TME1. Arbre couvrant.

BM Bui-Xuan

Problème de l'arbre couvrant dans un plan Euclidien : Etant donné un ensemble de points dans le plan, le problème de l'arbre couvrant minimal consiste à calculer un arbre (au sens de la théorie des graphes) passant par tous ces points, tel que la longueur totale des arêtes de l'arbre est la plus petite possible. Ce problème a des applications au delà des mots et de l'imaginaire populaire. On distingue deux problèmes de l'arbre couvrant :

— Le problème de l'arbre couvrant classique impose de plus que les sommets de l'arbre soient exactement les points de l'ensemble des points donné en entrée du problème. Ce problème peut être résolu de manière optimale en temps polynomial.

— Le problème de l'arbre de Steiner n'impose pas la position des sommets de l'arbre couvrant. Il s'agit d'un problème NP-commplet que nous allons revoir à la prochaine séance.

Pendant cette séance de TME, nous allons travailler essentiellement à propos du problème de l'arbre couvrant classique. Les algorithmes sont à implanter dans le canevas `TME1_cpa_20170207.jar`. Afin de pouvoir comparer les résultats entre eux, tout calcul qui suit doit être appliqué à un même fichier de données `input.points` disponible sur la page Web de l'UE.

Bonus/malus sur le projet principal de l'UE : A la fin de la séance de TME (12h45, cachet du serveur de messagerie faisant foi), prendre une capture d'écran du meilleur score obtenu par vos calculs, via le canevas GUI, et envoyer à `buixuan@lip6.fr` (maximum 2 courriels par groupe, maximum 2 personnes par groupe). N.B. : pour être valide, tout résultat doit être appliqué à exactement le fichier de données `input.points` disponible sur la page Web de l'UE.

1 Approximation

Notre premier objectif est de couvrir les points donnés au départ, peu importe le coût de l'arbre couvrant. Nous ne cherchons pas encore à fournir une solution optimale.

Exercice 1 – Approximation plus que naïve

Nous partons du principe où un chemin est un arbre.

1. Implanter un calcul partant d'un point aléatoire de l'ensemble de départ, visiter le point le plus proche de ce point. Tant qu'il existe encore des points non-visités, visiter le point le plus proche du dernier point visité. Retouner la liste des points par ordre de la visite. Cette liste représente un arbre couvrant (qui est un chemin) de coût généralement astronomique.
2. Admirer l'horrible score obtenu avec cette méthode sur le canevas du TME.

2 Arbre couvrant classique

Le problème de l'arbre couvrant classique est polynomial. Il a été montré que l'approche glouton résout ce problème de manière optimale.

Exercice 2 – Algorithme Kruskal

Le principe est le suivant :

- Construire une liste de toutes les arêtes possibles de l'arbre, appelées arêtes candidates : l'ensemble de toutes les paires constituées de deux points de l'ensemble des points de départ fera l'affaire.
- Trier la liste des arêtes candidates par ordre croissant selon leur poids (la distance entre les deux extrémités).
- Initialiser une liste "solution" à vide.
- Parcourir la liste des arêtes candidates par ordre croissant :
  - Si l'ajout de l'arête courante ne crée pas de cycle dans la solution, ajouter celle-ci dans la solution.
  - Répéter cette opération tant que la liste solution ne comporte pas tous les points de l'ensemble des points de départ.
- Traduire la liste "solution" en une structure d'arbre et retourner celle-ci.

1. Implanter l'algorithme Kruskal et analyser sa complexité. Pour le test des cycles dans une liste d'arêtes, on peut utiliser un système d'étiquettes : initialement tous les points ont des étiquettes distinctes deux à deux ; on ne peut ajouter une arêtes dont les extrémités comportent des mêmes étiquettes ; à chaque ajout d'arête dont les deux extrémités ont des étiquettes x et y, re-étiquetter tous les points ayant étiquette x en y. PS : la complexité au pire cas de cette méthode est horrible. En revanche, elle est rapide à coder et est suffisante pour notre instance du jour.

3 Arbre de Steiner

Le problème de l'arbre de Steiner étant NP-complet, nous allons tenter de fournir la meilleure approximation possible de la solution optimale (sans pouvoir savoir quelle est cette solution). Nous supposons qu'une solution valide a été obtenue : l'arbre de Kruskal fera l'affaire.

Exercice 3 – Règle du barycentre

Le principe est le suivant :

- Parcourir les sommets de l'arbre courant, déterminer une configuration où nous avons trois sommets A, B et C de l'arbre tel que B et C sont voisins de A.
- Soit I le barycentre de A, B, et C.
- Considérer l'arbre obtenu en remplaçant les arêtes AB et AC par les trois arêtes AI, BI et CI.
- Si le nouvel arbre a un meilleur score que l'ancien arbre, retenir le nouvel arbre à la place de l'ancien arbre.
- Répéter ces opérations jusqu'au point de stabilisation.

1. Implanter la règle du barycentre en veillant à ce que les calculs se terminent à temps... (il est par exemple inutile de considérer les points A, B et C où le triangle ABC croise une autre arête de l'arbre courant).

Exercice 4 – Règle Torricelli-Fermat

Le principe est le suivant : Se renseigner à propos du point de Torricelli-Fermat et revisiter l'implantation précédente afin d'obtenir de meilleurs scores sur le fichier test `input.points`.

Exercice 5 – Règle ahdoc

Proposer des règles supplémentaires afin d'améliorer le score. On peut penser à optimiser des configurations à 4 points, à 5 points, etc. Remarquer la dégradation phénoménale en temps de calcul au profit des améliorations du score. Peut on espérer qu'une optimisation des configurations à 8 points se termine en temps raisonnable sur une machine de l'ordre du Ghz avec le fichier test `input.points` ?