Compte Rendu

Travail Pratique 3

Algorithme génétique: Application au problème du voyageur

Métaheuristique en Optimisation

8INF852 – Groupe 1

Esmé James - JAME15539504
Wilfried Pouchous - POUW04069501

Sommaire

I - Organisation des fichiers

À la racine du dossier, contenant l'algorithme génétique, se trouve :

- Un fichier « *main.m »* qui correspond au point de départ du programme
- Un fichier « *GetUserInput.m »* pour la gestion des choix utilisateurs
- Un dossier ***AlgoGenetique*** qui contient plusieurs sous-dossiers dans lesquels sont rangés les différentes étapes de l'algorithme.

II - Fonctionnalités implémentées

1. Fonctionnement principal

Dans le main, un appel à la fonction *GetUserInput* permet d'afficher dans la console une liste de choix et permet de récupérer les entrées tapées par l'utilisateur. Un contrôle est ensuite effectué sur les choix pour s'assurer que le choix tapé correspond bien à une méthode implémentée.

S'il ne souhaite pas utiliser la fonction, l'utilisateur peut commenter l'appel à la fonction et rentrer les valeurs à la main dans le code dans la partie commentée en dessous de l'appel. Attention cependant, ces variables existent uniquement pour les tests des développeurs et ne sont pas soumises à des vérifications.

2. Étapes de l'algorithme

2.1. Initialisation

Dans l'initialisation nous générons tout d'abord des villes avec des coordonnées x et y aléatoires entre 0 et 100. Ces coordonnées sont stockées dans la matrice *M*.

Ensuite, nous générons une population initiale *Po* qui correspond à une permutation aléatoire des *n* villes. Ainsi, chaque individu de *Po* correspond à un ordre de visite des villes.

2.2. Gestion des redondances

Cette étape consiste à trouver et à enlever les individus qui sont redondant, c'est à dire les individus pour lesquels l'ordre des villes visitées est le même mais la ville de départ est différente.

La gestion des redondances se fait en trois étapes :

- **La recherche des redondances** dans la population initiale : cette étape nous renvoie les indices des couples d'individus redondant dans *Po*.
- **La suppression des doublons d'indices**. Si l'on obtient les indices redondants suivants : [1 2] et [2 1], on considère que ce n'est qu'une seule redondance et alors on doit en supprimer une.
- **La suppression des redondances dans la population**. Cela consiste à supprimer le deuxième indice de chaque couple d'individus redondants. Par exemple, si on a le coup d'indice [1 2], on supprimer l'élément de la population d'indice 2.

2.3. Evaluation

Une fois les redondances gérées, il faut évaluer la population. Pour cela on ajoute la ville de retour à la fin de chaque individu et on calcule la distance pour voyager entre ces villes et revenir.

A la fin de l'évaluation, on obtient une matrice *P* contenant les *n* villes visitées, la ville de retour et la distance qu'il faut pour parcourir ce cycle hamiltonien.

2.4. Sélection, croisements, mutations

Pour la sélection nous avons choisi une sélection par roulette (Roulette Wheel Selection). Les croisements et mutations suivants ont été implémentés :

Croisements :

- Croisement partiel
- Croisement basé sur la position
- Croisement basé sur le cycle

Mutations :

- Mutation d'échange
- Mutation d'insertion
- Mutation de déplacement
- Mutation d'inversion

III - Résultats

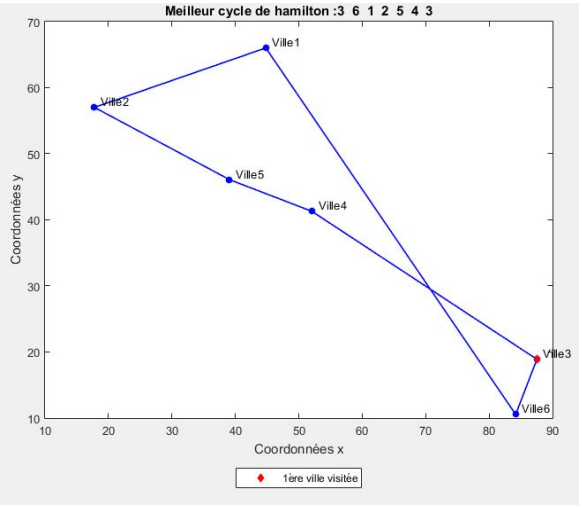
Nous testons les résultats en affichant le meilleur cycle hamiltonien trouvé sur un graphe.

1. Variation des paramètres

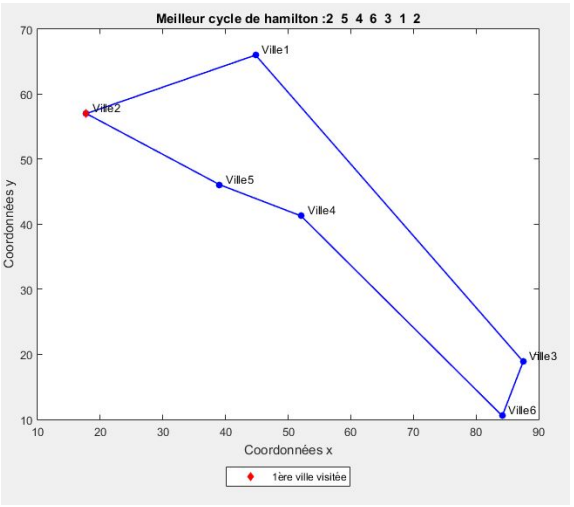
Nous testons les paramétrages avec la sélection RWS, le croisement par position et la mutation d'échange.

- **Premier paramétrage :**

Nombre de villes	*6*
Taille de la population	*20*
Gmax	*100*
Proba de croisement pc	*0.7*
Proba de mutation pm	*0.3*



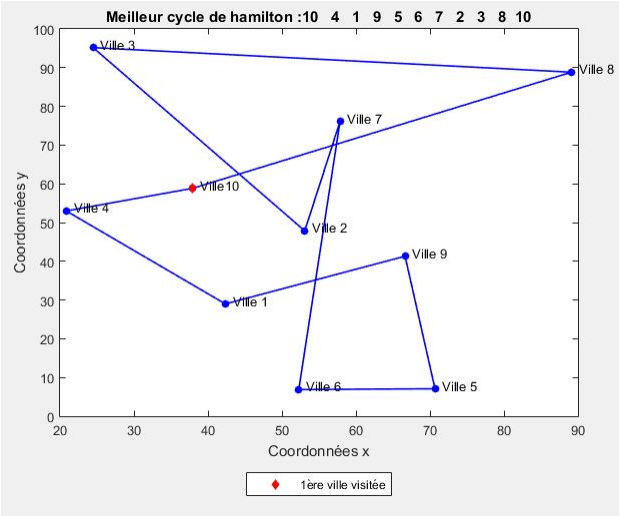
G = 1, Distance = 185.22



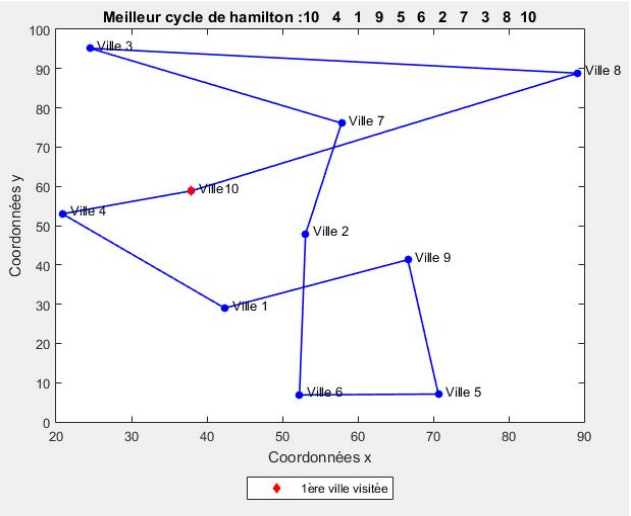
G = 5, Distance = 183.31

- **Deuxième paramétrage :**

Nombre de villes	*12*
Taille de la population	*10*
Gmax	*100*
Proba de croisement pc	*0.7*
Proba de mutation pm	*0.3*



G = 1, Distance = 408.1



G = 3, Distance = 362.779

2. Variation des opérateurs de croisement et mutation

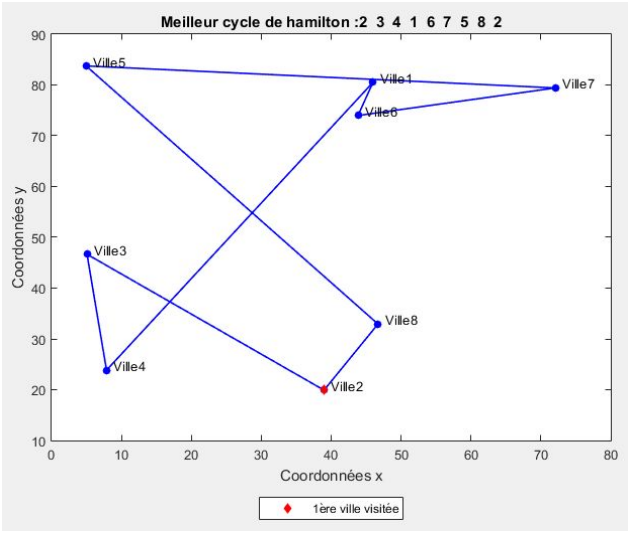
Après les tests sur les paramètres, nous choisissons les paramètres optimaux que nous allons utiliser pour les tests suivants :

Nombre de villes	*8*
Taille de la population	*10*
Gmax	*100*
Proba de croisement pc	*0.7*
Proba de mutation pm	*0.3*

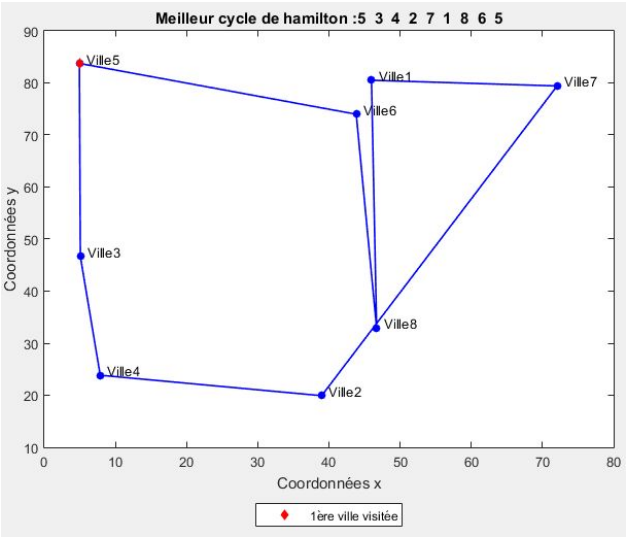
2.1. Variation des croisements

Dans ces tests nous faisons varier les croisements et gardons la même mutation : la mutation d'échange.

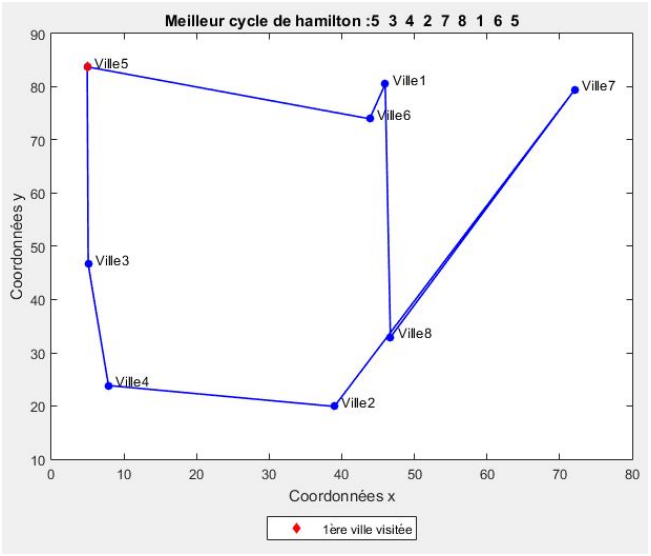
- **Croisement partiel :**



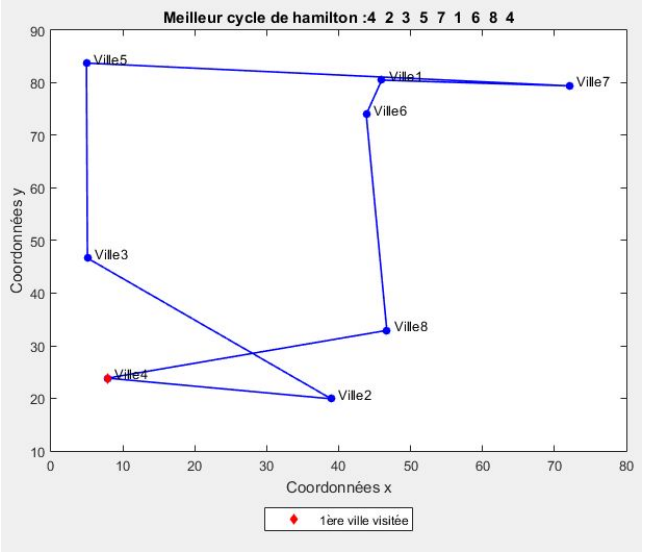
G = 1, Distance = 318.24



G = 10, Distance = 314.47



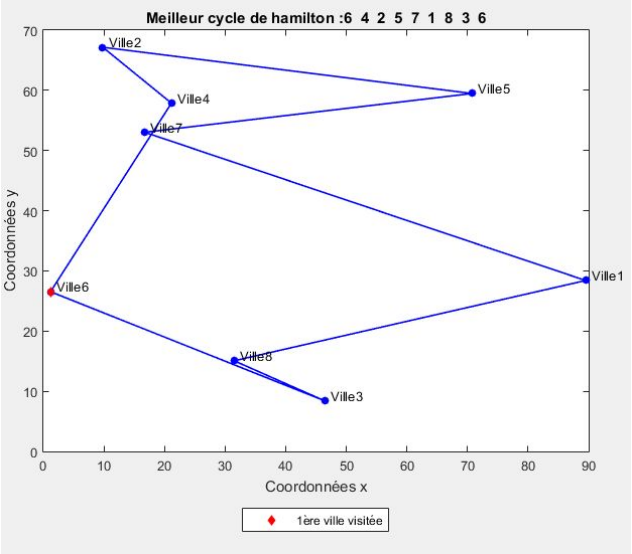
G = 13, Distance = 301.97



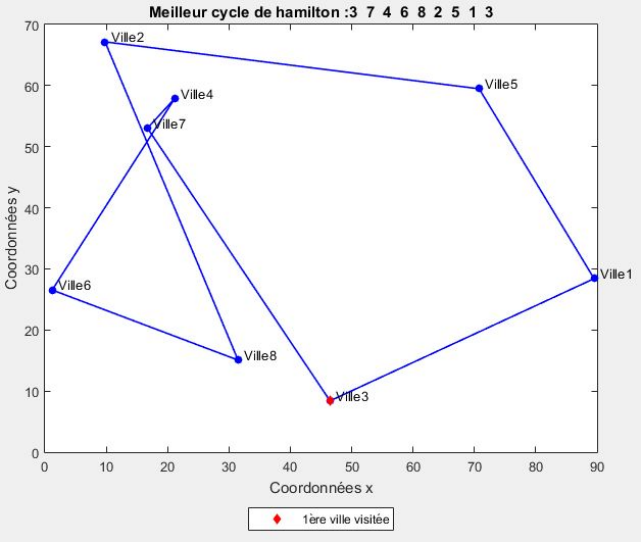
G = 20, Distance = 292.9

- **Croisement par position (utilisé dans les tests de paramétrages et mutations)**

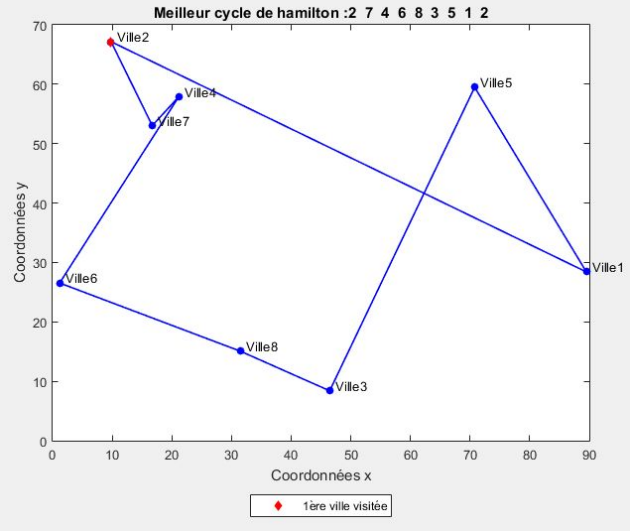
- **Croisement par cycle :**



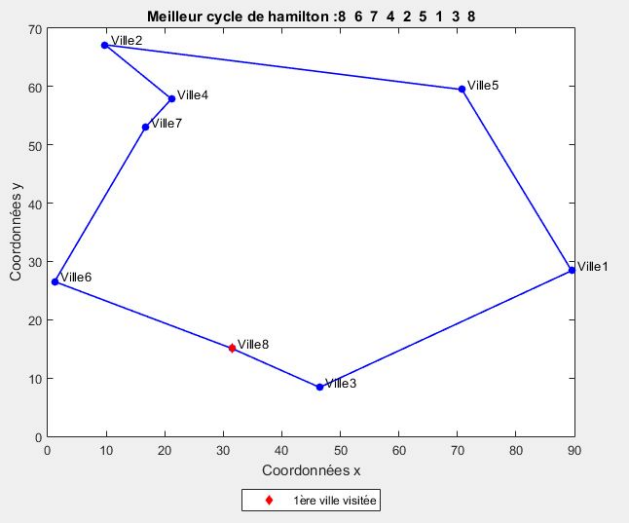
G = 1, Distance = 369.28



G = 3, Distance = 331,38



G = 14, Distance = 274.4



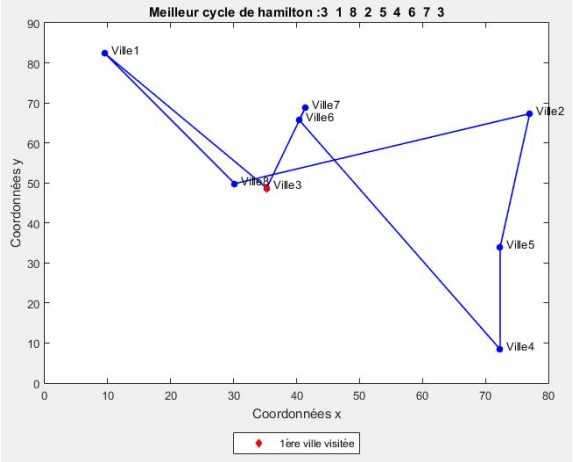
G = 23, Distance = 245.93

2.2. Variation des mutations

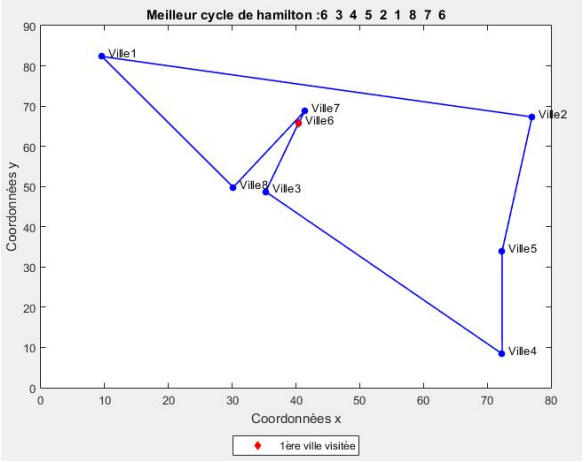
Dans ces tests nous faisons varier les mutations et gardons le même croisement : le croisement par position.

- **Mutation d'échange (utilisé dans les tests de paramétrage)**

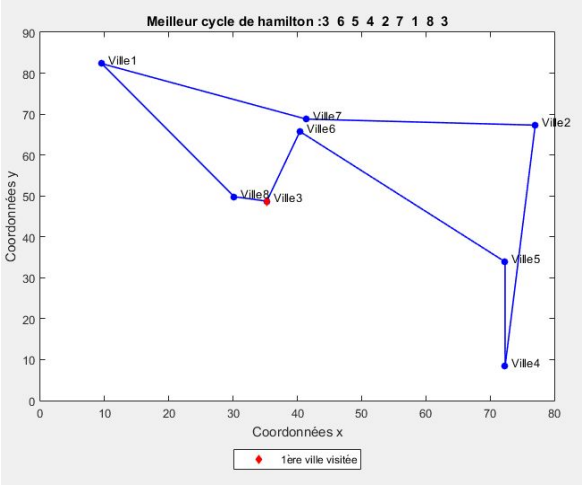
- **Mutation d'insertion :**



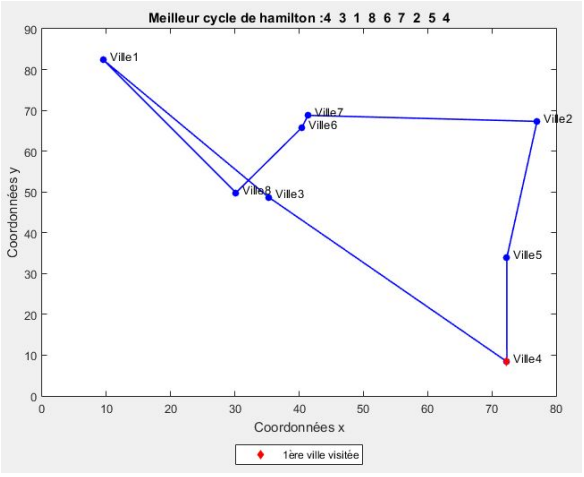
G = 1, Distance = 279.8



G = 3, Distance = 264.6

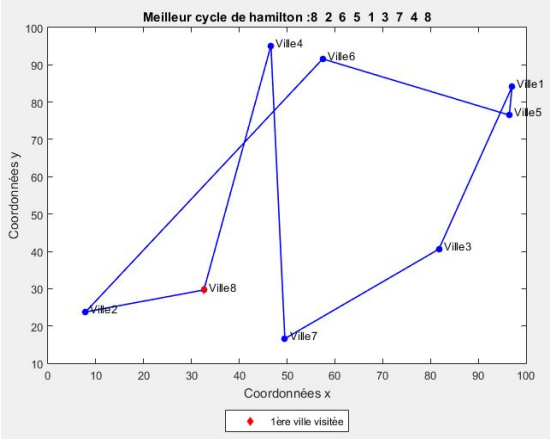


G = 13, Distance = 261.4

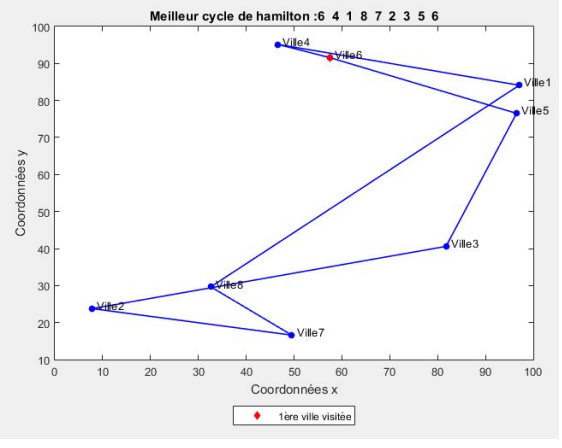


G = 25, Distance = 252.5

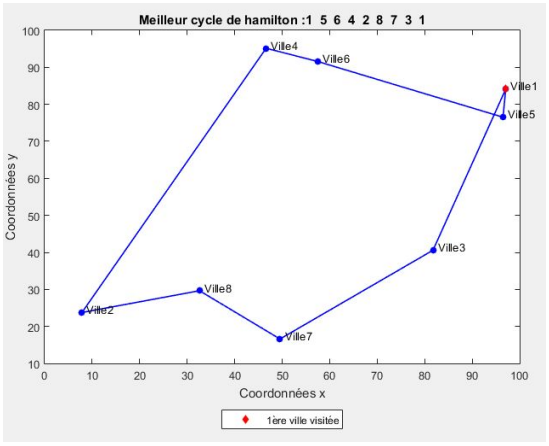
- **Mutation de déplacement :**



G = 1, Distance = 390.85

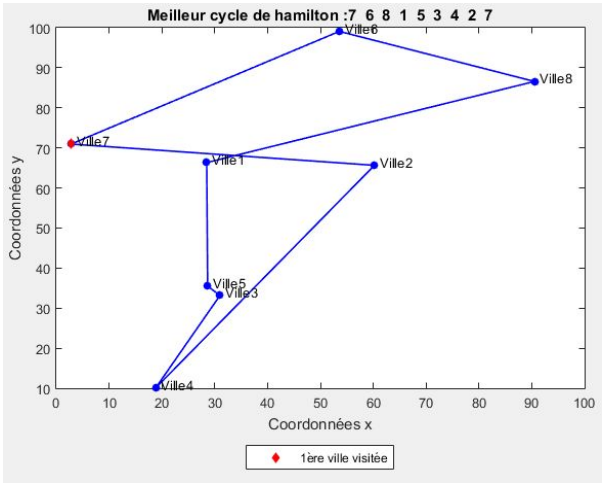


G = 2, Distance = 367.61

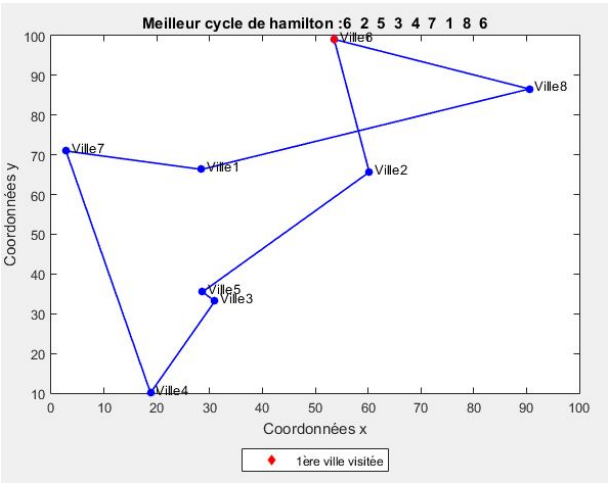


G = 6, Distance = 275.24

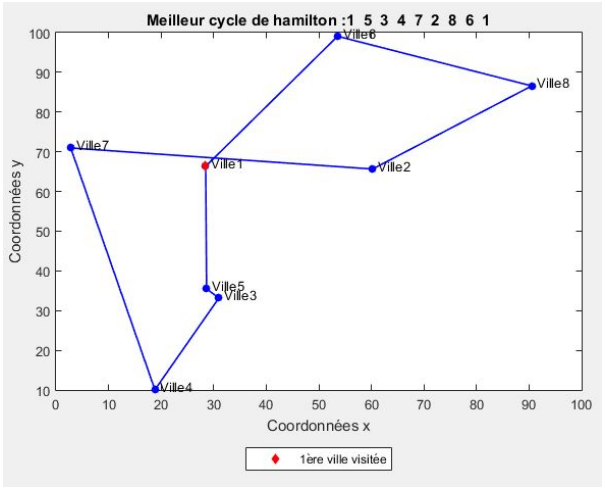
- **Mutation d'inversion :**



G = 1, Distance = 349.5



G = 4, Distance = 300.44



G = 7, Distance = 297.93

IV - Discussions

D’après les tests effectués, on peut en conclure plusieurs choses. Tout d’abord, il existe une corrélation entre le nombre de ville défini et la taille de la population. En effet, si le nombre de ville est bas mais que la taille de la population est grande, alors il y a de fortes chances de tomber dès le départ sur la solution optimale. Nous avons donc défini la taille de la population à 10 et le nombre de villes à 8 pour respecter cette corrélation et observer plus facilement l’impact des croisements et des mutations utilisés.

s

En ce qui concerne les croisements, nous avons jugé que les croisements par position et par cycle se valent et sont plus efficaces que le croisement partiel. En ce qui concerne les mutations, les mutations par déplacement et inversement sont plus efficaces que les deux autres. Nous pensons que cela vient du fait qu’elles peuvent déplacer un bloc de taille moyenne du cycle hamiltonien afin de changer l'enchaînement entre deux ou trois villes proches, ceci étant souvent le type de changement nécessaire pour obtenir la solution optimale.