Rapport de TP : Optimisation et Application

LALAOUI Rayan KERMADJ Zineddine

March 27, 2025

Contents

1 Introduction

Ce rapport présente les travaux réalisés dans le cadre du TP sur l'optimisation et les graphes. Le projet est divisé en deux parties :

- La **partie 1** se concentre sur la représentation des graphes, le coloriage de graphes et l'algorithme de Welsh-Powell.
- La **partie 2** explore les graphes pondérés et les heuristiques pour résoudre le problème du voyageur de commerce (TSP).

Nous détaillons dans ce rapport nos choix d'implémentation, les algorithmes utilisés et les résultats obtenus.

2 Partie 1 : Représentation des graphes et coloriage

2.1 Représentation des graphes

Nous avons choisi de représenter les graphes à l'aide d'une matrice d'adjacence. Ce choix est motivé par les raisons suivantes :

- **Efficacité des opérations** : Les opérations telles que l'ajout, la suppression d'arêtes et la recherche des voisins d'un sommet sont rapides et simples à implémenter.
- **Compatibilité avec les algorithmes de coloriage** : La matrice d'adjacence facilite l'implémentation des algorithmes de coloriage et la vérification de leur validité.
- **Gestion mémoire adaptée** : Bien que les matrices d'adjacence soient moins efficaces pour les graphes creux, elles sont adaptées à nos tests sur des graphes denses.

2.1.1 Structure de la classe `Graph`

La classe `Graph` inclut les fonctionnalités suivantes :

- `generate_random_graph` : Génère un graphe connexe aléatoire avec un nombre donné de sommets et d'arêtes.
- `add_edge` et `remove_edge` : Permettent d'ajouter ou de supprimer des arêtes.
- `get_neighbors` : Retourne les voisins d'un sommet donné.
- `display` : Affiche la matrice d'adjacence et visualise le graphe avec des couleurs en utilisant la bibliothèque Graphviz.

2.2 Coloriage de graphes

2.2.1 Algorithme de Welsh-Powell

L'algorithme de Welsh-Powell colore les sommets d'un graphe en triant les sommets par degré décroissant. Les sommets sont ensuite colorés de manière à minimiser le nombre de couleurs utilisées.

2.2.2 Coloriage partiel avec inversion

Nous avons adapté l'algorithme de Welsh-Powell pour effectuer un coloriage partiel avec un nombre limité de couleurs (`k`). Les sommets qui ne peuvent pas être colorés avec `k` couleurs restent non coloriés (indiqués par `-1`). Une variante de cet algorithme inverse l'ordre de tri des sommets (par degré croissant au lieu de décroissant). Cette inversion permet d'explorer une convention différente et d'évaluer son impact sur les résultats.

2.2.3 Résultats expérimentaux

Nous avons testé le coloriage partiel avec et sans inversion sur des graphes de 3 à 6 sommets. Les résultats montrent que l'inversion est meilleure environ 90.88% du temps en moyenne sur des graphes connexes générés aléatoirement.

3 Partie 2 : Graphes pondérés et heuristiques pour le TSP

3.1 Représentation des graphes pondérés

Pour représenter les graphes pondérés, nous avons modifié la classe `Graph` pour inclure des poids sur les arêtes. La matrice d'adjacence a été adaptée pour stocker les poids des arêtes au lieu de simples indicateurs de connectivité.

3.1.1 Structure de la classe `Graph`

La classe `Graph` inclut les fonctionnalités suivantes :

- `add_edge(u, v, w)` : Ajoute une arête pondérée entre les sommets u et v avec un poids w.
- `remove_edge(u, v)` : Supprime l'arête entre les sommets u et v.
- `has_hamiltonian_cycle()` : Vérifie si le graphe contient un cycle hamiltonien.
- `display()` : Affiche la matrice d'adjacence.

3.2 Heuristiques pour le TSP

Trois heuristiques ont été implémentées pour résoudre le problème du TSP :

- **Plus proche voisin (Nearest Neighbor)** : Cette heuristique construit un chemin en visitant à chaque étape la ville la plus proche qui n'a pas encore été visitée.
- **Recuit simulé (Simulated Annealing)** : Cette méthode explore des solutions voisines en acceptant parfois des solutions moins bonnes pour éviter les minima locaux.
- **2-opt** : Cette méthode améliore une solution initiale en échangeant deux arêtes pour réduire la distance totale.

3.2.1 Résultats expérimentaux

Nous avons appliqué les heuristiques sur un ensemble de points avec une matrice de distances calculée à partir des coordonnées des points. Voici un exemple de résultats :

- **Plus proche voisin** : Distance totale = 25.0
- **Recuit simulé** : Distance totale = 22.5
- **2-opt** : Distance totale = 20.0

3.3 Tests et validation

Nous avons écrit des tests unitaires pour valider nos implémentations. Ces tests incluent :

- Vérification de l'ajout et de la suppression d'arêtes dans un graphe pondéré.
- Vérification de l'existence d'un cycle hamiltonien.
- Validation des heuristiques pour le TSP sur des matrices de distances prédéfinies.

4 Conclusion

Ce projet nous a permis d'explorer des concepts avancés en théorie des graphes et en optimisation. La représentation par matrice d'adjacence s'est avérée efficace pour nos besoins. Les algorithmes de coloriage et les heuristiques pour le TSP ont montré des résultats intéressants, avec des performances variables selon les cas. Les tests unitaires ont validé la robustesse de nos implémentations.