Projet IPF S3 : Résolution approchée du problème de voyageur de commerce euclidien en Ocaml

Thibaut Milhaud

Janvier 2019

Table des matières

1 Introduction

L'objectif de ce projet est de coder un exécutable capable de trouver une solution correcte au problème du voyageur de commerce euclidien en 2D. Pour se faire nous allons utiliser des heuristiques de construction et d'optimisation de chemin. Le projet était décomposé en 2 parties. Dans la première chaque ville était reliée à toutes les autres tandis que dans la seconde certaines routes sont fermées. Toutefois, n'ayant pas réussi à réaliser un parser pour la seconde partie, je nais pas pu coder d'exécutable répondant au problème de la seconde partie.

2 Choix des structures

La difficulté dans le choix des structures a été comme souvent de trouver un compromis satisfaisant entre rapidité d'accès aux données et espace mémoire consommé. Dans un premier temps, j'ai décidé de d'attribuer à chaque ville un indice afin de travailler avec des entiers au lieu de chaînes de caractères pour augmenter la vitesse d'exécution. Ainsi pour éviter d'avoir une matrice d'adjacence très volumineuse j'ai choisi de représenter le problème dans deux structures de données complémentaires : les cartes et les chemins.

2.1 Carte

Description Une carte va contenir toutes les informations du début de problème et associer à chaque indice de ville les informations relatives à cette ville : le nom, les coordonnées, un booléen indiquant si oui ou non la ville est insérée dans le chemin associé à la carte.

Implémentation J'ai choisi de représenter les carte par des Map dont les clefs sont des entiers (villes) et les valeurs, un type enregistrement contenant les infos relatives à la ville. Dans la seconde partie, le type enregistrement est plus complexe et comporte un champ contenant une map des villes inaccessibles depuis la ville actuelle. Ainsi, si on essaie de calculer la distance entre 2 villes a et b, si b est dans la map d'exclusion de a on renvoie -1 pour signifier que les villes ne sont pas connectées. Malheureusement, je n'ai pas réussi à finir le parser pour la seconde partie du projet dans les temps, je ne sais donc pas si elle fonctionne.

2.2 Chemin

Description Un chemin est la structure qui va contenir notre solution courante tout au long de la résolution du problème. Dans un chemin chaque ville est associée a celle qui la précède et celle qui la suit dans le chemin.

Implémentation La encore, j'ai utilisé des Map ordonnées par des entiers et contenant des couples d'entiers (ville précédente, ville suivante).

3 Enveloppe convexe

QuickHull Pour calculer l'enveloppe convexe, j'ai choisi d'utiliser l'algorithme QuickHull. L'odée de cet algorithme est de diviser le problème en plusieurs sous problèmes plus simples. On commence donc par se donner deux points P, Q qui appartiennent à l'enveloppe convexe (celui d'abscisse minimale et celui d'abscisse maximale par exemple). On va maintenant ajouter à l'enveloppe H, le point le plus à gauche de PQ (droite vectorielle orientée) et on va répéter le même processus récursivement sur les droites PH et HQ. Puis lorsqu'on aura toute la partie gauche de l'enveloppe convexe, on appliquera le même procédé à QP.

Complexité La complexité de cet algorithme est en $O(n^2)$ dans le pire des cas d'après les ressources que j'ai trouvé en ligne.

4 Efficacité de la solution proposée

4.1 Points forts

Le point le plus satisfaisant de ce programme est la vitesse d'exécution, en effet, il propose une solution à un problème de 1000 ville en un temps de l'ordre d'une seconde avec les options ("FARTHEST" et "RANDOM", je n'ai pas réussi à comprendre pourquoi mon implémentation de "NEAREST" est moins efficace) et le temps est quasiment instantané pour des problème de moins de 500 villes.

4.2 Limites

Le principal problème que je verrais dans ce programme est que je n'ai pas de distance de référence à quoi comparer mes résultats quantitativement. De plus même si les variations de distance entre "NEAREST", "FARTHEST" et "RANDOM" très faibles, l'option "RANDOM" donne presque systématiquement des meilleurs résultats en temps et en distance que les deux autres ce qui est un peu décevant (et inquiétant pour mon inplémentation).

5 Conclusion

Pour conclure, je dirais que je suis assez satisfait du travail réalisé pendant ce projet même si le fait de m'y être pris tard m'a empêché de concrétiser la seconde partie. Une documentation sommaire se trouve dans le `README.md` expliquant l'organisation des fichiers ainsi que les instructions pour compiler et exécuter ce projet.