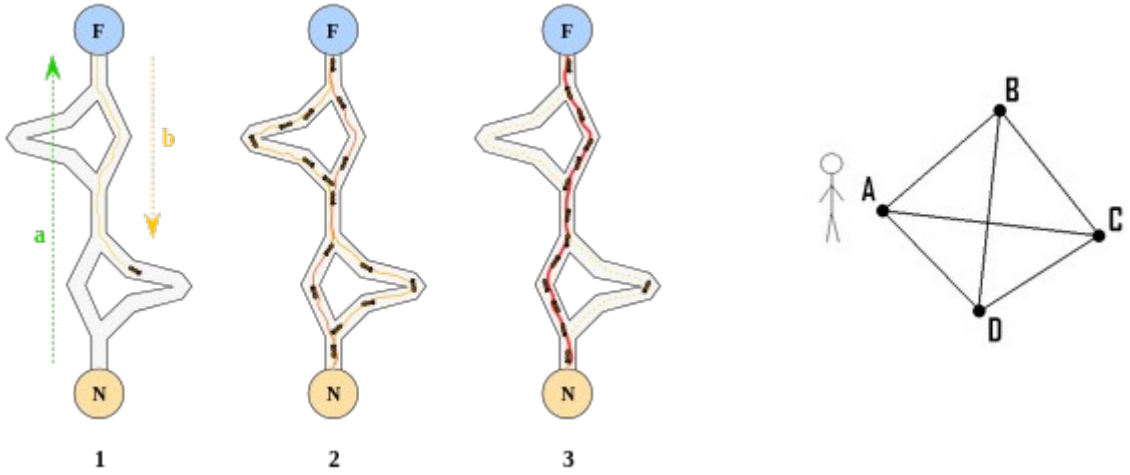
Programmation Parallèle – Parallélisation MPICH
Voyageur de commerce – Colonie de fourmis
Sarfraz Laïk
12312565



I/ Introduction

Le problème qui se pose pour un voyageur est de visiter toutes les villes d'une région donnée avec la plus petite distance pour économiser de l'énergie et du temps. Nous appliquons pour cela l'algorithme de la colonie de fourmis. Cette méthode consiste à mettre des fourmis dans chaque ville, et les faire déplacer dan chacune de celles ci puis de la faire revenir à sa position initiale.
Un lien entre deux villes i et j comportera :
— une distance d (ij)
— une quantité de phéromones laissée par les fourmis p(ij)
À chaque étape, les fourmis se déplacent vers une autre ville suivant la probabilité phéromones/distance [7] . Lorsqu'une fourmi choisit un chemin, l'arrête est mise à jour selon la relation p (ij) =p (ij)/2 + 100 ∗ (nb de foumis passées entre i et j).
J'ai utilisé la bibliothèque de graphe Lemon.

II/ Pseudo Code

Je créer une structure fourmi qui contient ;
- Un tableau de toutes les villes, c'est en fait un flag pour savoir si la ville a été visitée ou pas.
- Sa position initiale
- Sa position courante

Je place ensuite une fourmi par ville. C'est la fonction WhatCityShouldIVisit qui est la plus importante. La fourmi analyse chaque arrête en vérifiant que les villes à l'opposé de l'arrête ne sont pas encore visitées.. Si elle n'est pas visitée, la fourmi met (push) l'arrête (edge) qui mène à la ville dans un vector <pair>. Elle fait de même avec toutes les arrêtes susceptibles d'être visitées. Elle compare ensuite toutes les arrêtes et choisit celle qui a le plus ratio de phéromones/ distance. On fait cela avec toutes les fourmis. On répète cela autant de fois qu'on le voudra, avec une boucle for pour accentuer les résultats. A la fin, il y a un chemin qui se forme avec le plus de phéromones, c'est ce chemin là qui est le plus économique et rapide pour le voyageur.

III / Tableau de résultats

Les test suivants ont été réalisés sur un ordinateur 4 coeurs Intel(R) Core(TM) i5 CPU M 560 2.67GHz avec 6Go de RAM. La compilation a été effectuée avec mpic++ et g++ et l'exécution avec mpirun.mpich -np 4 ./[exécutable].

Nombre de villes	*Temps en séquentiel*	*Temps en parallèle MPICH*
100	1,2	0,3
200	9,659	1,976
500	39,7	154

Conclusions

La version MPICH permet de calculer 4 fois plus rapidement le chemin le plus rapide pour un commerçant.