre 0 et $n-1$, de la ville $i \in N$ dans le circuit.

4.1.3 Objectif

L'objectif consiste à minimiser la longueur du tour construit. La fonction objectif est donc donnée par :

$$\text{Min.} \quad \sum_{i \in N} \sum_{\substack{j \in N \\ j \neq i}} c_{ij} x_{ij}$$

4.1.4 Contraintes

- **Contraintes de sortie :** chaque ville est quittée exactement une fois.

$$\sum_{\substack{j \in N \\ j \neq i}} x_{ij} = 1, \quad \forall i \in N.$$

- **Contraintes d'entrée :** chaque ville est visitée exactement une fois.

$$\sum_{\substack{j \in N \\ j \neq i}} x_{ji} = 1, \quad \forall i \in N.$$

- **Fixation de la ville de départ :** la ville 0 est fixée à la première position du tour.

$$u_0 = 0.$$

- **Contraintes MTZ (élimination des sous-tours) :** si l'arc (i, j) est utilisé, alors la position de j est strictement supérieure à celle de i.

$$u_j \geq u_i + 1 - (n-1)(1 - x_{ij}), \quad \forall i, j \in N, i \neq j, j \neq 0.$$

- **Bornes sur les variables de position :** les villes autres que 0 occupent une position entre 1 et $n-1$.

$$1 \leq u_i \leq n-1, \quad \forall i \in N^*.$$

- **Contraintes d'intégralité :** les variables de décision sont binaires.

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in N, i \neq j.$$

4.1.5 Modèle MTZ complet

$$\text{Min.} \quad \sum_{i \in N} \sum_{\substack{j \in N \\ j \neq i}} c_{ij} x_{ij} \tag{1}$$

$$\text{s.c.} \quad \sum_{\substack{j \in N \\ j \neq i}} x_{ij} = 1 \qquad \forall i \in N, \tag{2}$$

$$\sum_{\substack{j \in N \\ j \neq i}} x_{ji} = 1 \qquad \forall i \in N, \tag{3}$$

$$u_0 = 0, \tag{4}$$

$$u_j \geq u_i + 1 - (n-1)(1 - x_{ij}) \qquad \forall i, j \in N, i \neq j, j \neq 0, \tag{5}$$

$$1 \leq u_i \leq n-1 \qquad \forall i \in N*, \tag{6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in N, i \neq j. \tag{7}$$

4.2 Formulation CUT

La formulation CUT repose sur l'ajout de contraintes de connexité visant à éliminer explicitement les sous-tours et à garantir que la solution obtenue correspond à un unique circuit hamiltonien.

4.2.1 Données

Identiques à celles du modèle MTZ, (N), (N^*), et (c_{ij}), en ajoutant :

- $\mathcal{S}$: l'ensemble de tous les sous-ensembles de sommets de N^*.

4.2.2 Variables

Identiques à celles du modèle MTZ, (x_{ij}) et (u_i).

4.2.3 Objectif

Identiques à celles du modèle MTZ, (1).

4.2.4 Contraintes

- Les contraintes (2), (3) et (7) sont identiques.
- **Contraintes de coupe (élimination des sous-tours) :** pour tout sous-ensemble strict S de villes, au moins un arc doit sortir de S afin de garantir la connexité du tour.

$$\sum_{i \in N \setminus S} \sum_{j \in S} x_{ij} \geq 1 \quad \forall S \in \mathcal{S}.$$

4.2.5 Modèle CUT complet

$$\text{Min.} \quad \sum_{i \in N} \sum_{\substack{j \in N \\ j \neq i}} c_{ij} x_{ij} \tag{8}$$

$$\text{s.c.} \quad \sum_{\substack{j \in N \\ j \neq i}} x_{ij} = 1 \qquad \forall i \in N, \tag{9}$$

$$\sum_{\substack{j \in N \\ j \neq i}} x_{ji} = 1 \qquad \forall i \in N, \tag{10}$$

$$\sum_{i \in N \setminus S} \sum_{j \in S} x_{ij} \geq 1 \qquad \forall S \in \mathcal{S}, \tag{11}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in N, i \neq j. \tag{12}$$

5 Méthodologie de résolution et d'expérimentation

5.1 Inégalités valides et coupes

Les contraintes d'élimination des sous-tours garantissent que chaque solution correspond à un circuit hamiltonien. Elles sont liées à l'enveloppe convexe du problème.

5.2 Algorithmes de séparation

Pour identifier les coupes violées, des algorithmes spécifiques sont appliqués aux solutions entières et aux solutions fractionnaires. Ces coupes sont ajoutées dynamiquement lors de la résolution (voir).

5.3 Cadre Branch-and-Cut

La résolution repose sur le schéma branch-and-cut, qui combine la recherche arborescente (branching) et l'ajout dynamique de coupes (cutting). Les callbacks de Gurobi permettent d'intervenir pendant la résolution pour injecter ces coupes.

5.4 Détails d'implémentation

Le modèle est implémenté avec l'API C++ de Gurobi. Les callbacks gèrent l'ajout dynamique des coupes. La limite de temps est fixée à 180 secondes (= 3 minutes) par instance.

5.5 Protocole expérimental

- Les tests ont été réalisés sur les dix instances (*voir tableau*).
- **Environnement matériel :** 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz et 15 GB RAM.
- **Environnement logiciel :** Gurobi (v13.0.1), compilateur C++ (v13.3.0).

6 Analyse et interprétation des résultats

6.1 Résultats Obtenus

Les pseudo-codes des algorithmes de séparation implémentés sont présentés en annexe (3). Les fonctions auxiliaires correspondantes figurent en (1 et 2). Les commandes pour exécuter les fichiers `.hpp/.cpp` sont en annexe (7).

	MTZ					CUT (entier)					CUT (fractionnaire)				
Instance	Obj	Bound	Nodes	Status	Time (s)	Obj	Bound	Cuts	Status	Time (s)	Obj	Bound	Cuts	Status	Time (s)
br17	39	39	980	OPT	0.42	39	39	31	OPT	0.02	39	39	13	OPT	0.00
ftv170	2755	2755	1233	OPT	13.85	-	2677	1821	TLR	180.02	2715.17	2715.17	54	OPT	0.01
ftv33	1286	1286	1	OPT	0.33	1286	1286	53	OPT	0.09	1286	1286	23	OPT	0.00
ftv35	1473	1473	174	OPT	0.71	1473	1473	114	OPT	0.18	1457.33	1457.33	28	OPT	0.00
ftv38	1530	1530	106	OPT	0.95	1530	1530	80	OPT	0.16	1514.33	1514.33	26	OPT	0.00
ftv44	1613	1613	189	OPT	0.72	1613	1613	86	OPT	0.24	1584.88	1584.88	17	OPT	0.00
ftv47	1776	1776	2497	OPT	2.13	1776	1776	73	OPT	0.29	1748.61	1748.61	32	OPT	0.00
ftv55	1608	1608	398	OPT	1.24	1608	1608	96	OPT	0.40	1584	1584	33	OPT	0.00
ftv64	1839	1839	4692	OPT	10.05	1839	1839	117	OPT	0.74	1807.50	1807.50	36	OPT	0.00
ftv70	1950	1950	3295	OPT	6.48	1950	1950	932	OPT	7.03	1909	1909	22	OPT	0.00

Table 1: MTZ vs CUT results (time limit 180 secondes)

* TL = Time Limit Reached

Le tableau suivant synthétise les performances moyennes observées pour les modèles.

Critère	**MTZ**	**CUT (entier)**	**CUT (fractionnaire)**
Instances optimales	10/10	9/10	10/10
Échecs (time limit)	0	1	0
Temps moyen (s)	3.688	18.917	0.001
Nœuds explorés (moyenne)	13565	–	–
Contraintes générées (moyenne)	–	340	28.4

Table 2: Comparaison synthétique des performances des formulations MTZ et CUT

6.2 Performance du modèle MTZ

- Le modèle MTZ résout toutes les instances à l'optimalité (10/10) dans la limite de temps fixée (180 s).
- Toutefois, le nombre moyen de nœuds explorés est élevé (13 565), ce qui traduit une relaxation linéaire relativement faible.
- Le temps de résolution reste modéré (3.688 s en moyenne), mais augmente sensiblement pour les grandes instances `ftv64, ftv70, ftv170`).
- La formulation MTZ est simple à implémenter, mais son efficacité repose fortement sur le branch-and-bound.

6.3 Performance du modèle CUT

6.3.1 CUT (solutions entières)

- Le modèle avec séparation uniquement sur solutions entières atteint l'optimalité sur 9 instances sur 10.
- L'instance `ftv170` atteint la limite de temps (180 s). Par ailleurs, en executant sans limite de temps, l'exécution n'est toujours pas terminée après 1 503 s (environ 25 minutes).
- Le nombre moyen de contraintes générées (340) reste important, ce qui traduit un effort significatif de séparation.
- Cette approche améliore la structure du modèle mais peut rester coûteuse sans séparation fractionnaire.

6.3.2 CUT (solutions fractionnaires)

- La séparation sur solutions fractionnaires permet de résoudre toutes les instances à l'optimalité (10/10).
- Les temps de résolution sont quasi nuls (0.001 s en moyenne), montrant une relaxation très forte.
- Le nombre moyen de coupes générées est faible (28.4), mais efficace.
- Cette stratégie réduit drastiquement l'effort de branchement grâce à un renforcement précoce de la relaxation.

6.4 Bilan

Le tableau 2 présentait une synthèse quantitative des performances moyennes. Le tableau suivant (3) propose une lecture comparative et qualitative des trois approches.

Critère	**MTZ**	**CUT (entier)**	**CUT (fractionnaire)**
Taux d'optimalité	10/10	9/10	10/10
Sensibilité au time limit	Faible	Élevée (`ftv170`)	Nulle
Temps de résolution	Modéré	Élevé (moyenne impactée)	Très faible
Effort de branchement	Très important	Réduit	Quasi nul
Nombre de coupes	–	Important	Faible
Force de la relaxation	Faible	Moyenne	Très forte
Implémentation	Simple	Plus complexe	Plus plus complexe

Table 3: Analyse comparative des formulations

- MTZ est robuste et simple, mais explore beaucoup de nœuds en raison d'une relaxation plus faible.
- CUT avec séparation entière améliore la validité du modèle mais reste sensible au temps limite.
- CUT avec séparation fractionnaire domine clairement en termes de temps et de qualité de borne.

7 Conclusion

Les résultats illustrent l'impact majeur du renforcement polyédral et du branch-and-cut sur la performance globale.
La qualié de la relaxation linéaire apparaît comme le facteur déterminant des performances observées, bien plus que la simple complexité apparente de la formulation (comme le projet de IPVE : Uncapacitated lot-sizing problem with setups).

Annexe

Fonctions auxiliaire

Algorithm 1: Détection d'un sous-tour dans une solution entière

Data: $n \geq 0$ ▷ nombre de villes, $x[i][j]$
Result: vrai si une liste des sous-tours violés S est trouvé, ou faux si aucun

```
1  Fonction findSubtour_S(n):
2      initialiser visite[i] ← faux pour tout i;
3      for i ← 0 to n − 1 do
4          if ville i non visitée then
5              cycle ← []                                   ▷ initialisation du cycle
6              current ← i;
7              while current non visité do
8                  visite[current] ← vrai                   ▷ marquer comme visité
9                  cycle ← cycle + [current];
10                 trouveNext ← faux;
11                 for j ← 0 to n − 1 do
12                     if current ≠ j et x[current][j] > 0.5 then
13                                 ▷ 0,5 pour tolérance numérique current ← j;
14                         trouveNext ← vrai;
15                         break;
16                 if pas de trouveNext then
17                     break;
18             if cycle fermé et taille(cycle) < n then
19                 S ← cycle;
20                 return vrai                              ▷ sous-tour trouvé
21     return faux                                          ▷ aucun sous-tour détecté
```

Algorithm 2: Détection d'un sous-tour dans une solution fractionnaire

Data: $n \geq 0$ ▷ nombre de villes, $x[i][j]$ solution fractionnaire
Result: vrai si une liste des sous-tours violés S est trouvé, ou faux si aucun

```
1  Fonction findFractionalCut_S(n):
2      Construire la matrice cap[i][j] : Si i = j alors cap[i][j] ← 0.0;
3      Sinon cap[i][j] ← x[i][j];
4      for sink ← 1 to n − 1 do
5          initialiser dist[0..n − 1];
6          val ← 0.0;
7          Calculer le min-cut dirigé de 0 vers sink : directed_min_cut(cap, n, 0,
             sink, val, dist);
8          if val < 1 − ϵ then
9                                          ▷ coupe violée détectée S ← [];
10             for v ← 0 to n − 1 do
11                 if dist[v] ≤ n − 1 then
12                     Ajouter v à S;
13             if S non vide et taille(S) < n then
14                 return vrai                          ▷ une coupe fractionnaire est trouvée
15     return faux                                                ▷ pas de coupe violée
```

Algorithmes de séparation implémentés

Algorithm 3: Callback Gurobi pour CUT

Data: Variables $x[i][j]$, noeud courant
Result: Ajouter lazy cut ou user cut

```
1  Fonction callbackCUT():
2      if where == MIPSOL then
3                                  ▷ solution entière xVal ← getSolution(x);
4          Si viol => addLazy(coupe);
5      if where == MIPNODE then
6                                ▷ solution fractionnaire xVal ← getNodeRel(x);
7          Si viol => addCut(coupe);
```

Exemples de commandes

On suppose que le dossier `data/`, contenant toutes les instances, se trouve à la racine du projet.

- **MTZ (par défaut)**
  `./TSP_Gurobi ../data/br17.atsp MTZ`
- **CUT (MIP entier avec coupes de sous-tours paresseuses)**
  `./TSP_Gurobi ../data/br17.atsp CUT`
- **CUT_LP (PL fractionnaire avec génération itérative de coupes)**
  `./TSP_Gurobi ../data/br17.atsp CUT_LP`