Ilhan ACHERCHOUR / Hakim AOUDIA
Semestre 4 - 2ème année de Licence

26/04/2022

Perfectionnement en C

Rapport du TP 9 - Le voyageur de commerce génétique

Sommaire :

Introduction

Ce TP9 est le dernier de cette année pour le perfectionnement à la programmation C. C'est pour cela que c'est un TP qui nous met au défi avec une notion nouvelle pour nous : **La**

programmation génétique. Avec cette méthode nous devions résoudre un problème connu qui est celui du voyageur de commerce.
Ce problème consiste à trouver la distance de chemin le plus court entre plusieurs villes données.

Il y a de nombreux algorithmes pour permettre de résoudre ce problème avec une solution plus ou moins optimale. C'est le cas de l'**algorithme glouton** qui permet de résoudre ce problème mais n'offre pas la meilleure distance.
Nous utiliserons donc la programmation génétique qui permet de trouver une solution assez optimale cependant on ne peux pas savoir s' il existe une meilleur solution.

Compilation

Nous avons réalisé un makefile pour ce projet. On compile à l'aide de la commande :
`make pvc` ou seulement `make`

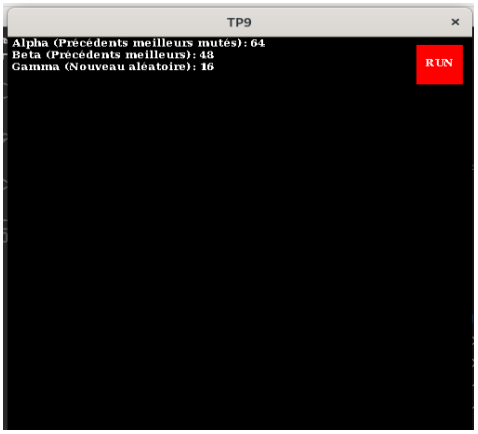
On lance le projet de cette façon :
`./pvc α β γ -X -Y` tel que
α : Réalise une mutations sur les α meilleures visites précédentes.
β : Récupére les β meilleurs visites précédentes.
γ : Génération de γ nouvelles visites aléatoires.
-X : - Soit **-c** (pour **clic**)
- Soit **-h** (pour **hasard**)
- Soit **-f** (pour **fichier**)
-Y : - Pour l'option **-h** on écrit le nombre de villes à générer.
- Pour l'option **-f** on écrit le fichier à utiliser.

Voici des exemples d'utilisation avec des bon paramètre :
```
./pvc 68 48 16 -c
./pvc 68 48 16 -h 50
./pvc 68 48 16 -f CARTE.pc
```

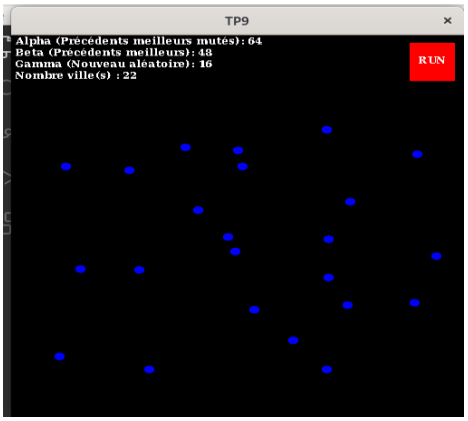
Documentation utilisateur

L'utilisateur à 3 choix pour lancer le programme.

Option -c avec clic de l'utilisateur

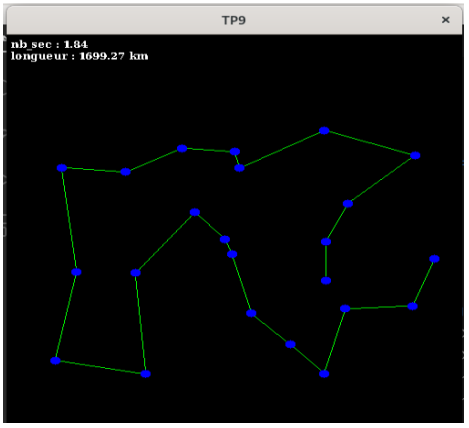


Il peut utiliser le programme avec la souris avec l'option -c. Cela va lui ouvrir une fenêtre avec écrit en haut à gauche les paramètres donnés et le nombre de villes sur l'écran.



Il peut cliquer sur l'écran tant qu'il le souhaite. Le nombre de villes va augmenter dynamiquement en haut à gauche. Quand il le souhaite, il peut lancer la recherche du chemin le plus court en appuyant sur le bouton Run en haut à droite à l'aide la souris.

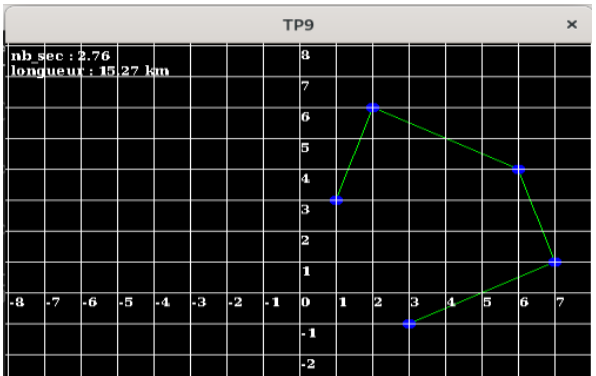
Une fois le calcule de distance lancé, il y a en haut à gauche la durée du programme ainsi que la meilleure distance entre les villes.



Option -h génération de villes aléatoires

L'option -h permet de générer des villes aléatoirement sur l'écran. Le nombre de villes est spécifié sur en paramètre après le paramètre -h. Lorsque l'utilisateur choisit cette option, il y a en haut à gauche la durée du programme et la distance entre les villes.

Option -f génération de villes en fonction d'un fichier



L'option -f permet d'ouvrir un fichier spécifié en paramètre. Ce fichier contient des noms de ville ainsi que leurs positions en abscisse puis en ordonnée. Une fois le programme lancé,

les villes sont affichées sur une fenêtre avec un quadrillage et un repère orthonormé.

On peut voir en haut à gauche la durée du programme ainsi que la distance totale.

Ici on a lancer le programme avec cette commande :

```
./pvc 68 48 16 -f CARTE.pcv
```

Documentation technique

L'algorthime principal du projet est celui décrit dans le TP. En résumé, on commence par récupérer des villes qu'on place dans des visites. Les visites sont des suites de villes permettant de calculer une distance totale.
Ensuite nous avons des listes de visites. Ces listes de visites sont **triés** pour connaître quelles sont les meilleures visites.

L'algorithme consiste à **muter** les alpha premiere visites, **récupérer** les bêta meilleurs visite et **générer** de nouvelles visites aléatoirement à chaque tour de boucle que nous appellerons génération.

Chaque génération est en théorie **soit meilleure soit égale** à la précédente génération. A chaque génération on trie de nouveau les suites de visites et on affiche la meilleure visite sur une fenêtre graphique.

Ce moyen de programmation ressemble à ce qui se passe dans la vie avec la sélection naturelle. Les individus sont triés par génération en supprimant les plus faibles et en gardant les meilleurs. On garde les meilleurs "gènes" dans notre cas, on garde les meilleurs chemins.

Moyen mis en oeuvre et difficultés

Commencement du TP

Tout d'abord on a commencé à comprendre ce qu'était la programmation génétique en **écoutant** les informations données par la professeur. On a **dessiné des exemples** sur feuilles aussi pour mieux appréhender les problèmes. On a vite compris la logique de l'algorithme. Il ne manquait plus qu'à créer les **structures adéquates** et les fonctions de

base (telles que le calcul de distance, la permutation dans une liste, la mutation de visites, l'échange de valeurs etc).

Implémentation moins complexe que prévu

Découverte de l'efficacité de la programmation génétique. On a été étonné par l'efficacité de cet algorithme. On pensait qu'il serait dur à mettre en place mais une fois qu'on a bien compris cet algorithme on a réussi à le mettre en place.

Problème de chemin.

Nos premières observations sur l'affichage des villes et du chemin qui les traverse nous ont montré un phénomène étrange : plusieurs visites possédaient exactement la même distance de parcours. **A cause d'une affectation** sur un type de structure. Cela génère des chemins totalement incohérents et **crée des croisements** dans tous les sens. Certaines villes étaient liées à plusieurs villes. On a géré le problème en affectant les valeurs des villes (x, y, nom) variables par variables.

Affichage

Une fois la phase de recherche de chemin fini nous sommes passés à l'affichage. On avait le choix entre la bibliothèque MLV ou SDL. Nous sommes restés avec la bibliothèque MLV car nous l'avons déjà utilisé de nombreuses autres fois pour différents projets précédemment. Nous étions donc déjà familier avec celle-ci.

Nous avons dû faire la **gestion de clic**, et l'affichage de texte (informations), rond (pour les villes) et de lignes (chemin entre villes).

Ce qui nous a posé problème était pour l'**affichage de ville donné dans un fichier**. En effet, dans un fichier les villes données en paramètre **possèdent de petites valeurs** comparé à la taille en pixel de notre fenêtre. Nous avons donc dû placer ces villes dans une fenêtre de dimension plus grande. On a alors **multiplié ces valeurs en fonction de la taille** de la fenêtre et de la valeur maximum des villes à l'aide d'une **"formule".**

De plus, on devait gérer des **valeurs négatives**. On a alors créé un quadrillage et un repère orthonormé. A l'aide d'une formule nous sommes parvenus à un résultat convenable qui se rapproche de l'affichage donnée dans l'exemple du TP.
Cette partie d'affichage ne nous a pas posé trop de problèmes.

Amélioration

Nous avons ajouté une option permettant de saisir des villes à l'aide de la souris comme vu dans la section **Documentation utilisateur.**

Conclusion

On a vraiment aimé réaliser ce TP car cela nous a permis d'apprendre une nouvelle notion intéressante. La programmation génétique est utile étant donné qu'elle se rapproche de la réalité dans la façon dont elle sélectionne les meilleurs éléments par le hasard et par des mutations. C'est une sorte de sélection naturelle réalisée par l'ordinateur. De plus ce TP nous a permis d'utiliser toutes les connaissances acquises dans les précédents TP dans un projet plus gros que les autres.