Rapport SAE Optimisation

Autour du tracé d'un métro circulaire

MAZOUZ Erij
MAHJOUB Mohamed Saber

Année universitaire 2026-2027

Université : **Sorbonne Paris Nord - Sup Gualilée**
Responsable du cours : **M.LETOCART Lucas**

Table des matières

1 Engagement de non-plagiat

Nous, soussignees **MAHJOUB Mohamed Saber**, **MAZOUZ Erij** etudiants en 2ème annee d'ecole d'ingenieur a Sup Galilee, declarons etre pleinement conscients que la copie de tout ou partie d'un document, quel qu'il soit, publie sur tout support existant, y compris sur Internet, constitue une violation du droit d'auteur ainsi qu'une fraude caracterisee. En consequence, nous declarons que notre rapport ne comporte aucun plagiat, et assure que nous avons cite explicitement, a chaque fois que nous en avons fait usage, toutes les sources utilisees pour le rediger.

Fait a **Cité Universitaire De Paris** , le **29/01/2026**

2 Introduction

En effet la conception un réseau de transport public est un défi majeur pour organiser et développer les grandes villes. L'enjeu ici c'est de trouver un tracé qui desservira les quartiers les plus peuplés, sans augmenter les coûts de construction et d'exploitation. Dans cette optique, le modèle de l'anneau-étoile est très utile pour imaginer une ligne circulaire.

Notre problème revient à choisir, parmi un ensemble de points , un nombre fixe de stations, disons p. Chaque habitant d'un quartier par exemple doit se diriger vers la station la plus proche à lui. Les stations sélectionnées sont ensuite reliées en une boucle unique, formant ainsi la ligne circulaire. L'objectif est de minimiser le coût total, qui inclut à la fois les distances à pied et la longueur de la boucle elle-même.

Progressivement dans ce projet, nous avons utiliser des données d'un problème classique du voyageur de commerce (TSP) contenant des coordonnées de 52 points. Et comme indiqué dans le PDF donnée le premier point doit être une station.
Pour résoudre ce problème, nous allons exploité deux approches.

La première c'est une modélisation mathématique précise, sous forme d'un PLNE on parle ici de la **Formulation Compacte**, qui permet d'obtenir des solutions optimales, mais pour des tailles de problème raisonnables.

La 2ème approche est fondé sur des méthodes heuristiques et métaheuristiques qui apportent des solutions approchées et peuvent traiter des données volumineusent.

Ce rapport détaille ces approches et à la fin une comparaison des résultats obtenus des 3 modèles est établie en terme de qualité des solutions que sur les temps de calcul.

3 Données et instances

Nous avons utilisé ce lien pour les instances à utiliser : `https://github.com/mastqe/tsplib/tree/master`

Dans le contexte de ce rapport, on a travaillé et comparé sur l'instance *berlin52.tsp*
`https://github.com/mastqe/tsplib/blob/master/berlin52.tsp`

Nous allons assumer que le premier sommet de coordonnées **(565.0 , 575.0)** est une station.

4 Difficulté du problème de l'anneau-étoile

Le problème de l'anneau-étoile (Ring-Star) est à la fois un problème de TSP combiné avec celui du p-médium **(que nous savons qu'ils sont eux même NP-hard)**. Il est donc très facile de démontrer qu'il s'agit d'un problème *NP-hard*. Pour ce faire, il suffit de démontrer qu'au moins un des deux problèmes (p-medium ou tsp) peut se réduire au problème de l'anneau étoile.

Théorème : Un problème **A** est **NP-hard** s'il existe **au moins un** problème NP-difficile qui s'y réduit polynomialement.

On rappelle qu'un problème *Ring-Star* est défini par :
- un graphe complet $G = (V, E)$,
- un coût de cycle c_{ij} pour chaque arête (i, j),
- un coût d'affectation d_{ij} pour l'affectation du sommet i au sommet j appartenant à l'anneau,
- l'objectif est de déterminer un cycle (anneau) $R \subseteq V'$ et une affectation des sommets hors de R à ceux de R, minimisant le coût total.

4.1 Réduction du TSP à Ring-Star

On considère le problème du *voyageur de commerce* (TSP) défini comme suit :
- une instance est un graphe complet $G' = (V', E')$ avec une fonction de coût $c' : E \to R^+$,
- l'objectif est de trouver un cycle hamiltonien de coût minimum.

En choisisant les bons paramètres de notre problème : $\mathbf{p} = \mathbf{n}$ (le nombre de stations que nous cherchons est le même que le nombre de points totale), on arrive à réduire TSP à Ring-Star

- On pose $V' = V$.
- Pour chaque arête $(i, j) \in E'$, on définit le coût de cycle :

$$c'_{ij} = c_{ij}.$$

$$\text{TSP} \leq_p \text{Ring-Star} \implies \text{Ring-Star est NP-difficile}$$

4.2 Réduction du p-médium à Ring-Star

Ceci est facultatif, mais pour renformer le raisonnement et le justificatf, on peut également réduire p-médium à Ring-Star.

5 Formulation pour le problème de l'anneau-étoile (Formulation compacte)

Dans cette partie, on s'intéresse à la formulation compacte du problème de l'anneau-étoile. On considère un ensemble de points $V = \{1, \ldots, n\}$ "dans notre cas n = 52 " pris d'une instance TSP.
À partir de ces points, on va sélectionner exactement p stations avec des contraintes précise comme :
Chaque point non station est affecté à une station.

Les stations sélectionnées sont reliées entre elles par un unique cycle (anneau), sans sous-cycles.
Sans oublier l'indication de l'enoncé que le sommet 1 doit être une station.

L'objectif de la formule compacte est de minimiser 2 choses à la fois :

$$Min \sum_{ij \in E} d_{ij} x_{ij} + \sum_{(i,j) \in V \times V} d_{ij} y_{ij}$$

Premièrement le coût des affectations des points vers les stations et deuxièmement le coût du cycle reliant les stations.

5.1 Variables de décision

Dans cette partie on prend le graphe complet $G = (V, E)$ associé aux points du fichier .tsp. Les distances euclidiennes entre les points i et j sont notées d_{ij}.

On distingue 3 types de variables de décision tel que :

Variables d'affectation :

$$y_{ij} = \begin{cases} 1 & \text{si le point } i \text{ est affecté à la station } j, \\ 0 & \text{sinon.} \end{cases}$$

Notamment si $y_{ii} = 1$ cela signifie que le point i est une station.

Variables de cycle :

$$x_{ij} = \begin{cases} 1 & \text{si l'arête } \{i, j\} \text{ est dans le cycle reliant les stations,} \\ 0 & \text{sinon.} \end{cases}$$

Dans le but que chaque station a degré 2 dans le cycle et les non-stations ont degré 0.

Variables de flot :

$$z_{ij} \geq 0$$

signifie la quantité de flot envoyée sur l'arc orienté (i, j). Un flot total d'une valeur de $p - 1$ part du sommet 1 et perd une unité à chaque station visitée.
–> Ces ça qui interdit les sous-cycles.

5.2 Fonction objectif

La fonction objectif vise à minimiser le coût total de notre solution :

$$\min \sum_{(i,j) \in E} d_{ij} x_{ij} + \sum_{i \in V} \sum_{j \in V} d_{ij} y_{ij}$$

Le premier terme correspond à la longueur totale du cycle entre les stations. Le deuxième représente le coût des affectations des points vers les stations respectives. On dispose d'un paramètre α pourrait être utiliser pour pondérer l'importance relative du cycle, mais dans notre cas, il est fixé à 1.

5.3 Contraintes

La formule compacte repose sur ces contraintes on va les expliquer une par une :
Nombre de stations :

$$\sum_{i \in V} y_{ii} = p$$

Choisir exactement p stations signifie p sommets ont $y_{ii} = 1$.
Affectation unique :
Chaque point est affecté à exactement une station (ou à lui-même s'il est station) :

$$\sum_{j \in V} y_{ij} = 1 \quad \forall i \in V$$

On n'oublie pas que chaque i est lié à un unique j.
Affectation vers une station :

$$y_{ij} \leq y_{jj} \quad \forall i \neq j$$

On ne peut affecter i à j que si j est une station.
Ramarque : Si $y_{jj} = 0$, aucun point ne peut être affecté à j.

Degré des sommets :
un sommet est de degré 2 s'il est station, et 0 sinon :

$$\sum_{j \in V, j \neq i} x_{ij} = 2y_{ii} \quad \forall i \in V$$

si i est une station ($y_{jj} = 1$) alors la somme est égal à 2 c'est à dire i a 2 voisins dans le cycle.

sinon la somme est égal à 0 –> aucune arête incidente choisie.

Contraintes de flot :
Le flot total qui sort de 1 vaut $p - 1$.
Le sommet 1 envoie un flot total de $p - 1$.

$$\sum_{j \in V \setminus \{1\}} z_{1j} = p - 1$$

Entrant = Sortant + Consommation.

$$\sum_{j \in V \setminus \{i\}} z_{ji} = \sum_{j \in V \setminus \{1,i\}} z_{ij} + y_{ii} \qquad \forall i \in V \setminus \{1\}$$

Si i est une station ($y_{ii} = 1$) : le flot "perd" une unité à i.
Si i n'est pas une station : $y_{ii} = 0$, donc l'entrant égal au sortant (c'est un simple transit).

Finalement, puisque on a commencer avec $p - 1$, on force le flot à visiter toutes les stations (sauf 1 "le sommet") pour consommer $p - 1$ unités.

Le flot ne peut passer que sur une arête sélectionnée :
Si $x_{ij} = 0$, alors aucun flot ne peut circuler sur l'arête signfie que :

$$z_{ij} = z_{ji} = 0.$$

Si $x_{ij} = 1$, alors on autorise le passage du flot avec une borne large :

$$z_{ij} + z_{ji} \leq (p-1)\, x_{ij}.$$

Ces contraintes garantissent l'existence d'un unique cycle contenant toutes les stations et éliminent les sous-cycles.

5.3.1 Contraintes d'intégrité

Les variables appartiennent aux domaines suivants :

$$x_{ij} \in \{0,1\} \quad \forall ij \in E,$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in V \times V,$$
$$z_{ij} \in [0, p-1] \quad \forall (i,j) \in V \times V \setminus \{1,j\}.$$

Le sommet 1 est imposé comme station :

$$y_{11} = 1, \qquad y_{1j} = 0 \quad \forall j \in V \setminus \{1\}.$$

5.4 Résultats

5.4.1 Résultats de la formulation compacte

La formulation compacte a été testée sur l'instance `berlin52` avec un nombre de stations dans notre cas on a pris $p = 9$ stations.
Le solveur retourne une solution optimale avec un coût total égal à :

$$10404.73$$

Les points choisis comme stations sont :

$$\{0,\ 1,5,\ 11,17,\ 18,19,\ 26,\ 47\}$$

De même, chaque point non-station est affecté à l'une de ces stations, de façon de minimiser la distance totale des affectations.

Selon notre solution les affectations des points aux stations sont les suivantes :

$(0->0)$, $(1->1)$, $(2->17)$, $(3->5)$, $(4->5)$, $(5->5)$, $(6->1)$, $(7->18)$, $(8->18)$,

$(9->18)$, $(10->11)$, $(11->11)$, $(12->26)$, $(13->26)$, $(14->5)$, $(15->19)$, $(16->17)$,

$(17->17),\ (18->18),\ (19->19),\ (20->17),\ (21->0),\ (22->19),\ (23->47),\ (24->5),$
$(25->26),\ (26->26),\ (27->26),\ (28->19),\ (29->19),\ (30->17),\ (31->0),\ (32->5),$
$(33->47),\ (34->0),\ (35->0),\ (36->47),\ (37->47),\ (38->47),\ (39->47),\ (40->18),$
$(41->1),\ (42->5),\ (43->0),\ (44->18),\ (45->47),\ (46->26),\ (47->47),\ (48->0),$
$(49->19),\ (50->11),\ (51->26)$

Les arêtes sélectionnées pour former le cycle entre les stations sont :

$$(0,17),\ (0,19),\ (1,19),\ (1,26),\ (5,11),\ (5,47),\ (11,26),\ (17,18),\ (18,47)$$

Ces arêtes définissent un unique cycle passant par l'ensemble des stations sélectionnées. Les contraintes de flot du problème garantissent l'absence de sous-cycles et assurent d'avoir un cycle.

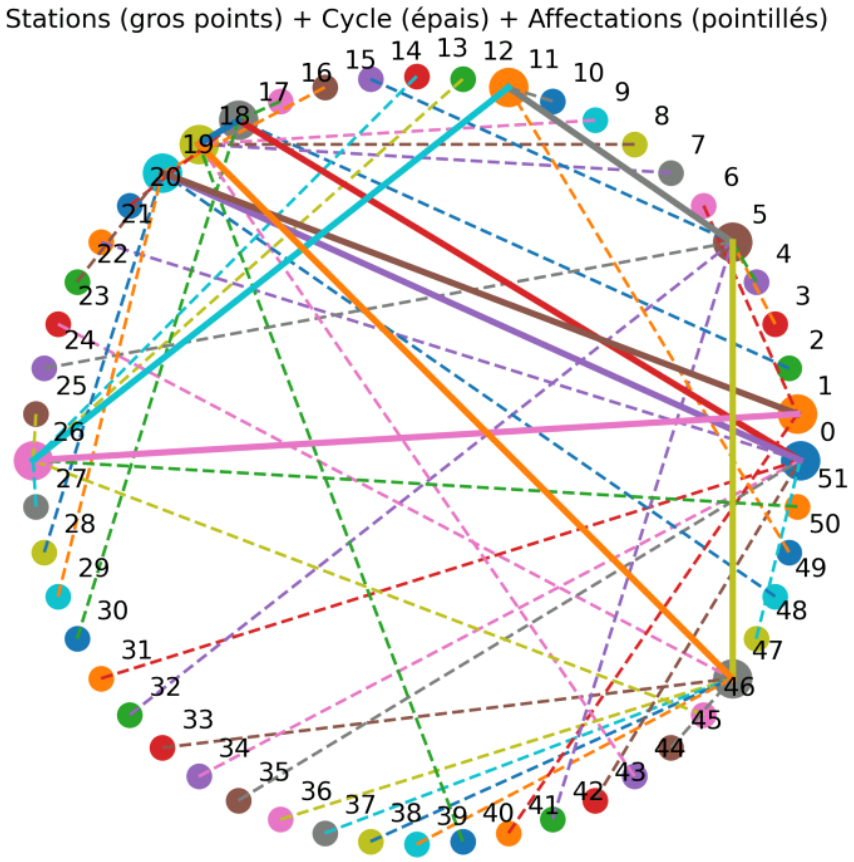

FIGURE 1 – Solution obtenue par la formulation compacte

D'après le résultat on remarque que les points sont globalement rattachés à la station la plus proche, ce qui est cohérent avec le but du problème , plus en détaille :

Les stations sont bien réparties.
Le cycle est unique et sans sous-cycles.

Chaque point est affecté à la station la plus proche.

Donc toutes les contraintes sont respectées ce qui confirme la validité de la formulation compacte et l'optimalité de notre solution .

6 Méthodes heuristiques

6.1 Principe générale de l'heuristique implémentée

L'heuristique générale du problème de *l'anneau-étoile (Ring-star problem)* que nous avons implémenté nous a servis à retrouver une solution réalisable, mais en aucun la plus optimale.

Son principe est simplement le suivant :

Il s'agit de la combinaison de deux sous-heuristiques qui s'appliquent l'une après l'autre : la denière étant l'heuristique du problème de *(Traveling Salesman Problem, TSP)* qui est appliqué après une première heuristique gloutonne du problème de p-médian,

6.2 L'heuristique gloutonne du p-médian

> ***Remarque :*** Notre implémentation utilise différentes structures de données (listes, dictionnaires et sets) gourmandes en mémoire ainsi que des parcours qui peuvent être facilement optimisés en utilisant des outils d'IA.

Il s'agit de la première étape de notre implémentation.

On se dispose tout d'abord d'un nuage de points éparpilés de façon aléatoire (pas forcément uniforme). La première chose serait de définir un cadre (ou un rectangle minimal général qui doit couvrir tous les points dont nous nous disposons). Ce rectangle général sera subdivisé en p **(étant le nombre de stations que nous cherchons à construire)** sous-rectangles de ***tailles égales (même hauteur et même largeur)***

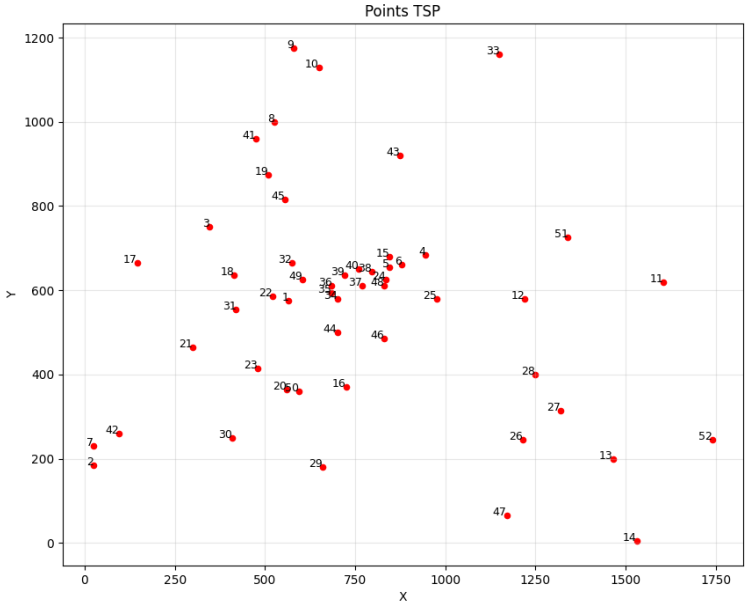

FIGURE 2 – Nuage des points de l'instance choisie

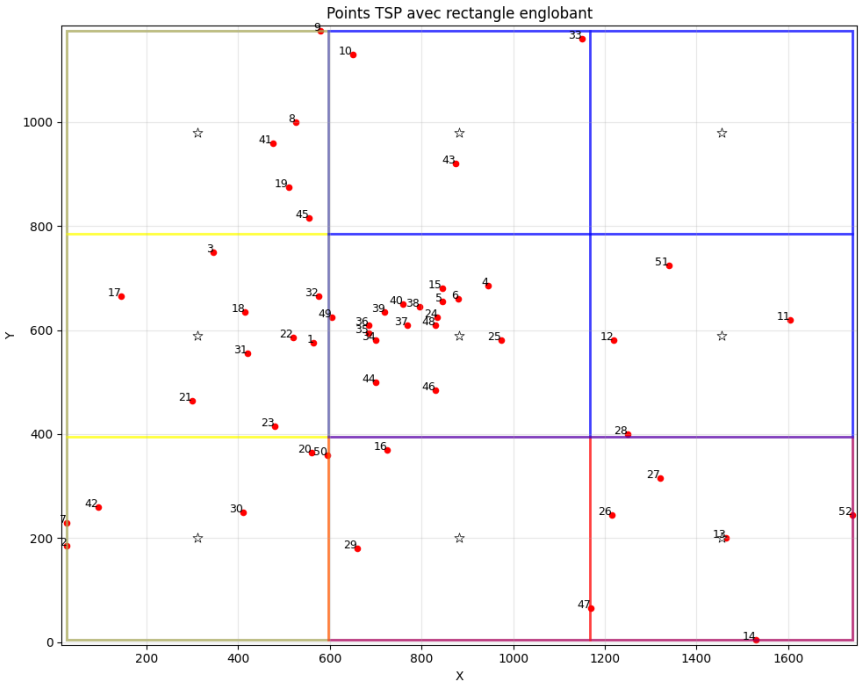

FIGURE 3 – Résultat du coupage pour l'heuristique gloutonne

Par définition, chaque sous-rectangle contiendra au minimum un point station par la logique qui suit **(le cas échéant est expliqué ci-dessous)** :

- 1 - On détermine le centre (x_c, y_c) de chaque sous-rectangle (par une simple formule qui somme le point inférieur gauche : avec la largeur, divisée par deux pour déterminer (x_c), et avec la longueur divisée par deux pour déterminer (y_c) ; chaque sous-rectangle est considéré comme un élement d'une liste.

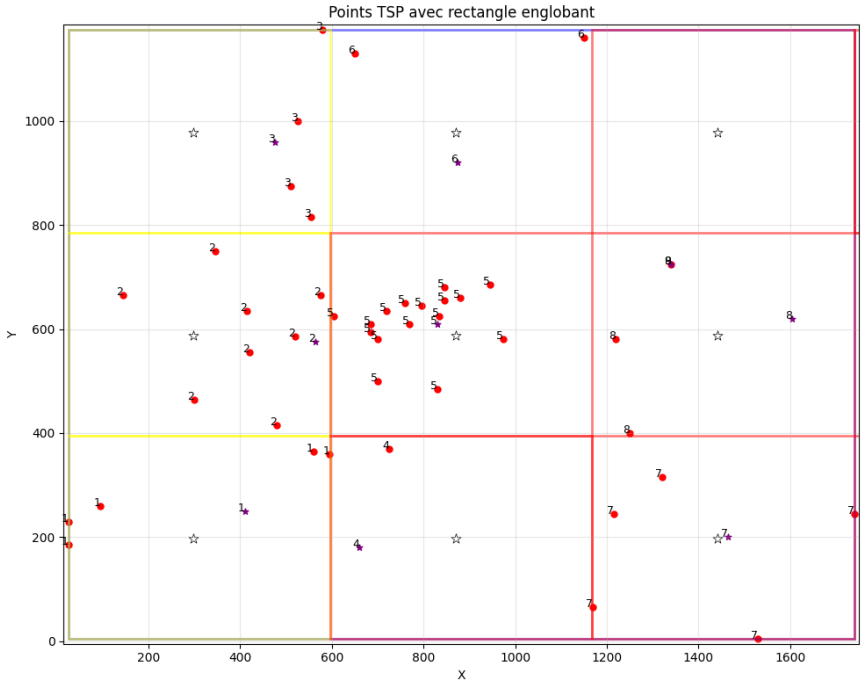

FIGURE 4 – Visualisation des stations

- 2 - On recense pour chaque sous-rectangle l'ensemble des points qui lui appartient (ceux avec des coordonnées qui ne dépassent pas ses bordures). Il s'agit d'une liste où chaque élément est un set de points.

Attention : le rectangle avec le premier sommet qui est défini par hypothèse comme station est exempté de cette étape.

Attention : Le cas où un rectangle est vide (aucun point ne lui appartient), on prend comme medium celui qui est le plus proche de son centre. Si ce point est déjà un medium, on réitère jusqu'à trouver un qui soit non-medium.

- 3 - On détermine pour chaque rectangle un et un seul point qui est le plus proche de son centre (en se basant sur la distance euclidienne) ; ce point est candidat pour être medium. On collectionne ces points dans une liste à part (qui sera par la suite parcouru pour le graphique)

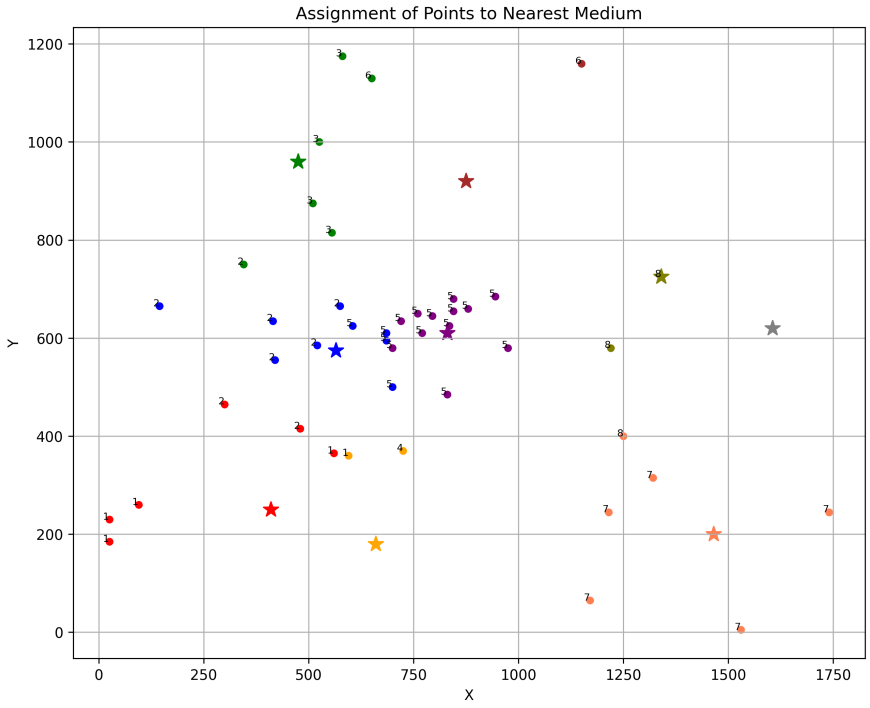

FIGURE 5 – Viusalisation des villes regroupés par station

- 4 - On affecte chaque point de l'ensemble des points non-medium à un point medium qui lui est le plus proche (toujours en utilisant la distance euclidienne)

6.3 L'heuristique du TSP

Il s'agit de la dernière étape de notre algorithme pour retrouver une solution au problème de Ring-Star.

6.3.1 L'heuristique du plus proche voisin

Le déroulement de l'algorithme, qui se base sur l'ensemble des p stations déjà retrouvés, est comme suit :

-1- On commence par le premier point medium (ex A), on cherche le point medium (station) non visité qui lui est le plus proche (B). On relie de façon qu'un trajet se crée avec comme départ le point A et comme arrivée le point B. Pour chaque segmement (trajet) on doit signaler que le point B a été visité, pour éviter de le revisiter de nouveau dans les prochains tours.

-2- On réitère jusqu'à ce que tous les points mediums soient visités.

-3- On retourne ensuite au point de départ A.

Ceci est representé sous forme d'une liaison liant chaque station avec un trait discontinu

NB : L'heuristique du plus proche voisin du TSP produit :
* Une tournée valide.
* Mais souvent avec des croisements (une étoile ou plusieurs sous-cycles)
* Solution non optimale

Pour résoudre cela, nous devons impérativement utiliser 2-OPT pour au final avoir un cycle

6.3.2 L'heuristique 2-OPT et la construction du cycle

L'algorithme 2-OPT est une **recherche locale** qui améliore une tournée.

Principe : si deux arêtes du cycle se croisent, alors les échanger réduit toujours la longueur totale.

- On part d'un cycle valide
- On enlève exactement **2 arêtes**
- On ajoute exactement **2 arêtes**
- Chaque sommet garde un degré 2
- Le graphe reste :
  → connexe
  → fermé
  → sans branche

- 4 - On lie maintenant tous les points non-medium (villes) ensemble par un trait continu.

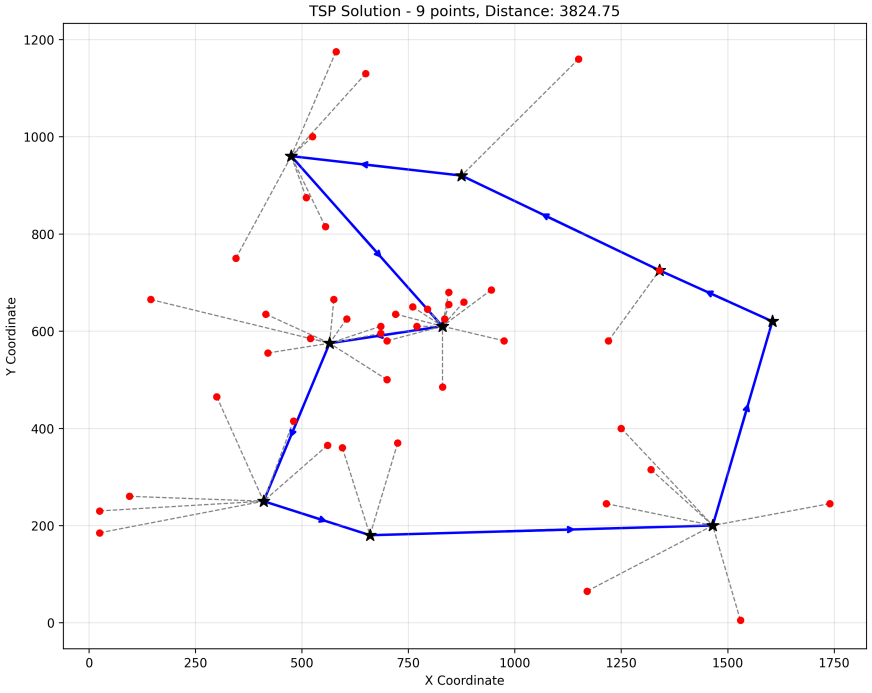

FIGURE 6 – Résultat de l'heuristique TSP (avec 2-OPT)

7 Méthodes métaheuristique

7.1 Introduction

Nous savons qu'une solution heuristique est une solution réalisable, possible mais pas la plus optimale. On pourra forcément faire mieux !

Une méthode métaheuristique va se baser sur la solution fournie par l'algorithme heuristique, et cherche à l'améliorer.

Nous avons choisis d'implémenter un algorithme de **Recherche locale stochastique (ou descente stochastique) (Randomized Local Search)**

7.2 Principe de l'algorithme Descente Stochastique

Notre algorithme est simple :

On réitère l'heuristique **Ring–Star**, on fait des substitutions aléatoires (on change une station par un point quelconque, on ne garde pas trace des mouvements déjà faits, et on retourne la meilleure solution rencontrée. Plus le nombre d'itérations est grand, plus on a la chance d'avoir un coût minimal.

> Lors de la substitution, si le point aléatoirement choisis pour devenir station est déjà une station, on réitère de nouveau jusqu'à ce qu'il soit un point ordinaire.

7.3 Résultat

Voici le résultat de notre problème, pour la même instance, après un nombre d'itérations de 1000.

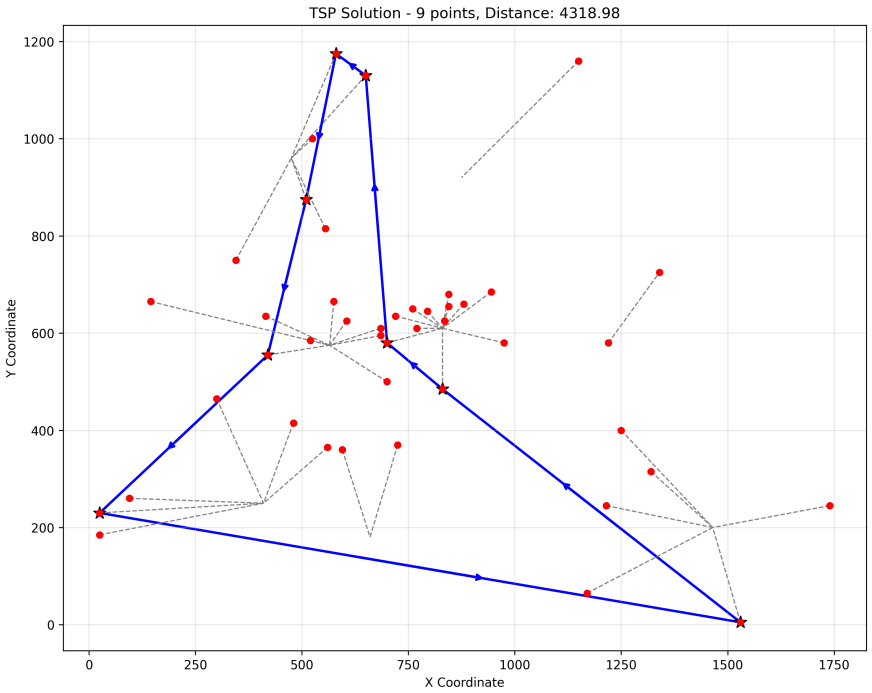

FIGURE 7 – Résultat de la métaheuristique

8 Discussion et comparaison

Dans cette partie finale, nous souhaitons comparer les différents résultats engendrés par les différentes approches.

Cette comparaison peut sembler être évidente, mais en fait, elle peut varier dépendant de la logique d'implémentation.

Nous tenons à rappeller que notre implémentation a utilisé plusieurs structures de données et son code est loin d'être optimisé pour avoir le temps d'exécution le plus minimal. Ces optimisations de code (bon choix ,réduction des structures de données et de variables) pourront être facilement faits à l'aide d'outils d'IAG.

Pour l'instance *berlin52.tsp*, nos constations sont les suivantes :

- L'heuristique a pris un peu plus de temps que la métaheuristique
- La formule compacte a un temps d'exécution quasi-exponentiel, s'élevant avec le nombre d'itérations. Dans notre contexte nous avons dû interrompre en limitant à un temps d'exécution bien établi (Nous recommandons de ce fait d'utiliser google colab qui a un solveur beaucoup plus performant pour PuLP et qui a permis de résoudre ce PLNE en quelques secondes)

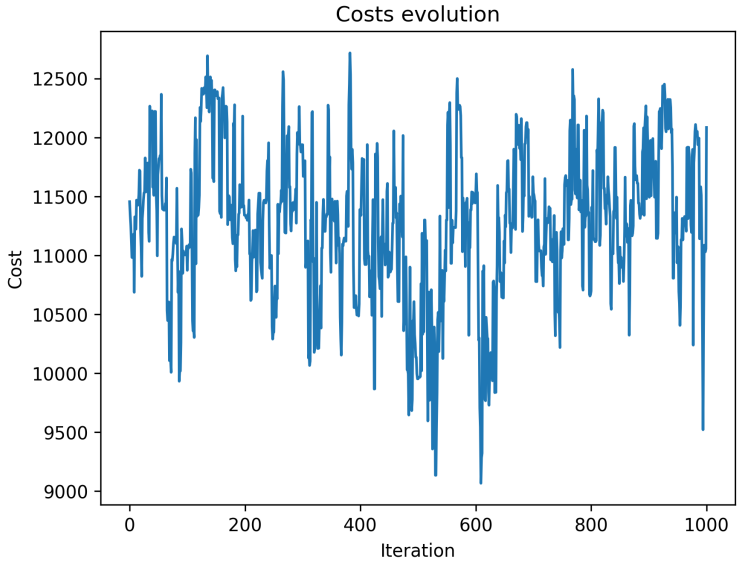

FIGURE 8 – Evolution du coût en fonction du nombre d'itérations de la métaheuristique

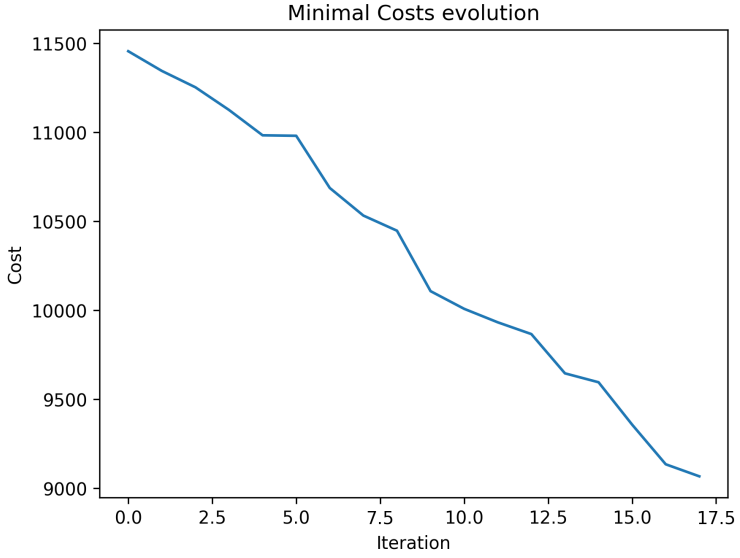

FIGURE 9 – Evolution du meilleur coût trouvé par la métaheuristique

En lancant les algorithmes, on peut trouver le temps d'exéuction de chaque algorithme dans le fichier **timing_results.txt**

Approche	**Temps d'exécution**	**Coût de la solution**	**Complexité en espace**	**Structure de données**
Métaheuristique	5s	environ 3000	Modérée	Listes, ensembles, dictionnaires
Heuristique	9.5s	11500	Modérée	Listes simples, tableaux
PLNE	11s (mais borné) (exponentiel)	Proche de l'optimal	Élevée	Matrices, graphes, solveur

TABLE 1 – Comparaison des méthodes de résolution pour le problème Ring–Star

Il peut sembler étrange q'une heuristique prend plus de temps qu'une métaheuristique. Mais dans notre implémentation, on a procédé à dessiner les rectangles, les mediums avec pas mal de boucles, dont certaines sont imbriquées. De plus, l'heuristique reconstruit la solution à chaque appel (recalcul du tsp, p-médian, et 2-OPT) alors que la métaheuristique ne modifie que **localement** la solution.

Ceci est la raison derrière le ralentissement en temps d'exécution.

Il est indéniable que le **PLNE** prendra le plus de temps, étant donnée que le domaine est extrêmement large.

9 Conclusion

En guise de conclusion, ce projet a constitué une excellente occassion de consolider nos connaissances acquises théoriquement et de les appliquer dans un vrai problème réaliste.

Plusieurs pistes d'amélioration sont envisageables, notamment l'amélioration de la qualité de notre code, utiliser une métaheuristique tabou...