Rapport mini-projet :

Résolution du problème de tournées de véhicules

Réalisé par :

LAHRECH Ilham

AZLAG Nouhaila

FAKHAM Chaimaa

Table des matières

[Introduction 3](#_Toc136257574)

[Description du problème 3](#_Toc136257575)

[Définitions 4](#_Toc136257576)

[VRP 4](#_Toc136257577)

[TSP 4](#_Toc136257578)

[Domaine D’application 5](#_Toc136257579)

[Variantes du VRP 5](#_Toc136257580)

[VRP avec capacité (CVRP) : 5](#_Toc136257581)

[VRP avec fenêtres de temps (VRPTW): 5](#_Toc136257582)

[VRP avec ramassage et livraison (VRPPD) : 5](#_Toc136257583)

[Formulations 7](#_Toc136257584)

[Formulation Graphique 7](#_Toc136257585)

[Formulation Mathématique 7](#_Toc136257586)

[modèle mathématique 7](#_Toc136257587)

[Indices 7](#_Toc136257588)

[Paramètres 7](#_Toc136257589)

[Variables 8](#_Toc136257590)

[Objectif : 8](#_Toc136257591)

[Variante CVRP 8](#_Toc136257592)

[La fonction objectif: 8](#_Toc136257593)

[Les contraintes : 9](#_Toc136257594)

[Contrainte1: 9](#_Toc136257595)

[Contrainte2: 9](#_Toc136257596)

[Contrainte3: 9](#_Toc136257597)

[Contrainte4: 9](#_Toc136257598)

[Méthode de Résolution du VRP(Exactes /Heuristique) 10](#_Toc136257599)

[Exemple de Résolution de VRP 10](#_Toc136257600)

[CPLEX: 11](#_Toc136257601)

[CPLEX est un outil informatique d’optimisation commercialise par IBM depuis son acquisition de l’entreprise français ILOG en 2009. Son nom fait référence au langage C et a l’algorithme de simplexe. 11](#_Toc136257602)

[CPLEX travail sur la version de Python 3.7 et 3.8 . 11](#_Toc136257603)

[CPLEX 22.10 installation sur Anaco nda Environnement 11](#_Toc136257604)

[La librairie DOCPLEX: 11](#_Toc136257605)

[Validation de la solution 11](#_Toc136257606)

[Application Pratique 11](#_Toc136257607)

[Résultat 16](#_Toc136257608)

**Conclusion ……………………………………………………………………………………………………………………………………18**

# Introduction

Le Problème de tournées de véhicules (VRP) est un problème d'optimisation combinatoire qui cherche à déterminer des itinéraires optimaux pour un ensemble de véhicules afin de livrer un ensemble de clients tout en respectant des contraintes spécifiques. Le VRP trouve des applications dans de nombreux domaines tels que la logistique, la distribution, le transport de marchandises, la gestion des flottes.

Cette présentation donne un aperçu du Problème de tournées de véhicules (VRP) en mettant l'accent sur sa définition, ses domaines d'application, sa formulation mathématique, ses variantes, les méthodes de résolution et une conclusion générale sur son importance.

# Description du problème

Le problème de tournées de véhicules (aussi appelé VRP pour Vehicle Routing Problem) est une classe de problèmes de recherche opérationnelle et d’optimisation combinatoire. Il s’agit de déterminer les tournées d’une flotte de véhicules afin de livrer une liste de clients, ou de réaliser des tournées d’interventions (maintenance, réparation, contrôles) ou de visites (visites médicales, commerciales, etc.). Le but de ce mini-projet est de minimiser le coût de livraison des biens en utilisant les algorithmes d’apprentissage. Ce problème est une extension classique du problème du voyageur de commerce, et fait partie de la classe des problèmes NP-complet.

# Définitions

## VRP

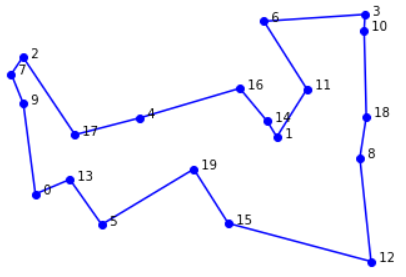
Le Problème de tournées de véhicules, noté VRP pour Véhicule Routing Problème est un problème appartenant à la classe des problèmes d’optimisation, il consiste à déterminer des tournées pour un ensemble de véhicules partant d'un dépôt central et devant visiter un certain nombre de clients. L'objectif est de minimiser la distance totale parcourue, le coût de transport, ou d'autres critères spécifiques tout en respectant des contraintes telles que la capacité des véhicules, les fenêtres de temps des clients.

## TSP

 Le problème du voyageur de commerce (The traveling salesman problem) (TSP) est l'un des problèmes d'optimisation combinatoire les plus connus étudiés dans la littérature de recherche opérationnelle.

Elle consiste à déterminer un tour qui commence et se termine à un nœud de base donné après avoir visité un ensemble de nœuds exactement une fois tout en minimisant la distance totale. Résoudre le problème TSP est crucial car il appartient à la classe des non-polynômes (NP)-complets.

Dans cette classe de problèmes, aucun algorithme en temps polynomial n'a été découvert. Siquelqu'un trouve un algorithme TSP efficace, il peut être étendu à d'autres problèmes de classe NP-complet. Malheureusement, à ce jour, personne n'a pu le faire. Le TSP est divisé en deux catégories, symétrique et asymétrique, en fonction de la distance entre deux nœuds. En TSP asymétrique (ATSP), la distance d'un nœud à un autre est différent de la distance inverse, et en TSP symétrique (STSP), cette distance est la même. Comme mentionné précédemment, la TSP consiste à déterminer un circuit de distance minimale passant par chaque sommet une et une seule fois. Un tel circuit est appelé tour ou circuit hamiltonien (ou cycle).



# Domaine D’application

Le VRP est largement utilisé dans les opérations logistiques et la gestion des transports. Il est utilisé pour planifier les itinéraires des véhicules de livraison, optimiser les tournées des camions de collecte des déchets, organiser les déplacements des techniciens de maintenance, etc. Il est également utilisé dans le domaine du e-commerce, où les entreprises cherchent à optimiser la livraison de leurs produits aux clients de manière efficace et rentable.

## Variantes du VRP

### VRP avec capacité (CVRP) :

Contrainte : Chaque véhicule a une capacité maximale qu'il ne peut pas dépasser lors de la livraison des marchandises aux clients. Objectif : Minimiser le nombre de véhicules utilisés tout en respectant les contraintes de capacité.

### VRP avec fenêtres de temps (VRPTW):

Contrainte : Chaque client a une fenêtre de temps spécifiée pendant laquelle il doit être livré. Objectif : Minimiser le temps total de parcours tout en respectant les contraintes de fenêtres de temps.

### VRP avec ramassage et livraison (VRPPD) :

Contrainte : Les véhicules doivent ramasser des marchandises à certains endroits avant de les livrer à d'autres endroits. Objectif : Minimiser le temps total de parcours et respecter les contraintes de ramassage et de livraison.

VRP à plusieurs dépôts (MDVRP) :

Contrainte : Il y a plusieurs dépôts disponibles pour les véhicules.

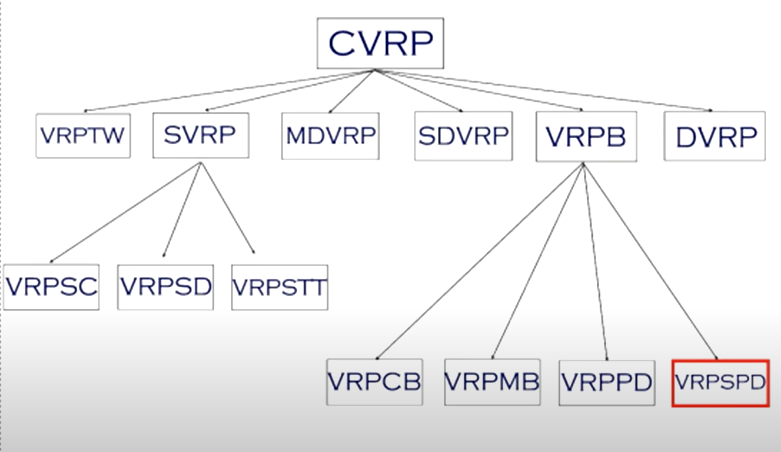
Objectif : Minimiser le temps total de parcours en utilisant efficacement les dépôts disponibles.

VRP avec contraintes de durée de travail (VRPWT) :

Contrainte : Il y a plusieurs dépôts Contrainte : Les conducteurs des véhicules ont des contraintes de durée de travail maximale. Objectif : Minimiser le temps total de parcours tout en respectant les contraintes de durée de travail des conducteurs. Temps.

VRP dynamique (DVRP) :

Contrainte : Les demandes des clients et les conditions du réseau routier peuvent changer au fil du temps. Objectif : Adapter les itinéraires des véhicules en temps réel pour répondre aux demandes changeantes et minimiser les coûts.



### Formulations

### Formulation Graphique

De nombreuses formulations du VRP existent dans la littérature. Le point commun de toutes ces formulations est la représentation graphique du problème de tournées de véhicules sous forme d'un graphe orienté.

La représentation graphique du VRP classique peut être décrite comme suit.

Soit G = (X, U) un graphe orienté où :

X est l'ensemble des sommets du graphe G, qui représente les clients et le(s) dépôt(s) du VRP.

On a n + 1 sommets du graphe , où 0 représente le(s) dépôt(s) . Chaque client i appartenant à X \ {0} a une demande de produit di, qui correspond à la quantité qu'il faut livrer ou collecter.

X =n U {0} et (|X| = n + 1).

U est l'ensemble des arcs du graphe G, qui représente les chemins reliant les clients entre eux et le dépôt du VRP.

(|U|= (n + 1)n/2) arcs possibles dans le graphe.

Dans cette représentation, une production peut être effectuée sur les sommets (resp arcs) pour définir la quantité demandée par le client (resp distance séparant deux clients ou le temps de déplacement …

) .

### Formulation Mathématique

C'est un problème de d'optimisation mathématique c'est à dire qu'il constitué de :

Variable de décision .

D’une fonction objective.

Des contraintes à satisfaire.

La variable de décision Xijk=1 si l'arc(i,j)est parcouru par K ème véhicule .

### modèle mathématique

### Indices

I: L'ensemble de villes (clients).

K: L'ensemble de véhicules.

### Paramètres

Cij : Cout du trajet entre ville i et ville j .

di : la demande du client i .

qk:la capacité du véhicule k .

### Variables

Cij : Cout du trajet entre ville i et ville j .

di : la demande du client i .

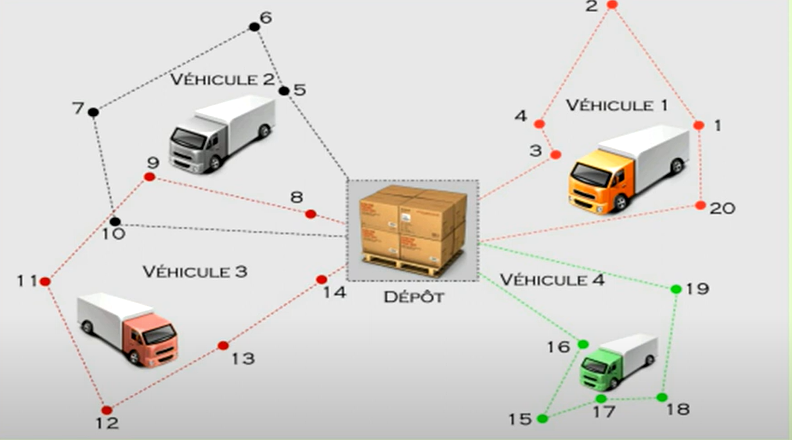
qk:la capacité du véhicule k .

### Objectif :

Minimiser la fonction objectif représentant le coût total de la tournée des véhicules .

## Variante CVRP

L’objectif du CVRP est de minimiser le coût total, c-à-dire la somme des distances ou des temps de parcours des tournées, tout en respectant la contrainte de capacité des véhicules : la quantité de marchandises livrées sur une tournée ne doit pas dépasser la capacité du véhicule qui l’assure. La figure au-dessous , représente un exemple de problème 1 de VRP avec 20 clients, résolu avec 4 véhicules .



*exemple : problème avec 20 clients et 4 véhicules.*

### La fonction objectif:

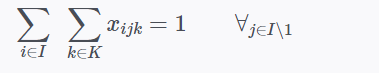
Capture d’écran

## Les contraintes :

## Contrainte1:

## 

La contrainte 1, assure que chaque nœud est entré une fois par un véhicule.



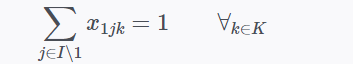
## Contrainte2:

La contrainte 2, assure la sortie de chaque véhicule est le point depuis il entre.



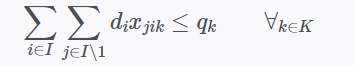
## Contrainte3:

La contrainte 3 Chaque véhicule quitte le dépôt.



## Contrainte4:

La contrainte 4, assure que la somme de livraison par véhicule, ne dépasse pas la capacité du véhicule.



Contraintes 5 :

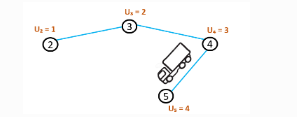
Les sous tours:

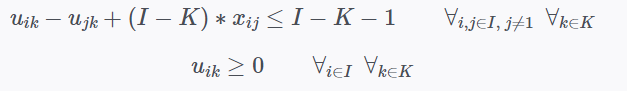
Éliminer les sous tours à l'aide de la méthode Miller–Tucker–Zemlin (MTZ) .

La formulation Miller-Tucker-Zemlin (MTZ) utilise une variable supplémentaire. La variable s'appelle ui et obtient une valeur pour chaque nœud, à l'exception du dépôt. Si un véhicule roule du nœud i au nœud j, la valeur de uj doit être plus grand que la valeur de ui.

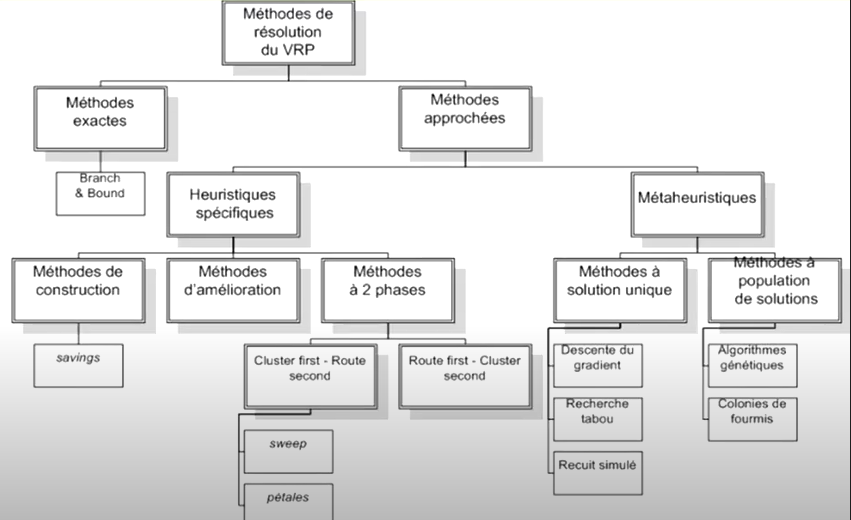


Ainsi, chaque fois qu'un nouveau nœud est visité, la valeur de ui augmente .





# Méthode de Résolution du VRP(Exactes /Heuristique)



## Exemple de Résolution de VRP

* La programmation linéaire : cplex.
* L’heuristique : k-means (K-moyenne).
* Les méta-heuristiques a population de solution : les algorithmes génétique .

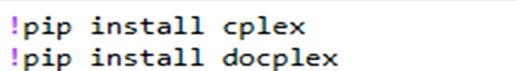
## CPLEX:

## CPLEX est un outil informatique d’optimisation commercialise par IBM depuis son acquisition de l’entreprise français ILOG en 2009. Son nom fait référence au langage C et a l’algorithme de simplexe.

## CPLEX travail sur la version de Python 3.7 et 3.8 .

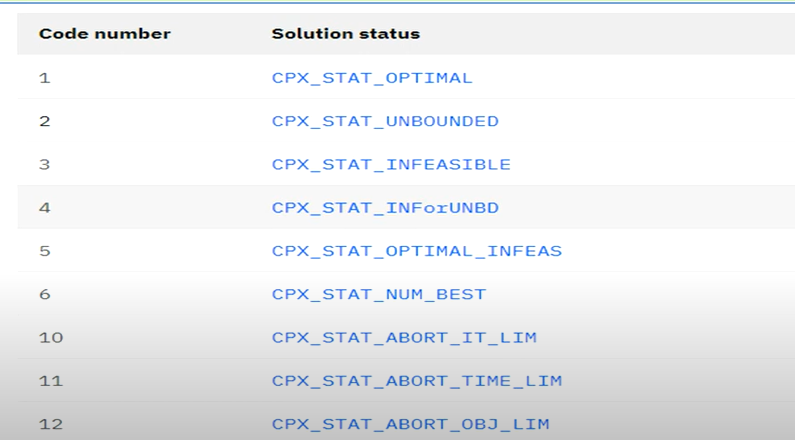
## CPLEX 22.10 installation sur Anaco nda Environnement

## **La librairie DOCPLEX:**



## Validation de la solution

## 



## Application Pratique

from docplex.mp.model import Model

import matplotlib.pyplot as plt

from tkinter import Frame

import numpy as np

from tkinter import Tk, Label, Entry, Button

from PIL import Image, ImageTk

nbVille = None

nbVehicules = None

def get\_values():

    global nbVille1, nbVehicules

    nbVille1 = int(entry\_clients.get())

    nbVehicules = int(entry\_vehicules.get())

    # Ferme la fenêtre et continue avec le reste du code en utilisant les valeurs entrées par l'utilisateur

    root.destroy()

# Crée la fenêtre principale

root = Tk()

root.title("Entrée des valeurs")

image = Image.open("D:\CI\_GI\S3\JEE\camion-removebg-preview.png")

photo = ImageTk.PhotoImage(image)

# Crée un widget Label pour afficher l'image en arrière-plan

background\_label = Label(root, image=photo)

background\_label.place(x=0, y=0, relwidth=1, relheight=1)

window\_width = 300

window\_height = 200

root.geometry(f"{window\_width}x{window\_height}")

# Crée un cadre pour contenir les éléments

frame = Frame(root)

frame.place(relx=0.5, rely=0, anchor='center')

# Crée les libellés et les champs de saisie pour les clients et les véhicules

label\_clients = Label(root, text="Nombre de clients :")

label\_clients.pack()

entry\_clients = Entry(root)

entry\_clients.pack()

label\_vehicules = Label(root, text="Nombre de véhicules :")

label\_vehicules.pack()

entry\_vehicules = Entry(root)

entry\_vehicules.pack()

# Crée le bouton pour valider les valeurs entrées

button\_valider = Button(root, text="Valider", command=get\_values)

button\_valider.pack()

# Lance la boucle principale de l'interface

root.mainloop()

rnd = np.random

rnd.seed(0)

# Index Range ville

nbVille = nbVille1 + 1

ville = range(nbVille)

vehicules = range(nbVehicules)

# Coût entre les nœuds i et j

cij = rnd.rand(nbVille, nbVille)

# Demande de chaque client i

di = rnd.randint(1, 10, nbVille)

di[0] = 0

# Capacité de chaque véhicule k

qk = rnd.randint(0, 200, nbVehicules)

#qk = [10.1 , 80.1 , 110.1, 50]

print(qk)

# Coordonnées des villes

loc\_x = rnd.rand(len(ville)) \* 200 #[100,155,25,13,25,96,162]

loc\_y = rnd.rand(len(ville)) \* 100 #[60,94,79,50,11,17,17]

for i in ville:

    plt.annotate('$D\_%d=%.4f$' % (i, di[i]), (loc\_x[i] + 2, loc\_y[i]))

image = plt.imread("D:\CI\_GI\S3\JEE\map.jpg")

plt.imshow(image, extent=[0, 200, 0, 100])

plt.axis('equal')

model = Model('CVRP')

# Variables de décision

x = model.binary\_var\_cube(keys1=ville, keys2=ville, keys3=vehicules, name='x')

u = model.integer\_var\_matrix(keys1=ville, keys2=vehicules, lb=0, ub=nbVille, name='u')

# Fonction objectif

model.minimize(model.sum(cij[i, j] \* x[i, j, k] for i in ville for j in ville for k in vehicules))

# Contraintes

for j in ville:

    if j > 0:

         model.add\_constraint(model.sum(x[i, j, k] for i in ville for k in vehicules) == 1, ctname='cnt1')

for j in ville:

    for k in vehicules:

        model.add\_constraint(model.sum(x[i, j, k] for i in ville) == model.sum(x[j, i, k] for i in ville), ctname='c2')

for k in vehicules:

    model.add\_constraint(model.sum(x[0, j, k] for j in ville if j > 0) == 1, ctname='c3')

for k in vehicules:

    model.add\_constraint(model.sum(di[j] \* x[i, j, k] for i in ville for j in ville if j > 0) <= qk[k], ctname='c4')

for i in ville:

    for j in ville:

        if j > 0:

            for k in vehicules:

                model.add\_constraint(u[i, k] - u[j, k] + (nbVille - nbVehicules) \* x[i, j, k] <= (nbVille - nbVehicules - 1),

                                     ctname='c5')

print('solution')

solution = model.solve(log\_output=True)

solution.display()

solution.get\_objective\_value()

A = [(i, j, k) for i in ville for j in ville for k in vehicules if i != j]

arcs\_trajet = [a for a in A if x[a].solution\_value > 0.9]

# Afficher le depot avec leurs images

img = plt.imread("D:\CI\_GI\S3\JEE\depot.jpg")

img\_width = 15

img\_height = 10

plt.imshow(img, extent=[loc\_x[0] - img\_width / 2, loc\_x[0] + img\_width / 2, loc\_y[0] - img\_height / 2, loc\_y[0] + img\_height / 2])

# Charger les images des clients

client\_images = [plt.imread(f"D:\CI\_GI\S3\JEE\client\_{i}.jpg") for i in ville[1:]]

# Afficher les clients avec leurs images

for i in ville:

    if i > 0:

        img = client\_images[i - 1]

        img\_width = 6  # Largeur de l'image en unité de coordonnées

        img\_height = 6  # Hauteur de l'image en unité de coordonnées

        plt.imshow(img, extent=[loc\_x[i] - img\_width / 2, loc\_x[i] + img\_width / 2, loc\_y[i] - img\_height / 2, loc\_y[i] + img\_height / 2])

colors = ['b', 'g', 'r', 'c', 'm', 'y']  # Liste des couleurs disponibles

# linestyle=[':','--',':','-.']    , linestyle = linestyle[k  % len(linestyle)]

for i, j, k in arcs\_trajet:

    plt.plot([loc\_x[i], loc\_x[j]], [loc\_y[i], loc\_y[j]], c=colors[k  % len(colors)], alpha = 0.65 , linewidth=2.5 )

plt.axis('equal')

print('Numéro de véhicule')

for k in vehicules:

    for a in A:

        if x[a].solution\_value > 0.9:

            if k == a[-1]:

                print(a[-1], '             ', a[0], '             ', a[1])

cout = f"Coût total du VRP : {solution.get\_objective\_value():.3f}"

plt.scatter(loc\_x, loc\_y, label=f"Clients : {nbVille1}")

#plt.scatter(loc\_x, loc\_y, label=f"Véhicules : {nbVehicules}")

for k in vehicules :

    plt.scatter(loc\_x, loc\_y, label=f"Quantité du véhicules {k+1}: {qk[k]}")

plt.scatter(loc\_x, loc\_y, label= cout)

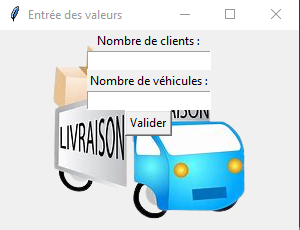
plt.legend()

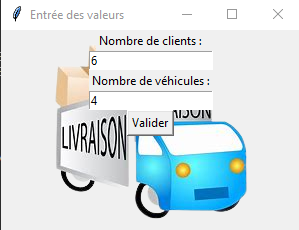
plt.show()

## 

## Résultat

Interface dans laquelle on a saisi le nombre de client et de véhicule :

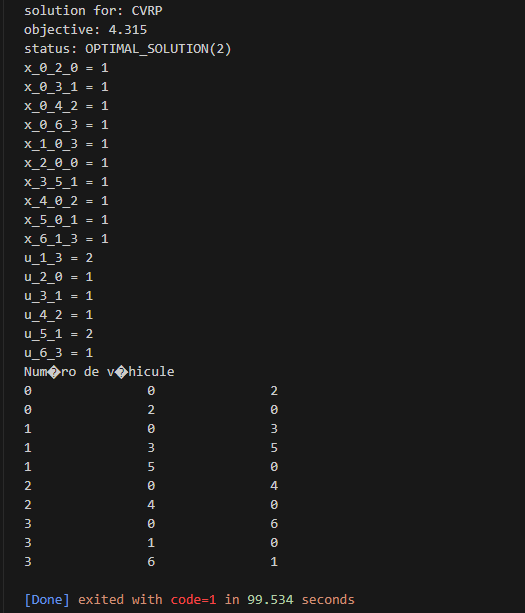




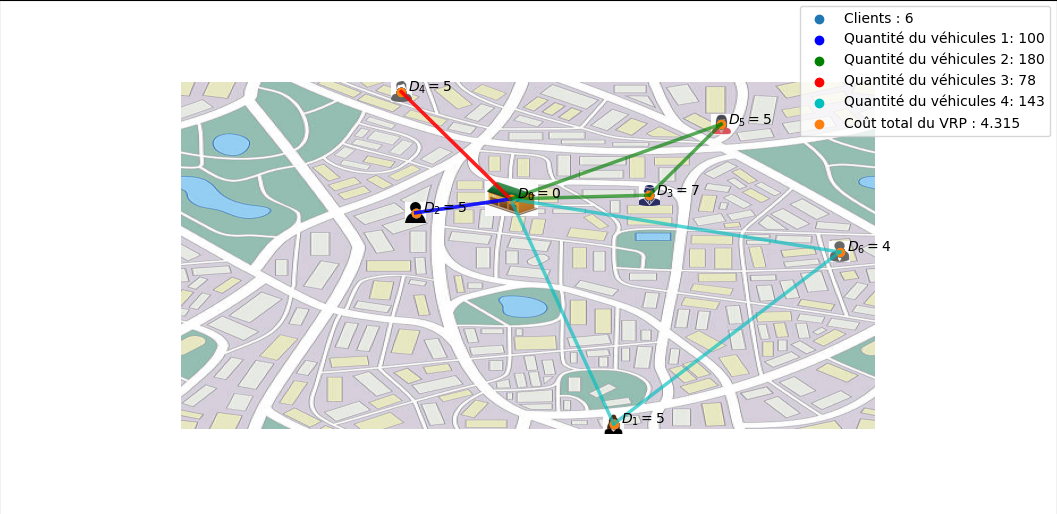
*exemple : problème avec 6 clients et 4 véhicules.*

Dans ce exemple on doit minimiser le coût total pour 6 client et 4 véhicule en respectant la contrainte de capacité des véhicules :

Affichage de console Apres L’exécution :



Affichage de VRP:



Conclusion :

Le Problème de tournées de véhicules (VRP) est un problème d'optimisation complexe avec de nombreuses applications pratiques. Le VRP nécessite une modélisation précise des contraintes et des objectifs spécifiques, ainsi que des méthodes de résolution efficaces pour trouver des solutions optimales ou proches de l'optimal.

Les avancées technologiques, les algorithmes d'optimisation et les techniques de modélisation ont permis d'améliorer la résolution du VRP, contribuant ainsi à une meilleure gestion des ressources et à des opérations logistiques plus efficaces.