Résolution du problème du voyageur de commerce

Équipe 14

- Thomas CHUNG-HOW
- Sidrit VEJSELI
- Max ROGEL
- Hichem SANHAJI

Description

Le projet consiste à résoudre le problème du voyageur de commerce à l'aide d'un programme codé en C, et ce à l'aide de plusieurs méthodes différentes.

Le code principal se trouve dans `/src/main.c`.

Le code en Python pour les tests se trouve dans `/test`.

Compilation

```
make
```

Exécution

```
./bin/main
```

Paramètres

Paramètre	Description
`-f <fichier>`	Spécifie le chemin du fichier `.tsp` contenant les données du problème.
`-m <méthode>`	Définit la méthode de résolution à utiliser.
`-o <fichier>`	(Optionnel) Spécifie le fichier de sortie pour enregistrer les résultats.
`-c`	(Optionnel) Affiche la tournée canonique.
`-h`	Affiche l'aide.

Méthodes de résolution (pour `-m`)

Méthode	Description	Complexité
`bf`	Force brute : explore toutes les permutations possibles. Exact mais très lent.	$O(n!)$

Méthode	Description	Complexité
`nn`	Plus proche voisin (nearest neighbour) : construit un chemin en choisissant à chaque étape la ville la plus proche.	$O(n^2)$
`rw`	Marche aléatoire (random walk) : génère un chemin aléatoire.	$O(n)$
`2optnn`	2-opt avec initialisation par le plus proche voisin : améliore le chemin initial en décroisant toutes les arrêtes.	$O(n^3)$
`2optrw`	2-opt avec initialisation par la marche aléatoire : idem mais à partir d'un chemin aléatoire.	$O(n^3)$
`ga <nb_individus> <nb_générations> <taux_mutation>`	Algorithme génétique générique : populations d'individus évoluant avec sélection, croisement et mutation.	$O(n^2)$
`gadpx <nb_individus> <nb_générations> <taux_mutation>`	Algorithme génétique avec DPX.	$O(n^2)$
`all`	Toutes les méthodes sauf la force brute.	

Exemples d'exécution

Exemple 1 — Résolution de `att12.tsp` par la méthode de force brute :

```
./bin/main -f data/tsp/att12.tsp -c -m bf
```

Exemple 2 — Résolution de `gr431.tsp` par la méthode génétique générique :

```
./bin/main -f data/tsp/gr431.tsp -c -m ga 20 20 .3
```

Lancement des tests Python

Des tests unitaires sont fournis pour valider le bon fonctionnement du programme principal `main.c`.

Préparation de l'environnement de test

Avant de lancer les tests, il faut configurer l'environnement Python :

```
./test/setup_env.sh
source ./venv/bin/activate
```

Exécution des tests

Une fois l'environnement activé, lancez les tests avec :

```
python3 ./test/test_tsp_c.py
```

Désactivation de l'environnement virtuel

```
deactivate
```

Affichage interactif des données

Récupération des données pour l'affichage

```
make clean
make interactive
./bin/main -o ./test/donnees.txt ...
```

Affichage avec Python

```
python3 ./affichage_interactif/affichage_interactif.py
```