



Le génie pour l'industrie

SYS817-01

**Systèmes de distribution et de transport intelligents**

**Devoir 1**

Travail présenté à :

Mustapha Ouhimmou

Fadwa Maslouhi

Par :

Achraf Moustafi – MOUA23299901

Pierre-Adrien Lefèvre – LEFP14059805

Le

01 Mars 2024

Session : Hiver 2024

# TABLE DES FIGURES

1.1	Figure après l'itération 1	6
1.2	Figure après la dernière itération	6
1.3	Figure après l'itération 1	7
1.4	Figure après l'itération 6	7
1.5	Figure après la dernière itération	8
1.6	Figure après l'itération 1	8
1.7	Figure après l'itération 5	9
1.8	Figure après la dernière itération	9
1.9	Figure après l'itération 1	10
1.10	Figure après l'itération 5	10
1.11	Figure après la dernière itération	11
1.12	Figure après l'itération 1	11
1.13	Figure après l'itération 5	12
1.14	Figure après la dernière itération	12
1.15	Arbre de départ	13
1.16	Arbre de recouvrement minimum ( MST ) avec algorithme Kruskal	13
1.17	Arbre de recouvrement minimum (MST) avec arêtes doublées	14
1.18	Chemin final de l'algorithme Double tree	14
3.1	Visualiation des clients avec les fenêtres de temps	33
3.2	Visuliation des itinéraires des véhicules	34
4.1	Meilleure solution connu	38
4.2	ANLS Simple	39
4.3	String removals	39
4.4	Meilleure solution connu	40
4.5	ANLS Simple	40
4.6	String removals	41
4.7	Simple ANLS	41
4.8	String removals	42
4.9	Meilleure solution connu	42
4.10	ANLS Simple	43
4.11	String removals	43

# LISTE DES TABLEAUX

1.1	Matrice des distances . . . . .	3
1.2	Matrice des distances entre les points . . . . .	4
1.3	Récapitulatif des performances des algorithmes . . . . .	15
2.1	Récapitulatif des performances des algorithmes pour 14 villes . . . . .	20
2.2	Récapitulatif des performances des algorithmes pour 150 villes . . . . .	23
2.3	Récapitulatif des performances des algorithmes pour 1037 villes . . . . .	27

# TABLE DES MATIÈRES

<b>1 Application de l'heuristique de Clarke et Wright pour le Routage des Véhicules</b>	<b>2</b>
Introduction . . . . .	3
1.1 Application de l'heuristique de Clarke et Wright . . . . .	3
1.1.1 Matrice des distances : . . . . .	3
1.1.2 Matrice des économies : . . . . .	3
1.2 Application de l'heuristique du plus proche voisin : . . . . .	7
1.3 Application de l'heuristique Insertion la plus proche : . . . . .	8
1.3.1 Itération 1 : . . . . .	8
1.3.2 Itération 5 : . . . . .	9
1.3.3 Itération 9 : . . . . .	9
1.4 Application de l'heuristique Insertion la plus éloignée : . . . . .	10
1.4.1 Itération 1 : . . . . .	10
1.4.2 Itération 5 : . . . . .	10
1.4.3 Itération 9 : . . . . .	11
1.5 Minimisé le cout d'insertion : . . . . .	11
1.5.1 Itération 1 : . . . . .	11
1.5.2 Itération 5 : . . . . .	12
1.5.3 Itération 9 : . . . . .	12
1.6 Algorithme de Kruskal et double Tree : . . . . .	13
1.7 Tableau récapitulatif des différentes distances : . . . . .	15
<b>2 Heuristiques Avancées pour le Problème du Voyageur de Commerce</b>	<b>16</b>
Introduction . . . . .	18
2.1 Avec 14 villes . . . . .	18
2.1.1 Clarke et Wright . . . . .	18
2.1.1.1 2-OPT . . . . .	18
2.1.1.2 3-OPT . . . . .	18
2.1.2 Plus proches voisins : . . . . .	18
2.1.2.1 2-OPT . . . . .	18
2.1.2.2 3-OPT . . . . .	19
2.1.3 Insertion la plus éloignée . . . . .	19
2.1.3.1 2-OPT . . . . .	19

2.1.3.2	3-OPT . . . . .	19
2.1.4	Tabou . . . . .	19
2.1.5	Concorde . . . . .	19
2.1.6	Tableau récapitulatif pour 14 villes et analyse de résultat : . . . . .	20
2.2	Avec 150 villes . . . . .	20
2.2.1	Clark and Wright : . . . . .	20
2.2.1.1	2-OPT : . . . . .	21
2.2.1.2	3-OPT : . . . . .	21
2.2.2	Plus proches voisins : . . . . .	21
2.2.2.1	2-OPT : . . . . .	21
2.2.2.2	3-OPT : . . . . .	22
2.2.3	Insertion la plus éloignée : . . . . .	22
2.2.3.1	2-OPT : . . . . .	22
2.2.3.2	3-OPT : . . . . .	22
2.2.4	Tabou : . . . . .	23
2.2.5	Concorde : . . . . .	23
2.2.6	Tableau récapitulatif pour 150 villes et analyse des résultats : . . . . .	23
2.3	Avec 1037 villes . . . . .	24
2.3.1	Clark and Wright : . . . . .	24
2.3.1.1	2-OPT : . . . . .	24
2.3.1.2	3-OPT : . . . . .	24
2.3.2	Plus proches voisins : . . . . .	25
2.3.2.1	2-OPT : . . . . .	25
2.3.2.2	3-OPT : . . . . .	25
2.3.3	Insertion la plus éloignée : . . . . .	25
2.3.3.1	2-OPT : . . . . .	25
2.3.3.2	3-OPT : . . . . .	25
2.3.4	Tabou : . . . . .	25
2.3.5	Concorde : . . . . .	26
2.3.6	Tableau récapitulatif pour 1037 villes : . . . . .	27
<b>3</b>	<b>Variantes du Problème de Routage de Véhicules</b>	<b>29</b>
	Introduction . . . . .	30
3.1	Définition de la variante : . . . . .	30
3.2	Formulation Mathématiques : . . . . .	31
3.2.1	Définition : . . . . .	31
3.2.2	Objectif : Minimiser la somme des coûts de toutes les liaisons utilisées : . . . . .	31
3.2.3	Contraintes . . . . .	31
3.3	Code : . . . . .	33
3.4	Résolution : . . . . .	34
3.5	Explication : . . . . .	35

<b>4 Approches Métaheuristiques du Problème de Routage de Véhicules avec Capacité Limitée</b>	<b>36</b>
Introduction	37
4.1 Explication de l'ALNS	37
4.2 Réponses avec différents jeux de données	38
4.2.1 Avec 32 villes :	38
4.2.2 Avec 242 villes :	40
4.2.3 Avec 1001 points :	42
4.3 Analyse des Performances	43
4.3.1 Avec 32 Villes	43
4.3.2 Avec 242 Villes	44
4.3.3 Avec 1001 Points	44

# CHAPITRE 1

## APPLICATION DE L'HEURISTIQUE DE CLARKE ET WRIGHT POUR LE ROUTAGE DES VÉHICULES

### Sommaire

---

<b>Introduction</b>	3
<b>1.1 Application de l'heuristique de Clarke et Wright</b>	3
1.1.1 Matrice des distances :	3
1.1.2 Matrice des économies :	3
<b>1.2 Application de l'heuristique du plus proche voisin :</b>	7
<b>1.3 Application de l'heuristique Insertion la plus proche :</b>	8
1.3.1 Itération 1 :	8
1.3.2 Itération 5 :	9
1.3.3 Itération 9 :	9
<b>1.4 Application de l'heuristique Insertion la plus éloignée :</b>	10
1.4.1 Itération 1 :	10
1.4.2 Itération 5 :	10
1.4.3 Itération 9 :	11
<b>1.5 Minimisé le cout d'insertion :</b>	11
1.5.1 Itération 1 :	11
1.5.2 Itération 5 :	12
1.5.3 Itération 9 :	12
<b>1.6 Algorithme de Kruskal et double Tree :</b>	13
<b>1.7 Tableau récapitulatif des différentes distances :</b>	15

---

## Introduction

Dans le contexte de notre exercice visant à résoudre le problème du voyageur de commerce à l'aide de différentes heuristiques, nous avons adopté une approche en générant les coordonnées des villes de manière aléatoire via le langage de programmation Python à l'aide OpenAI (2024) , en utilisant spécifiquement sa bibliothèque random.

Le point de départ de notre analyse, le dépôt situé à la coordonnée (21, 30), ainsi que les emplacements des neuf autres villes, ont été déterminés en générant aléatoirement des valeurs dans un intervalle compris entre 1 et 50. Cette démarche garantit l'unicité de notre jeu de données et souligne l'importance de la flexibilité et de l'adaptabilité des méthodes heuristiques dans la résolution de problèmes d'optimisation de tournées.

L'objectif principal de cet exercice est de mettre en évidence l'efficacité des différentes l'heuristique lorsqu'elles sont appliquées à des données générées de manière aléatoire, reflétant ainsi les défis rencontrés dans les applications réelles de logistique et de gestion des transports. En utilisant Python comme outil de simulation et d'analyse, nous visons à développer une compréhension approfondie des mécanismes sous-jacents des heuristiques, tout en comparant leurs solutions.

Ville 1 à, (21, 30) Ville 2 à (39, 15), Ville 3 à (31, 45), Ville 4 à (35, 41), Ville 5 à (35, 20), Ville 6 à (6, 46), Ville 7 à (28, 33), Ville 8 à (13, 16), Ville 9 à (1, 31), et Ville 10 à (21, 34). .

### 1.1 Application de l'heuristique de Clarke et Wright

#### 1.1.1 *Matrice des distances :*

Voici la matrice des distances sous forme de tableau :

TABLE 1.1 – Matrice des distances

Point	0	1	2	3	4	5	6	7	8	9
0	0.00000	23.4307	18.0278	17.8045	17.2047	21.9317	7.61577	16.1245	20.0250	4.00000
1	23.4307	0.00000	31.0483	26.3059	6.40312	45.2769	21.0950	26.0192	41.2311	26.1725
2	18.0278	31.0483	0.00000	5.65685	25.3180	25.0200	12.3693	34.1321	33.1059	14.8661
3	17.8045	26.3059	5.65685	0.00000	21.0000	29.4279	10.6301	33.3017	35.4401	15.6525
4	17.2047	6.40312	25.3180	21.0000	0.00000	38.9487	14.7648	22.3607	35.7351	19.7990
5	21.9317	45.2769	25.0200	29.4279	38.9487	0.00000	25.5539	30.8058	15.8114	19.2094
6	7.61577	21.0950	12.3693	10.6301	14.7648	25.5539	0.00000	22.6716	27.0740	7.07107
7	16.1245	26.0192	34.1321	33.3017	22.3607	30.8058	22.6716	0.00000	19.2094	19.6977
8	20.0250	41.2311	33.1059	35.4401	35.7351	15.8114	27.0740	19.2094	0.00000	20.2237
9	4.00000	26.1725	14.8661	15.6525	19.7990	19.2094	7.07107	19.6977	20.2237	0.00000

#### 1.1.2 *Matrice des économies :*

Voici la matrice des distances sous forme de tableau :

TABLE 1.2 – Matrice des distances entre les points

	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>
<b>0</b>	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
<b>1</b>	0.0	46.8614	10.4102	14.9293	34.23228	0.0855	9.95147	13.5360	2.2246	1.2582
<b>2</b>	0.0	10.4102	36.0556	30.17545	9.9145	14.9395	13.27427	0.0202	4.9469	7.1617
<b>3</b>	0.0	14.9293	30.17545	35.6090	14.0092	10.3083	14.79017	0.6273	2.3894	6.1520
<b>4</b>	0.0	34.23228	9.9145	14.0092	34.4094	0.1877	10.05567	10.9685	1.4946	1.4057
<b>5</b>	0.0	0.0855	14.9395	10.3083	0.1877	43.8634	3.99357	7.2504	26.1453	6.7223
<b>6</b>	0.0	9.95147	13.27427	14.79017	10.05567	3.99357	15.23154	1.06867	0.56677	4.5447
<b>7</b>	0.0	13.5360	0.0202	0.6273	10.9685	7.2504	1.06867	32.2490	16.9401	0.4268
<b>8</b>	0.0	2.2246	4.9469	2.3894	1.4946	26.1453	0.56677	16.9401	40.0500	3.8013
<b>9</b>	0.0	1.2582	7.1617	6.1520	1.4057	6.7223	4.5447	0.4268	3.8013	8.0

Voici les différentes itérations :

#### Itération 0 :

- Itinéraire avant fusion #1 : [1]
- Itinéraire avant fusion #2 : [2]
- Itinéraire avant fusion #3 : [3]
- Itinéraire avant fusion #4 : [4]
- Itinéraire avant fusion #5 : [5]
- Itinéraire avant fusion #6 : [6]
- Itinéraire avant fusion #7 : [7]
- Itinéraire avant fusion #8 : [8]
- Itinéraire avant fusion #9 : [9]

#### Itération 1 :

- Après fusion des itinéraires pour les clients 1 et 4
- Itinéraire après fusion #1 : [1, 4]
- Itinéraire après fusion #2 : [2]
- Itinéraire après fusion #3 : [3]
- Itinéraire après fusion #4 : None
- Itinéraire après fusion #5 : [5]
- Itinéraire après fusion #6 : [6]
- Itinéraire après fusion #7 : [7]
- Itinéraire après fusion #8 : [8]
- Itinéraire après fusion #9 : [9]

#### Itération 2 :

- Après fusion des itinéraires pour les clients 2 et 3
- Itinéraire après fusion #1 : [1, 4]
- Itinéraire après fusion #2 : [2, 3]
- Itinéraire après fusion #3 : None
- Itinéraire après fusion #4 : None
- Itinéraire après fusion #5 : [5]
- Itinéraire après fusion #6 : [6]
- Itinéraire après fusion #7 : [7]
- Itinéraire après fusion #8 : [8]
- Itinéraire après fusion #9 : [9]

#### Itération 3 :

- Après fusion des itinéraires pour les clients 5 et 8
- Itinéraire après fusion #1 : [1, 4]
- Itinéraire après fusion #2 : [2, 3]
- Itinéraire après fusion #3 : None
- Itinéraire après fusion #4 : None
- Itinéraire après fusion #5 : [5, 8]
- Itinéraire après fusion #6 : [6]
- Itinéraire après fusion #7 : [7]
- Itinéraire après fusion #8 : None
- Itinéraire après fusion #9 : [9]

**Itération 4 :**

Après fusion des itinéraires pour les clients 7 et 8

Itinéraire après fusion #1 : [1, 4]

Itinéraire après fusion #2 : [2, 3]

Itinéraire après fusion #3 : None

Itinéraire après fusion #4 : None

Itinéraire après fusion #5 : None

Itinéraire après fusion #6 : [6]

Itinéraire après fusion #7 : [7, 8, 5]

Itinéraire après fusion #8 : None

Itinéraire après fusion #9 : [9]

**Itération 5 :**

Après fusion des itinéraires pour les clients 2 et 5

Itinéraire après fusion #1 : [1, 4]

Itinéraire après fusion #2 : [3, 2, 5, 8, 7]

Itinéraire après fusion #3 : None

Itinéraire après fusion #4 : None

Itinéraire après fusion #5 : None

Itinéraire après fusion #6 : [6]

Itinéraire après fusion #7 : None

Itinéraire après fusion #8 : None

Itinéraire après fusion #9 : [9]

**Itération 6 :**

Après fusion des itinéraires pour les clients 1 et 3

Itinéraire après fusion #1 : [4, 1, 3, 2, 5, 8, 7]

Itinéraire après fusion #2 : None

Itinéraire après fusion #3 : None

Itinéraire après fusion #4 : None

Itinéraire après fusion #5 : None

Itinéraire après fusion #6 : [6]

Itinéraire après fusion #7 : None

Itinéraire après fusion #8 : None

Itinéraire après fusion #9 : [9]

**Itération 7 :**

Après fusion des itinéraires pour les clients 4 et 6

Itinéraire après fusion #1 : [7, 8, 5, 2, 3, 1, 4, 6]

Itinéraire après fusion #2 : None

Itinéraire après fusion #3 : None

Itinéraire après fusion #4 : None

Itinéraire après fusion #5 : None

Itinéraire après fusion #6 : None

Itinéraire après fusion #7 : None

Itinéraire après fusion #8 : None

Itinéraire après fusion #9 : [9]

**Itération 8 :**

Après fusion des itinéraires pour les clients 6 et 9

Itinéraire après fusion #1 : [7, 8, 5, 2, 3, 1, 4, 6, 9]

Itinéraire après fusion #2 : None

Itinéraire après fusion #3 : None

Itinéraire après fusion #4 : None

Itinéraire après fusion #5 : None

Itinéraire après fusion #6 : None

Itinéraire après fusion #7 : None

Itinéraire après fusion #8 : None

Itinéraire après fusion #9 : None

La route est [1, 8, 9, 6, 3, 4, 2, 5, 7, 10, 1] avec un parcours de 140.36703075108707

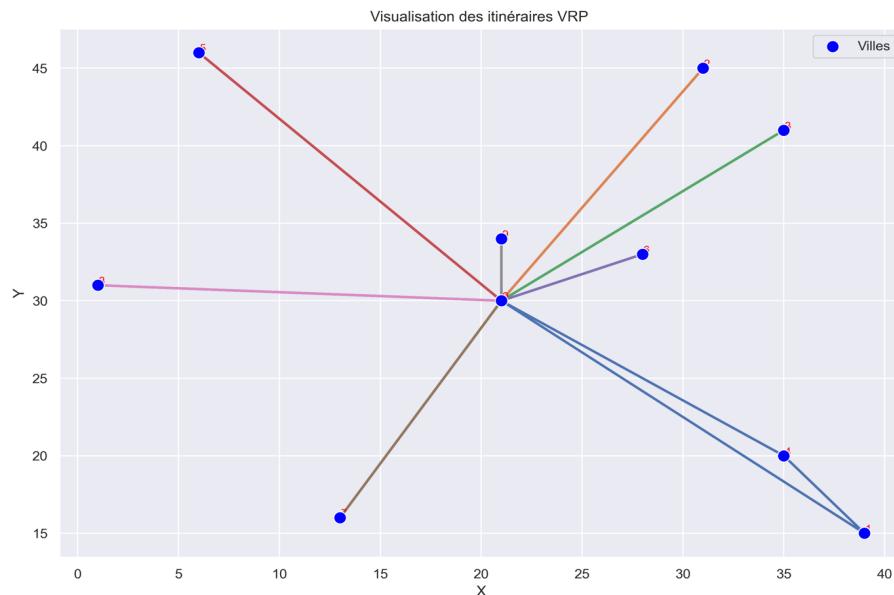


FIGURE 1.1 – Figure après l’itération 1

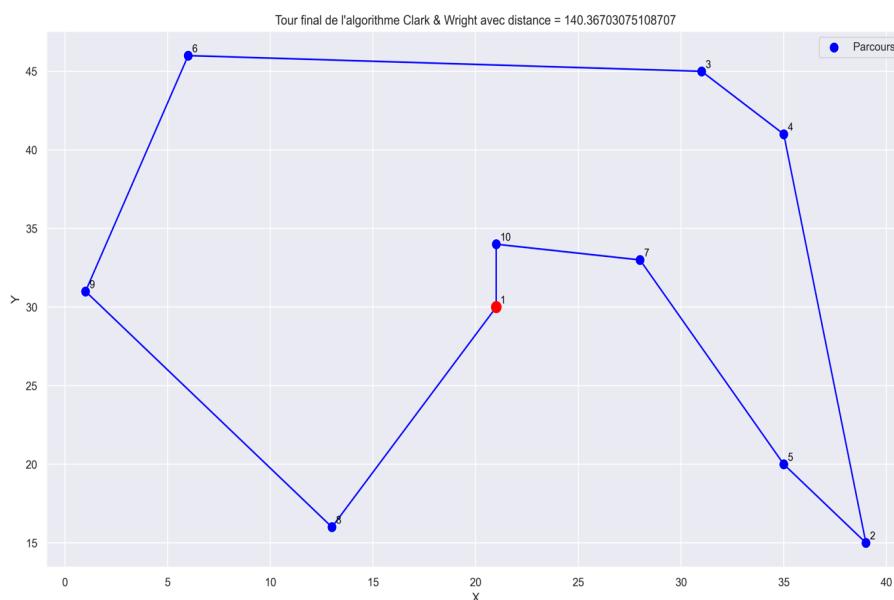


FIGURE 1.2 – Figure après la dernière itération

## 1.2 Application de l'heuristique du plus proche voisin :

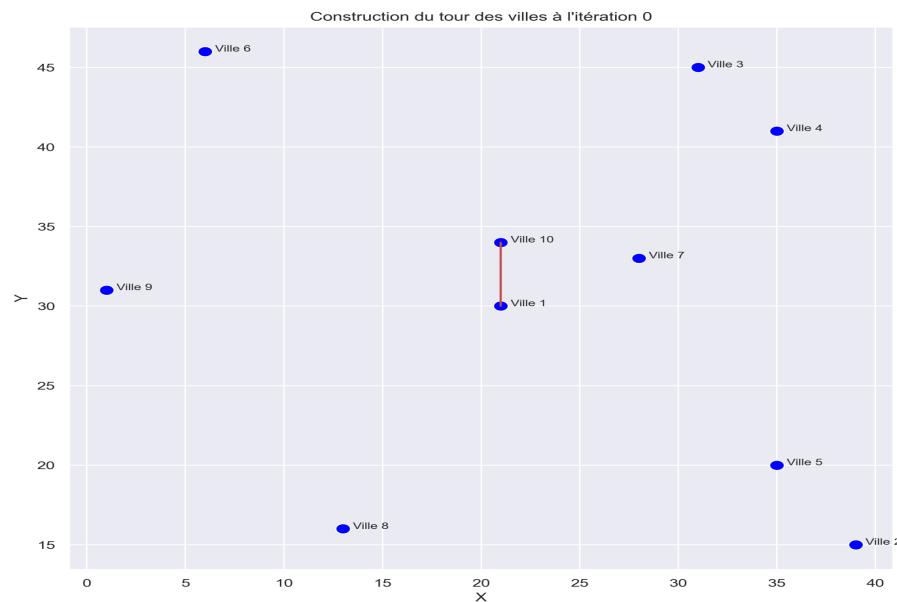


FIGURE 1.3 – Figure après l'itération 1

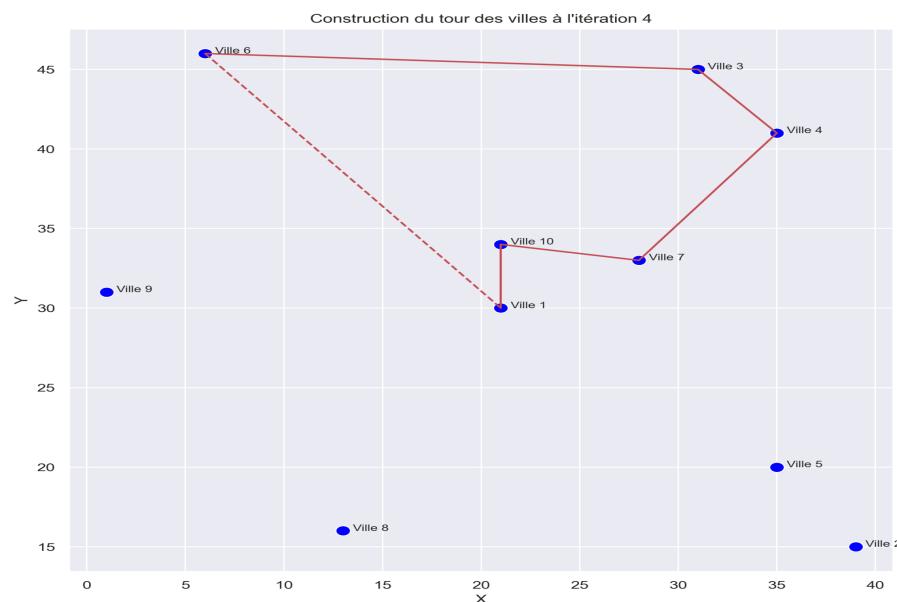


FIGURE 1.4 – Figure après l'itération 6

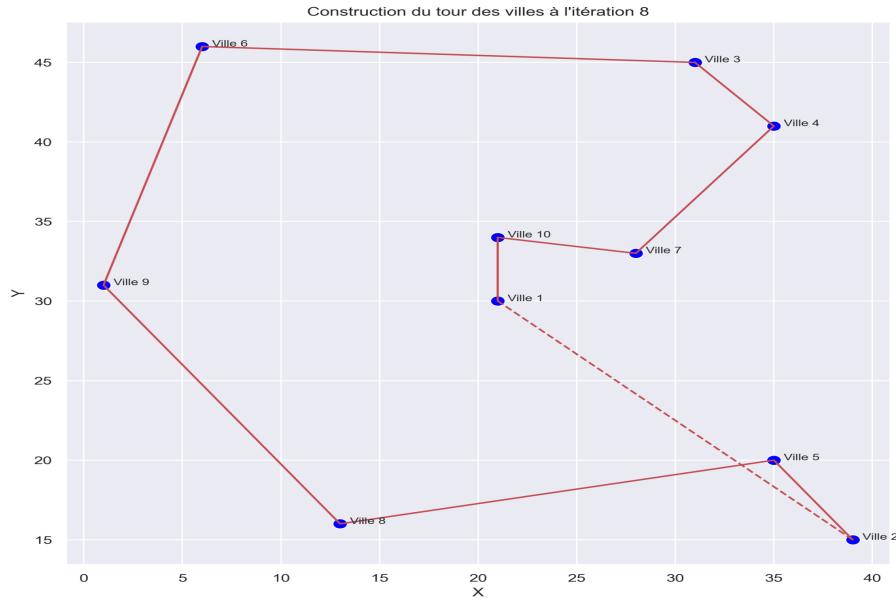


FIGURE 1.5 – Figure après la dernière itération

Tour : [0, 9, 6, 3, 2, 5, 8, 7, 4, 1, 0]

Distance : 139.59337393377726

### 1.3 Application de l'heuristique Insertion la plus proche :

#### 1.3.1 Itération 1 :

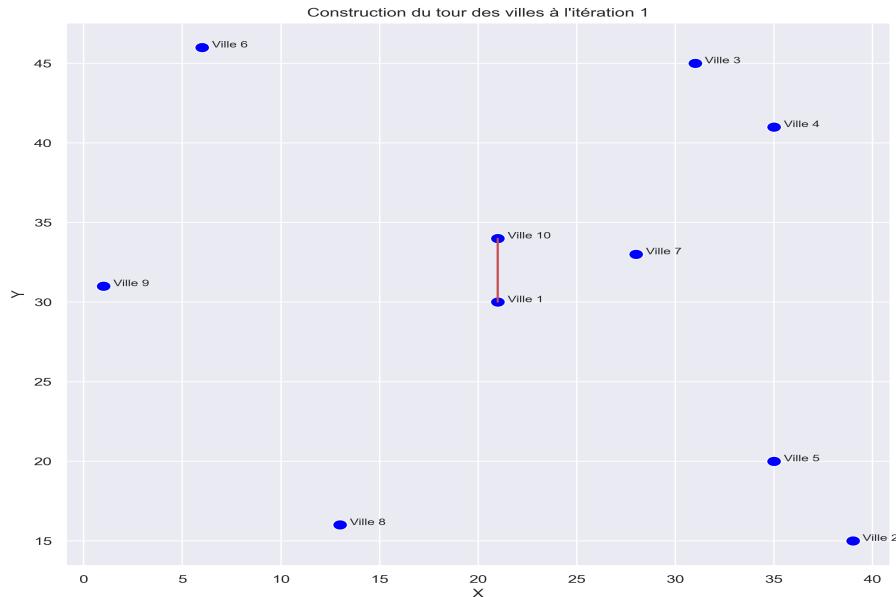


FIGURE 1.6 – Figure après l'itération 1

### 1.3.2 Itération 5 :

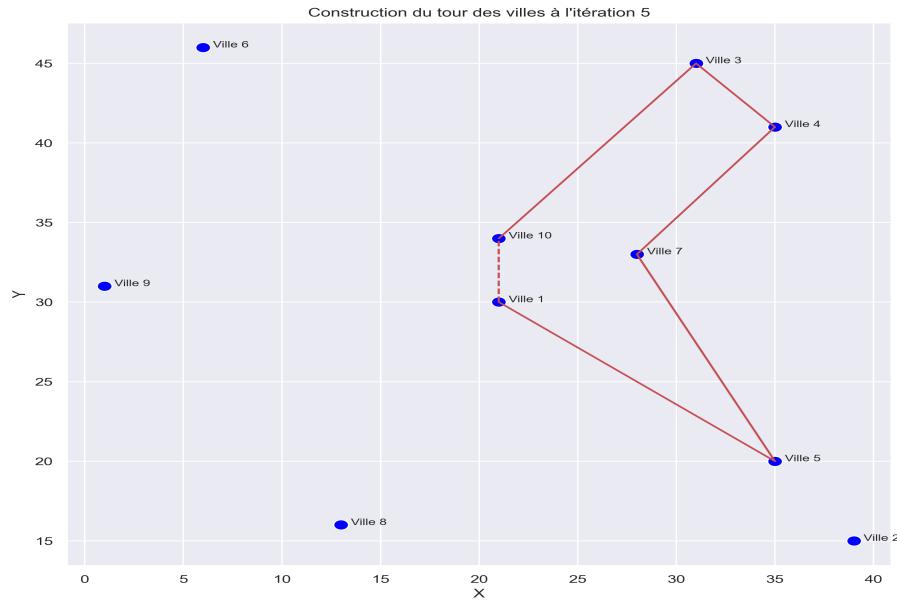


FIGURE 1.7 – Figure après l'itération 5

### 1.3.3 Itération 9 :

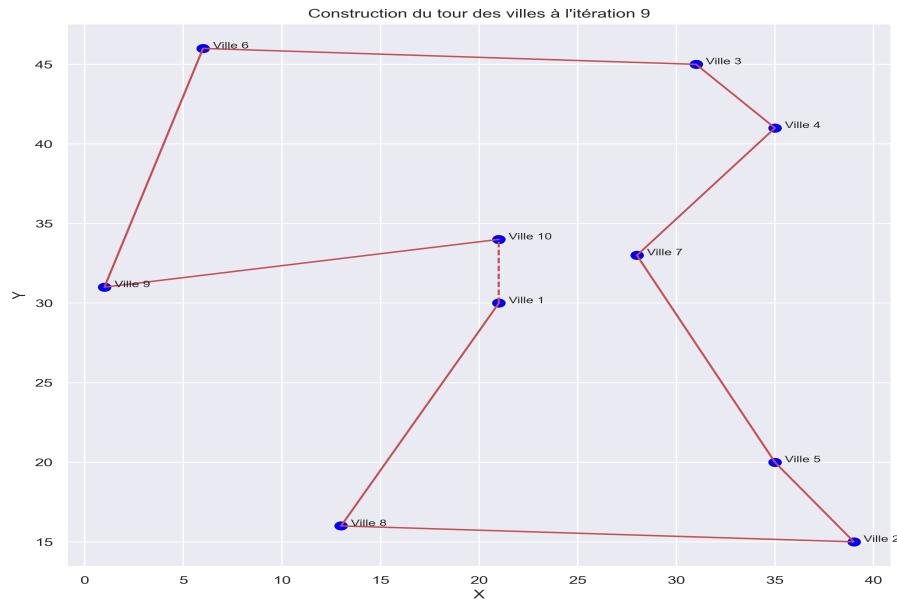


FIGURE 1.8 – Figure après la dernière itération

Tour final : [9, 8, 5, 2, 3, 6, 4, 1, 7, 0]

Coût total du tour final : 140.65381524239794

## 1.4 Application de l'heuristique Insertion la plus éloignée :

### 1.4.1 Itération 1 :

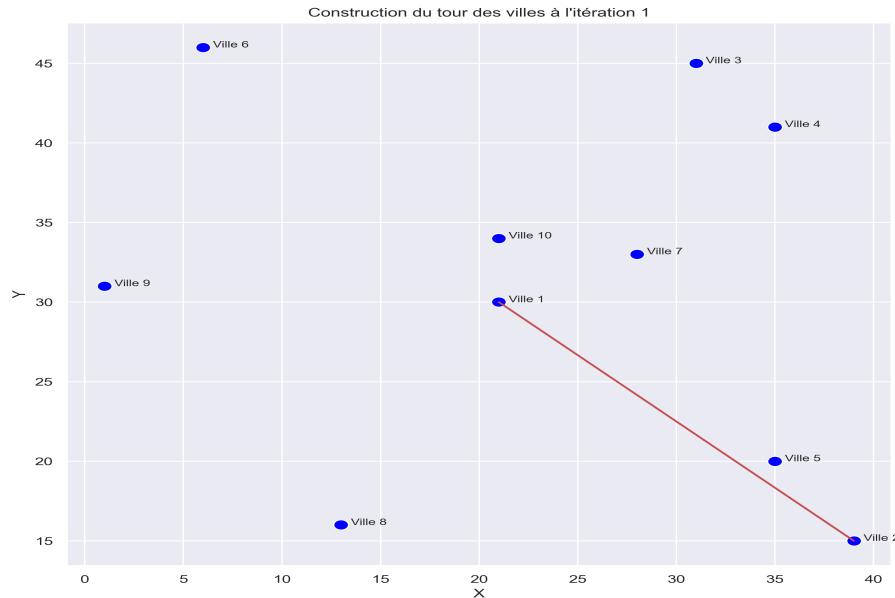


FIGURE 1.9 – Figure après l'itération 1

### 1.4.2 Itération 5 :

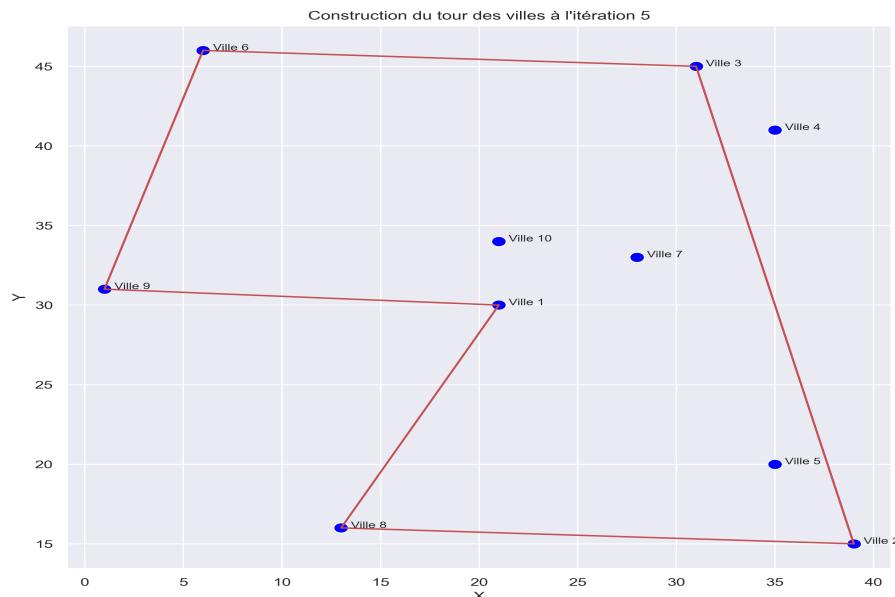


FIGURE 1.10 – Figure après l'itération 5

### 1.4.3 Itération 9 :

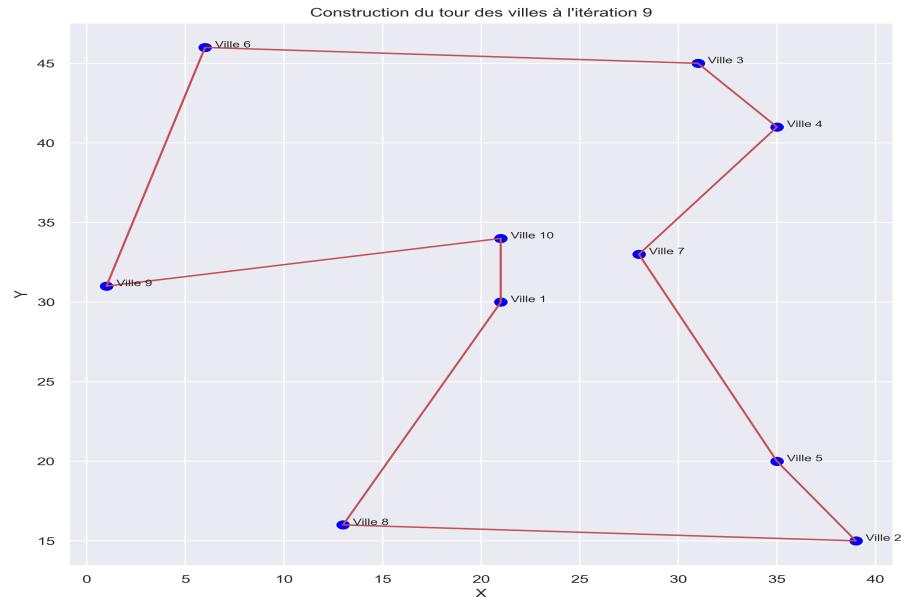


FIGURE 1.11 – Figure après la dernière itération

Tour final : [0, 9, 8, 5, 2, 3, 6, 4, 1, 7, 0]

Coût total du tour final : 144.65381524239794

### 1.5 Minimisé le cout d'insertion :

#### 1.5.1 Itération 1 :

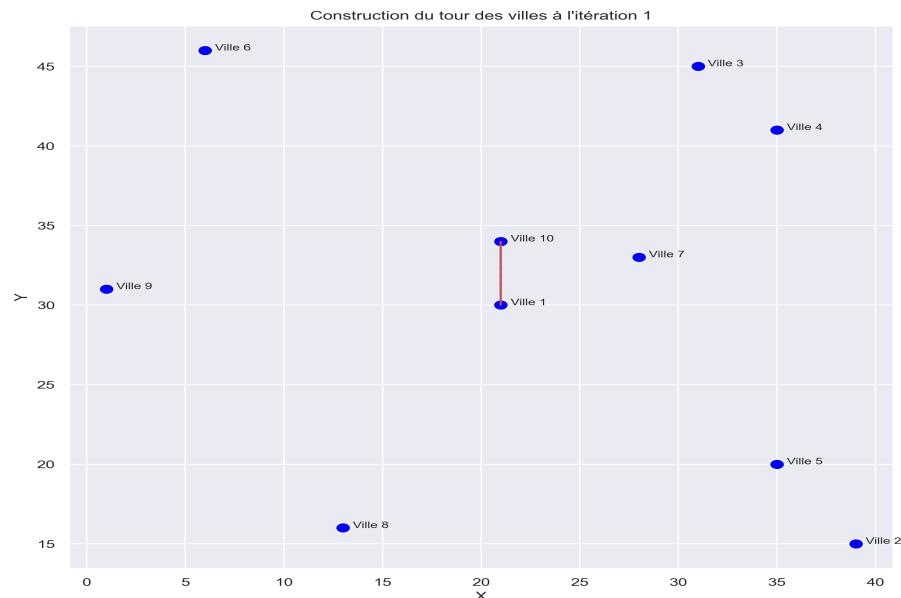


FIGURE 1.12 – Figure après l'itération 1

### 1.5.2 Itération 5 :

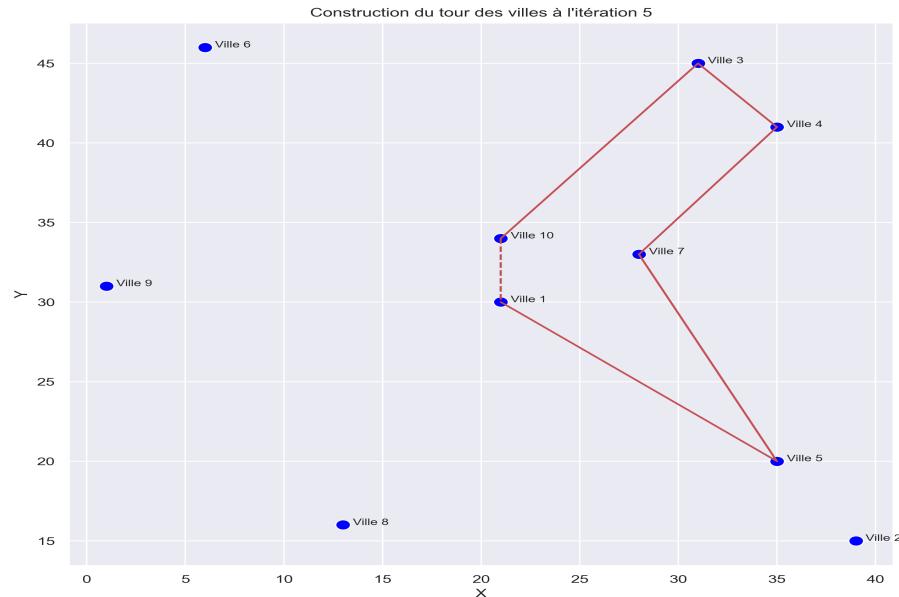


FIGURE 1.13 – Figure après l'itération 5

### 1.5.3 Itération 9 :

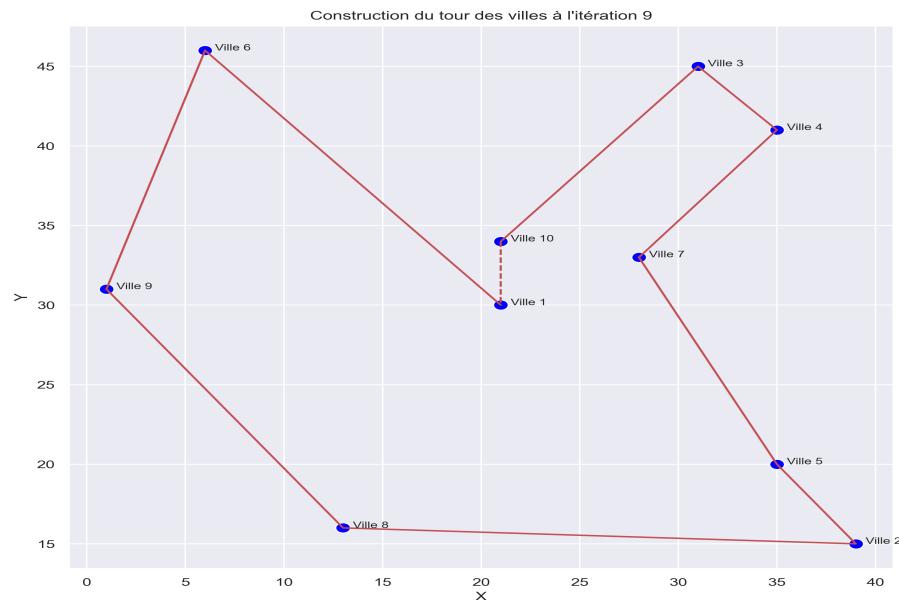


FIGURE 1.14 – Figure après la dernière itération

Tour final : [0, 5, 8, 7, 1, 4, 6, 3, 2, 9, 0]

Coût total du tour final : 139.2927129823289

### 1.6 Algorithme de Kruskal et double Tree :

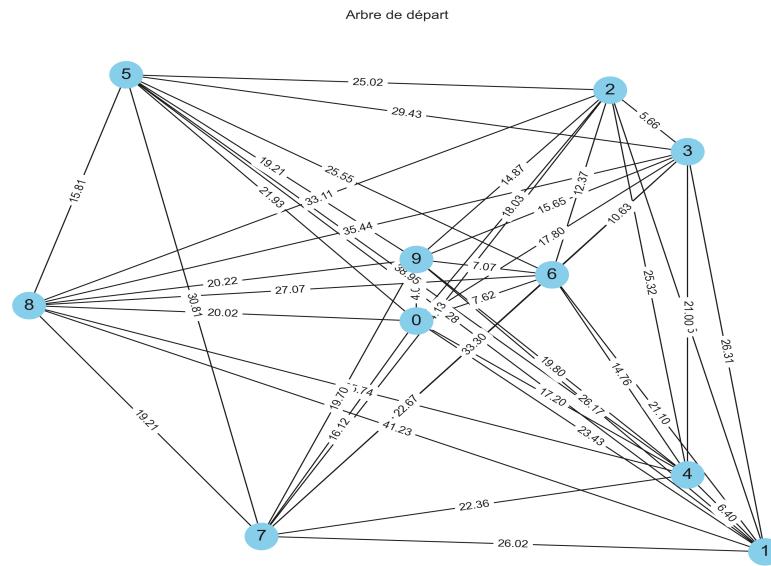


FIGURE 1.15 – Arbre de départ

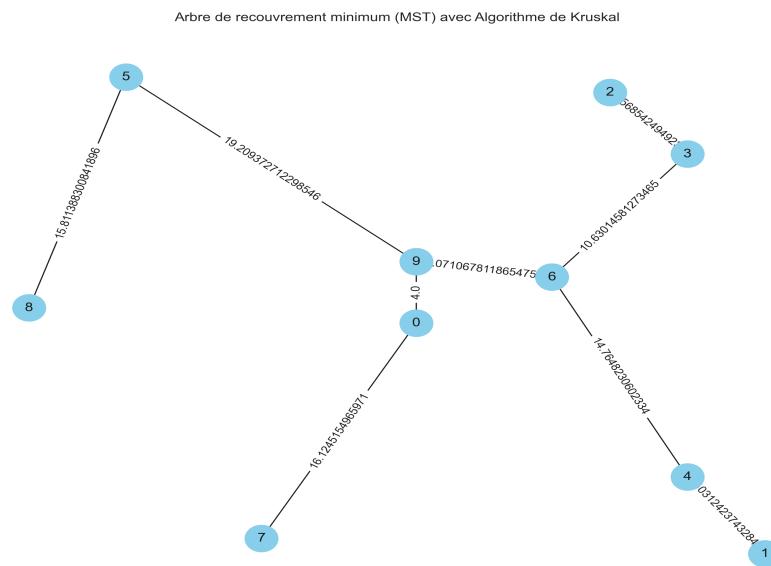


FIGURE 1.16 – Arbre de recouvrement minimum ( MST ) avec algorithme Kruskal

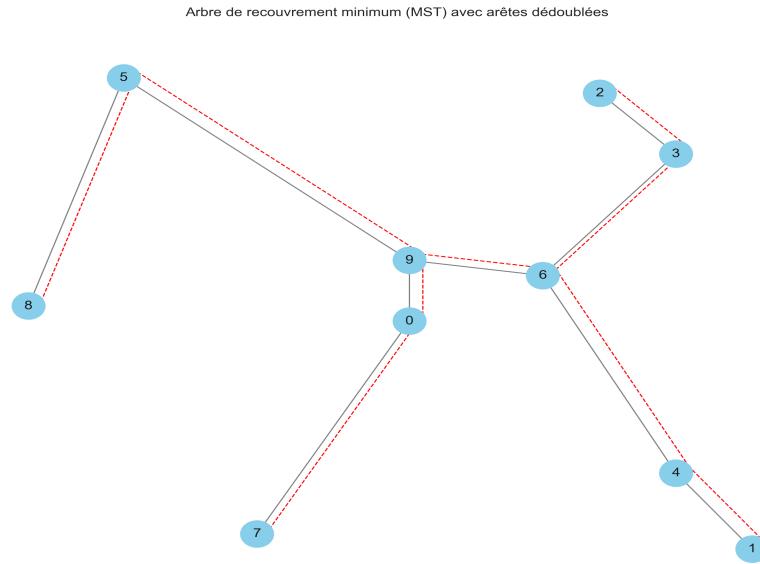


FIGURE 1.17 – Arbre de recouvrement minimum (MST) avec arêtes doublées

Chemin Eulérien : [0, 7, 0, 9, 6, 4, 1, 4, 6, 3, 2, 3, 6, 9, 5, 8, 5, 9, 0]

Élimination des chemins déjà visités :

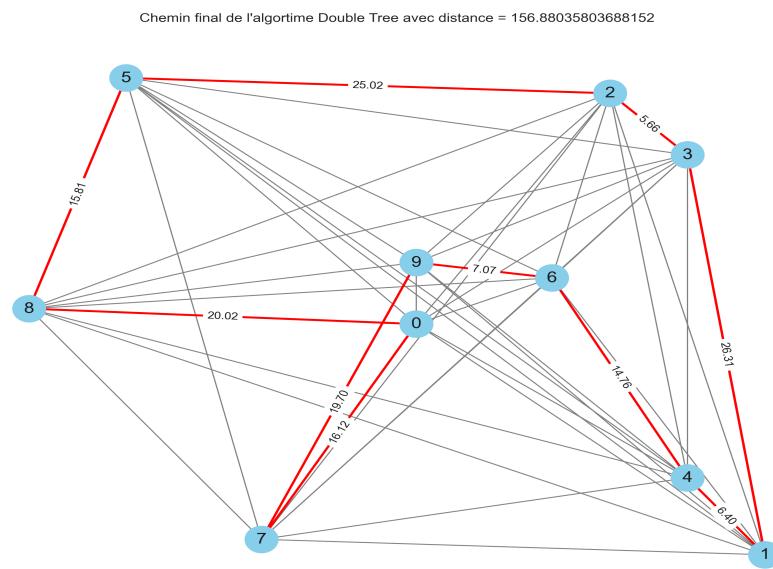


FIGURE 1.18 – Chemin final de l'algorithme Double tree

Distance = 156.88035803688152 Tour = [0, 7, 9, 6, 4, 1, 3, 2, 5, 8, 0]

### 1.7 Tableau récapitulatif des différentes distances :

TABLE 1.3 – Récapitulatif des performances des algorithmes

Algorithmme	Tour Final	Coût Total (Distance)
Clark and Wright	[1, 8, 9, 6, 3, 4, 2, 5, 7, 10, 1]	140.367
Heuristique du Plus Proche Voisin	[0, 9, 6, 3, 2, 5, 8, 7, 4, 1, 0]	139.593
Insertion la Plus Proche	[0, 9, 8, 5, 2, 3, 6, 4, 1, 7, 0]	140.654
Insertion la Plus Éloignée	[0, 9, 8, 5, 2, 3, 6, 4, 1, 7, 0]	144.654
Minimiser le Coût d'Insertion	[0, 5, 8, 7, 1, 4, 6, 3, 2, 9, 0]	139.293
Algorithme de Kruskal + Double Tree	[0, 7, 9, 6, 4, 1, 3, 2, 5, 8, 0]	156.880

#### Analyse des résultats :

**1. Efficacité en termes de coût total :** L'algorithme "Minimiser le Coût d'Insertion" présente le coût total le plus bas (139.293), ce qui indique qu'il est le plus efficace pour minimiser la distance totale parcourue dans ce cas spécifique. Ceci est suivi de près par l'Heuristique du Plus Proche Voisin avec un coût de 139.593.

**2. Comparaison des parcours :** En comparant les tours finaux, on peut remarquer que la séquence des visites varie considérablement entre les algorithmes, ce qui reflète leurs différentes stratégies d'optimisation. Par exemple, l'Heuristique du Plus Proche Voisin commence et termine au point 0, optimisant le chemin en se concentrant sur les voisins immédiats.

**3. Impact de la stratégie d'insertion :** Les algorithmes d'insertion, tels que "Insertion la Plus Proche" et "Insertion la Plus Éloignée", montrent des coûts totaux et des parcours différents, ce qui met en évidence l'impact de la stratégie d'insertion sur le résultat final. L'Insertion la Plus Éloignée a le coût le plus élevé (144.654), suggérant que cette stratégie peut ne pas être la plus efficace dans tous les cas.

**4. Performance relative des approches :** Bien que l'Algorithme de Kruskal + Double Tree soit connu pour son approche méthodique dans la construction d'arbres couvrants minimaux, son coût total est le plus élevé (156.880) dans ce cas, ce qui peut indiquer des limites lorsque appliqué directement à des problèmes de tournées spécifiques sans adaptations supplémentaires.

**5. Optimisation globale versus locale :** Les différences de coûts totaux et de parcours suggèrent une tension entre l'optimisation globale et locale. Certains algorithmes peuvent trouver des solutions localement optimales qui ne sont pas globalement optimales, comme le montre la variation des coûts totaux.

# CHAPITRE 2

## HEURISTIQUES AVANCÉES POUR LE PROBLÈME DU VOYAGEUR DE COMMERCE

### Sommaire

---

<b>Introduction</b>	18
<b>2.1 Avec 14 villes</b>	18
2.1.1 Clarke et Wright	18
2.1.1.1 2-OPT	18
2.1.1.2 3-OPT	18
2.1.2 Plus proches voisins :	18
2.1.2.1 2-OPT	18
2.1.2.2 3-OPT	19
2.1.3 Insertion la plus éloignée	19
2.1.3.1 2-OPT	19
2.1.3.2 3-OPT	19
2.1.4 Tabou	19
2.1.5 Concorde	19
2.1.6 Tableau récapitulatif pour 14 villes et analyse de résultat :	20
<b>2.2 Avec 150 villes</b>	20
2.2.1 Clark and Wright :	20
2.2.1.1 2-OPT :	21
2.2.1.2 3-OPT :	21
2.2.2 Plus proches voisins :	21
2.2.2.1 2-OPT :	21
2.2.2.2 3-OPT :	22
2.2.3 Insertion la plus éloignée :	22
2.2.3.1 2-OPT :	22
2.2.3.2 3-OPT :	22
2.2.4 Tabou :	23
2.2.5 Concorde :	23

2.2.6	Tableau récapitulatif pour 150 villes et analyse des résultats :	23
<b>2.3</b>	<b>Avec 1037 villes</b>	<b>24</b>
2.3.1	Clark and Wright :	24
2.3.1.1	2-OPT :	24
2.3.1.2	3-OPT :	24
2.3.2	Plus proches voisins :	25
2.3.2.1	2-OPT :	25
2.3.2.2	3-OPT :	25
2.3.3	Insertion la plus éloignée :	25
2.3.3.1	2-OPT :	25
2.3.3.2	3-OPT :	25
2.3.4	Tabou :	25
2.3.5	Concorde :	26
2.3.6	Tableau récapitulatif pour 1037 villes :	27

---

## Introduction

Dans cette question, nous abordons le problème du voyageur de commerce (TSP), qui consiste à trouver le chemin le plus court permettant de visiter une série de villes tout en revenant au point de départ. Le but est de comparer différentes approches heuristiques, incluant les heuristiques de construction et d'amélioration telles que la recherche locale k-opt, avec des solutions optimales ou quasi-optimales obtenues via le logiciel Python à l'aide OpenAI ( 2024 ) et le logiciel Concorde. Nous testerons ces méthodes sur diverses instances de TSP pour évaluer leur efficacité et applicabilité. La recherche inclura l'implémentation de stratégies de résolution, l'utilisation d'outils de programmation adaptés, et la documentation des démarches et résultats obtenus.

### 2.1 Avec 14 villes

#### 2.1.1 Clarke et Wright

La route est : [1, 10, 9, 11, 13, 7, 12, 6, 5, 4, 3, 14, 2, 8, 1]

La distance totale est : 31.208766207101625

Le temps d'exécution total est de 0.004805803298950195 secondes.

Le temps d'exécution par point est de 0.0003432716642107282 secondes par point.

#### 2.1.1.1 2-OPT

La meilleure route est : [1, 10, 9, 11, 13, 7, 12, 6, 5, 4, 3, 14, 2, 8, 1]

La distance du tour est de : 31.208766207101625

Le temps d'exécution total est de 0.0057621002197265625 secondes.

Le temps d'exécution par point est de 0.0004115785871233259 secondes par point.

#### 2.1.1.2 3-OPT

La route est : [1, 10, 9, 11, 8, 13, 7, 12, 6, 5, 4, 3, 14, 2, 1]

La distance totale est : 30.878503892588

Le temps d'exécution total est de 0.5855669975280762 secondes.

Le temps d'exécution par point est de 0.041826214109148295 secondes par point.

#### 2.1.2 Plus proches voisins :

La route est : [1, 8, 11, 9, 10, 2, 14, 12, 6, 7, 13, 3, 4, 5, 1]

La distance totale est : 38.68810773852191

Le temps d'exécution total est de 0.004899024963378906 secondes.

Le temps d'exécution par point est de 0.00034993035452706475 secondes par point.

#### 2.1.2.1 2-OPT

La meilleure route est : [1, 11, 9, 10, 2, 14, 3, 4, 5, 6, 12, 7, 13, 8, 1]

La distance du tour est de : 31.22691510942754

Le temps d'exécution total est de 2.6349620819091797 secondes.

Le temps d'exécution par point est de 0.18821157727922713 secondes par point.

### 2.1.2.2 3-OPT

La route est : [1, 10, 9, 11, 8, 13, 7, 12, 6, 5, 4, 3, 14, 2, 1]

La distance totale est : 30.878503892588

Le temps d'exécution total est de 7.449847936630249 secondes.

Le temps d'exécution par point est de 0.5321319954735892 secondes par point.

### 2.1.3 Insertion la plus éloignée

La route est : [1, 2, 14, 3, 4, 5, 6, 12, 7, 13, 8, 11, 9, 10, 1]

La distance totale est : 30.878503892588004

Le temps d'exécution total est de 0.0007569789886474609 secondes.

Le temps d'exécution par point est de 5.406992776053292e-05 secondes par point.

### 2.1.3.1 2-OPT

La meilleure route est : [1, 10, 9, 11, 8, 13, 7, 12, 6, 5, 4, 3, 14, 2, 1]

La distance du tour est de : 30.878503892588

Le temps d'exécution total est de 0.0012218952178955078 secondes.

Le temps d'exécution par point est de 8.727822984967913e-05 secondes par point.

### 2.1.3.2 3-OPT

La route est : [1, 10, 9, 11, 8, 13, 7, 12, 6, 5, 4, 3, 14, 2, 1]

La distance totale est : 30.878503892588

Le temps d'exécution total est de 0.027459144592285156 secondes.

Le temps d'exécution par point est de 0.0019613674708775113 secondes par point.

### 2.1.4 Tabou

Solution initiale : Insertion la plus éloignée

Nb itérations= 500

Liste Tabu = 5

La route est : [1, 2, 14, 3, 3, 4, 5, 6, 12, 7, 13, 8, 11, 9, 10, 1]

La distance totale est : 30.878503892588

Le temps d'exécution total est de 1.5905389785766602 secondes.

Le temps d'exécution par point est de 0.11360992704119001 secondes par point.

### 2.1.5 Concorde

Heuristique LK = 30

Optimal= 30

### 2.1.6 Tableau récapitulatif pour 14 villes et analyse de résultat :

TABLE 2.1 – Récapitulatif des performances des algorithmes pour 14 villes

Méthode / Amélioration	Route	Distance Totale	Temps d'Exécution Total (s)	Temps d'Exécution par Point (s)
Clark and Wright	[1, 10, 9, 11, 13, 7, 12, 6, 5, 4, 3, 14, 2, 8, 1]	31.21	0.0048	0.00034
+ 2-OPT	[1, 10, 9, 11, 13, 7, 12, 6, 5, 4, 3, 14, 2, 8, 1]	31.21	0.0058	0.00041
+ 3-OPT	[1, 10, 9, 11, 8, 13, 7, 12, 6, 5, 4, 3, 14, 2, 1]	30.88	0.5856	0.04183
Plus Proches Voisins	[1, 8, 11, 9, 10, 2, 14, 12, 6, 7, 13, 3, 4, 5, 1]	38.69	0.0049	0.00035
+ 2-OPT	[1, 11, 9, 10, 2, 14, 3, 4, 5, 6, 12, 7, 13, 8, 1]	31.23	2.6349	0.18821
+ 3-OPT	[1, 10, 9, 11, 8, 13, 7, 12, 6, 5, 4, 3, 14, 2, 1]	30.88	7.4498	0.53213
Insertion la Plus Éloignée	[1, 2, 14, 3, 4, 5, 6, 12, 7, 13, 8, 11, 9, 10, 1]	30.88	0.0008	0.00005
+ 2-OPT	[1, 10, 9, 11, 8, 13, 7, 12, 6, 5, 4, 3, 14, 2, 1]	30.88	0.0012	0.00009
+ 3-OPT	[1, 10, 9, 11, 8, 13, 7, 12, 6, 5, 4, 3, 14, 2, 1]	30.88	0.0275	0.00196
Tabou	[1, 2, 14, 3, 4, 5, 6, 12, 7, 13, 8, 11, 9, 10, 1]	30.88	1.5905	0.11361
Concorde	Optimal	30	-	-

**1. Efficacité des méthodes d'amélioration :** - -OPT et 3-OPT : Ces méthodes d'amélioration, lorsqu'appliquées après l'initialisation par Clark and Wright ou Plus Proches Voisins, réduisent la distance totale, ce qui montre leur efficacité pour optimiser le parcours. Par exemple, l'application du 3-OPT sur la base de Clark and Wright et Plus Proches Voisins aboutit à une distance totale de 30.88, une amélioration notable par rapport aux solutions initiales.

**2. Impact sur le temps d'exécution :** - L'ajout des méthodes d'optimisation 2-OPT et 3-OPT augmente significativement le temps d'exécution total, particulièrement évident avec le 3-OPT appliqué au Plus Proches Voisins, passant de 0.0049s à 7.4498s. Cela indique que, bien que ces méthodes améliorent la qualité de la solution, elles requièrent un temps de calcul plus important.

**3. Performance de l'Insertion la Plus Éloignée :** - L'Insertion la Plus Éloignée, même sans amélioration, atteint une distance totale de 30.88, équivalente à celle obtenue après les améliorations par 2-OPT et 3-OPT. Ceci souligne l'efficacité de cette méthode pour obtenir une bonne solution initiale avec un temps d'exécution minimal (0.0008s).

**4. Efficacité relative de Concorde :** - Concorde, en tant que solveur, trouve une solution optimale avec une distance totale de 30.00, servant de référence pour évaluer l'efficacité des autres méthodes. Cela montre que, malgré les améliorations apportées par les méthodes 2-OPT et 3-OPT, il reste une marge d'optimisation pour atteindre l'optimalité.

**5. Temps d'exécution par point :** - L'analyse du temps d'exécution par point offre un aperçu supplémentaire sur l'efficacité computationnelle des méthodes. Par exemple, l'Insertion la Plus Éloignée avec amélioration par 2-OPT montre une augmentation modeste du temps par point (de 0.00005s à 0.00009s), suggérant une optimisation efficace en termes de temps de calcul par rapport à l'amélioration apportée.

## 2.2 Avec 150 villes

### 2.2.1 Clark and Wright :

La route est : [1, 98, 103, 7, 84, 30, 34, 76, 73, 48, 63, 10, 113, 3, 62, 94, 88, 21, 22, 104, 4, 115, 44, 71, 45, 68, 91, 106, 13, 74, 31, 27, 49, 72, 112, 64, 80, 14, 77, 23, 38, 32, 67, 43, 12, 101, 41, 39, 57, 60, 66, 17, 11, 61, 36, 69, 24, 53, 40, 42, 9, 28, 6, 37, 2, 19, 99, 114, 102, 108, 70, 86, 29, 81, 110, 47, 20, 51, 109, 25, 58, 50, 55, 65, 85, 18, 75, 26, 56, 83, 90, 46, 54, 92, 105, 111, 16, 15, 78, 59, 79, 52, 33, 93, 35, 96, 89, 8, 97, 100, 5, 107, 95, 82, 87, 1]

La distance totale est : 6022.258541978591

Le temps d'exécution total est de 0.02292799949645996 secondes.

Le temps d'exécution par point est de 0.00019937390866486924 secondes par point.

### 2.2.1.1 2-OPT :

La meilleure route est : [1, 87, 82, 95, 107, 5, 100, 97, 8, 89, 96, 35, 93, 33, 52, 79, 59, 78, 15, 16, 111, 105, 92, 54, 46, 90, 83, 56, 26, 75, 18, 85, 65, 55, 50, 58, 25, 20, 51, 109, 47, 110, 81, 29, 86, 108, 70, 102, 114, 99, 19, 2, 37, 6, 28, 9, 42, 40, 53, 24, 69, 36, 61, 11, 17, 66, 60, 39, 57, 41, 101, 12, 43, 67, 32, 38, 23, 77, 14, 80, 64, 112, 72, 49, 27, 31, 74, 13, 106, 91, 68, 45, 71, 44, 115, 4, 104, 22, 21, 88, 94, 62, 3, 113, 10, 63, 48, 73, 76, 34, 30, 84, 7, 103, 98, 1]

La distance du tour est de : 5972.687040414349

Le temps d'exécution total est de 0.7919769287109375 secondes.

Le temps d'exécution par point est de 0.006886755901834239 secondes par point.

### 2.2.1.2 3-OPT :

La route est : [1, 98, 103, 82, 95, 107, 5, 100, 97, 7, 84, 30, 8, 89, 96, 35, 93, 52, 111, 105, 33, 92, 54, 46, 90, 83, 56, 26, 75, 18, 85, 65, 55, 58, 25, 20, 47, 110, 81, 29, 86, 50, 70, 108, 102, 114, 99, 19, 2, 37, 6, 28, 9, 42, 40, 53, 24, 69, 36, 61, 11, 17, 66, 60, 39, 57, 41, 101, 12, 51, 109, 43, 67, 32, 38, 23, 77, 14, 80, 78, 15, 16, 59, 79, 94, 88, 21, 64, 112, 72, 49, 27, 31, 74, 13, 106, 91, 68, 45, 71, 44, 115, 4, 104, 22, 62, 3, 113, 10, 63, 48, 73, 76, 34, 87, 1]

La distance totale est : 5741.113077567531

Le temps d'exécution total est de 746.150839805603 secondes.

Le temps d'exécution par point est de 6.488268172222635 secondes par point.

### 2.2.2 Plus proches voisins :

La route est : [1, 98, 103, 34, 87, 76, 73, 48, 63, 30, 84, 7, 8, 89, 96, 35, 93, 52, 33, 105, 111, 16, 59, 79, 78, 15, 80, 14, 77, 32, 23, 38, 67, 43, 109, 51, 20, 25, 110, 81, 29, 86, 108, 70, 50, 58, 55, 65, 85, 18, 75, 26, 56, 83, 90, 46, 54, 92, 47, 42, 9, 28, 6, 37, 2, 19, 99, 114, 102, 40, 53, 24, 12, 101, 41, 57, 39, 60, 66, 17, 11, 61, 36, 69, 27, 31, 49, 72, 112, 64, 44, 71, 45, 4, 104, 115, 21, 88, 94, 10, 113, 3, 62, 22, 68, 91, 106, 13, 74, 97, 100, 5, 107, 95, 82, 1]

La distance totale est : 6551.75966787569

Le temps d'exécution total est de 0.0034961700439453125 secondes.

Le temps d'exécution par point est de 3.0401478643002717e-05 secondes par point.

### 2.2.2.1 2-OPT :

La meilleure route est : [1, 98, 103, 82, 95, 107, 5, 100, 97, 93, 35, 52, 33, 105, 111, 16, 59, 79, 78, 15, 80, 14, 77, 32, 23, 38, 67, 43, 109, 51, 20, 25, 46, 54, 92, 90, 83, 56, 26, 75, 18, 85, 65, 55, 50, 58, 81, 110, 29, 86, 108, 70, 102, 114, 99, 19, 2, 37, 6, 28, 9, 42, 47, 40, 53, 12, 24, 69, 36, 61, 11, 17, 66, 60, 57, 39, 101, 41, 27, 31, 49, 72, 64, 112, 74, 13, 106, 91, 68, 45, 71, 44, 21, 115, 4, 104, 22, 62, 3, 113, 10, 88, 94, 96, 89, 8, 7, 84, 30, 63, 48, 73, 76, 34, 87, 1]

La distance du tour est de : 5744.806646852654

Le temps d'exécution total est de 3.3054656982421875 secondes.

Le temps d'exécution par point est de 0.028743179984714675 secondes par point.

### 2.2.2.2 3-OPT :

La route est : [1, 98, 103, 82, 95, 107, 5, 100, 97, 7, 84, 30, 8, 89, 96, 35, 93, 52, 33, 105, 111, 16, 59, 79, 78, 15, 80, 14, 77, 23, 38, 32, 67, 43, 109, 51, 20, 25, 90, 46, 54, 92, 56, 83, 26, 75, 18, 85, 65, 55, 58, 50, 86, 108, 70, 102, 114, 99, 19, 29, 81, 110, 47, 2, 37, 6, 28, 9, 42, 40, 53, 12, 24, 69, 36, 61, 11, 17, 66, 60, 57, 39, 101, 41, 27, 31, 49, 72, 64, 112, 74, 13, 106, 91, 68, 104, 4, 45, 71, 44, 115, 21, 22, 62, 3, 88, 94, 10, 113, 63, 48, 73, 76, 34, 87, 1]

La distance totale est : 5692.045761379183

Le temps d'exécution total est de 842.9087133407593 secondes.

Le temps d'exécution par point est de 7.32964098557182 secondes par point.

### 2.2.3 Insertion la plus éloignée :

La route est : [1, 98, 103, 82, 95, 107, 5, 100, 97, 8, 7, 84, 30, 89, 96, 35, 93, 52, 33, 111, 105, 92, 54, 46, 90, 83, 56, 26, 75, 18, 85, 65, 55, 58, 50, 86, 29, 108, 70, 102, 114, 99, 19, 2, 37, 6, 28, 9, 42, 47, 110, 81, 25, 20, 51, 109, 43, 12, 24, 53, 40, 69, 36, 61, 11, 17, 66, 60, 39, 57, 41, 101, 67, 32, 38, 23, 72, 49, 27, 31, 74, 112, 64, 13, 106, 91, 68, 71, 44, 115, 45, 4, 104, 22, 21, 78, 15, 80, 14, 77, 16, 59, 79, 88, 94, 62, 3, 113, 10, 63, 48, 73, 76, 34, 87, 1]

La distance totale est : 5806.944572987945

Le temps d'exécution total est de 0.08188390731811523 secondes.

Le temps d'exécution par point est de 0.0007120339766792629 secondes par point..

### 2.2.3.1 2-OPT :

La meilleure route est : [1, 98, 103, 82, 95, 107, 5, 100, 97, 8, 7, 84, 30, 89, 96, 35, 93, 52, 111, 105, 33, 92, 54, 46, 90, 83, 56, 26, 75, 18, 85, 65, 55, 58, 50, 86, 29, 108, 70, 102, 114, 99, 19, 2, 37, 6, 28, 9, 42, 47, 110, 81, 25, 20, 51, 109, 43, 12, 24, 53, 40, 69, 36, 61, 11, 17, 66, 60, 39, 57, 41, 101, 67, 32, 38, 23, 72, 49, 27, 31, 74, 112, 64, 13, 106, 91, 68, 71, 44, 115, 45, 4, 104, 22, 21, 78, 15, 80, 14, 77, 16, 59, 79, 88, 94, 62, 3, 113, 10, 63, 48, 73, 76, 34, 87, 1]

La distance du tour est de : 5787.512747255148

Le temps d'exécution total est de 0.453812837600708 secondes.

Le temps d'exécution par point est de 0.0039461985878322435 secondes par point.

### 2.2.3.2 3-OPT :

La route est : [1, 98, 103, 82, 95, 107, 5, 100, 97, 7, 84, 30, 8, 89, 96, 35, 93, 52, 111, 105, 33, 92, 54, 46, 90, 83, 56, 26, 75, 18, 85, 65, 55, 58, 50, 86, 29, 108, 70, 102, 114, 99, 19, 2, 37, 6, 28, 9, 42, 47, 110, 81, 25, 20, 51, 109, 43, 12, 24, 53, 40, 69, 36, 61, 11, 17, 66, 60, 39, 57, 41, 101, 67, 32, 38, 23, 77, 14, 80, 64, 112, 72, 49, 27, 31, 74, 13, 106, 91, 68, 45, 71, 44, 115, 4, 104, 22, 21, 78, 15, 16, 59, 79, 88, 94, 62, 3, 113, 10, 63, 48, 73, 76, 34, 87, 1]

La distance totale est : 5676.720978527594

Le temps d'exécution total est de 345.64822697639465 secondes.

Le temps d'exécution par point est de 3.005636756316475 secondes par point.

#### 2.2.4 *Tabou* :

Solution initiale : Insertion la plus éloignée Nb itérations = 10 000 Liste Tabu = 30 La route est : [1, 98, 103, 82, 95, 107, 5, 100, 97, 8, 7, 84, 30, 89, 96, 35, 93, 52, 111, 105, 33, 92, 54, 46, 90, 83, 56, 26, 75, 18, 85, 65, 55, 58, 50, 86, 29, 108, 70, 102, 114, 99, 19, 2, 37, 6, 28, 9, 42, 47, 110, 81, 25, 20, 51, 109, 43, 12, 24, 53, 40, 69, 36, 61, 11, 17, 66, 60, 39, 57, 41, 101, 67, 32, 38, 23, 27, 31, 49, 72, 64, 112, 74, 13, 106, 91, 68, 68, 45, 71, 44, 115, 4, 104, 22, 3, 62, 21, 78, 15, 80, 14, 77, 16, 59, 79, 88, 94, 113, 10, 63, 48, 73, 76, 34, 87, 1]

La distance totale est : 5731.459577270324

Le temps d'exécution total est de 7764.144521951675 secondes.

Le temps d'exécution par point est de 67.51430019088413 secondes par point.

#### 2.2.5 *Concorde* :

Heuristique LK = 5584

Optimal = 5584

#### 2.2.6 *Tableau récapitulatif pour 150 villes et analyse des résultats* :

TABLE 2.2 – Récapitulatif des performances des algorithmes pour 150 villes

Algorithm	Total Distance	Total Execution Time (s)	Execution Time per Point (s)
Clark and Wright	6022.26	0.02293	0.00020
2-OPT (C&W)	5972.69	0.79198	0.00689
3-OPT (C&W)	5741.11	746.15084	6.48827
Plus proches voisins	6551.76	0.00350	0.00003
2-OPT (PPV)	5744.81	3.30547	0.02874
3-OPT (PPV)	5692.05	842.90871	7.32964
Insertion la plus éloignée	5806.94	0.08188	0.00071
2-OPT (IPE)	5787.51	0.45381	0.00395
3-OPT (IPE)	5676.72	345.64823	3.00564
Tabou	5731.46	7764.14452	67.51430
Concorde	5584.00	N/A	N/A

#### Performance :

**1. Optimalité** : Concorde, étant un solveur, fournit la solution optimale avec une distance totale de 5584.00. Cela sert de benchmark pour évaluer l'efficacité des autres méthodes.

**2. Amélioration des distances** : L'application des méthodes 2-OPT et 3-OPT montre une réduction significative de la distance totale pour les solutions initiales générées par Clark and Wright (CW), Plus proches voisins (PPV), et Insertion la plus éloignée (IPE). Par exemple, l'application de 3-OPT à la solution initiale de CW réduit la distance de 6022.26 à 5741.11.

**3. Temps d'exécution** : Il y a une augmentation notable du temps d'exécution total avec l'application des méthodes d'amélioration, particulièrement visible avec 3-OPT. Par exemple, le temps d'exécution total passe de 0.02293s pour Clark and Wright seul à 746.15084s avec l'application de 3-OPT.

**4. Efficacité temporelle par point :** Le temps d'exécution par point augmente également avec les méthodes d'amélioration, indiquant un coût computationnel plus élevé pour obtenir des améliorations de distance. Le temps d'exécution par point pour 3-OPT appliqué à Clark and Wright est de 6.48827s, un ordre de grandeur plus élevé que pour l'algorithme initial seul.

**5.Comparaison des algorithmes initiaux :** Avant l'amélioration, les algorithmes montrent des performances variées en termes de distance totale, avec le Plus proches voisins ayant la plus haute distance initiale (6551.76) et l'Insertion la plus éloignée ayant une performance relativement meilleure (5806.94).

#### **Implications :**

**1. Choix de l'algorithme initial :** Le choix de l'algorithme pour générer la route initiale a un impact significatif sur la qualité de la solution finale après amélioration. Les algorithmes qui commencent avec des distances plus courtes tendent à aboutir à de meilleures solutions après l'application de 2-OPT ou 3-OPT.

**2. Coût computationnel vs. amélioration de la distance :** Bien que les méthodes d'amélioration puissent réduire la distance totale, elles viennent avec un coût computationnel élevé. Le choix entre le temps d'exécution et l'optimisation de la distance dépend des contraintes spécifiques du problème et des ressources disponibles.

**3. Efficacité de l'amélioration :** Les méthodes d'amélioration 2-OPT et 3-OPT sont efficaces pour améliorer les solutions initiales, mais l'écart entre ces solutions améliorées et la solution optimale de Concorde suggère qu'il existe un potentiel d'amélioration encore plus grand, soit par des ajustements de ces méthodes, soit par l'utilisation de solveurs spécialisés comme Concorde pour des problèmes spécifiques.

## **2.3 Avec 1037 villes**

### **2.3.1 *Clark and Wright :***

La distance totale est : 46852.47860417494

Le temps d'exécution total est de 2.0676681995391846 secondes.

Le temps d'exécution par point est de 0.001993894117202685 secondes par point.

#### **2.3.1.1 2-OPT :**

La distance du tour est de : 46513.46078483456

Le temps d'exécution total est de 602.5276379585266 secondes.

Le temps d'exécution par point est de 0.5810295448008935 secondes par point.

#### **2.3.1.2 3-OPT :**

La distance totale est : 46852.47860417494

Le temps d'exécution total est de 600.001177072525 secondes.

Le temps d'exécution par point est de 0.5785932276494937 secondes par point

**2.3.2 *Plus proches voisins :***

La distance totale est : 51861.70527442242

Le temps d'exécution total est de 0.2231762409210205 secondes.

Le temps d'exécution par point est de 0.00021521334707909402 secondes par point.

**2.3.2.1 *2-OPT :***

La distance du tour est de : 50212.86897345179

Le temps d'exécution total est de 607.8117790222168 secondes.

Le temps d'exécution par point est de 0.5861251485267278 secondes par point.

**2.3.2.2 *3-OPT :***

La distance totale est : 51808.983921774256

Le temps d'exécution total est de 600.0020091533661 secondes.

Le temps d'exécution par point est de 0.5785940300418188 secondes par point.

**2.3.3 *Insertion la plus éloignée :***

La distance totale est : 47072.06151355614

Le temps d'exécution total est de 39.18679690361023 secondes.

Le temps d'exécution par point est de 0.037788618036268304 secondes par point.

**2.3.3.1 *2-OPT :***

La distance du tour est de : 46831.24894566641

Le temps d'exécution total est de 600.0132591724396 secondes.

Le temps d'exécution par point est de 0.5786048786619475 secondes par point.

**2.3.3.2 *3-OPT :***

La distance totale est : 47072.06151355614

Le temps d'exécution total est de 600.0005040168762 secondes.

Le temps d'exécution par point est de 0.5785925786083667 secondes par point.

**2.3.4 *Tabou :***

Solution initiale : Insertion la plus éloignée

Temps max = 6h

Liste Tabu = 75

La distance totale est : 47060

Le temps d'exécution total est de 7764.14 secondes.

Le temps d'exécution par point est de 67.5143 secondes par point.

**2.3.5 *Concorde* :**

Heuristique LK = 42254

Optimal = 42157

### 2.3.6 Tableau récapitulatif pour 1037 villes :

TABLE 2.3 – Récapitulatif des performances des algorithmes pour 1037 villes

#	Algorithm	Distance totale	Temps d'exécution total (s)	Temps d'exécution par point (s)
1	Clark and Wright	46852.48	2.07	0.00199
2	2-OPT (C&W)	46513.46	602.53	0.58103
3	3-OPT (C&W)	46852.48	600.00	0.57859
4	Plus proches voisins	51861.71	0.22	0.00022
5	2-OPT (PPV)	50212.87	607.81	0.58613
6	3-OPT (PPV)	51808.98	600.00	0.57859
7	Insertion la plus éloignée	47072.06	39.19	0.03779
8	2-OPT (IPE)	46831.25	600.01	0.57860
9	3-OPT (IPE)	47072.06	600.00	0.57859
10	Tabou	47060	7764.14	67.5143
11	LK	42254	N/A	N/A
12	Concorde	42157	N/A	N/A

#### Observations Clés

**Limitation du Temps d'Exécution :** La contrainte de 600 secondes imposée à la plupart des algorithmes pourrait ne pas permettre à certains d'entre eux, particulièrement les méthodes 3-OPT et l'algorithme Tabou, d'atteindre leur potentiel d'optimisation maximal. Cela se reflète dans les distances totales qui ne montrent pas d'amélioration significative ou, dans certains cas, une augmentation de la distance après l'application de 3-OPT.

#### Efficacité des Algorithmes :

**1. Clark and Wright :** Avant toute amélioration, cet algorithme présente une distance totale compétitive (46852.48) avec un temps d'exécution très faible (2.07s), ce qui le rend efficient pour des solutions rapides mais pas nécessairement optimales.

**2. Plus proches voisins :** Cet algorithme a la distance initiale la plus élevée (51861.71), et même après amélioration, il ne parvient pas à atteindre les performances des autres méthodes, soulignant son inefficacité relative dans ce cas.

**3. Insertion la plus éloignée :** Avec une distance initiale légèrement supérieure à celle de Clark and Wright mais avec un temps d'exécution significativement plus long (39.19s), cette méthode offre une base légèrement moins efficace pour les améliorations ultérieures.

#### Impact des Améliorations (2-OPT et 3-OPT) :

Les améliorations par 2-OPT montrent une réduction notable de la distance totale pour Clark and Wright et Insertion la plus éloignée, mais avec un coût temporel élevé.

L'application de 3-OPT ne montre pas d'amélioration significative de la distance dans le temps imposé, ce qui suggère que la limitation de temps empêche ces méthodes d'explorer pleinement l'espace des solutions.

#### Implications et Recommandations

**Sélection de l'Algorithm :** Le choix de l'algorithme doit être influencé par la priorité entre le temps d'exécution et la minimisation de la distance. Pour des solutions rapides, Clark and Wright offre un bon équilibre, tandis que Concorde est préférable pour l'optimalité sans contrainte de temps.

**Impact de la Limite de Temps :** La contrainte de 600 secondes limite l'efficacité des méthodes d'optimisation, particulièrement pour 3-OPT et l'algorithme Tabou. Il peut être utile d'ajuster cette limite

en fonction des besoins spécifiques du problème et des capacités de calcul disponibles.

**Utilisation des Méthodes d'Amélioration :** Bien que les méthodes d'amélioration telles que 2-OPT et 3-OPT puissent améliorer les résultats, leur efficacité est fortement dépendante du temps d'exécution alloué. Une planification soignée de la capacité de calcul et des délais est essentielle pour tirer le meilleur parti de ces méthodes.

**Cas Particulier de l'Algorithmme Tabou :** L'algorithme Tabou montre un temps d'exécution extrêmement élevé pour une amélioration marginale de la distance, ce qui suggère une inefficacité dans le contexte de cette limitation de temps. Son utilisation doit être soigneusement évaluée en fonction des objectifs spécifiques et des ressources disponibles.

Cette analyse met en lumière l'importance de considérer la contrainte de temps dans le choix et l'application des algorithmes et méthodes d'amélioration, ainsi que l'impact significatif que cette contrainte peut avoir sur les résultats obtenus.

# CHAPITRE3

## VARIANTES DU PROBLÈME DE ROUTAGE DE VÉHICULES

### Sommaire

---

<b>Introduction</b> . . . . .	<b>30</b>
<b>3.1 Définition de la variante :</b> . . . . .	<b>30</b>
<b>3.2 Formulation Mathématiques :</b> . . . . .	<b>31</b>
3.2.1 Définition : . . . . .	31
3.2.2 Objectif : Minimiser la somme des coûts de toutes les liaisons utilisées : . . . . .	31
3.2.3 Contraintes . . . . .	31
<b>3.3 Code :</b> . . . . .	<b>33</b>
<b>3.4 Résolution :</b> . . . . .	<b>34</b>
<b>3.5 Explication :</b> . . . . .	<b>35</b>

---

## Introduction

Dans cette question, nous explorons une variante spécifique du problème de tournées de véhicules (VRP), différente de la variante classique de routage de véhicules avec contraintes de capacité (CVRP) abordée précédemment. Le but est de sélectionner une variante, parmi celles discutées en classe ou identifiées dans la littérature, et de la définir clairement. Cette exploration comprend la formulation mathématique du problème choisi, le développement d'un modèle correspondant en utilisant un langage de programmation mathématique de choix, et la résolution du modèle à l'aide d'un solveur d'optimisation standard comme Cplex ou Gurobi. Nous nous sommes notamment inspirés de l'approche proposée par Ruthmair (2024) pour résoudre le problème de routage de véhicules. Nous limiterons notre étude à un ensemble de 50 clients, avec des coordonnées prises dans la bibliothèque TSPLIB, afin de tester et valider l'approche choisie. Cette question vise non seulement à approfondir la compréhension des différentes variantes du VRP mais aussi à développer des compétences pratiques en modélisation et résolution de problèmes d'optimisation complexes.

### 3.1 Définition de la variante :

Dans le CVRPTW, l'objectif est de minimiser le coût total (qui peut être le temps total de parcours, la distance totale parcourue, ou une combinaison de ces éléments et d'autres coûts) tout en construisant des itinéraires pour une flotte de véhicules devant servir un ensemble de clients. Chaque client doit être visité une seule fois par un seul véhicule, et la visite doit se faire dans une fenêtre de temps spécifiée. Les véhicules partent d'un dépôt et doivent y retourner après avoir terminé leurs tournées.

#### Contraintes Spécifiques du CVRPTW :

- Fenêtres de temps : Chaque client a une fenêtre de temps définie, souvent représentée par une heure de début et une heure de fin. Les véhicules doivent arriver et commencer le service dans cette fenêtre. S'ils arrivent trop tôt, ils doivent attendre ; s'ils arrivent trop tard, la visite est considérée comme non réalisable.
- Capacité des véhicules : Comme dans le VRP classique, chaque véhicule a une capacité maximale de chargement qui ne doit pas être dépassée.
- Temps de service : Chaque client a un temps de service associé, représentant le temps nécessaire pour effectuer la livraison ou la collecte.
- Coûts : Le coût total peut inclure la distance totale parcourue, le temps total de parcours, et parfois des pénalités pour l'attente ou le non-respect des fenêtres de temps.

#### Objectifs du CVRPTW :

L'objectif principal reste la minimisation du coût total tout en respectant les contraintes de capacité des véhicules et les fenêtres de temps des clients. Cela peut inclure la réduction du nombre de véhicules utilisés, la minimisation de la distance totale parcourue, ou la réduction du temps total de parcours.

#### Applications du CVRPTW :

Le CVRPTW est largement utilisé dans la logistique de distribution, où le respect des horaires de livraison est crucial. Il trouve des applications dans la distribution de marchandises, la collecte de déchets, les services de maintenance à domicile, et toute situation où le timing de service est un facteur critique.

### 3.2 Formulation Mathématiques :

#### 3.2.1 Définition :

- un dépôt 0 où tous les véhicules sont situés au début,
- un ensemble de clients  $C = \{1, \dots, N\}$  (nous définissons les emplacements  $L = \{0\} \cup C$ ),
- une demande  $d_i \geq 0$  pour chaque client  $i \in C$  (le dépôt n'a pas de demande, c'est-à-dire,  $d_0 = 0$ ),
- une fenêtre de temps  $[a_i, b_i]$  avec  $0 \leq a_i \leq b_i$  pour chaque emplacement  $i \in L$  (y compris le dépôt),
- un temps de service  $s_i \geq 0$  pour chaque client  $i \in C$  (le dépôt a un temps de service nul, c'est-à-dire,  $s_0 = 0$ ),
- des coûts ( $c_{ij} \geq 0$ ) et un temps ( $t_{ij} \geq 0$ ) pour voyager de l'emplacement  $i$  à l'emplacement  $j$ ,
- et une flotte homogène  $K = \{1, \dots, |K|\}$  de véhicules avec une capacité de charge  $Q > 0$ .

La formulation mathématique compacte du problème de tournées de véhicules avec fenêtres de temps (VRPTW) se présente comme suit :

#### 3.2.2 Objectif : Minimiser la somme des coûts de toutes les liaisons utilisées :

$$\min \sum_{i,j \in L} c_{ij} x_{ij}$$

où ( $c_{ij}$ ) représente le coût de déplacement du lieu ( $i$ ) au lieu ( $j$ ), et ( $x_{ij}$ ) est une variable binaire indiquant si le trajet de ( $i$ ) à ( $j$ ) est utilisé ( $x_{ij} = 1$ ) ou non ( $x_{ij} = 0$ ).

#### 3.2.3 Contraintes

- Chaque client est visité exactement une fois, ce qui signifie qu'il existe exactement une connexion entrante et une connexion sortante pour chaque client :
  - Pour toute connexion entrante vers le client ( $j$ ) :  $\sum_{i \in L} x_{ij} = 1 \quad \forall j \in C$
  - Pour toute connexion sortante du client ( $i$ ) :  $\sum_{j \in L} x_{ij} = 1 \quad \forall i \in C$
- Utilisation des Véhicules : Au plus  $|K|$  véhicules peuvent être utilisés, où  $|K|$  est le nombre total de véhicules disponibles :
 
$$\sum_{j \in C} x_{0j} \leq |K|$$

#### Contrainte de Capacité :

On peut également calculer une borne inférieure sur le nombre de véhicules nécessaires, qui est la somme de toutes les demandes divisées par la capacité du véhicule. Cette contrainte n'est pas nécessaire

mais aide parfois à améliorer les performances :

$$\sum_{j \in C} x_{0j} \geq \left\lceil \frac{\sum_{i \in C} d_i}{Q} \right\rceil$$

où

$$\left( \sum_{j \in C} x_{0j} \right)$$

représente le nombre total de voyages qui partent du dépôt vers les clients. C'est-à-dire, pour chaque client ( $j$ ), un véhicule part du dépôt (0) vers le client ( $j$ ), et ( $x_{0j}$ ) est une variable binaire qui prend la valeur 1 si un tel trajet est réalisé.

$$\left\lceil \frac{\sum_{i \in C} d_i}{Q} \right\rceil$$

calcule le nombre minimal de voyages nécessaires pour satisfaire toutes les demandes des clients, étant donné que chaque véhicule a une capacité limitée ( $Q$ ). Le numérateur ( $\sum_{i \in C} d_i$ ) est la demande totale de tous les clients, et le dénominateur ( $Q$ ) est la capacité de charge de chaque véhicule. Le résultat de cette division est arrondi au nombre entier supérieur pour garantir que le résultat est un nombre entier de véhicules pour transporter toute la demande, même si la dernière charge ne remplit pas complètement un véhicule.

### **Charge**

Nous introduisons des variables continues  $y_{ij} \in [0, Q]$  pour chaque connexion ( $i, j$ ) afin de dénoter la charge du véhicule après avoir pris en charge la demande chez  $i$  et en se dirigeant vers l'emplacement  $j$ . Nous pouvons également exprimer les nouvelles variables avec les précédentes de manière non linéaire, c'est-à-dire,  $y_{ij} = y_i \cdot x_{ij}$ . Dans une interprétation différente, ces variables modélisent le flux de marchandises sur la connexion ( $i, j$ ) ce qui suggère d'utiliser des contraintes de conservation de flux :

$$\begin{aligned} y_{0j} &= 0 \quad \forall j \in C, \\ \sum_{i \in L} y_{ij} + d_j &= \sum_{i \in L} y_{ji} \quad \forall j \in C, \\ y_{ij} &\geq d_i x_{ij} \quad \forall i \in C, j \in L, i \neq j, \\ y_{ij} &\leq (Q - d_j) x_{ij} \quad \forall i \in C, j \in L, i \neq j. \end{aligned}$$

### **Temps**

Nous introduisons des variables  $z_{ij} \in [0, b_i]$  pour chaque connexion ( $i, j$ ) afin de dénoter l'heure de début du service chez  $i$  (qui doit être dans sa fenêtre de temps) en se dirigeant immédiatement vers

l'emplacement  $j$ . Ensuite, nous pouvons définir le système de flux suivant :

$$\begin{aligned} z_{0j} &= 0 & \forall j \in C \\ \sum_{i \in L} [z_{ij} + (s_i + t_{ij})x_{ij}] &\leq \sum_{i \in L} z_{ji} & \forall j \in C \\ z_{ij} &\geq a_i x_{ij} & \forall i, j \in L, i \neq j \\ z_{ij} &\leq b_i x_{ij} & \forall i, j \in L, i \neq j \end{aligned}$$

Notez que la conservation de flux peut ne pas tenir avec égalité si le début de la fenêtre de temps chez  $j$  est plus tard.

### 3.3 Code :

Le code est fourni en annexe sous le nom de modèle CVRPTW. Voici les paramètres en entrée :

```
numCustomers = 50 Nombre de clients
maxNumVehicles = 10 Nombre maximal de véhicules
vehicleCapacity = 30 Capacité de chaque véhicule
demandRange = (2, 5) Intervalle des demandes des clients
timeHorizon = 100 Horizont de temps total
timeWindowWidthRange = (10, 15) Intervalle de la largeur des fenêtres de temps
serviceTimeRange = (2, 4) Intervalle des temps de service
```

Voici les clients avec des demandes et des fenêtres de temps aléatoires :



FIGURE 3.1 – Visualisation des clients avec les fenêtres de temps

### 3.4 Résolution :

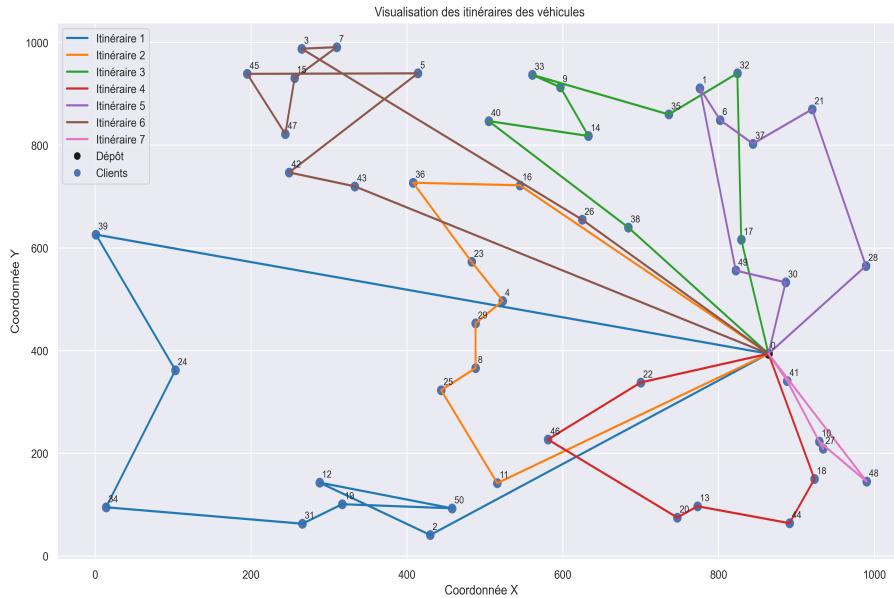


FIGURE 3.2 – Visuliation des itinéraires des vehicules

Optimal solution found (tolerance 1.00e-04) Best objective 1.161800000000e+04, best bound 1.161800000000e+04, gap 0.0000

La solution contient 7 itinéraires :

**Itinéraire 1 :** 0 -> Client 2 (Charge :0, Temps :8) -> Client 12 (Charge :4, Temps :14) -> Client 50 (Charge :6, Temps :23) -> Client 19 (Charge :9, Temps :30) -> Client 31 (Charge :12, Temps :35) -> Client 34 (Charge :15, Temps :61) -> Client 24 (Charge :19, Temps :67) -> Client 39 (Charge :21, Temps :75) -> 0 (Charge :25/30, Temps :92)

**Itinéraire 2 :** 0 -> Client 16 (Charge :0, Temps :7) -> Client 36 (Charge :4, Temps :14) -> Client 23 (Charge :9, Temps :38) -> Client 4 (Charge :11, Temps :42) -> Client 29 (Charge :13, Temps :45) -> Client 8 (Charge :18, Temps :55) -> Client 25 (Charge :22, Temps :58) -> Client 11 (Charge :25, Temps :63) -> 0 (Charge :30/30, Temps :86)

**Itinéraire 3 :** 0 -> Client 17 (Charge :0, Temps :4) -> Client 32 (Charge :3, Temps :13) -> Client 35 (Charge :6, Temps :17) -> Client 33 (Charge :11, Temps :24) -> Client 9 (Charge :16, Temps :28) -> Client 14 (Charge :19, Temps :42) -> Client 40 (Charge :23, Temps :53) -> Client 38 (Charge :25, Temps :69) -> 0 (Charge :27/30, Temps :76)

**Itinéraire 4 :** 0 -> Client 22 (Charge :0, Temps :3) -> Client 46 (Charge :5, Temps :36) -> Client 20 (Charge :9, Temps :45) -> Client 13 (Charge :11, Temps :50) -> Client 44 (Charge :16, Temps :56) -> Client 18 (Charge :19, Temps :67) -> 0 (Charge :22/30, Temps :82)

**Itinéraire 5 :** 0 -> Client 28 (Charge :0, Temps :4) -> Client 21 (Charge :2, Temps :11) -> Client 37 (Charge :6, Temps :24) -> Client 6 (Charge :10, Temps :29) -> Client 1 (Charge :14, Temps :35) -> Client 49 (Charge :16, Temps :51) -> Client 30 (Charge :20, Temps :57) -> 0 (Charge :24/30, Temps :76)

**Itinéraire 6 :** 0 -> Client 43 (Charge :0, Temps :9) -> Client 42 (Charge :3, Temps :14) -> Client 5 (Charge :7, Temps :20) -> Client 45 (Charge :11, Temps :26) -> Client 47 (Charge :13, Temps :45)

-> Client 15 (Charge :15, Temps :65) -> Client 7 (Charge :18, Temps :69) -> Client 3 (Charge :20, Temps :72) -> Client 26 (Charge :25, Temps :81) -> 0 (Charge :28/30, Temps :89)

**Itinéraire 7 :** 0 -> Client 48 (Charge :0, Temps :4) -> Client 10 (Charge :3, Temps :11) -> Client 27 (Charge :7, Temps :50) -> Client 41 (Charge :9, Temps :59) -> 0 (Charge :14/30, Temps :76) Clients non visités : None

### 3.5 Explication :

La mention "Optimal solution found" indique que Gurobi a réussi à trouver la solution optimale pour le problème donné avec une tolérance de 1.00e-04, ce qui signifie une précision très élevée dans les résultats obtenus. "Best objective 1.161800000000e+04" réfère à la valeur optimale de la fonction objectif du problème, qui pourrait représenter, par exemple, le coût total minimisé, la distance totale parcourue, ou tout autre critère d'optimisation spécifié dans le modèle.

"Best bound" est identique à la "Best objective", indiquant que l'estimation la plus basse possible (bound) coïncide avec l'objectif atteint, confirmant l'optimalité de la solution.

"Gap 0.0000" montre qu'il n'y a pas d'écart entre la meilleure solution trouvée et la borne inférieure estimée pour l'objectif, renforçant la conclusion que la solution est véritablement optimale.

Il y a 7 itinéraires différents qui sont expliqués dans la partie résolution.

# CHAPITRE4

## APPROCHES MÉTAHEURISTIQUES DU PROBLÈME DE ROUTAGE DE VÉHICULES AVEC CAPACITÉ LIMITÉE

### Sommaire

---

<b>Introduction</b> . . . . .	<b>37</b>
<b>4.1 Explication de l'ALNS</b> . . . . .	<b>37</b>
<b>4.2 Réponses avec différents jeux de données</b> . . . . .	<b>38</b>
4.2.1 Avec 32 villes : . . . . .	38
4.2.2 Avec 242 villes : . . . . .	40
4.2.3 Avec 1001 points : . . . . .	42
<b>4.3 Analyse des Performances</b> . . . . .	<b>43</b>
4.3.1 Avec 32 Villes . . . . .	43
4.3.2 Avec 242 Villes . . . . .	44
4.3.3 Avec 1001 Points . . . . .	44

---

## Introduction

Dans cette question, nous allons aborder une approche alternative pour résoudre le problème de tournées de véhicules avec capacité limitée (CVRP) en utilisant une métaheuristique différente des méthodes traditionnelles comme le Tabou, le recuit simulé ou les algorithmes génétiques. Nous sélectionnerons la métaheuristique ANLS. Nous expliquerons comment cette métaheuristique fonctionne et l'appliquerons ensuite sur des instances de différentes tailles du CVRP, extraites de la TSPLIB, pour évaluer son efficacité et sa capacité à trouver des solutions optimales ou proches de l'optimal. Notre approche a été inspirée par l'application pratique de l'heuristique ALNS dans le contexte du problème de routage de véhicules, comme démontré par Wouda (2024)

### 4.1 Explication de l'ALNS

L'Adaptive Large Neighborhood Search (ALNS) est une métaheuristique pour la résolution de problèmes d'optimisation combinatoire, proposée initialement par Ropke et Pisinger en 2006. Elle vise à résoudre diverses variantes du problème de tournées de véhicules (VRP). L'ALNS est caractérisée par son approche itérative qui intègre des opérateurs de destruction et de réparation pour parcourir l'espace de recherche.

Dans le cadre de ce travail, l'auteur a développé une implémentation de l'ALNS destinée au problème de tournées de véhicules avec capacité limitée (CVRP). Une classe 'CvrpState' a été introduite afin de modéliser l'état de la solution, comprenant une liste de routes et une liste de clients non assignés. Pour explorer l'espace de recherche, des opérateurs de destruction et de réparation ont été mis en place.

L'opérateur de destruction conçu, nommé 'random\_removal', élimine aléatoirement un certain nombre de clients de la solution en cours. Quant à l'opérateur de réparation, dénommé 'greedy\_repair', il réinsère les clients non assignés dans la solution actuelle selon une logique gloutonne. L'auteur a aussi mis en œuvre une fonction 'nearest\_neighbor' pour créer une solution initiale.

L'utilisation de la bibliothèque 'alns' a permis de structurer l'ALNS, intégrant l'opérateur de destruction et l'opérateur de réparation. Une fonction de sélection a été définie pour choisir l'opérateur de destruction à utiliser à chaque itération, et une fonction d'acceptation a été établie pour décider de l'adoption ou non de nouvelles solutions. L'ALNS a ensuite été exécuté durant un temps défini, avec une présentation des résultats obtenus.

Pour améliorer l'approche, l'auteur explore une méthode plus sophistiquée pour les opérateurs de destruction et de réparation dans le CVRP, nommée Slack Induction by Substring Removal (SISR), introduite par Christiaens et Vanden Berghe en 2020.

Contrairement à la suppression aléatoire, l'opérateur de destruction dans SISR élimine des sous-chaînes de routes basées sur leur proximité. Un mécanisme de "clignotement" est ajouté à l'opérateur de réparation glouton, permettant d'omettre certaines vérifications d'insertion.

L'auteur a implémenté une version simplifiée de cet opérateur de suppression de chaînes, remplaçant l'opérateur de destruction aléatoire. Les paramètres MAX\_STRING\_REMOVALS et MAX\_STRING\_SIZE régulent le nombre maximal de sous-chaînes à retirer et la taille maximale de ces sous-chaînes. La fonction 'string\_removal' reçoit en entrée l'état actuel de la solution et un état aléatoire, et produit un nouvel état détruit par suppression de sous-chaînes de routes autour d'un client choisi

aléatoirement. La fonction ‘remove\_string‘ est utilisée pour éliminer une sous-chaîne spécifique d’une route.

En résumé, l’ALNS est une métaheuristique pour la résolution de problèmes d’optimisation combinatoire qui utilise des opérateurs de destruction et de réparation pour explorer l’espace de recherche.

## 4.2 Réponses avec différents jeux de données

### 4.2.1 Avec 32 villes :

Meilleure solution connue :

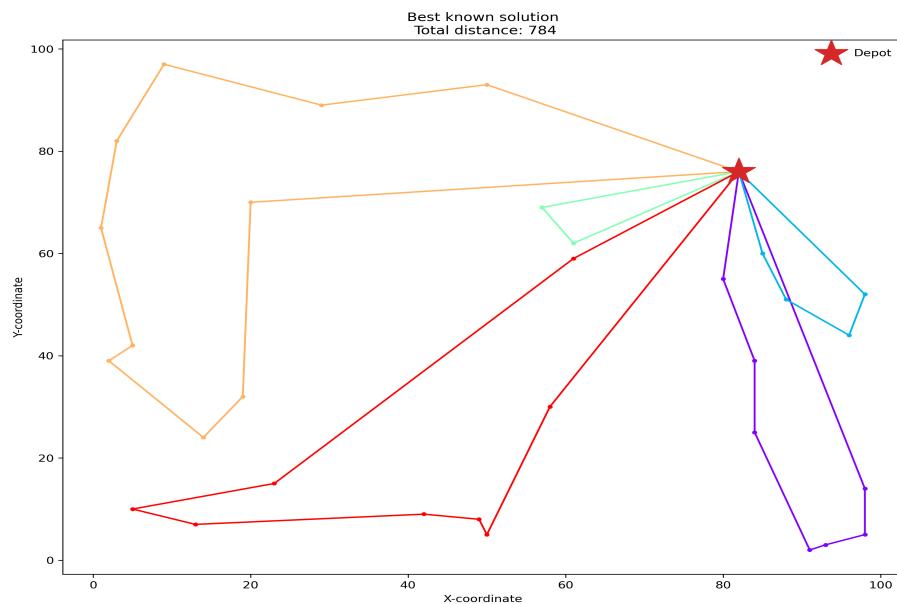


FIGURE 4.1 – Meilleure solution connue

ANLS Simple :

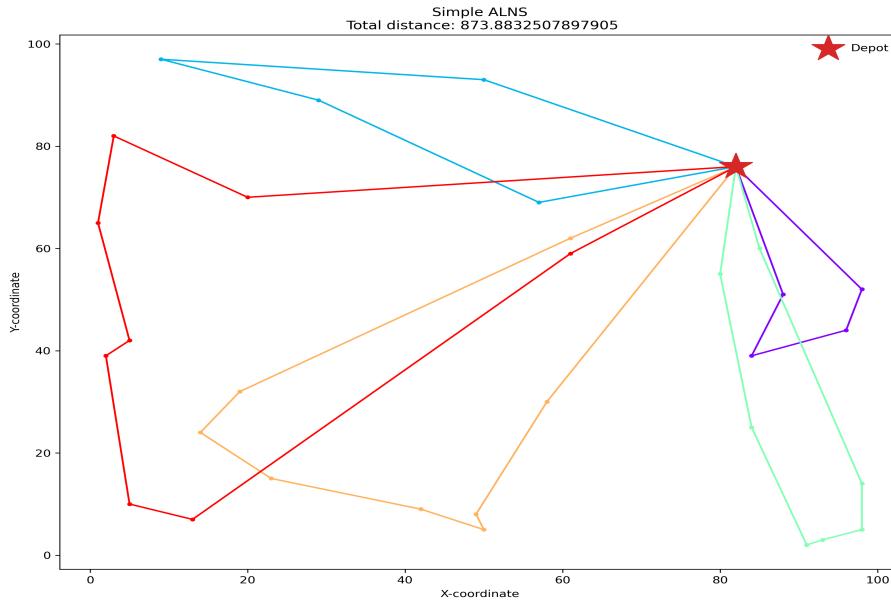


FIGURE 4.2 – ANLS Simple

Distance total = 873.88 11.5

ANLS avec Slack Induction by Substring Removal (SISR)

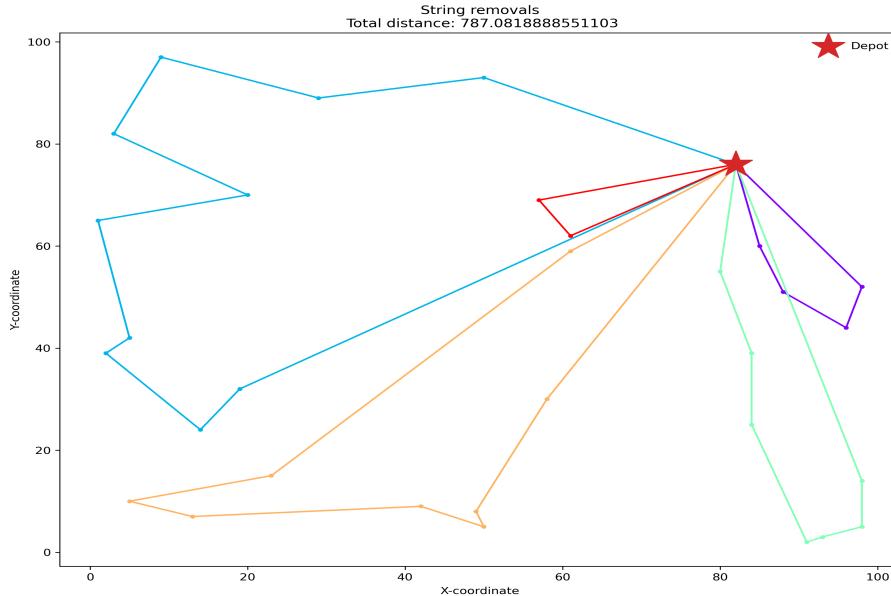


FIGURE 4.3 – String removals

Distance : 787.0818888551103. C'est à 0.4% de optimal

#### 4.2.2 Avec 242 villes :

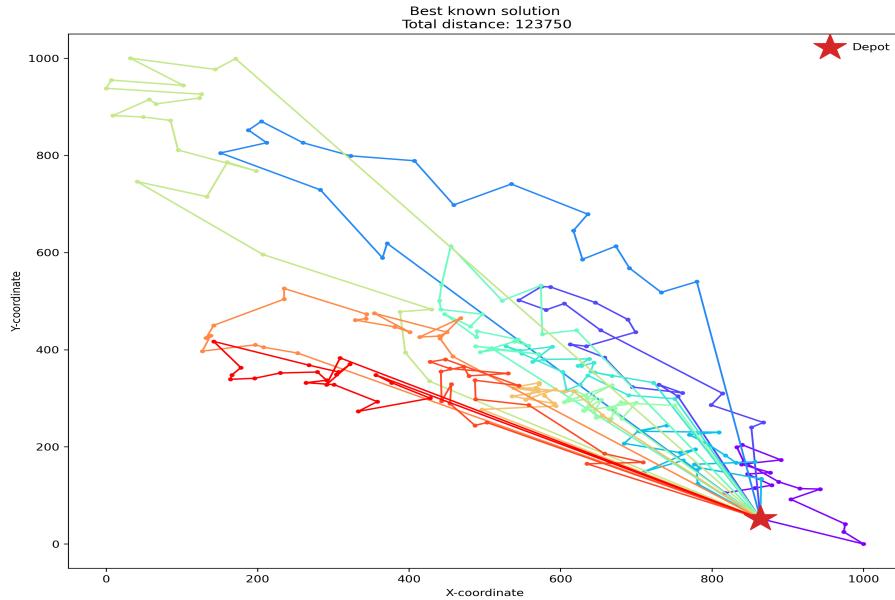


FIGURE 4.4 – Meilleure solution connu

Meilleure solution : 123750

ANLS Simple :

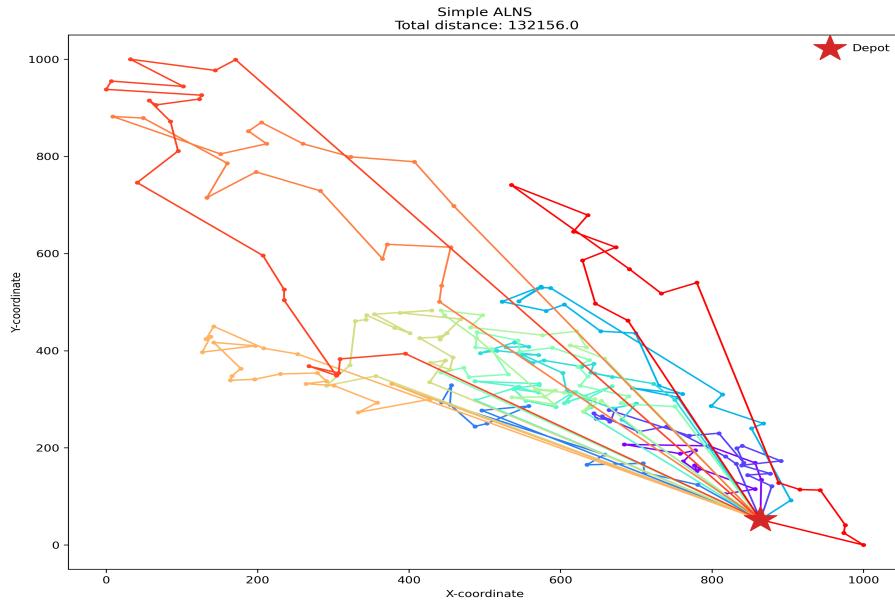


FIGURE 4.5 – ANLS Simple

Distance 132156 à 6.8% de l'optimal ANLS avec Slack Induction by Substring Removal (SISR)

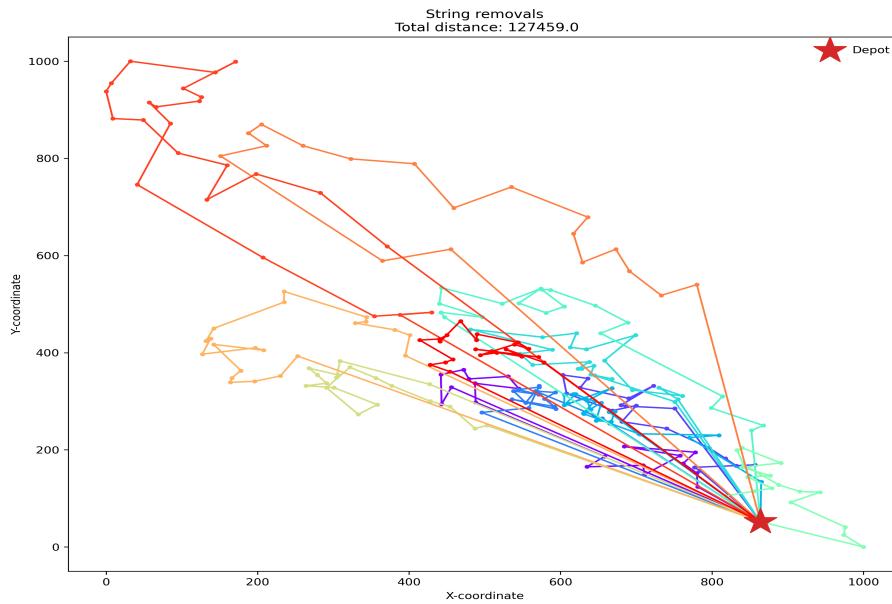


FIGURE 4.6 – String removals

Distance 127459 à 3% de la solution optimal.

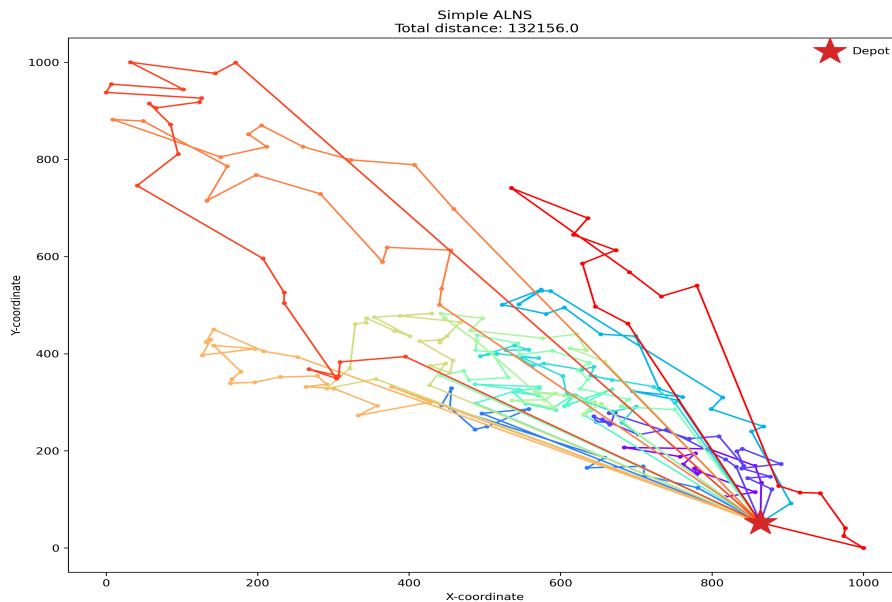


FIGURE 4.7 – Simple ANLS

Meilleure solution : 123750 ANLS Simple :

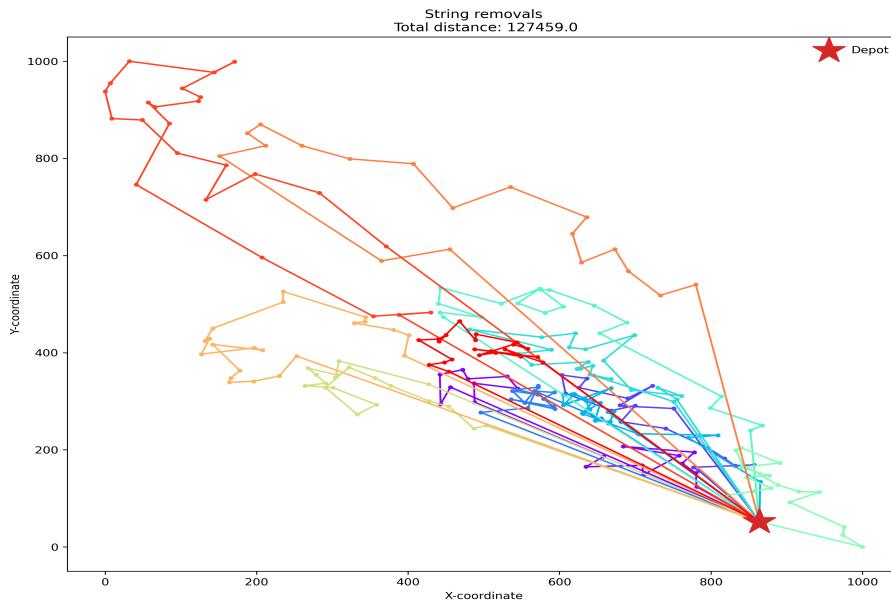


FIGURE 4.8 – String removals

Distance 132156 à 6.8% de l'optimal ANLS avec Slack Induction by Substring Removal (SISR)

#### 4.2.3 Avec 1001 points :

Meilleure solution connu : 72355

Simple ANLS :

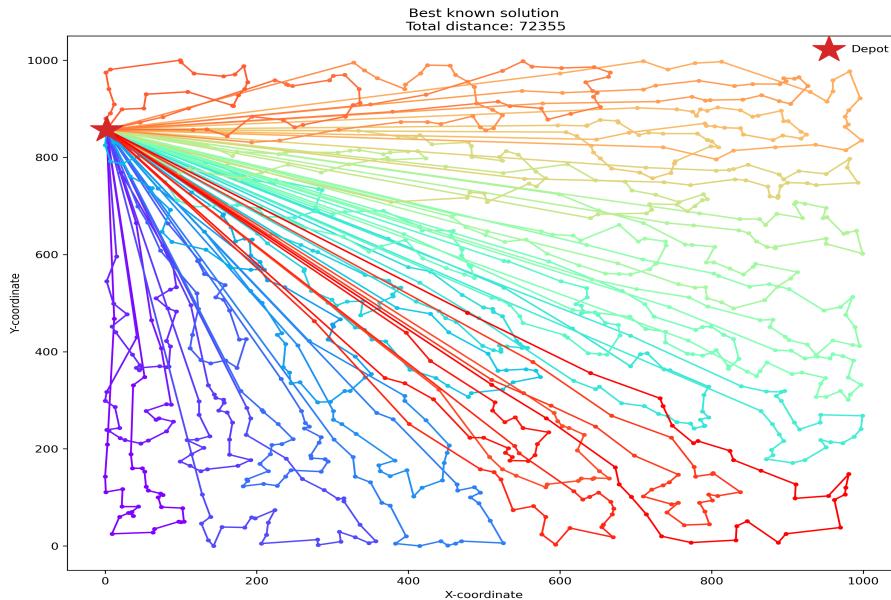


FIGURE 4.9 – Meilleure solution connu

Meilleure solution connu : 72355

Simple ANLS :

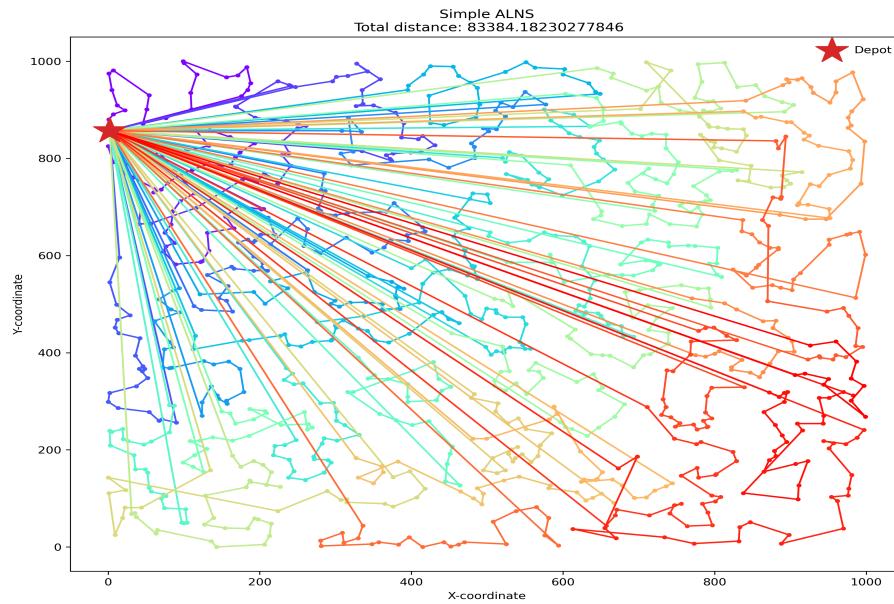


FIGURE 4.10 – ANLS Simple

Solution 83384 à 15% de la meilleure solution. ANLS avec Slack Induction by Substring Removal (SISR)

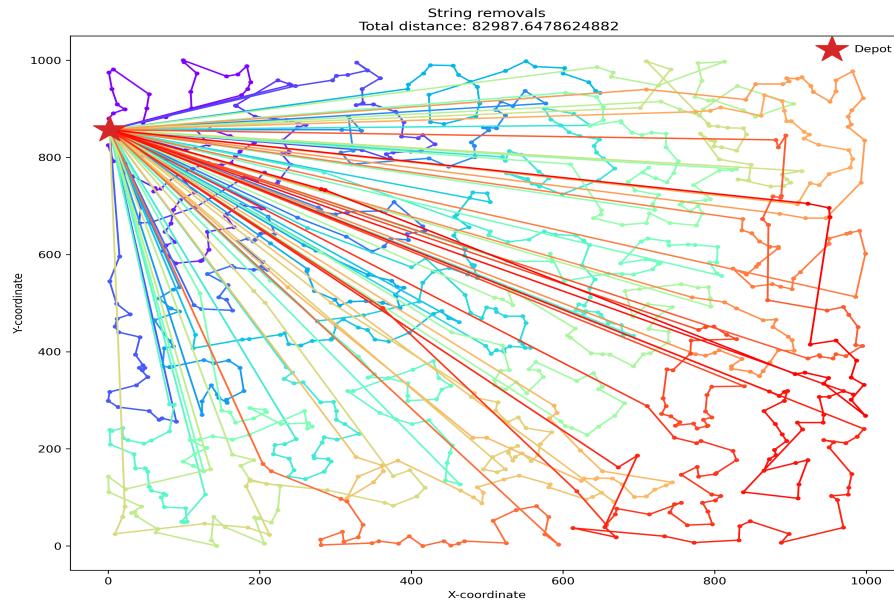


FIGURE 4.11 – String removals

Solution 82987 à 14.7% de la solution optimale.

### 4.3 Analyse des Performances

#### 4.3.1 Avec 32 Villes

**ANLS Simple :** La distance totale obtenue est de 873.88, ce qui représente 11.5% au-dessus de l'optimal. Ceci montre une performance relativement bonne, mais il y a clairement une marge d'amélio-

ration.

**ANLS avec SISR :** La distance est réduite à 787.08, soit seulement 0.4% au-dessus de l'optimal, démontre une amélioration significative en intégrant la méthode SISR. Cela souligne l'efficacité de la suppression ciblée de sous-chaînes et de l'insertion avec clignotement dans l'amélioration de la qualité de la solution.

#### 4.3.2 Avec 242 Villes

**ANLS Simple :** La distance de 132156, soit à 6.8% de l'optimal, indique que la performance de l'ANLS simple diminue avec l'augmentation de la taille du problème.

**ANLS avec SISR :** L'amélioration de la distance à 127459, à 3% de l'optimal, montre encore une fois l'efficacité de la méthode SISR, bien que l'écart par rapport à l'optimal soit plus grand comparé au cas de 32 villes. Cela peut indiquer que la complexité croissante du problème affecte l'efficacité des opérateurs de destruction et de réparation.

#### 4.3.3 Avec 1001 Points

**ANLS Simple :** La solution trouvée à 83384, soit 15% au-dessus de l'optimal, montre une dégradation notable de la performance avec l'augmentation de la taille de l'ensemble de données.

**ANLS avec SISR :** Une légère amélioration est observée avec une distance de 82987, soit 14.7% au-dessus de l'optimal. Bien que l'amélioration soit modeste, cela suggère que la méthode SISR conserve une certaine efficacité même pour de grands ensembles de données.

### Conclusion et Recommandations

**Efficacité de SISR :** L'intégration de la méthode Slack Induction by Substring Removal (SISR) dans l'ALNS améliore systématiquement la performance par rapport à l'ANLS simple, soulignant l'importance d'opérateurs de destruction et de réparation sophistiqués dans la métaheuristique ALNS.

**Impact de la Taille du Problème :** L'efficacité relative des deux approches diminue avec l'augmentation de la taille du problème. Cela indique que des ajustements ou des améliorations supplémentaires des opérateurs pourraient être nécessaires pour maintenir l'efficacité sur de plus grands ensembles de données.

**Recherche Future :** Des recherches supplémentaires pourraient explorer l'intégration d'autres opérateurs de destruction et de réparation, ou l'ajustement des paramètres existants de SISR pour améliorer davantage la performance, surtout pour les ensembles de données de grande taille.

# BIBLIOGRAPHIE

- [1] OpenAI. (2024). ChatGPT : OpenAI ChatGPT model documentation. Retrieved from <https://openai.com>
- [2] Ruthmair, M. (2024). VRP GitHub Repository. Retrieved from <https://github.com/ruthmair/vrp/blob/main/README.md>
- [3] Wouda, N. (2024). ALNS GitHub Repository : Capacitated Vehicle Routing Problem Example. Retrieved from [https://github.com/N-Wouda/ALNS/blob/master/examples/capacitated\\_vehicle\\_routing\\_problem.ipynb](https://github.com/N-Wouda/ALNS/blob/master/examples/capacitated_vehicle_routing_problem.ipynb)

