# Contrôle continu - Complexité - Master MIASHS 2016

Sommaire

[Contrôle continu - Complexité - Master MIASHS 2016 1](#_Toc466568208)

[Introduction 1](#_Toc466568209)

[Structure et fonctionnement du programme 1](#_Toc466568210)

[Structure du programme 1](#_Toc466568211)

[Fonctionnement de l'algorithme 2](#_Toc466568212)

[Résolution du problème à deux branches 3](#_Toc466568213)

[Description de la solution 3](#_Toc466568214)

[Remarques 3](#_Toc466568215)

[Résolution du problème du voyageur de commerce 4](#_Toc466568216)

[Description de la solution 4](#_Toc466568217)

[Remarques 4](#_Toc466568218)

[Conclusion 5](#_Toc466568219)

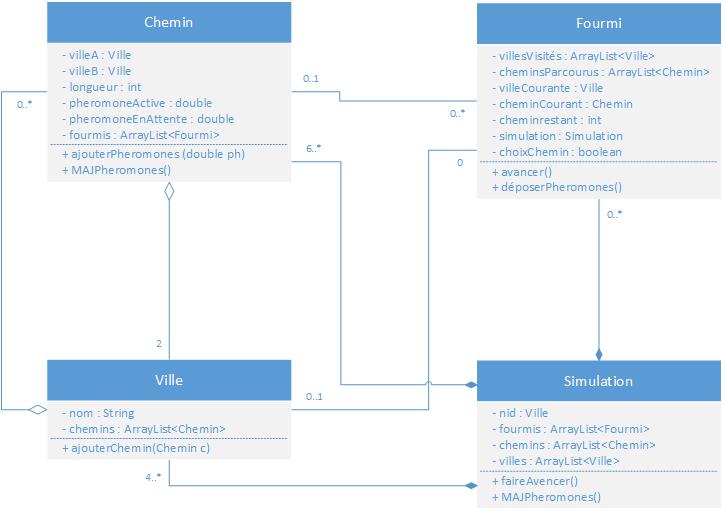
## Introduction

Le but de ce projet est de mettre en place algorithme à colonie de fourmis pour trouver, dans un graphe, le plus court cycle hamiltonien d'un graphe. Pour ce faire, un cas particulier comprenant deux sommets et deux arêtes est d'abord résolut, puis le problème du voyageur de commerce avec n villes.

## Structure et fonctionnement du programme

### Structure du programme

Le programme est structuré de la manière suivante :



On a donc :

* Une classe Ville qui représente les commets du graphe. Une Ville est caractérisée par un nom, ainsi que les chemins qui relient cette ville.
* Une classe Chemin qui représente les arêtes du graphe. Un chemin est définit par sa longueur, les deux villes qu'il relie, la quantité de phéromones ainsi que les fourmis présentent sur le chemin.
* Une Fourmi qui représente les fourmis envoyées dans le graphe à chaque pas. Une fourmi est caractérisée par la liste des villes qu'elle a visité, la liste des chemins par lesquels elle est passée, elle ville dans laquelle elle se trouve ou la dernière ville visitée, le chemin sur lequel elle se trouve, si elle est sur un chemin, la distance qu'il reste à la fourmi pour arriver au bout du chemin courant, ainsi qu'un booléen qui définit le manière avec laquelle elle va choisir un chemin lorsqu'elle aura le choix.
* Une classe Simulation qui supervise les autres composants comprenant le nid, c'est-à-dire la ville d'où partent les fourmis, ainsi que les liste des fourmis, villes et chemins.

### Fonctionnement de l'algorithme

A chaque pas, un nombre k de fourmis sont envoyées dans le graphe, à partir du nid, pour parcourir un cycle hamiltonien du graphe. Sur leur circuit, les fourmis vont se retrouver à des embranchement et vont donc devoir choisir un chemin, ainsi que poser des phéromones pour influencer les autres fourmis.

#### Déplacement des fourmis

Une fourmis peut être soit sur une ville, soit sur un chemin.

Lorsqu'une fourmi arrive sur un chemin son paramètre cheminRestant est initialisé avec la longueur du chemin et est décrémenter à chaque nouveau pas, jusqu'à ce que celui-ci soit nul. Les fourmis se déplacent donc à une vitesse de 1 unité de longueur par pas. Lorsque cet attribut est nul, la fourmi est alors arrivée dans une ville. Elle va donc devoir choisir dans quelle direction elle va aller.

Une fourmi va pouvoir choisir son chemin de deux manière différentes :

* Elle regarde les différents chemins qui s'offrent à elle, puis elle choisit le chemin qui possède le plus de phéromones. (Option 1)
* Elle calcul des probabilités selon la formule vue en cours :

Si la ville à laquelle mène le chemin n'a pas été visitée, sinon le chemin n'est pas pris en compte. Grâce à un nombre aléatoire, elle choisit un chemin. Les coefficient α et β permettent de modifier l'importance des phéromone(α) et de la longueur des chemins (β) (Option 2)

Le choix d'une des deux options est définit en début de simulation.

#### Dépôt et évaporation des phéromones

La classe Fourmi possède un attribut statique Q qui représente une quantité de phéromone. Cet attribut est initialisé à 1. C'est lorsqu'une fourmi a terminé son circuit qu'elle dépose une quantité de phéromones sur chacun des chemins qu'elle a parcouru. Cette quantité est égale à l'attribut Q divisé par la longueur totale du circuit. Ainsi, les phéromones d'une fourmi qui aura parcouru un grand cycle auront moins d'impacte sur les autres fourmis, que ceux provenant d'une fourmi ayant fait un circuit court.

De plus à chaque pas, les phéromones présents sur un chemin au pas précédent vont s'évaporer selon la formule vue en cours :

Dans cette formule, représente les phéromones présentent au pas précédent, les nouveaux phéromones et est un coefficient définit par l'utilisateur. S'il est nul, les phéromones s'évaporent complètement à chaque pas, et s'il est égale à 1, les phéromones ne s'évaporeront pas du tout.

## Résolution du problème à deux branches

### Description de la solution

Le programme permettant de résoudre ce problème se trouve dans la méthode main de la classe Simulation. Il s'exécute avec les arguments suivants :

* Le nombre de fourmis créées à chaque pas
* la longueur du premier chemin
* la longueur du second chemin
* le coefficient d'évaporation des phéromones
* l'option à utiliser (1 pour l'option 1, sinon l'option 2 est sélectionnée)

A chaque pas, il écrit dans la console le nombre de fourmis présentent sur chacune des branches. Il s'arrête lorsque l'une des deux branches est vide.

### Remarques

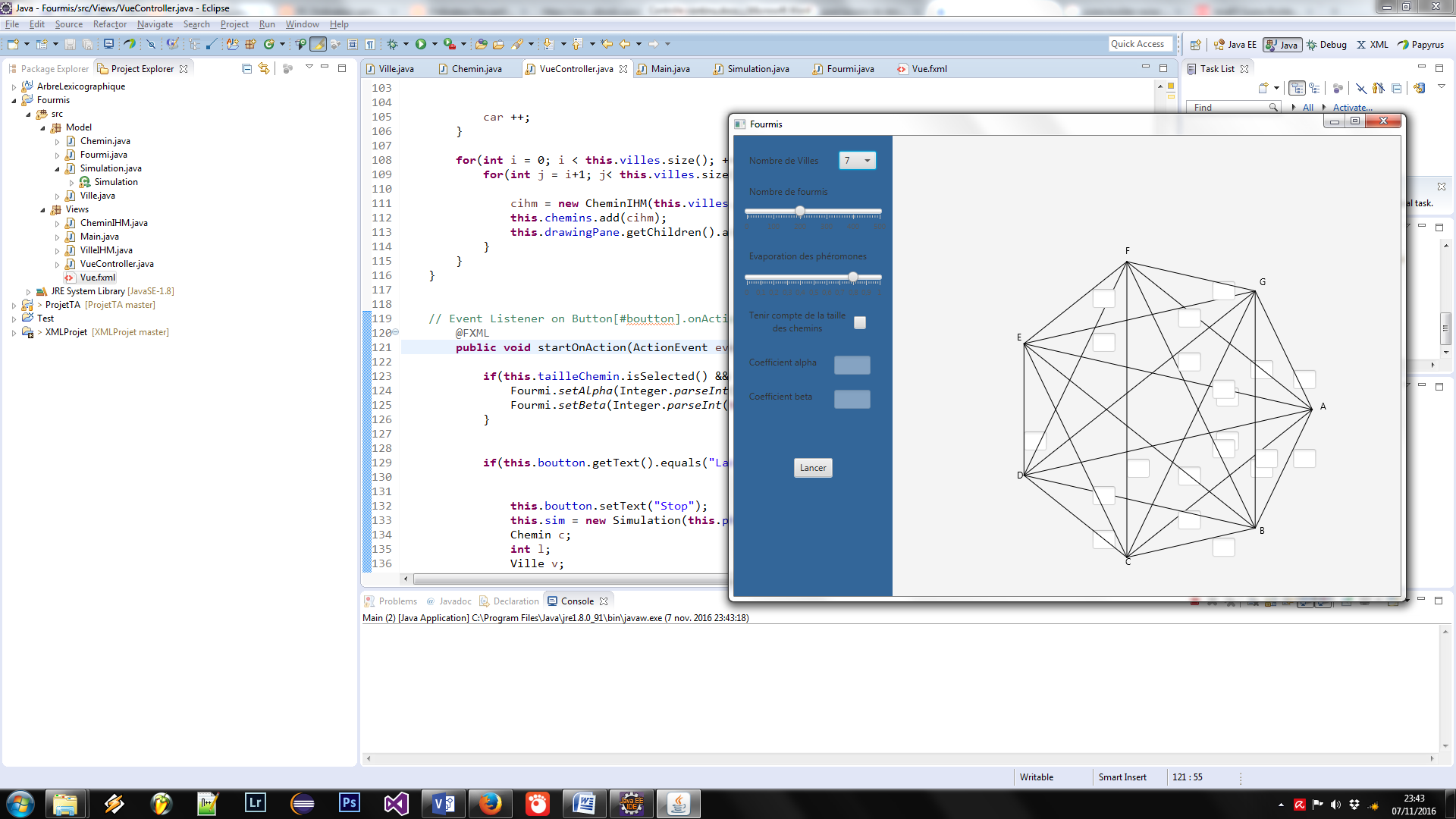
Lorsque l'on utilise l'option 1 avec des branches de taille égale et de taille petite, on s'aperçoit que la branche sur laquelle les fourmis restent est la branche sur laquelle il y a le plus de fourmis au premier pas dans la plupart des cas. En effet, si plus de fourmi prennent un chemin, plus de phéromones y seront déposées, et les fourmis suivantes prendront donc ce chemin. Avec l'option 2, les fourmi décident leur chemin selon des probabilités, il y a donc plus de chances pour qu'elles empruntent l'autre chemin, mais globalement les observations précédentes restent les même.

Lorsque l'on paramètre deux chemins de taille différente, quelque soit le nombre de fourmis envoyées à chaque pas et quelque soit l'option choisit, les fourmis finissent toujours toutes sur la branche la plus courte.

## Résolution du problème du voyageur de commerce

### Description de la solution

Le programme permettant de résoudre le problème du voyageur de commerce se trouve dans la classe Main du package Views.

Lorsque le programme est lancé, l'interface suivante apparaît : 

Dans l'ordre, les boutons servent à modifier :

* Le nombre de villes du graphe.
* Le nombre de fourmis créées à chaque pas.
* Le coefficient d'évaporation des phéromones
* Utiliser l'option 2 et tenir compte de la longueur des chemins
* modifier le coefficient α
* modifier le coefficient β

Lorsqu'un nombre de ville est choisit, le graphe correspondant est alors affiché dans le panel à droite. Le graphe créé est complet. Sur chacune des arêtes un champs texte est affiché, lié à l'arête par le coin supérieur droit. Ce champs permet de modifier la longueur de l'arête. Si le champs n'est pas rempli, a longueur par défaut d'une arête est de 20.

Le bouton "Lancer" permet de démarrer la simulation. Lorsqu'une simulation est en cours, les arêtes sur lesquels il n'y a pas de fourmis deviennent clair, ce qui permet de visualiser lorsque les fourmis ont trouvé un chemin. De plus, le nombre de fourmis sur chaque arête s'affiche dans la console à chaque pas.

### Remarques

On observe que, plus le nombre de fourmis créées à chaque pas est important, plus le programme va mettre de temps à trouver une solution. En effet, comme les premières fourmis forment un cycle de manière aléatoire, plus elles sont nombreuses, plus elle vont rester longtemps sur chacun des chemins, dont les plus longs. Mais, si le nombre de fourmis envoyées dans le graphe est trop bas et que le graphe possède plusieurs cycle de longueur proche, il arrive que le programme donne une solution fausse. Il est donc risqué de donner de mettre un nombre de fourmis trop bas.

Si le coefficient d'évaporation des phéromones est nul, les fourmis vont mettre du temps à trouver une solution car seul les phéromones émis au pas courant auront un impact sur le comportement des autres fourmis. Il faut donc trouver une valeur adéquate pour ce paramètre.

De plus, on observe qu'une solution est trouvée, en général, plus rapidement lorsque l'option 2 est utilisée. Celle-ci tient compte de la longueur des chemins, alors que l'option 1 n'en tient pas compte, ce qui est logique.

## Conclusion

Ce projet m'aura permis de découvrir plus amplement l'algorithme à colonie de fourmis, car le temps passé sur celui-ci par les différents tests effectués permettent de comprendre de manière approfondit son fonctionnement.

De plus, implémenter ce modèle, basé sur le comportement d'êtres vivants à été pour moi une expérience très intéressante. En effet, l'idée que l'ont puisse utiliser des comportements naturels et réels pour résoudre des problèmes informatiques est, à mon goût, captivante.