# Rapport sur le projet modèles mathématiques
# “Traveling Salesman Problem (TSP)”

## 1. Introduction

Ce rapport présente une comparaison entre deux méthodes de résolution du problème du voyageur de commerce (TSP) : la formulation MTZ (Miller-Tucker-Zemlin) et la génération de contraintes. Nous avons généré aléatoirement des instances de différentes tailles et analysé les performances de chaque méthode en termes de temps de résolution et de qualité des solutions.

## 2. Contexte

Le problème du voyageur de commerce (TSP) est un problème de programmation linéaire en nombres entiers (PLNE) où l'objectif est de déterminer le chemin optimal pour visiter un ensemble de villes une seule fois chacune et revenir au point de départ. Pour cela, nous avons utilisé deux méthodes différentes : la formulation MTZ (Miller-Tucker-Zemlin) et la génération de contraintes. La formulation MTZ utilise des variables supplémentaires et des contraintes spécifiques pour éliminer les sous-tours, tandis que la méthode de génération de contraintes ajoute progressivement des contraintes pour exclure les sous-tours détectés. L'objectif de ce projet est de comparer ces deux méthodes en termes de performance et d'efficacité, afin de tirer des conclusions sur leurs avantages respectifs après une analyse comparative des résultats obtenus.

## 3. Structure de l'équipe et Méthodologie de travail

Notre équipe de projet est composée de deux membres : Keskes Nazim et Tarek Yacine Atbi. Nous avons réparti les tâches de manière structurée pour optimiser notre efficacité et garantir la qualité du travail.

- **Génération des instances:** Nazim a pris en charge la génération des instances. Il a généré 5 instances pour chaque taille spécifiée (10, 20, 30, 50, et 100 villes), totalisant ainsi 25 instances. Chaque instance a été créée en générant aléatoirement des points sur un plan et en calculant les distances euclidiennes arrondies.

- **Résolution du problème avec la génération de contraintes:** Nazim a également mis en œuvre la méthode de génération de contraintes pour résoudre les 25 instances. Il a noté les temps de résolution et a vérifié si les solutions optimales étaient obtenues dans la limite de temps impartie.

- **Résolution du problème avec la formulation MTZ (Miller-Tucker-Zemlin)** : Tarek Yacine a travaillé sur l'application de la formulation MTZ pour résoudre les mêmes 25 instances. Il a mesuré et comparé les temps de résolution obtenus avec ceux de la méthode de génération de contraintes.

- **Analyse comparative et bilan** : Nous avons collaboré ensemble pour analyser les résultats obtenus par les deux méthodes. Cette collaboration nous a permis de comparer les performances, d'identifier les points forts et faibles de chaque approche, et de tirer des conclusions sur leur efficacité respective.

## 4. Implementation

Maintenant que nous avons bien défini notre problème, nous devons d’abord générer les instances aléatoirement puis le résoudre avec les deux méthodes et enfin comparer les résultats obtenus par les deux méthodes.

### 4.1. Génération des instances

Pour évaluer les performances de nos méthodes de résolution, nous avons généré cinq jeux de cinq instances de tailles différentes ,étant respectivement composées de 10, 20, 30, 50 et 100 villes , totalisant **25 instances** au total. Ces instances varient en taille, allant de 10 à 100 villes, ce qui nous permet de couvrir un large éventail de scénarios. Chaque instance a été créée en générant aléatoirement des points (x, y) sur un plan, avec des coordonnées entières comprises entre 0 et 100.

Ensuite, les distances entre les villes ont été calculées en utilisant la distance euclidienne, avec arrondissement à l'entier le plus proche. Voici la formule de la distance euclidienne utilisé :

$$Distance((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Sachant que l'une des propriétés de la distance euclidienne est la suivante :

$$Distance((x_1, y_1), (x_2, y_2)) = Distance((x_2, y_2), (x_1, y_1))$$

Cette approche garantit une diversité suffisante dans les instances pour évaluer les performances des méthodes de résolution dans différentes conditions et tailles de problèmes. Voici un exemple d'un résultat obtenu de la matrice de distance pour un nombre de villes **n = 10 :**

| **villes** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 58 | 71 | 57 | 67 | 54 | 52 | 55 | 81 | 71 |
| **2** | 58 | 0 | 33 | 19 | 103 | 80 | 21 | 54 | 83 | 28 |
| **3** | 71 | 33 | 0 | 51 | 92 | 67 | 20 | 36 | 58 | 6 |
| **4** | 57 | 19 | 51 | 0 | 113 | 91 | 37 | 69 | 99 | 46 |
| **5** | 67 | 103 | 92 | 113 | 0 | 25 | 85 | 57 | 52 | 67 |
| **6** | 54 | 80 | 67 | 91 | 25 | 0 | 61 | 32 | 35 | 72 |
| **7** | 52 | 21 | 20 | 37 | 85 | 61 | 0 | 34 | 63 | 18 |
| **8** | 55 | 54 | 36 | 69 | 57 | 32 | 34 | 0 | 31 | 40 |
| **9** | 81 | 83 | 58 | 99 | 52 | 35 | 63 | 31 | 0 | 64 |
| **10** | 71 | 28 | 6 | 46 | 67 | 72 | 18 | 40 | 64 | 0 |

**Tableau :** un exemple d'une matrice de distance d'une instance de 10 villes.

## 4.2. Résolution avec la méthode de génération de contraintes

Pour résoudre le problème du voyageur de commerce avec la méthode de génération de contraintes, nous commençons par formuler le problème initial comme un PLNE (programmation linéaire en nombres entiers) spécifique. Le problème revient à minimiser une fonction objectif Z.

Les variables de décision sont les $x_{ij}$ , tels que $x_{ij} = 1$ si l'arc (i,j) est dans le tour et 0 sinon. voici le programme :

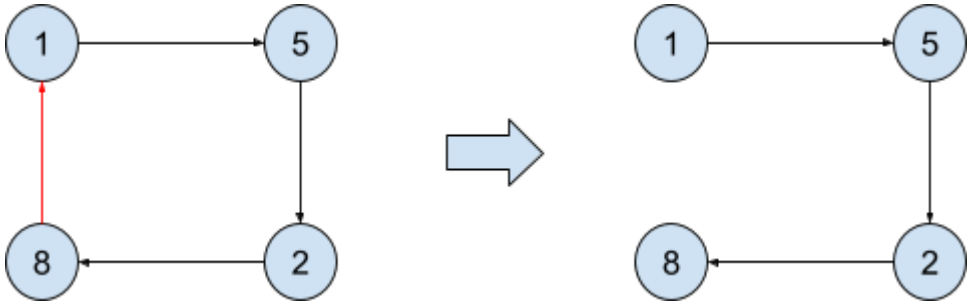
$$\begin{cases} Min\ Z = \sum_{ij} c_{ij} \times x_{ij} \\ \sum_{j=1}^{n} x_{ij} = 1\ ;\ \forall i \in [1;n] \\ \sum_{i=1}^{n} x_{ij} = 1\ ;\ \forall j \in [1;n] \\ x_{ii} = 0\ ;\ \forall i \in [1;n] \\ 0 \leq x_{ij} \leq 1\ ;\ \forall i \in [1;n]\ ,\ \forall j \in [1;n] \\ x_{ij} \in N;\ \forall i \in [1;n]\ ,\ \forall j \in [1;n] \end{cases}$$

En théorie de graphe, le problème TSP revient a trouver un circuit hamiltonien qui comporte le coût minimal. Pour se faire, nous sommes obligés de rajouter des nouvelles contraintes.
Avec cette méthode nous désirons donc rajouter des contraintes qui éliminent les sous tours (càd les cycles hamiltoniens qu'il commence par le sommet de départ 1). Voici le pseudo algorithme que nous avons utilisé :

**Algorithme de génération de contraintes** :

- On résout le problème sans contraintes d'éléminisation de sous-tour
- Soit sol la solution résultant de cette optimisation
- **tant qu**'il existe un sous tour dans sol **faire**
  - ajouter une contrainte qui élimine un ou plusieurs sous tour de la solution courante
  - Ré-optimiser avec cette nouvelle contrainte
  - Soit sol la solution résultant de cette optimisation
  - Détecter si sol contient un nouveau sous-tour
- **fait**

Prenons l'exemple suivant :




Donc, la contrainte a ajouté dans chaque itération sera la suivante :

$$\sum_{s(i,j) \in S} x_{ij} \leq |S| - 1 ; s \in S$$

Tels que S représente l'ensemble des sommets de sous tour trouvé.

Voici les résultats obtenus après avoir appliqué cette méthode sur les 25 instances :

| Le nombre de villes n | Instance | La solution optimale obtenue ? | Le temps limite n'est pas écoulé ? | Le temps estimé (s) |
|---|---|---|---|---|
| n = 10 | 1 | oui | oui | 0.265 |
| | 2 | oui | oui | 0.042 |
| | 3 | oui | oui | 0.008 |
| | 4 | oui | oui | 0.098 |
| | 5 | oui | oui | 0.056 |
| n = 20 | 1 | oui | oui | 0.059 |
| | 2 | oui | oui | 0.366 |

| | | | | |
|---|---|---|---|---|
| | 3 | oui | oui | 0.581 |
| | 4 | oui | oui | 0.114 |
| | 5 | oui | oui | 0.245 |
| n = 30 | 1 | oui | oui | 4.7 |
| | 2 | oui | oui | 10.2 |
| | 3 | oui | oui | 25.8 |
| | 4 | oui | oui | 3.44 |
| | 5 | oui | oui | 1.02 |
| n = 50 | 1 | non | non | 300 |
| | 2 | oui | oui | 44 |
| | 3 | oui | oui | 95.6 |
| | 4 | oui | oui | 140 |
| | 5 | oui | oui | 80.1 |
| n = 100 | 1 | non | non | 300 |
| | 2 | non | non | 300 |
| | 3 | non | non | 300 |
| | 4 | non | non | 300 |
| | 5 | non | non | 300 |

**Tableau:** Les résultats obtenus avec la méthode de génération des contraintes

## 4.3. Résolution avec la formulation MTZ (Miller-Tucker-Zemlin)

Pour résoudre le problème du voyageur de commerce avec la formulation MTZ (Miller-Tucker-Zemlin), nous débutons en formulant le problème initial comme un PLNE spécifique, similaire au programme initial précédent, mais cette fois-ci, nous introduisons de nouvelles variables entières Ui et leurs contraintes associées.

Les variables de décision sont les $x_{ij}$ , tels que $x_{ij} = 1$ si l'arc (i,j) est dans le tour et 0 sinon. De plus, on ajoute des nouvelles variables $u_i$ pour chaque sommet . voici le programme :

$$
\begin{cases}
Min\ Z = \sum_{ij} c_{ij} \times x_{ij} \\
\sum_{j=1}^{n} x_{ij} = 1\ ;\ \forall i \in [1;n] \\
\sum_{i=1}^{n} x_{ij} = 1\ ;\ \forall j \in [1;n] \\
x_{ii} = 0\ ;\ \forall i \in [1;n] \\
0 \leq x_{ij} \leq 1\ ;\ \forall i \in [1;n]\ ,\ \forall j \in [1;n] \\
x_{ij} \in N;\ \forall i \in [1;n]\ ,\ \forall j \in [1;n] \\
u_1 = 1 \\
2 \leq u_i \leq n\ ;\ \forall i \neq 1 \\
u_i - u_j + 1 \leq n(1 - x_{ij})\ ;\ \forall i \neq 1,\ \forall j \neq 1
\end{cases}
$$

Voici les résultats obtenus après avoir appliqué cette méthode sur les 25 instances :

| Le nombre de villes n | Instance | La solution optimale obtenue ? | Le temps limite n'est pas écoulé ? | Le temps estimé (s) |
|---|---|---|---|---|
| n = 10 | 1 | oui | oui | 0.101 |
| | 2 | oui | oui | 0.448 |
| | 3 | oui | oui | 0.133 |
| | 4 | oui | oui | 0.166 |
| | 5 | oui | oui | 0.286 |
| n = 20 | 1 | oui | oui | 4.71 |
| | 2 | oui | oui | 13.4 |
| | 3 | oui | oui | 10 |

| | | | | |
|---|---|---|---|---|
| | 4 | oui | oui | 43 |
| | 5 | oui | oui | 2.12 |
| n = 30 | 1 | oui | oui | 227 |
| | 2 | oui | oui | 79.6 |
| | 3 | non | non | 300 |
| | 4 | oui | oui | 106 |
| | 5 | oui | oui | 14.7 |
| n = 50 | 1 | non | non | 300 |
| | 2 | non | non | 300 |
| | 3 | non | non | 300 |
| | 4 | non | non | 300 |
| | 5 | non | non | 300 |
| n = 100 | 1 | non | non | 69.3 |
| | 2 | non | non | 84.5 |
| | 3 | non | non | 97.3 |
| | 4 | non | non | 52.7 |
| | 5 | non | non | 51.3 |

**Tableau:** Les résultats obtenus avec la formulation MTZ

## 4.4. Analyse comparative et bilan entre les deux méthodes

Pour comparer les méthodes de génération de contraintes et de formulation MTZ (Miller-Tucker-Zemlin), nous avons appliqué chacune d'elles à 25 instances aléatoires du problème du voyageur de commerce (TSP) de tailles variées (10, 20, 30, 50 et 100 villes).

**Tableaux de bords comparatives**

Les tableaux de bord ajoutent une dimension visuelle à notre analyse comparative des méthodes de résolution du problème du voyageur de commerce (TSP). Voici

les tableaux de bord que nous avons obtenus en se basant sur les résultats de chaque méthode :

- Le premier dashboard propose une comparaison directe des temps de résolution moyens entre les deux méthodes pour chaque taille d'instance.

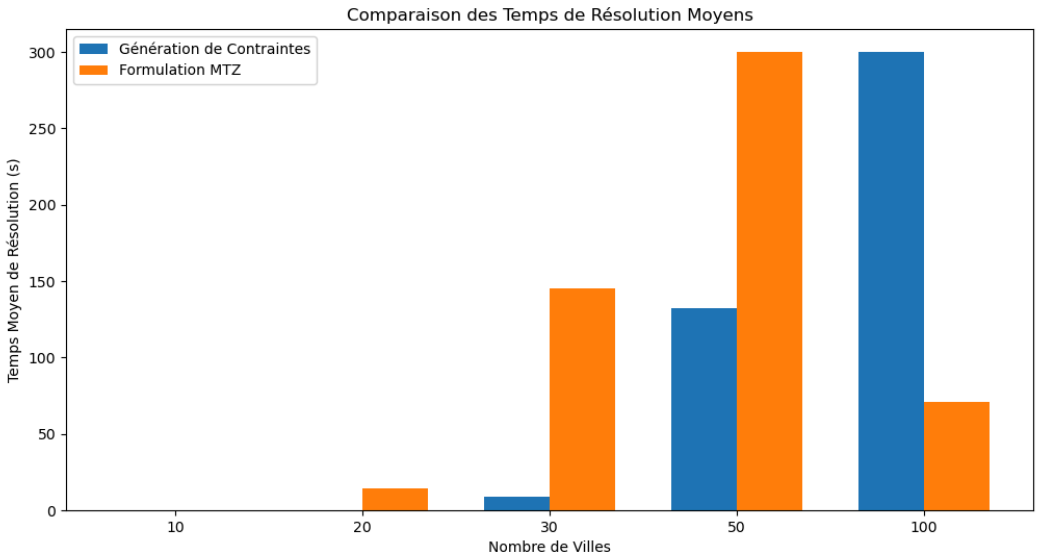



**Figure 1:** Comparaison des temps de résolution moyen en fonction de nombre de villes n.

- Le deuxième dashboard présente l'évolution des temps de résolution moyens en fonction du nombre de villes.

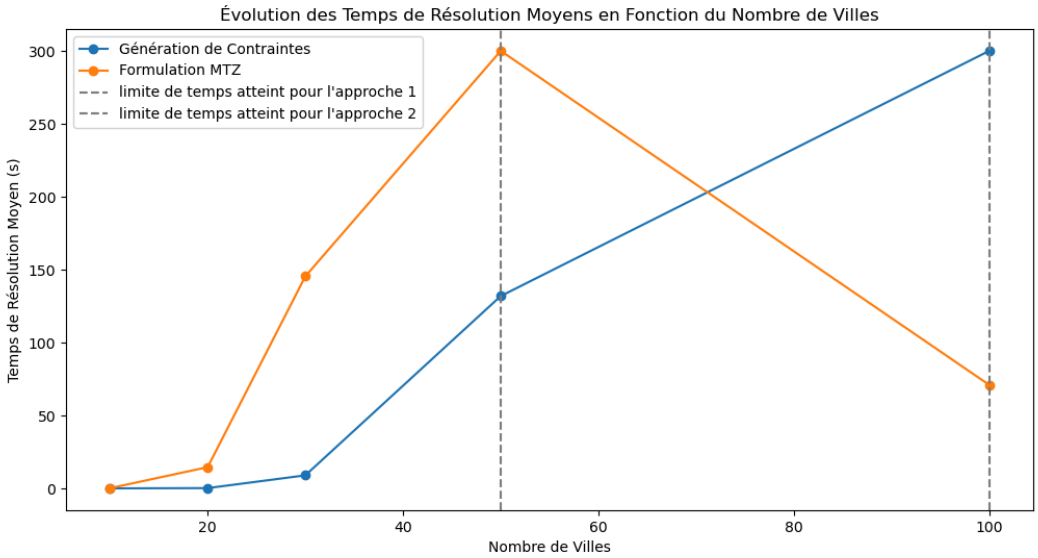



**Figure 2:** Évolution des temps de résolution moyen en fonction de nombre de villes n.

Maintenant, voici une analyse comparative basée sur les résultats obtenus précédemment ainsi que les tableaux de bord ci-dessus.

- **Instances de taille 10 villes**

  - *Génération de contraintes:* La méthode a résolu toutes les instances de taille 10 villes avec des temps de résolution très courts, allant de 0.008 à 0.265 secondes.
  - *Formulation MTZ :* La méthode a également résolu toutes les instances, avec des temps de résolution allant de 0.101 à 0.448 secondes.

  Pour cette taille d'instance, les performances des deux méthodes sont très proches, avec un léger avantage pour la génération de contraintes en termes de temps de résolution.

- **Instances de taille 20 villes**
  - *Génération de contraintes:* La méthode a résolu toutes les instances avec des temps allant de 0.059 à 0.581 secondes.
  - *Formulation MTZ :* La méthode a résolu toutes les instances avec des temps allant de 2.12 à 43 secondes.

  La génération de contraintes est plus rapide pour cette taille d'instance.

- **Instances de taille 30 villes**
  - *Génération de contraintes :* La méthode a résolu toutes les instances avec des temps allant de 1.02 à 25.8 secondes.
  - *Formulation MTZ :* La méthode a résolu 4 des 5 instances, avec des temps allant de 14.7 à 227 secondes, mais a atteint la limite de temps pour une instance.
    La génération de contraintes est plus fiable et rapide pour cette taille d'instance. En revanche, bien que la formulation MTZ ait permis de résoudre presque toutes les instances, les temps de résolution se rapprochent souvent de la limite maximale.

- **Instances de taille 50 villes**
  - *Génération de contraintes:* La méthode a résolu 4 des 5 instances, avec des temps allant de 44 à 140 secondes, atteignant la limite de temps pour une instance.
  - *Formulation MTZ :* La méthode n'a pu résoudre aucune instance dans le temps imparti de 300 secondes.

La génération de contraintes est nettement plus performante pour cette taille d'instance. Cependant, les temps de résolution se rapprochent de la limite maximale, indiquant que cette méthode commence à montrer ses limites pour des instances de cette taille.

- **Instances de taille 100 villes**
  - *Génération de contraintes :* La méthode n'a pu résoudre aucune instance dans le temps imparti de 300 secondes.
  - *Formulation MTZ :* la méthode n'a pu résoudre aucune instance dans le temps imparti de 300 secondes. De plus, la résolution pour plusieurs instances termine avant d'arriver à la limite de temps, mais sans obtenir de solution optimale.

  Les deux méthodes sont limitées pour cette taille d'instance. Aucune des deux méthodes n'a réussi à résoudre les instances dans le temps imparti, la méthode MTZ ayant même échoué à produire des solutions optimales avant l'écoulement du temps.

## 4.5. Résultats obtenus par les deux méthodes

- **Génération de contraintes :** Cette méthode montre une performance supérieure en termes de temps de résolution pour les petites et moyennes instances (10, 20 et 30 villes). Elle reste plus efficace et fiable jusqu'à 50 villes, mais devient inefficace pour les très grandes instances (100 villes).

- **Formulation MTZ :** Bien que performante pour les petites instances (10 villes), cette méthode devient progressivement moins efficace à mesure que la taille des instances augmente, avec des performances particulièrement limitées pour les grandes instances (50 et 100 villes).

En conclusion, la méthode de génération de contraintes s'avère plus performante dans la plupart des cas, en particulier pour les petites et moyennes tailles de problèmes. La formulation MTZ, bien qu'utile pour les très petites instances, montre des limitations significatives pour les tailles plus grandes. Pour des problèmes de très grande taille, des méthodes heuristiques ou des approches hybrides, que nous aborderons dans le prochain paragraphe, pourraient être nécessaires pour obtenir des résultats en temps raisonnable.

## 5. Méthodes Alternatives pour la résolution du Problème du Voyageur de Commerce

Dans le contexte des très grandes instances du Problème du Voyageur de Commerce (TSP), où ni la méthode de génération de contraintes ni la formulation MTZ ne parviennent à trouver des solutions optimales dans un temps raisonnable (dans ce cas, 300 secondes), l'exploration d'alternatives devient nécessaire. Voici un aperçu des méthodes heuristiques et des approches hybrides qui peuvent être envisagées :

- **Méthodes Heuristiques**

  Les méthodes heuristiques, bien qu'elles ne garantissent pas une solution optimale, offrent des solutions satisfaisantes dans des délais plus courts. Parmi celles-ci, on retrouve des techniques telles que l'algorithme du Nearest Neighbor, l'algorithme de Christofides, la Recherche Tabou et les Algorithmes Génétiques.

  - *Algorithme de Nearest Neighbor :* Un point de départ simple qui construit une tournée en ajoutant à chaque étape la ville la plus proche non encore visitée.
  - *Algorithme de Christofides :* Une heuristique qui fournit une solution dont la longueur ne dépasse pas 1,5 fois la longueur de la tournée optimale.
  - *Recherche Tabou :* Une méthode qui explore l'espace des solutions tout en évitant de revenir sur des solutions récemment explorées pour échapper aux minima locaux.
  - *Algorithmes génétiques :* Inspirés de la sélection naturelle, ces algorithmes utilisent des opérations comme la mutation, le croisement et la sélection pour évoluer vers de meilleures solutions.

- **Approches Hybrides**

  Les approches hybrides combinent différentes techniques pour améliorer la performance globale. Par exemple, on peut envisager de combiner des méthodes exactes avec des heuristiques, ou d'explorer des algorithmes métaheuristiques hybrides. De plus, la décomposition et la résolution du problème en sous-problèmes plus petits peuvent également être une stratégie efficace.

En somme, face aux défis posés par les très grandes instances du TSP, où les méthodes exactes atteignent leurs limites, les méthodes heuristiques et les approches hybrides se présentent comme des alternatives prometteuses pour trouver des solutions de qualité dans des délais acceptables.

## 6. Conclusion

L'étude comparative entre la méthode de génération de contraintes et la formulation MTZ dans la résolution du Problème du Voyageur de Commerce (TSP) a mis en lumière leurs performances respectives sur des instances de différentes tailles. Alors que la génération de contraintes a démontré une efficacité notable pour les petites et moyennes instances, montrant des temps de résolution courts et fiables, la formulation MTZ a rencontré des limitations croissantes à mesure que la taille des instances augmentait.

Pour les très grandes instances du TSP, aucune des deux méthodes n'a pu fournir des solutions optimales dans un temps raisonnable, soulignant ainsi la nécessité d'explorer des alternatives. Les méthodes heuristiques et les approches hybrides se présentent comme des voies prometteuses pour résoudre ces défis.

En conclusion, cette étude souligne l'importance de choisir la méthode de résolution appropriée en fonction de la taille et de la complexité du problème. Alors que les méthodes exactes restent efficaces pour les petites et moyennes instances, les méthodes heuristiques et hybrides sont des alternatives viables pour les problèmes de très grande taille. Cette recherche ouvre la voie à de futures explorations visant à améliorer les performances de résolution du TSP et d'autres problèmes d'optimisation combinatoire.