Projet PP1 2014: Problème du voyageur de commerce

February 10, 2014

1 Présentation

Dans le cadre de ce projet, on s'intéresse au problème classique du *voyageur de commerce*. Étant donnés un ensemble $V = \{1, 2, \ldots, n\}$ et w une fonction (que nous appellerons *fonction de distance*) allant des couples d'éléments de V dans $\mathbb{R}$. Etant donnée une permutation σ[1] sur les éléments de V, on appelle *longueur* de σ la somme

$$w(\sigma(n), \sigma(1)) + \sum_{i=1}^{n-1} w(\sigma(i), \sigma(i+1)).$$

Sans perte de généralité on peut se restreindre aux permutations σ telles que $\sigma(1) = 1$.

Le problème du voyageur de commerce est de trouver la permutation σ ayant la plus petite longueur possible.

Dans le cadre de ce projet, on se place dans le cas où la fonction de distance w est une métrique. C'est-à-dire que pour tout i, j, k dans V,

1. $w(i, j) = w(j, i) \geq 0$,
2. $w(i, j) = 0$ si et seulement si $i = j$,
3. $w(i, j) \leq w(i, k) + w(k, j)$.

Lorsque que w est un métrique, on parle alors du problème de voyageur de commerce métrique (metric Travel Saleman Problem).

2 Travail demandé

Le but de ce projet est de fournir un programme fonctionnant en ligne de commande permettant de calculer des solutions (pas forcément optimales) au problème du voyageur de commerce métrique.

Besoins fonctionnels:

[1] une permutation de taille n est une application bijective de V dans V. Une permutation permet de définir un ordre quelconque sur les éléments de V.

1. développer une heuristique simple (la plus simple étant “nearest neighbour”),
2. développer un algorithme d’approximation (le plus simple étant basé sur le “minimum spanning tree”),
3. développer un algorithme exact par recherche exhaustive,
4. développer un algorithme exact utilisant la technique de “branch and bound”.
5. pouvoir charger des instances de problèmes au format `TSPLIB` [tsp] dans le cas où les instances sont codées sous la forme de matrices complètes: `EDGE_WEIGHT_FORMAT: FULL_MATRIX`. Les solutions devrons également être retournées en respectant le format `TSPLIB`.
6. optionnel: fournir une visualisation graphique de la route calculée (en utilisant la section `DISPLAY_DATA_SECTION`). Vous pourrez pour cela utiliser par exemple la bibliothèque MLV [mlv] ou la bibliothèque SDL [sdl].

Besoins non-fonctionnels:

1. Votre projet devra se compiler et s’installer facilement.
2. Votre code devra être correctement documenté. Chaque membre du groupe devra documenter complètement au moins une fonction.
3. Les parties centrales de votre code devrons être testées. Chaque membre du groupe devra produire au moins un test. L’exécution des tests devra être automatisée.

2.1 Organisation

Le projet est travaillé et étudié en trinôme mais la notation est individuelle. L’enseignant évaluera la contribution de chacun des membres du trinôme au travail commun. L’enseignant se réservera la possibilité de modifier la composition de chacun des trinômes. Afin de permettre un travail profitable, il est conseillé de ne pas créer de groupes avec des niveaux trop différents.

Le projet doit être réalisé à l’aide des différents outils disponibles dans votre environnement de développement, à savoir, profiling de code, debugging, documentation automatique de code, gestion de versions (dans votre répertoire d’accueil ou sur le site savane du CREMI: `https://services.emi.u-bordeaux1.fr/projet/savane/`.

Deux jours avant chaque TD-PP1, une brève synthèse (1/2 page environ) des travaux effectués depuis la séance précédente sera transmise (mail ou papier) à l’enseignant. On précisera les points que l’on désire plus particulièrement aborder avec l’enseignant chargé de suivre le projet. Chaque document contiendra la date et les noms du trinôme.

2.2 Soutenance

La présentation finale du projet se fera en salle de TD. Cette présentation se fera autour d'une démonstration et d'un compte-rendu écrit présentant :

- les objectifs atteints et ceux qui ne le sont pas;
- l'exposé d'un problème technique rencontré et sa résolution;
- quelques exemples montrant que vous avez su éviter des redondances dans votre code.

2.3 Évaluation

La note est composée de 2 éléments. Ces deux éléments sont cumulatifs: pour obtenir la note finale, on ajoute les différents éléments.

- Contrôle continu: une note est attribuée à l'issue de chaque séance de TP. Cette note sera établie à partir des critères suivants: respect des consignes; évaluation de la synthèse et du code fourni; travail accompli depuis la séance précédente; évolution du projet. La note de contrôle continu sera la moyenne de ces notes.
- Projet: une note sur 200 est attribuée à l'issue de la soutenance. Cette note sera établie à partir des critères suivants:
  - Évaluation de la présentation orale et de la démonstration de l'exécution (20 pts)
  - Réponses aux questions posées sur le projet (20 pts)
  - Évaluation du rapport sur le projet (40 pts)
  - Qualité du code fourni (80 pts) (non duplication de code, concision et lisibilité du code, modularité et qualité du découpage, généricité, réponse aux spécifications, extensions éventuelles).
  - Utilisation adéquate des outils vus en EDD (40 pts).

2.4 Présence

La présence aux TD, à la soutenance et au devoir surveillé terminal est obligatoire. Toute absence injustifiée donnera la note 0 pour l'épreuve concernée. En cas de circonstances exceptionnelles, contacter son enseignant de TD par mail au plus tard le jour de la séance de TD concernée ou le jour du devoir surveillé.

References

[mlv] Bibliothèque graphique mlv. `http://www-igm.univ-mlv.fr/~boussica/mlv/index.html`.

[sdl] Simple directmedia layer. http://www.libsdl.org/.

[tsp] Tsplib 95. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/DOC.PS.