**Travail pratique** **Heuristiques 2-opt pour le TSP**

## Objectifs

L'objectif de ce travail pratique est de programmer les variantes « première amélioration » et « meilleure amélioration » de l'heuristique 2-opt pour le problème du voyageur de commerce symétrique. Les performances des deux approches seront ensuite évaluées sur les jeux de données fournis afin de proposer une stratégie de recherche d'une bonne tournée pour le problème du voyageur de commerce symétrique à partir des méthodes mises en oeuvre dans ce laboratoire et le précédent.

## Heuristique 2-opt

Rappelons que l'heuristique 2-opt est un algorithme d'amélioration basé sur des modifications locales d'un cycle hamiltonien représentant une tournée dans un problème de voyageur de commerce. Une modification locale, appelée « 2-échange », consiste à retirer deux arêtes du cycle actuel et à les remplacer par deux nouvelles arêtes de manière à conserver un cycle hamiltonien.

Plus spécifiquement un 2-échange $(i, j)$ consiste à retirer du cycle l'arête $\{i, i+1\}$ (reliant les villes aux positions $i$ et $i+1$ dans la tournée actuelle) ainsi que l'arête $\{j, j+1\}$ (reliant les villes aux positions $j$ et $j+1$) et à les remplacer par les arêtes $\{i, j\}$ et $\{i+1, j+1\}$. Un 2-échange est améliorant si la longueur du nouveau cycle est plus petite que celle du cycle avant échange.

La variante « première amélioration » de l'heuristique 2-opt consiste à rechercher et appliquer à chaque itération le premier 2-échange améliorant trouvé alors que la variante « meilleure amélioration » de l'heuristique 2-opt consiste à rechercher et appliquer à chaque itération le 2-échange entraînant la plus grande diminution de la longueur de la tournée actuelle. Dans les deux cas, l'algorithme s'arrête lorsqu'aucun 2-échange ne permet plus d'améliorer la solution courante. Les pseudocodes des deux variantes sont présentés en figures 1 et 2.

**Algorithme 1** Heuristique TwoOptFirstImprovement

**Données :** Un cycle hamiltonien représentant une tournée dans un problème de voyageur de commerce symétrique.

**Résultat :** Un nouveau cycle hamiltonien, correspondant à un minimum local de la structure de voisinage définie par les 2−échanges.

```
1: procédure TwoOptFirstImprovement
2:     répéter
3:         Rechercher le premier 2-échange améliorant pour la tournée actuelle et l'appliquer
4:     jusqu'à ce qu'il n'existe plus de 2-échanges améliorants
5:     Retourner la nouvelle tournée
6: fin procédure
```

**Algorithme 2** Heuristique TwoOptBestImprovement

**Données :** Un cycle hamiltonien représentant une tournée dans un problème de voyageur de commerce symétrique.

**Résultat :** Un nouveau cycle hamiltonien, correspondant à un minimum local de la structure de voisinage définie par les 2−échanges.

1: **procédure** TwoOptBestImprovement
2: **répéter**
3: Rechercher le meilleur 2-échange pour la tournée actuelle
4: Effectuer le 2-échange s'il est améliorant
5: **jusqu'à ce qu'**il n'existe plus de 2-échanges améliorants
6: Retourner la nouvelle tournée
7: **fin procédure**

### Travail à effectuer

Premièrement vous devez compléter l'implémentation des classes `TwoOptFirstImprovement` et `TwoOptBestImprovement`.

Ceci fait vous mettrez en place un programme principal permettant d'évaluer la qualité et les performances (longueur des tournées et temps de calcul) des deux variantes en fonction de la tournée de départ fournie. Plus précisément, pour chacun des six jeux de données fournis et pour les deux variantes de l'heuristique 2-opt,

- ▷ Vous évaluerez vos algorithmes en partant d'une tournée initiale aléatoire (fournie par la classe `RandomTour`) ou d'une tournée obtenue à l'aide des deux heuristiques de plus proches voisins développées lors du laboratoire précédent
- ▷ Pour chacune des trois possibilités vous calculerez des performances moyennes en calculant 10 tournées à chaque fois.
- ▷ Pour les tournées initiales aléatoires vous utiliserez à chaque fois la graine (*seed*) `0x134B3BD` pour la première tournée.
- ▷ Pour les tournées initiales obtenues à l'aide des heuristiques de plus proches voisins vous utiliserez comme ville de départ les 10 premières villes de la tournée aléatoire obtenue avec la graine `0x134B3BD`.

Finalement, en vous appuyant sur les résultats obtenus, vous proposerez et défendrez l'approche de résolution (choix de la tournée initiale et de la variante 2-opt) qui vous semble la plus pertinente pour calculer efficacement une solution approchée pour un problème du voyageur de commerce symétrique à partir des méthodes développées dans ce laboratoire et le précédent.

### Modalités et délais

- ▷ Le travail de programmation est à effectuer par groupe de deux, en Java, version 17.
- ▷ L'archive contenant les sources du projet est disponible sur le site Cyberlearn du cours tout comme les jeux de données à analyser.
- ▷ Vous devez rendre une archive (au format `zip`) contenant toutes les sources de votre projet complété ainsi qu'une note (au format `txt`, `pdf` ou `md`) contenant la sortie console de vos résultats (pas de copie d'écran !) et votre stratégie d'optimisation argumentée.
- ▷ Vous devez rendre votre travail sur Cyberlearn au plus tard le **dimanche 3 décembre 2023** (avant minuit).