Notice d'utilisation de la classe PVC_points

ZENAGUI MOHAMEDELAKEF

28 décembre 2025

1 Introduction

La classe `PVC_points` permet de résoudre le **Problème du Voyageur de Commerce (PVC)** sur un ensemble de points 2D. Elle fournit plusieurs algorithmes pour générer un cycle optimal ou approché :

- **PPP** : Point le plus proche.
- **OptPPP** : PPP optimisé avec l'algorithme 2-opt.
- **OptPrim** : Arbre couvrant minimal + parcours DFS pour générer un cycle.
- **HDS** : Heuristique de la demi-somme (branch-and-bound).

2 Installation

Pour utiliser la classe, installer les packages nécessaires :

```
pip install numpy matplotlib
```

3 Création d'un objet PVC_points

```
from src.PVC_points import PVC_points

pvc = PVC_points()
```

4 Chargement ou génération des points

4.1 Depuis une liste de points

```
points = [(0,0), (1,2), (3,1), (4,5)]
pvc.charger_de_liste(points)
```

4.2 Depuis un fichier texte

Le fichier doit contenir un point par ligne au format `(x, y)` :

```
(0,0)
(1,2)
(3,1)
(4,5)
```

```
pvc.charger_de_fichier("data/points.txt")
```

4.3 Générer des points aléatoires en mémoire

```
points_alea = PVC_points.generer_points(10)  # 10 points aleatoires
pvc.charger_de_liste(points_alea)
```

4.4 Générer un fichier de points aléatoires

```
PVC_points.generer_fichier(10, nom="points_alea")
# cree ./data/points_alea.txt
```

5 Algorithmes disponibles

Les méthodes utilisent un **point de départ aléatoire** défini lors du chargement des points.

5.1 PPP - Point le plus proche

```
cycle_ppp = pvc.PPP()
print("Cycle PPP :", cycle_ppp)
```

5.2 OptPPP - PPP optimisé 2-opt

```
cycle_opt = pvc.OptPPP()
print("Cycle OptPPP :", cycle_opt)
```

5.3 OptPrim - Arbre couvrant minimal + DFS

```
cycle_prim = pvc.OptPrim()
print("Cycle OptPrim :", cycle_prim)
```

5.4 HDS - Heuristique demi-somme

```
cycle_hds = pvc.HDS()
print("Cycle HDS :", cycle_hds)
```

6 Longueur d'un cycle

```
longueur = pvc.longueur(cycle_ppp)
print("Longueur du cycle PPP :", longueur)
```

7 Tracer un cycle

```
PVC_points.plot_cycle(cycle_ppp, titre="Cycle PPP")
PVC_points.plot_cycle(cycle_opt, titre="Cycle OptPPP")
PVC_points.plot_cycle(cycle_hds, titre="Cycle HDS")
```

8 Exemple complet

```python
from src.PVC_points import PVC_points

points = PVC_points.generer_points(15)
pvc = PVC_points()
pvc.charger_de_liste(points)

cycle_ppp = pvc.PPP()
cycle_opt = pvc.OptPPP()
cycle_hds = pvc.HDS()

print("PPP :", cycle_ppp)
print("OptPPP :", cycle_opt)
print("HDS :", cycle_hds)

print("Longueur PPP :", pvc.longueur(cycle_ppp))
print("Longueur OptPPP :", pvc.longueur(cycle_opt))
print("Longueur HDS :", pvc.longueur(cycle_hds))

PVC_points.plot_cycle(cycle_ppp, "Cycle PPP")
PVC_points.plot_cycle(cycle_opt, "Cycle OptPPP")
PVC_points.plot_cycle(cycle_hds, "Cycle HDS")
```

9 Conseils d'utilisation

- Le nombre minimal de points pour PPP, OptPPP et HDS est **3**.
- L'algorithme HDS peut être très lent pour **plus de 12 points**.
- La variable `seed_` garantit un point de départ fixe pour PPP et OptPPP si les points ne sont pas rechargés.
- Les cycles sont automatiquement fermés lors du tracé pour visualiser la tournée complète.