PROJET : LE TOUR DU MONDE LE MOINS CHER

Table des matières

Introduction

Le problème du tour du monde le moins cher - plus connu sous le nom de “Traveling Salesman Problem” ou TSP - est un problème NP-hard très connu, et qui peut se résumer ainsi :

Soit un graph dont les sommets représentent des villes, et dont les arêtes représentent les distances entre les villes. Quel est le chemin le plus court reliant tous les sommets du graph ?

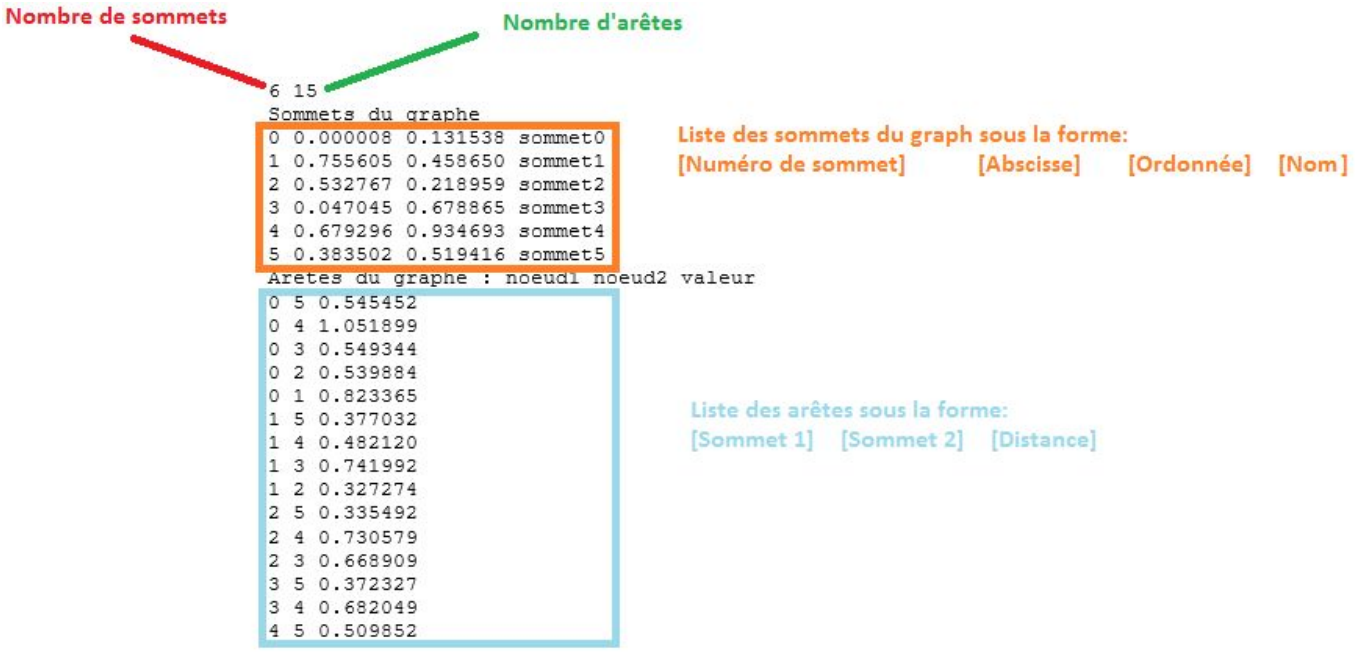
Dans ce projet, nous avions pour objectif de tenter de résoudre ce problème avec “l’algorithme des fourmis”. Le principe est le suivant : des fourmis parcourent le graph en déposant des phéromones sur les arêtes qu’elles empruntent. La présence de phéromones sur une arête augmente la probabilité pour qu’une fourmis emprunte cette dernière. Au bout d’un certain temps, le chemin le plus court aura été parcouru un plus grand nombre de fois. Ainsi l’algorithme sera capable de renvoyer le chemin de plus faible coût trouvé.

Spécifications

Données

Données d’entrée

Le but du projet étant d’estimer le plus court chemin pour parcourir tous les sommets d’un graph, nous avions à notre disposition des fichiers textes dans lesquels se trouvaient les nécessaire à la construction des graphs. Ces fichiers se présentaient sous la forme suivant :



L'intérêt de travailler avec des fichiers textes dont la syntaxe est formatée ainsi est qu’il est alors très

simple d'extraire toutes les informations nécessaires à la création de graph :

- Nombre de sommets & d'arêtes.
- Positions des sommets.
- Distances entre les sommets (coût d'une arête).

Structures

Le projet faisant intervenir des graph, il fut nécessaire de créer les structures associés à un tel concept :

- Structure NOEUD :

```
struct NOEUD {
  int numero;
  char ville[100];
  double x;
  double y;
  Liste voisins;
};
```

- Structure ARETE :

```
struct ARRETE {
  int numero; //numero de depart
  int arr; //numero d arrive
  double cout;
  double pheromones;
};
```

- Structure GRAPH : pointeur vers un noeud.

D'autres structures basiques telles que les files et les listes ont dû être implantées.

De plus, la méthode de résolution choisie pour le projet étant l'algorithme à colonie de fourmis, il a également été nécessaire d'implanter la structure FOURMI :

```
struct FOURMI {
  File solution;
  int ville_depart;
  int ville_courante;
};
```

Fonctions

(Pour plus de detail, se référer aux fichiers headers correspondants)

Les fonctions liées à la manipulation de graph (fichier graph.c graph.h)

- creer_arrete : fait ce que son nom indique, retourne un pointeur sur ARRETE.
- creer_noeud : fait ce que son nom indique, retourne un pointeur sur NOEUD.
- creer_graph : fait ce que son nom indique, retourne un GRAPH.
- ajout_noeud : crée un noeud selon les paramètres envoyés et retourne un GRAPH.
- initialiser_graph : initialise un GRAPH à partir d'un fichier.
- supprimer_graph : fait ce que son nom indique.

Les fonctions liées à l'évolution des phéromones (fichier tsp.c tsp.h)

- initialisation_pheromones : initialise les phéromones de chaque arête à $\varepsilon = 10^{-5}$
- evaporer_pheromones :

Les fonctions liées l'action des fourmis (fichier tsp.c tsp.h)

- choix_prochaine_ville : choisi selon une loi de probabilité donné la prochaine ville sur laquelle une fourmi va aller. Retourne un pointeur sur l'arête correspondante.
- proba : calcul et retourne la proba (double) pour une fourmi d'aller sur une ville donnée en paramètre
- is_in_tabu : renvoi 1 si la fourmi est déjà passé par le sommet envoyé en paramètre, 0 sinon.

Tests

Nous prévoyions de tester toutes les fonctions nécessaires à l'algorithme de TSP. C'est à dire :

- La création du graph à partir d'un fichier
- L'initialisation des phéromones
- Le calcul des probabilités de mouvement des fourmis
- Le choix aléatoire selon les probabilités de la ville suivante pour les fourmis
- L'évaporation des phéromones

D'autre part nous utilisions les fonctions de manipulation des files et listes que nous avions déjà créées et testées dans les BE précédents. Nous n'avions donc pas prévu de les tester à nouveau.

Répartition du travail et planning prévu

La séance 1 :

- Création des structures de données
- Etablissement de la structure du code et du projet
- Création des fonctions de manipulation des structures

Julien	Thomas
- Création de la structure ARRETE - Création de la structure Fourmi - Fonction creer_arrete - Fonction ajout_arrete - Fonction initialiser_graph - Fonction supprimer_graph	- Création de la structure NŒUD - Structure GRAPH - Fonction creer_noeud - Fonction ajout_noeud - Fonction creer_graph

Séance 2 :

Julien	Thomas
- Fonction initialisation_fourmis - Fonction initialisation_pheromones	- Algorithme de test des premières fonctions - Fonction proba

Séance 3 :

Julien	Thomas
- Fonction evaporer_pheromones - Adapter l'algo de test	- Fonction choix_ prochaine_ville - Fonction is_in_tabu

Séance 4 :

Julien	Thomas
- Algorithme de TSP	- Affichage graphique

Implantation

Etat du logiciel

Le logiciel est fonctionnel, mais il n'est pas pleinement optimisé et assez lent dès que le nombre de sommet du graphe dépasse 100. Il effectue 10 cycles en plaçant un même nombre de fourmis sur chaque sommet du graphe (2 pr défaut). Ces fourmis parcourt un chemin "pseudo-aléatoire" dont la probabilité de choix d'une direction dépend à la foi de la distance (inverse du coût du chemin) à parcourir dans une direction et des phéromones déposées par les autres fourmis lors des cycles précédents. Le chemin choisit reliant tous les sommets du graphe est celui présentant le moindre cout total.

Le résultat est retourné en console en affichant le graphe (chaque sommet suivi de chacune de ses arêtes), le chemin de moindre coût et son coût. Un affichage graphique complémentaire à également été réalisé affichant le graphe, le meilleur chemin en rouge et son coût.

Tests effectués

Nous avons réalisés tous les tests que nous prévoyons, c'est à dire que nous avons créé un graph complet simple de quatres sommets qui nous a permis de tester :

- La lecture du fichier et la création du graph correspondant
- L'initialisation des phéromones sur chacunes des arêtes
- l'évaporation des phéromones sur les arêtes

Nous avons également testé le fonctionnement de la structure "Fourmi" qui nous a par ailleur permis de tester :

- La fonction is_in_tabu permettant d'évaluer si une fourmi était déjà passé par un sommet
- La fonction de probabilité de choix de chaque sommet
- La fonction de choix de la direction suivante pour une fourmi en fonction de ces probabilités

Toutes ces fonctions semblent fonctionnelles à l'issue de ces tests.

Exemple d'exécution

Avec le fichier "graphe11.txt" placé au préalable dans le dossier "fichier" du projet.

Nous lancons le programme en se plaçant à la racine du projet et en exécutant la commande :

./exec graphe11.txt

Le programme va alors initialiser le graphique à partir du fichier et l'afficher.
Nous obtenons alors l'affichage suivant dans la console :

```
noeud 0 sommet0 0.000008 0.131538
        -> 0 1 0.823365 0.000000
        -> 0 2 0.539884 0.000000
        -> 0 3 0.549344 0.000000
        -> 0 4 1.051899 0.000000
        -> 0 5 0.545452 0.000000
noeud 1 sommet1 0.755605 0.458650
        -> 1 2 0.327274 0.000000
        -> 1 3 0.741992 0.000000
        -> 1 4 0.482120 0.000000
        -> 1 5 0.377032 0.000000
        -> 1 0 0.823365 0.000000
noeud 2 sommet2 0.532767 0.218959
        -> 2 3 0.668909 0.000000
        -> 2 4 0.730579 0.000000
        -> 2 5 0.335492 0.000000
        -> 2 1 0.327274 0.000000
        -> 2 0 0.539884 0.000000
noeud 3 sommet3 0.047045 0.678865
        -> 3 4 0.682049 0.000000
        -> 3 5 0.372327 0.000000
        -> 3 2 0.668909 0.000000
        -> 3 1 0.741992 0.000000
        -> 3 0 0.549344 0.000000
noeud 4 sommet4 0.679296 0.934693
        -> 4 5 0.509852 0.000000
        -> 4 3 0.682049 0.000000
        -> 4 2 0.730579 0.000000
        -> 4 1 0.482120 0.000000
        -> 4 0 1.051899 0.000000
noeud 5 sommet5 0.383502 0.519416
        -> 5 4 0.509852 0.000000
        -> 5 3 0.372327 0.000000
        -> 5 2 0.335492 0.000000
        -> 5 1 0.377032 0.000000
        -> 5 0 0.545452 0.000000
```

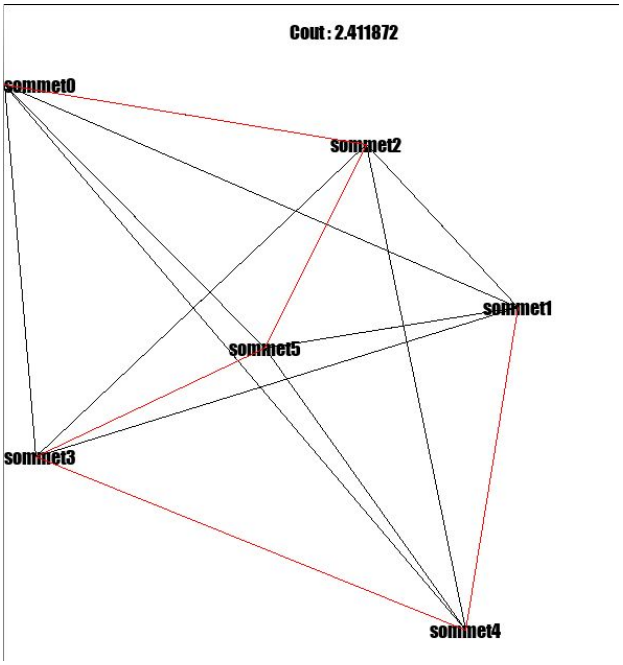
avec la syntaxe de la forme :
noeud numero_de_noeud nom_du_noeud poxition_x position_y
(arête 1) -> numero_noeud_depart numero_noeud_arrivée cout pheromones

Le programme initialise alors 2 fourmis par sommet et toutes les phéromones à 1x10^-5 sur chacune des arêtes. Les 10 cycles de l'algorithme déterminant le meilleur chemin par la méthode des fourmis déposant des phéromones sont lancés. L'affichage du résultat se fait sous cette forme en console :

```
Cycle 1 sur 10
Cycle 2 sur 10
Cycle 3 sur 10
Cycle 4 sur 10
Cycle 5 sur 10
Cycle 6 sur 10
Cycle 7 sur 10
Cycle 8 sur 10
Cycle 9 sur 10
Cycle 10 sur 10


Chemin le plus court reliant tous les sommets :
0<--->2<--->5<--->3<--->4<--->1
Cout : 2.411872
```

et également sous cette forme dans une fenêtre graphique distincte :



Les optimisations et les extensions réalisées

La seule optimisation que nous avons faite est dans l'algorithme de TSP qui prends beaucoup de temps, nous avons réduit le nombre de passage dans la boucle principale en arrêtant le calcul pour la fourmi k dès que le coût devient supérieur au coût du meilleur chemin en intercalant un break sur condition.

Par ailleurs nous avons réalisés un affichage graphique plus visuel avec les librairies SDL, SDL_draw et SDL_ttf qui affiche le graph et ses arêtes sur une fenêtre graphique, le meilleur chemin reliant tous les nœuds en rouge et son coût.

Suivi

Problèmes rencontrés

Nous n'avons pas rencontrés de problèmes particuliers lors de ce projet si ce n'est les bugs habituels suite à la mauvaise direction d'un pointeur, l'oublie d'un point virgule ou d'une lettre au nom d'une variable.

Planning effectif

Séance 1 :

Julien	Thomas
- Création de la structure ARRETE - Création de la structure Fourmi - Fonction creer_arrete - Fonction ajout_arrete - Fonction initialiser_graph - Fonction supprimer_graph	- Création de la structure NŒUD - Structure GRAPH - Fonction creer_noeud - Fonction ajout_noeud - Fonction creer_graph

Séance 2 :

Julien	Thomas
- Fonction initialisation_fourmis - Fonction initialisation_pheromones	- Fonction proba - Algorithme de test des premières fonctions

Séance 3 :

Julien	Thomas
- Fonction evaporer_pheromones - Adapter l'algo de test	- Fonction choix_ prochaine_ville - Fonction is_in_tabu

Séance 4 :

Julien	Thomas
- Algorithme de TSP	- Affichage graphique

Nous avons globalement bien suivi notre planning, les tâches en verte sont celles ayant été complétées en séances, celles en orange ont été terminé soi lors de la séance suivante, soit chez nous. La création d'un Makefile et du fichier Readme ainsi que leur adaptation à été faite tout au long du projet, en séance et chez nous.

Qu'avons-nous appris et que faudrait-il de plus ?

Ce projet nous as appris l'importance d'une bonne organisation des fichiers lors d'un projet informatique, aussi bien leur structure interne (fichiers .h et .c structurés) que leur rangement dans le fichier du projet (un dossier pour les sources, un pour les headers, un pour les fichiers de données...).

Il nous est apparu particulièrement important d'organiser de manière lisible nos fichiers .h car ce sont eux que nous consultions en priorité lorsque nous avions besoin d'une précision sur une structure crée par le binome ou bien que nous avions oublié (ce fut le cas pour la structure Fourmi que nous n'avons pratiquement pas touché avant la dernière séance).

Suggestion d'améliorations du projet

Le code de notre projet n'est pas pleinement optimisé et ce serait une piste d'amélioration possible. Néanmoins l'amélioration qui serait peut être la plus nécessaire serait l'ajout de la gestion de graph partiellement incomplet et des tests de bonne constitution des fichiers que nous considérons comme valides dans notre implémentation.

Conclusion

Ce projet nous a permis d'appréhender l'importance de bien organiser son code et son projet, et de développer fonctionnalité apres fonctionnalité. Cette méthode nous a permis de travailler assez rapidement et efficacement. Notre projet fonctionne à ce jour mais mérite d'être optimisé et amélioré pour pouvoir vérifier la validité des fichiers de données et de traiter le cas de graphes incomplets.

Par ailleur notre programme comportant une partie graphique, nous avons pu apprendre les bases de la librairie SDL et de ses compagnes (TTf, Draw...) et de comprendre comment est formé un affichage graphique.