ABDELLAOUI Abderrahim

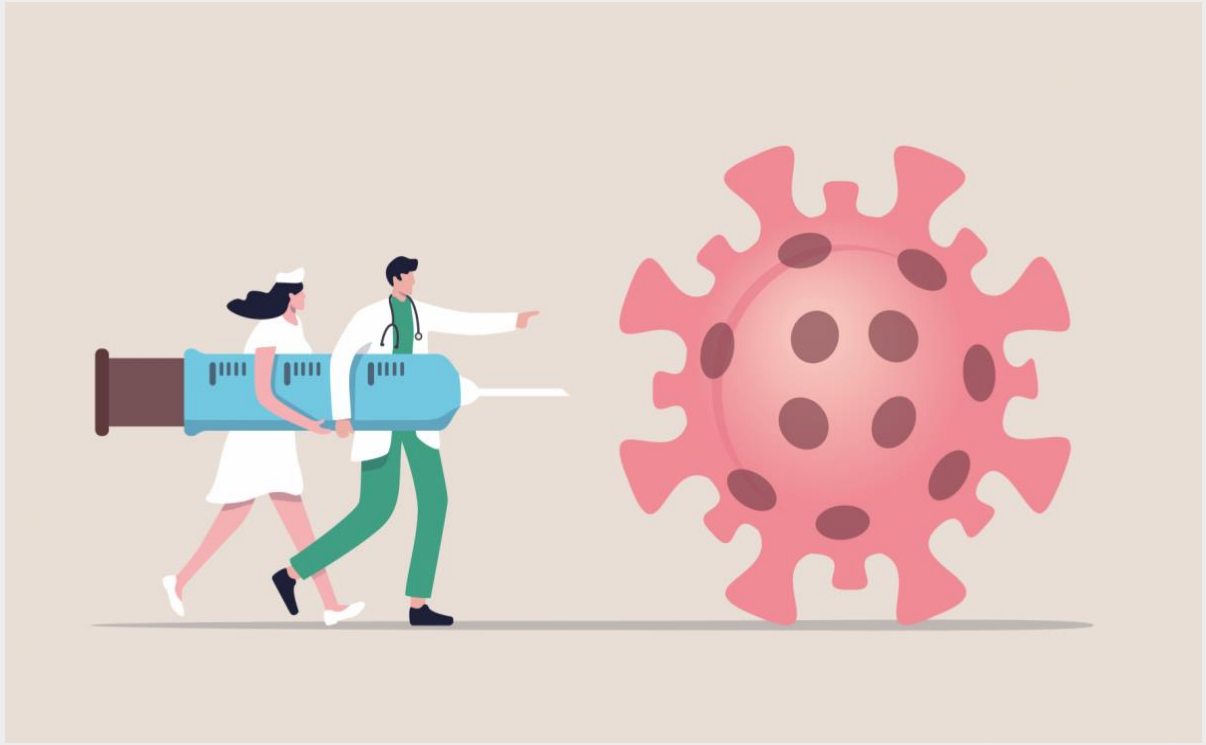
Le voyageur de commerce

N° d'insription 21921

Sommaire

I- Introduction

I- Introduction

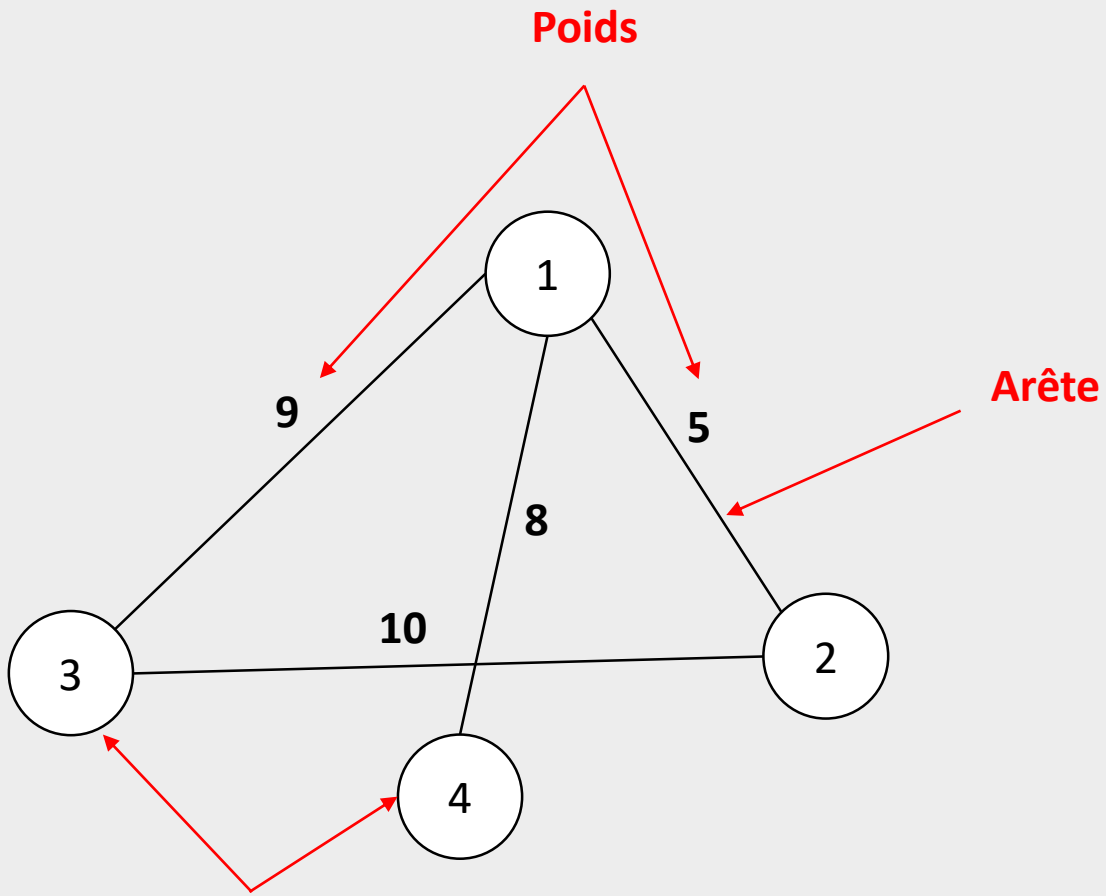


- Minimiser le coût de transport.
- Garantir la sécurité des individus.
- Assurer la distribution du vaccin le plus tôt possible.

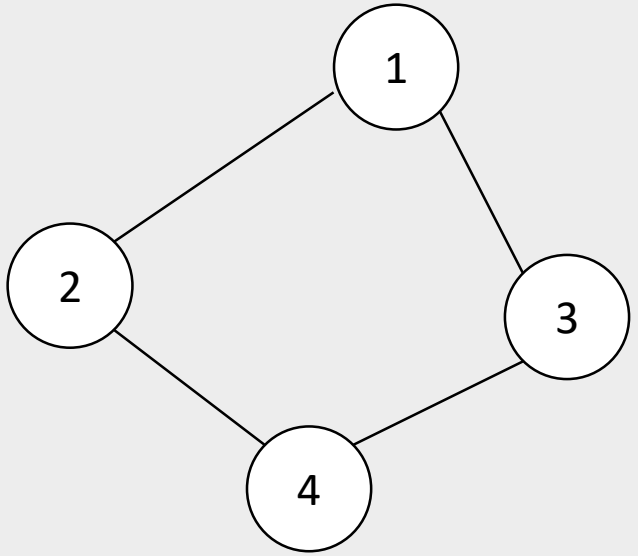
I- Introduction Problématique?

II- Partie mathématique

- Un **graphe fini G = (S,A)** est défini par l'ensemble fini des **sommets** S={$s_1, s_2, \dots \dots s_n$}**,** et par l'ensemble fini A de couple de sommets $(s_i, s_j) \in S^2$ appelés **arêtes**.
- Un graphe **pondéré**, est un graphe dont les arêtes sont affectées d'un nombre appelé **poids** (ou coût).
- Un graphe est dit **non-orienté** si les couples de sommets **ne sont pas ordonnés.**


Poids
Arête
1
2
3
4
9
5
8
10
Sommets

- **Une chaîne** est une suite ordonnée $(e_1, \dots \dots, e_k)$ de k arêtes.
- Un **cycle** est une chaîne dont les extrémités sont égales.
- Un cycle **C** est **hamiltonien** ,s'il passe par chacun de ces sommets exactement une fois.



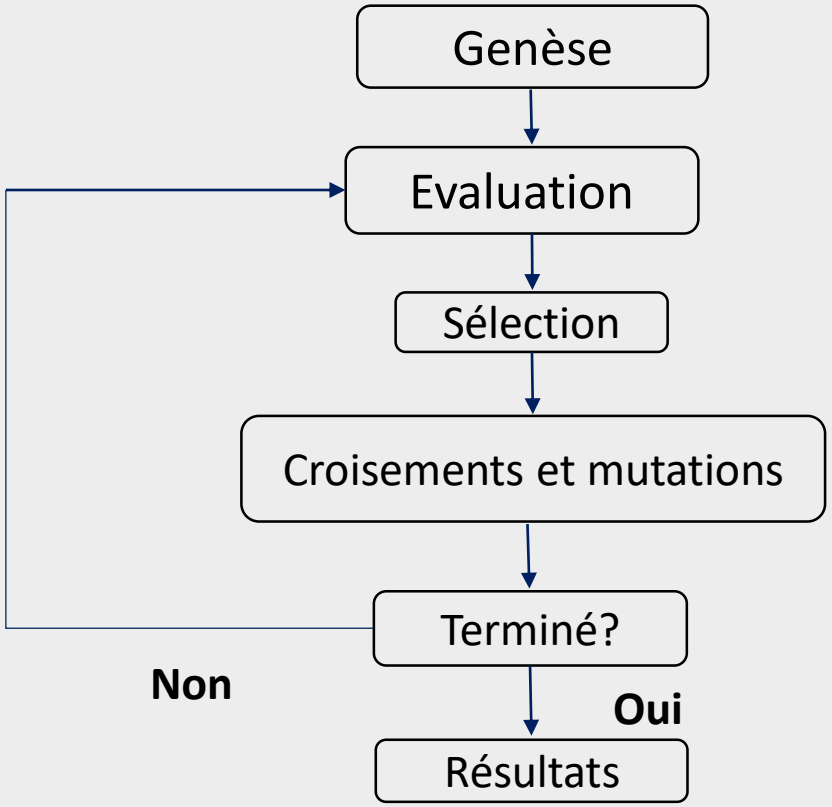
- Soit **(G,A)** un graphe pondéré non-orienté qui possède n sommets numérotés de 1 à n. On appelle **matrice d'adjacence** du graphe la matrice **M=($m_{i,j}$)** tel que:

$\forall (i,j) \in \{1, \ldots\ldots, n\}^2,$

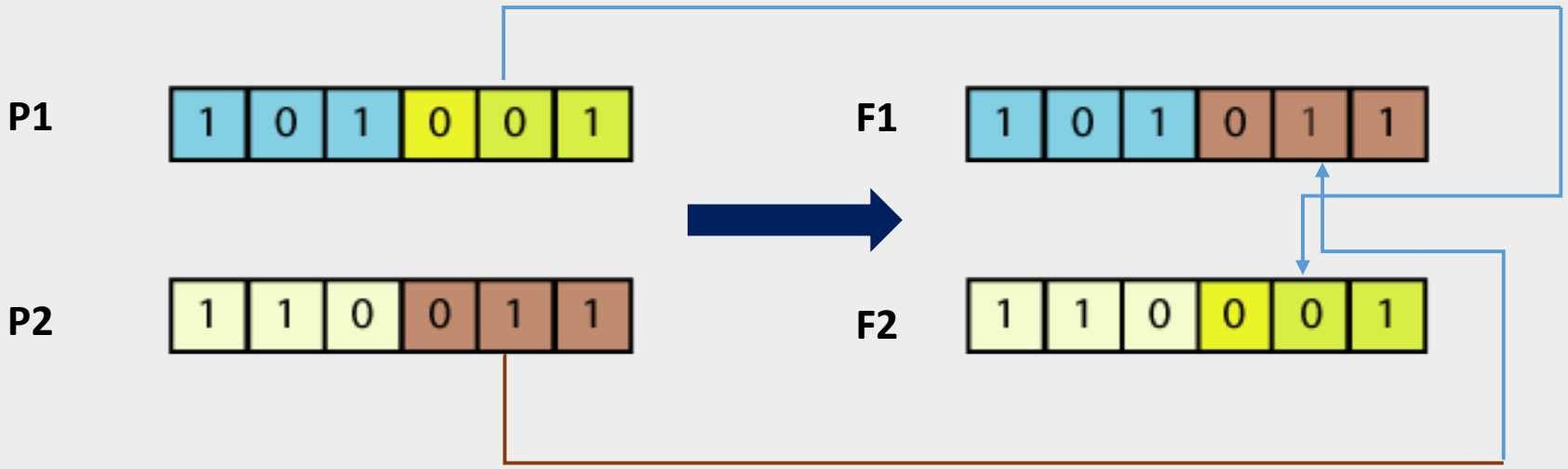
- **$m_{i,j}$ = le poids de l'arête**, si $(s_i, s_j) \in A$
- **$m_{i,j}$ = 0 ,** sinon

III- Partie informatique

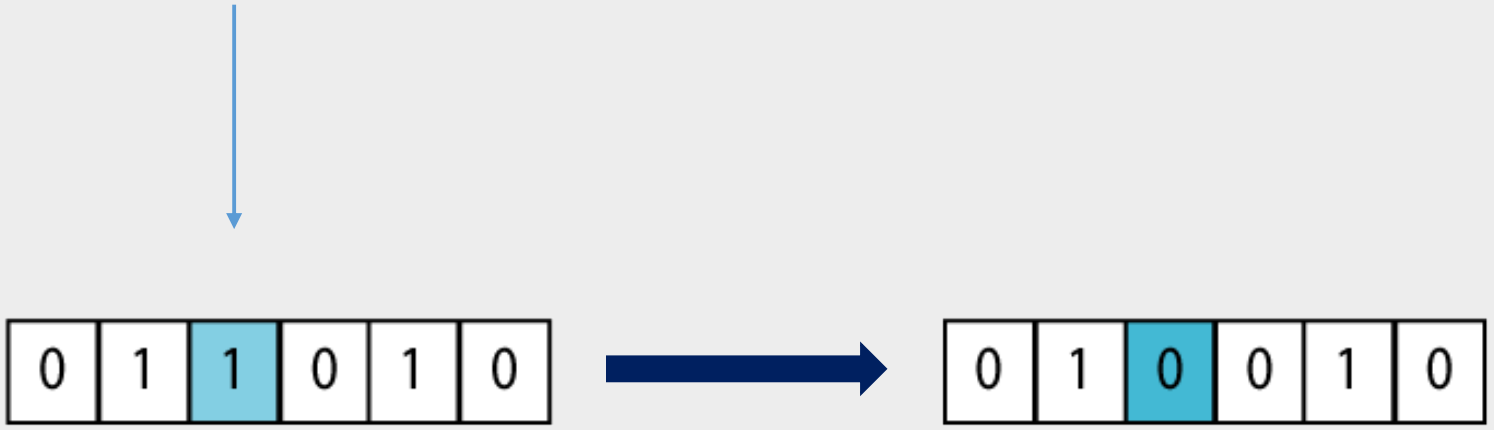
- **Chromosome :** le porteur de l'information génétique modélisé par un code génétique.
- **Gène** : partie d'un chromosome.
- **Individu** : le produit de l'activité des gènes, il est réduit à un chromosome.
- **Population**: ensemble d'individus.
- **Fitness**: la performance de l'individu.
- **Parents**: individus peuvent se reproduire et former un nouveau individu.


Genèse
Evaluation
Sélection
Croisements et mutations
Terminé?
Non
Oui
Résultats

- **Sélection par rang :** classer les individus selon leur fitness (performance) et choisir toujours les N meilleurs individus possédant un bon score.


P1
1 0 1 0 0 1
P2
1 1 0 0 1 1
F1
1 0 1 0 1 1
F2
1 1 0 0 0 1

Gène à muter



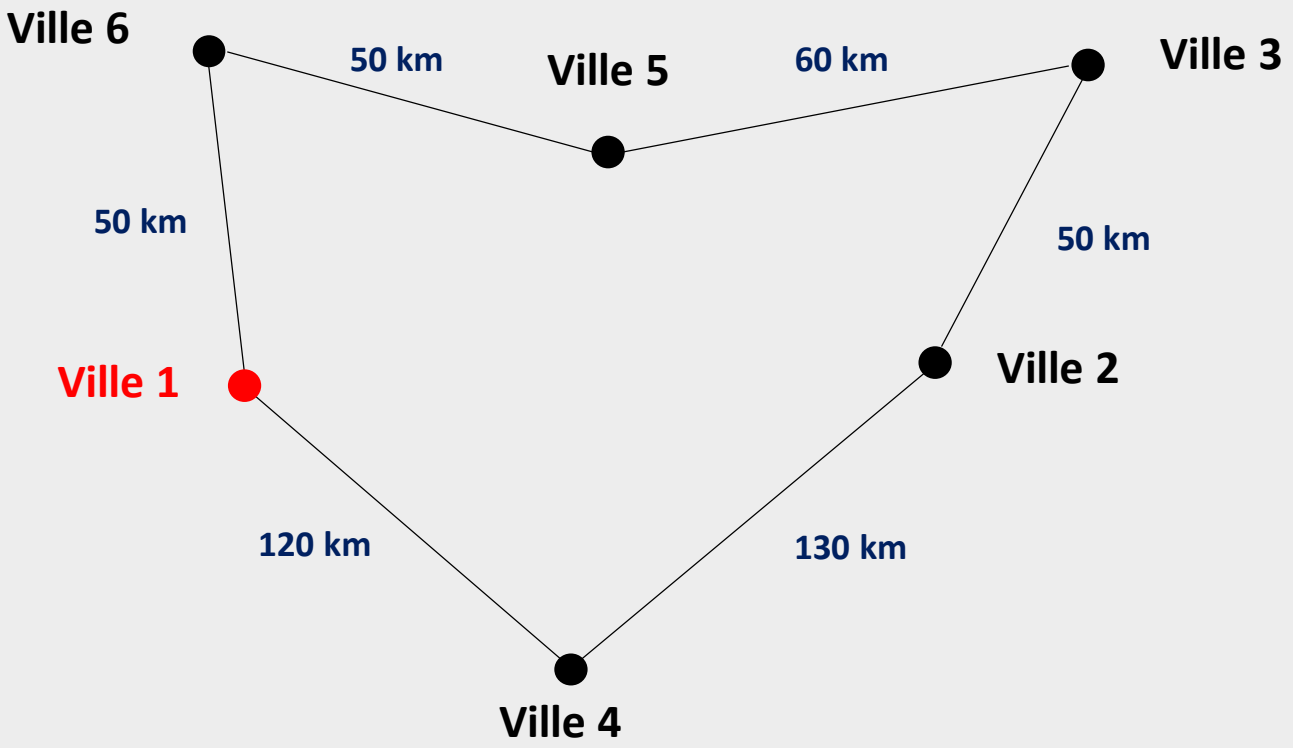
Chromosome initial **Chromosome muté**

- **Le problème du voyageur de commerce** peut être modélisé à l'aide d'un graphe.
- **Un sommet = une ville**.
- **Une arête = le passage d'une ville à une autre**.
- **Un poids = une distance.**
- **Problématique**? Trouver le plus court **cycle hamiltonien** passant par tous les sommets une unique fois.

- **Gène** = entier = ville.
- **Individu** = trajet.
- **Population** = ensemble de trajets.
- **Fitness** = la distance totale du trajet.
- **Croisement** = fusion de deux trajets.
- **Mutation** = changement d'un entier dans le trajet par un autre aléatoirement.

	Ville 1	Ville 2	Ville 3
Ville 1	0	$d_{1,2}$	$d_{1,3}$
Ville 2	$d_{2,1}$	0	$d_{2,3}$
Ville 3	$d_{3,1}$	$d_{3,2}$	0

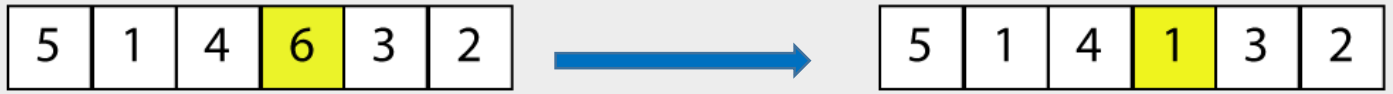
La distance entre la ville 2 et la ville 3



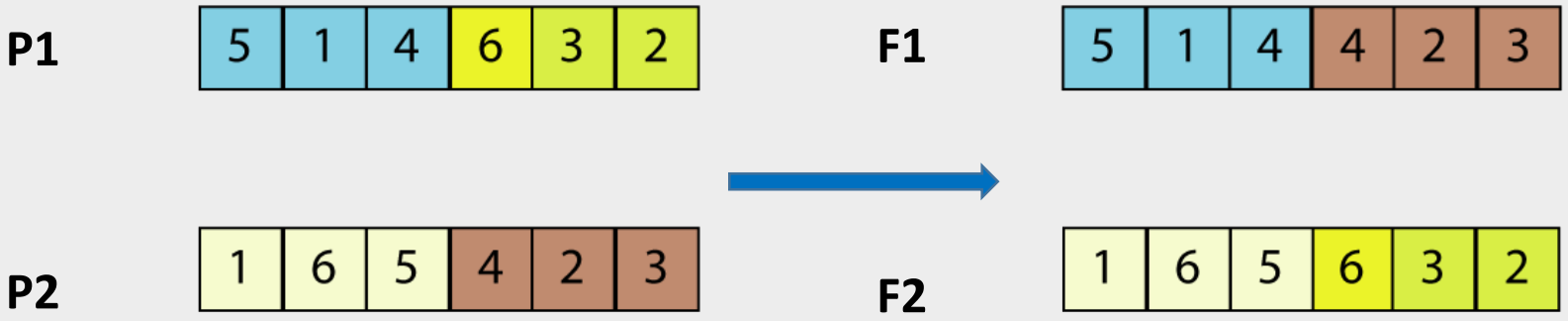
Trajet:

1	6	5	3	2	4	1

Fitness (distance totale) : 460km



Avant mutation **Après mutation**

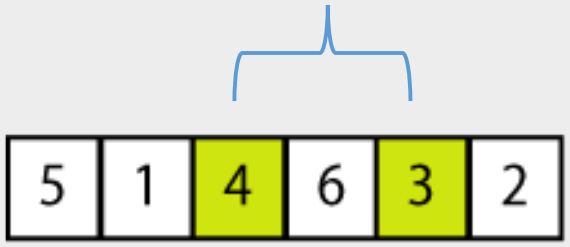


Avant croisement **Après croisement**

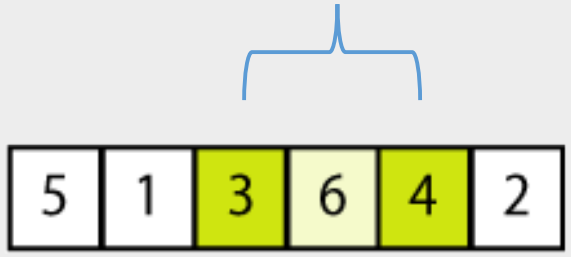
➢ Risque d'avoir un trajet qui ne passera pas par tous les villes.

- **Mutation d'inversion:**

Gènes choisies aléatoirement



Inversion des gènes



Avant mutation

Après mutation

- **Croisement proposé:**

P1

5	1	4	6	3	2

P2

1	6	5	4	2	3

- **Parcourir la 2éme partie de P2:**

4	2	3

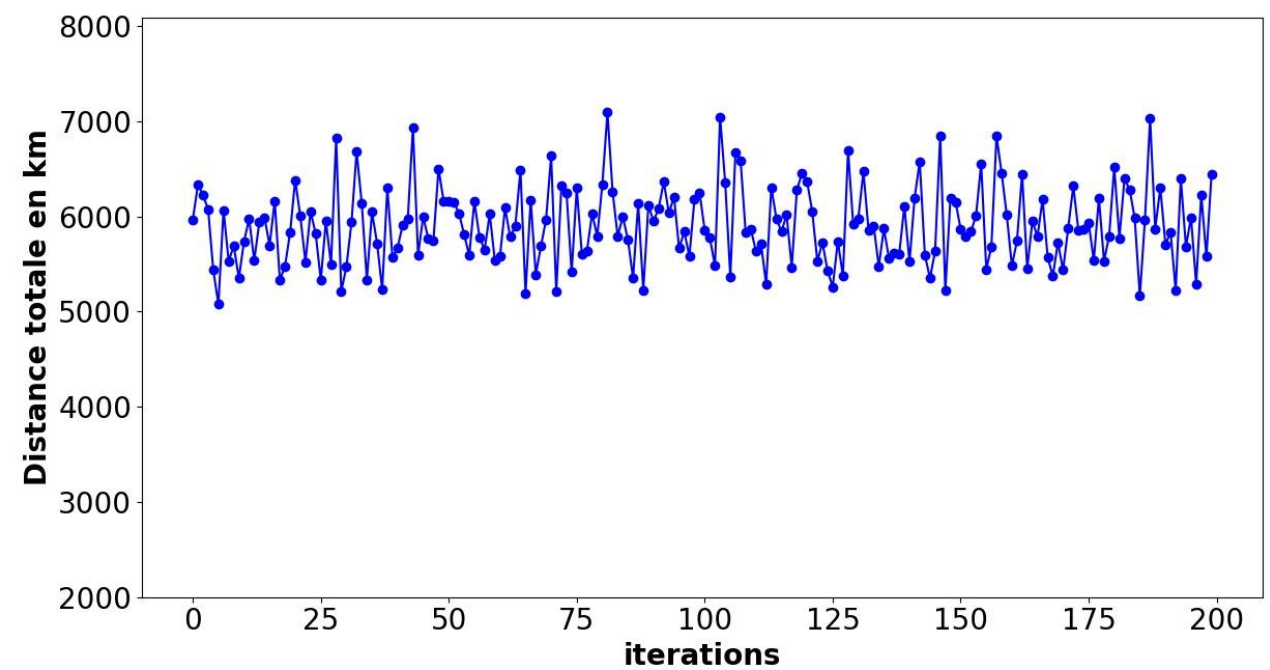
4 est déjà dans la partie 1 de P1, on va la remplacer par 6

2 et 3 ne sont pas dans la partie 1 de P1 + la partie ajoutée, on les conserve.

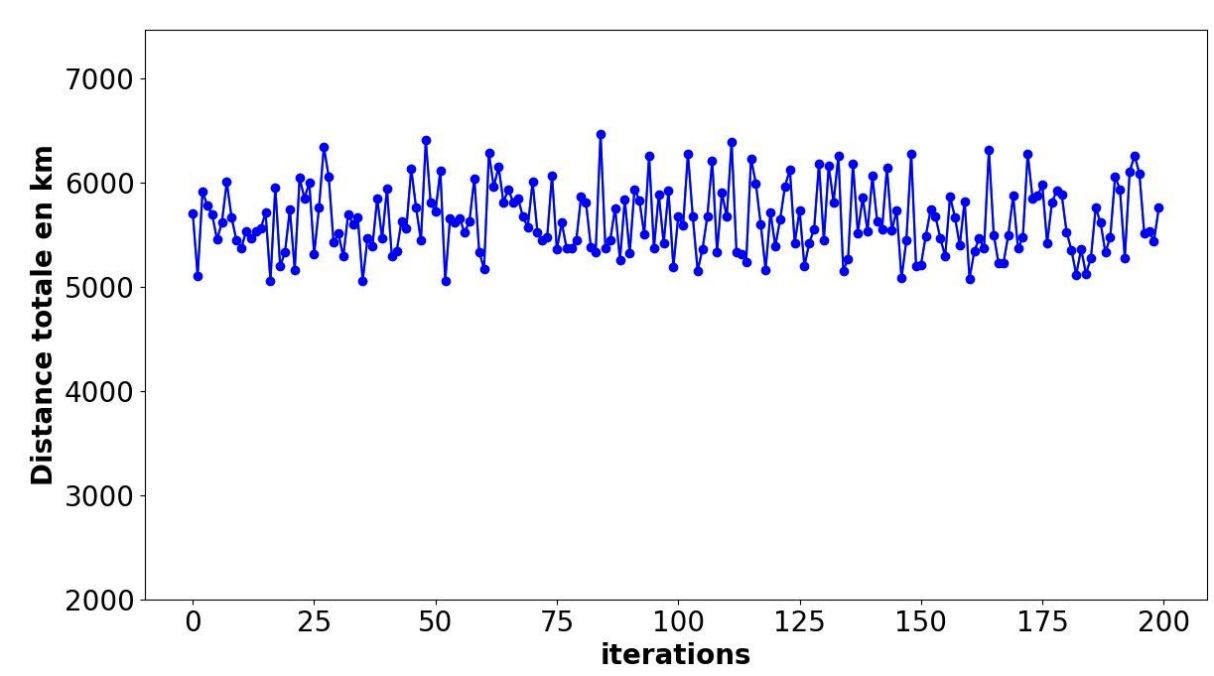
- **Résultat final:**

5	1	4	6	2	3

IV- Résultats et Conclusion: Résultats

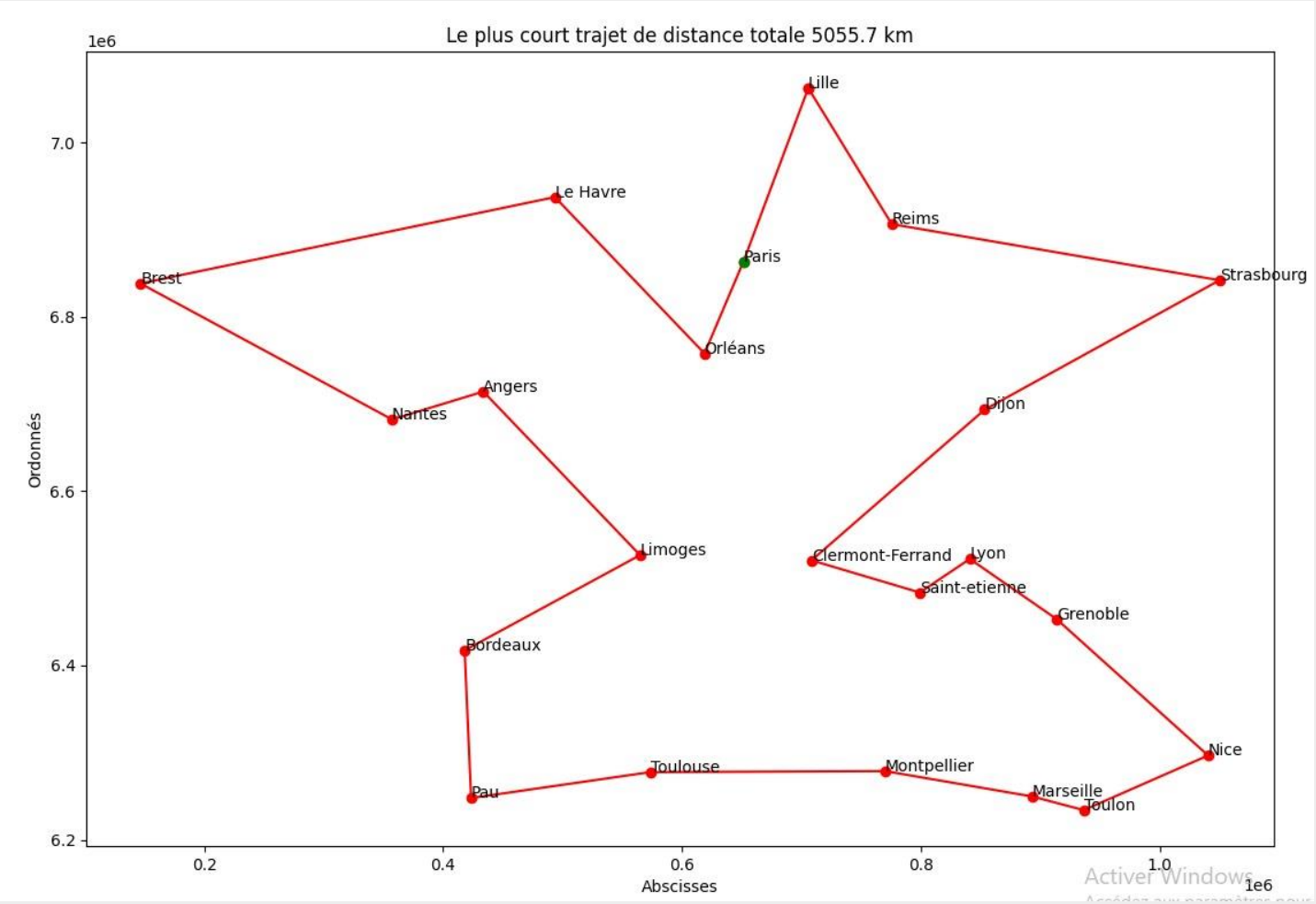


Par l'algorithme génétique

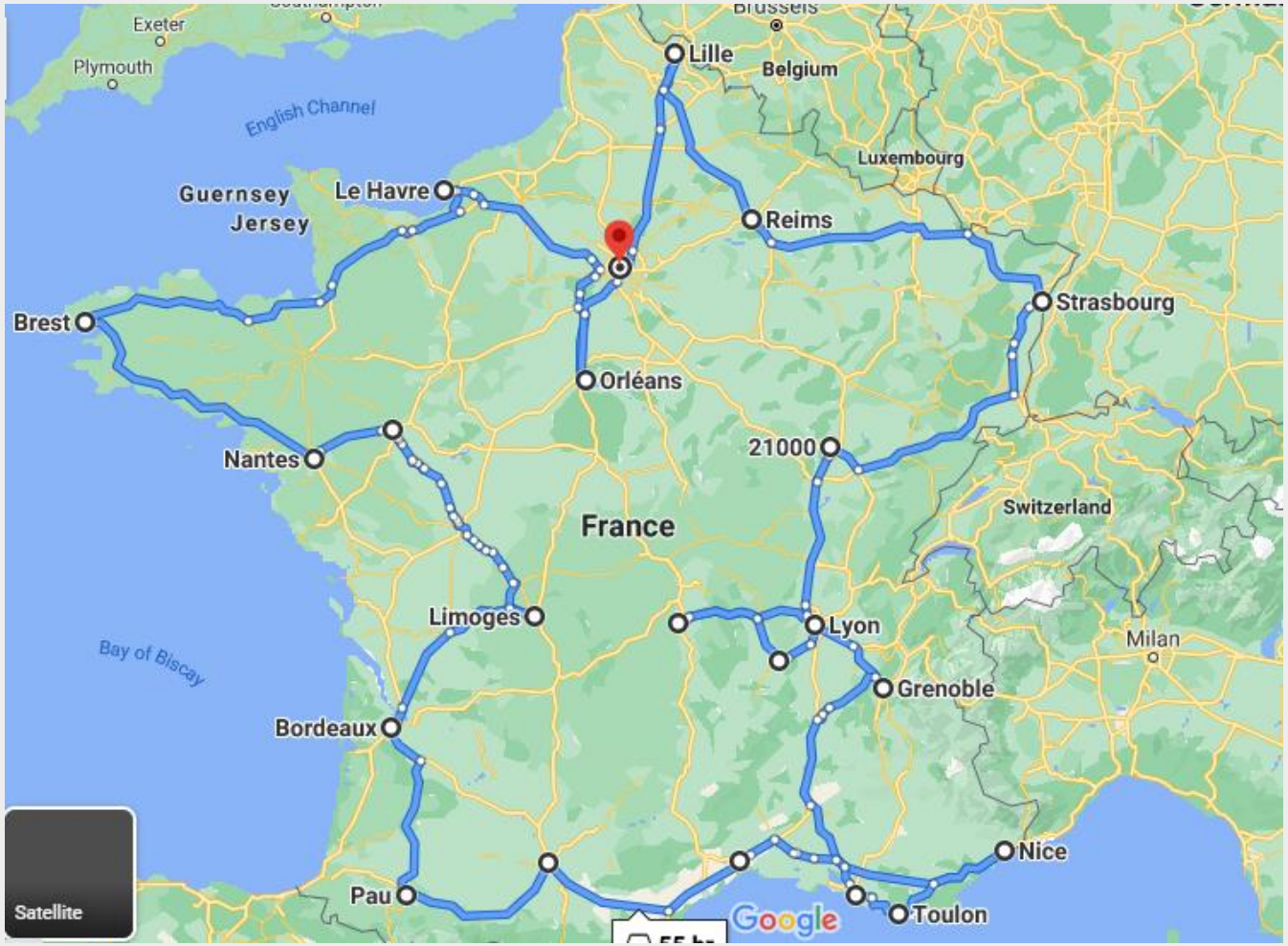


Par l'algorithme 2-opt

IV- Résultats et Conclusion: Résultats



IV- Résultats et Conclusion: Conclusion



(Google maps)

IV- Résultats et Conclusion: Annexe

Annexe

Déclaration des variables

```
import random
import matplotlib.pyplot as plt
from random import randint,sample
import time

#déclaration des variables
villes=["paris","nantes","nice","marseille","montpellier","monaco","lyon","bordeaux","grenoble",'lille']
Y=[706.582, 524.670, 142.625, 91.778, 121.136, 147.262, 364.824, 259.679, 295.439, 904.619]
X=[651.796,357.759,1039.617,894.098,770.262,1051.777,841.418,418.295,914.279,705.591]
M=[
   [0,385,894,772,749,915,492,546,585,215],
   [385,0,1143,986,825,1165,685,347,793,600],
   [894,1143,0,185,326,21,404,819,297,1068],
   [772,986,185,0,169,206,326,555,208,1005],
   [749,825,326,169,0,348,303,485,295,964],
   [915,1165,21,206,348,0,425,840,318,1089],
   [492,685,404,326,303,425,0,842,109,664],
   [546,347,819,555,485,840,842,0,675,792],
   [585,793,297,208,295,318,109,675,0,771],
   [215,600,1068,1005,964,1089,664,792,771,0],
]
```

Fonctions de base

```python
#fonctions de base
def create_ind():
    l=len(M)
    lis=random.sample(list(range(l)),k=l)
    return "".join([str(i) for i in lis])

def create_pop(n):
    return [create_ind() for i in range(n)]

def distance(a,b):
    a,b=int(a),int(b)
    return M[a][b]

def fitness(ind):
    indu=ind+ind[0]
    s=0
    for i in range(len(indu)-1):
        s+=distance(indu[i],indu[i+1])
    return s
```

Opérateurs génétiques

```python
#opérateurs génétiques
def selection(pop):
    sor=sorted([(fitness(pop[i]),i) for i in range(len(pop))])
    return [pop[p[1]] for p in sor[:5]]

def cross_over(pop):
    p1=pop[0]
    n=len(p1)//2
    p1_2=p1[n:]
    p=p1[:n]
    li=[p1]
    for i in range(1,len(pop)):
        p2_2=pop[i][n:]
        p1_1=p1[:n]
        for k in range(len(p2_2)):
            if p2_2[k] not in p1_1:
                p1_1+=p2_2[k]
            else:
                i=0
                while p1_2[i] in p1_1:
                    i=(i+1)%n
                p1_1+=p1_2[i]
        li.append(p1_1)
    return li

def mutation(pop):
    for i in range(len(pop)):
            ind=list(pop[i])
            ran1=randint(0,len(ind)-1)
            ran2=randint(0,len(ind)-1)
            ind[ran1],ind[ran2]=ind[ran2],ind[ran1]
            pop[i]="".join(ind)
```

Fonction génétique:

```
#fonction génétique
test=[]
temps=[]
def genetique():
    n=12
    l=[]
    pop=create_pop(n)
    m=(fitness(pop[0]),pop[0])
    for i in range(10000):
        pop=selection(pop)
        m1=(fitness(pop[0]),pop[0])
        if m1[0]<m[0]:
            m=m1
        pop=cross_over(pop)
        mutation(pop)
        pop+=create_pop(n-5)
    test.append(m)

for i in range(40):
    t1=time.time()
    genetique()
    t=time.time()-t1
    temps.append(t)
X2=list(range(40))
Y2=[i[0] for i in test]
```

Affichage des résultats:

```
#affichage des resultats
fig,ax1=plt.subplots()
ax1.set_xlabel('iterations')
ax1.set_ylabel('Distance en km')
ax1.plot(X2,Y2,'bo-')
ax1.set_ylim([0,max(Y2)+1000])
ax2=ax1.twinx()
ax2.plot(X2,temps,'go-')
ax2.set_ylim([0,max(temps)+1])
ax2.set_ylabel('Temps en s')

plt.show()
m=min(test)
print(m[0])
o=m[1][0]
d=m[1]+o
X1=[X[int(i)] for i in d]
Y1=[Y[int(i)] for i in d]
for i in range(len(X)):
    plt.annotate(villes[i],(X[i],Y[i]))
plt.plot(X1,Y1,'ro-')
plt.title("Le plus court trajet")
plt.ylabel("Y")
plt.xlabel("X")
plt.plot(X[0],Y[0],'go')
plt.show()
```