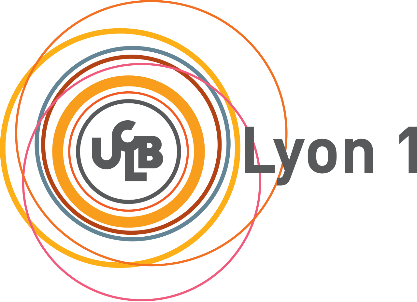
**Rapport Projet Optimisation**

Informatique par Apprentissage

Année scolaire 2017 – 2018

4ème Année - Semestre 8





|  |  |
| --- | --- |
| Jérôme Alibert  Fanny Daudies | Stephane Bonnevay |

Polytech Lyon

Bâtiment Istil,

15 Boulevard André Latarget,

69100 Villeurbanne

04 72 43 12 24

Sommaire

[Fonctionnement de l’application 1](#_Toc517020446)

[Partie Données 1](#_Toc517020447)

[Partie Algorithme 1](#_Toc517020448)

[Explication algorithme Tabou 2](#_Toc517020449)

[Introduction 2](#_Toc517020450)

[Exemple de valeurs retournées 2](#_Toc517020451)

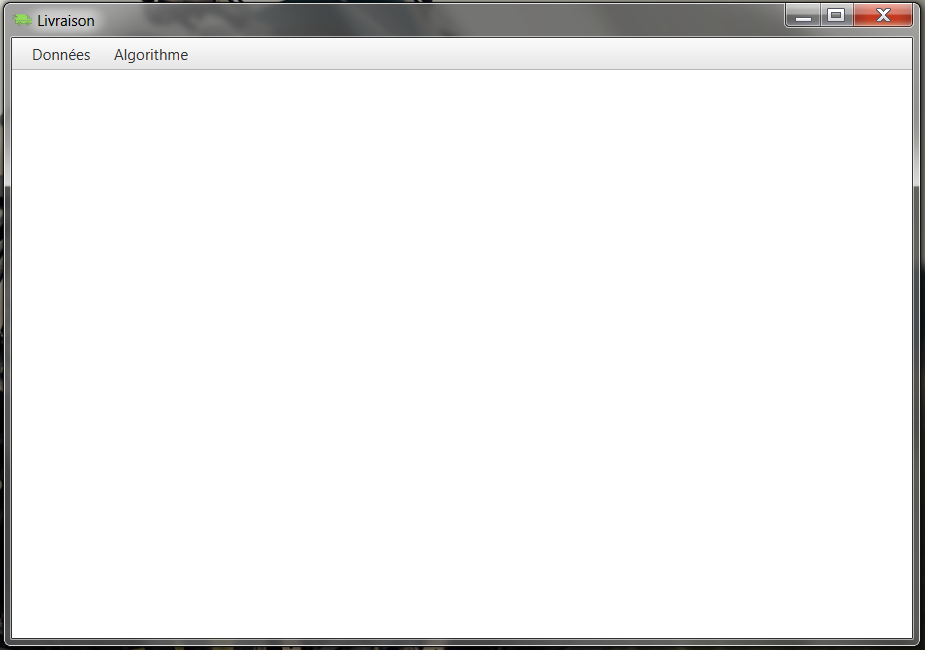
[Explication algorithme Génétique 3](#_Toc517020452)

[Introduction 3](#_Toc517020453)

[Exemple de valeurs retournées 4](#_Toc517020454)

# Fonctionnement de l’application

Notre application se présente avec une interface graphique permettant d’afficher les différents clients ainsi que les circuits générés.



## Partie Données

La lecture des fichiers de données se fait par le menu « Données ». Il est possible de choisir les 5 fichiers de données envoyés.

## Partie Algorithme

Ici on retrouve l’initialisation des circuits pour l’algorithme Tabou ainsi que l’exécution de l’algorithme Tabou et l’algorithme Génétique.

# Explication algorithme Tabou

## Introduction

Lors de l’initialisation des circuits pour l’algorithme Tabou, ces circuits prennent tous l’entrepôt en première position ainsi qu’un client sélectionné aléatoirement. Ensuite, les circuits sont remplis (en prenant en compte la quantité maximale du camion) par des clients le plus proches possible du centre de gravité (centre recalculé à chaque nouveau client).

Sur l’interface, il faut donc penser à initialiser les circuits avant de démarrer l’algorithme Tabou. Lorsque le bouton Tabou du menu est cliqué, une boite de dialogue s’ouvre demandant la taille de la liste Tabou ainsi que le nombre maximum d’itérations voulus

## Exemple de valeurs retournées

Les exemples suivants sont exécutés avec le jeu de donnée 1 (data01)

|  |  |  |  |
| --- | --- | --- | --- |
| Taille liste tabou | Nombre d’itération | Fitness de départ (aléatoire) | Fitness optimisée |
| 5 | 500 | 1383.66 | 860.40 |
| 1481.94 | 915.43 |
| 1316.30 | 877.67 |
| 10 | 500 | 1616.85 | 914.38 |
| 1463.17 | 906.57 |
| 1590 | 881.15 |
| 10 | 1000 | 1609.83 | 893.09 |
| 1329.54 | 834.30 |
| 1320.93 | 917.86 |

PS. Lors de nos test de développement, nous n’avons pas eu de fitness inférieur à environ 805.

# Explication algorithme Génétique

## Introduction

Contrairement à l’algorithme Tabou, l’algorithme génétique n’a besoin que de l’initialisation des données. Lors du lancement de cet algorithme, le nombre d’individu ainsi que le nombre d’itération (de génération) max seront demandés. L’initialisation de cet algo se fait donc aléatoirement, en créant N circuits pour commencer.

Nous avons décidé d’utiliser un codage de nos clients à travers les différents circuits. On sait donc dans quel circuit se trouve le client.

Au début d’une génération, on ne garde que les N/2 meilleurs circuits, ce qui correspond aux circuits ayant le fitness la plus faible possible. On crée ensuite, à partie de cette moitié de population, N/2 nouveau circuit. Ces nouveaux circuits sont créés à partir de deux circuits sélectionné aléatoirement.

On génère aléatoirement un nombre compris entre 1 et le nombre de client, ce qui permet de récupérer M1 client du premier circuit puis ensuite M2 client du deuxième. Ces deux parties de client nous donne un nouveau codage de client.

La partie suivante consiste à construire correctement un nouveau circuit avec les données de codage de ses deux parents. Pour cela on vérifie la quantité à livrer, et on garde également l’information de position du client dans le circuit parent. Ceci permet de garder le client à la même place qu’un de ses parents pour tendre vers une meilleurs solution.

La dernière étape de création d’un nouveau circuit est la gestion des clients exclus. En effet, avec la contrainte de quantité par camion, certains clients n’ont pas pu être insérés dans un circuit. C’est ici la partie « mutation » de notre algorithme. Contrairement aux autres algorithmes génétiques, nos mutations se font pratiquement tous le temps, car nous avons une condition supplémentaire, la quantité maximale à livrer. Il est très rare de tomber sur un fils où tous les clients ont réussi à intégrer un circuit.

Ces différentes étapes sont répétées N/2 fois pour chaque génération. Plus le nombre de génération et la taille de la population seront élevés, plus l’algorithme tendra vers une solution optimale mais prendra plus de temps.

Il faut tout de même prendre en compte que les valeurs que nous avons eu jusqu’ici n’était pas plus optimisées qu’avec la méthode Tabou.

## Exemple de valeurs retournées

Une fois encore les exemples ci-dessous sont récupéré avec le jeu de donnée data01.

|  |  |  |  |
| --- | --- | --- | --- |
| Population | Génération | Fitness initiale (plus petite valeur générée aléatoirement | Fitness optimisée |
| 10 | 5 | 1549.31 | 1184.88 |
| 1527.63 | 1178.52 |
| 1230.76 | 1105.81 |
| 10 | 10 | 1430.20 | 1178.83 |
| 1420.69 | 1226.52 |
| 1290.02 | 1217.88 |
| 500 | 500 | 1632.61 | 1087.40 |
| 1691.67 | 1088.79 |
| 1459.14 | 1119.68 |

L’exécution pour une population de 500 avec 500 génération prend environ 30 secondes.