CAPACITATED VEHICLE ROUTING PROBLEM

Problème d'acheminement de véhicules avec capacité

Projet d'optimisation discrète

Rémi Martinez

Baptiste Aubert

Année 2021 - 2022

POLYTECH LYON

SOMMAIRE

Introduction

Le problème d'acheminement de véhicules avec capacité (également connu sous le nom de CVRP — Capacitated Vehicle Routing Problem) est l'un des nombreux problèmes d'acheminement de véhicules (VRP — Vehicle Routing Problem). Celui que nous allons traiter ici se concentre sur la création de l'itinéraire idéal pour chaque véhicule d'une flotte afin que le maximum de livraisons ait lieu et que le maximum d'espace de chargement soit utilisé.

Il y a de nombreuses variables à prendre en compte pour le problème d'optimisation. Chaque véhicule a une capacité de transport limitée et tous les clients doivent être livrés, tout en minimisant les distances parcourues par les livreurs.

Pour traiter ce problème, nous avons décidé d'utiliser deux métaheuristiques : le **recuit simulé** et **la méthode Tabou**. Nous avons mis en place un programme Java avec une interface qui permet d'afficher les différents graphes, mais aussi la possibilité de lancer plusieurs simulations consécutivement.

Vous retrouverez dans ce rapport comment nous avons modélisé le problème, généré des solutions à partir des différents algorithmes, puis analysé les résultats. En fin de rapport se trouve des explications pour lancer et tester le programme chez vous, comment utiliser l'interface graphique, et comment exporter les données des simulations au format CSV.

1. Modélisation & mise en place de la structure du code

Nous avons décidé de partir sur une structure modèle-vue-contrôleur pour ce projet.
👉 Pour le modèle nous sommes partis sur 3 classes principales pour modéliser ce problème:

- `Client.java` représente une ligne d'un fichier graph `.txt` importé. Un client regroupe comme principales informations: `numéro, positionX, positionY, quantité`
- `Vehicle.java` représente dans le problème un véhicule. Dans un véhicule on va retrouver l'itinéraire qu'il parcourt (`visit`) ainsi que la distance totale de cette itinéraire (`length`) et la quantité (`quantity`) qui ne doit pas dépasser 100 pour ce problème.
  Nous avons également redéfinit les méthodes d'ajout et de suppression de client à la tournée d'un véhicule afin que la longueur d'une `visit` soit automatiquement mise à jour en fonction du client rajouté sans avoir besoin de recalculer l'ensemble des points de la tournée.

  Pour des questions d'optimisations, nous avons choisi d'implémenter le calcul de la longueur directement à l'aide de la fonction `add()` d'un client dans un véhicule. Cela permet de ne pas avoir à recalculer la distance totale à chaque fois.
- `Graph.java` cette classe sert simplement à définir un graph complet qui va avoir une `clientList` ainsi qu'une liste de véhicules `véhicles`. Cette classe va aussi permettre de définir différentes méthodes utiles pour le graph.

👉 La vue n'a pas de classe à proprement parler car nous avons réalisé l'interface avec JavaFX. Un fichier permet de gérer tout ça, il est situé dans `src/main/resources/cvrp/cvrp.fxml`.

👉 Pour interagir avec cette interface, le contrôleur est situé dans `src/main/java/cvrp/controller/RoutingController.java`. Il permet de faire fonctionner les boutons, zoomer sur le graphe, lancer la simulation avec les classes du modèle, etc.

2. Nombre de véhicules minimum à utiliser 🚚

Nous avons au sein du constructeur de notre classe `Graph.java` le calcul du nombre minimum de véhicule pour pouvoir livrer tout les colis en prenant en compte qu'un véhicule à une capacité de 100. Nous avons recensé pour chaque graph le nombre minium de véhicules dans ce tableau et cela nous permet de comparer avec les solutions que nous obtenons après lancement des algorithmes. On compare aussi lorsqu'on rempli les véhicule à fond ou non aux lancements

Nom du graph	Nombre de clients	Nombre de véhicules minimum
A3205.txt	32	5
A3305.txt	33	5
A3306.txt	33	6
A3405.txt	34	5
A3605.txt	36	5
A3705.txt	37	5
A3706.txt	37	6
A3805.txt	38	5
A3905.txt	39	5
A3906.txt	39	6
A4406.txt	44	6
A4506.txt	45	6
A4507.txt	45	7
A4607.txt	46	7
A5307.txt	53	7
A5407.txt	54	7
A5509.txt	55	9
A6009.txt	60	9
A6109.txt	61	9
A6208.txt	62	8
A6310.txt	63	10
A6409.txt	64	9
A6509.txt	65	9
A6909.txt	69	9
A8010.txt	80	10
c101.txt	101	10

c201.txt	101	10
r101.txt	101	8

3. Générateur de solutions aléatoires 🔀

La génération aléatoire nous avions proposé deux solutions pour la générations aléatoires.

1. La première était de tirer les clients aléatoirement dans la liste et les affecter à véhicule jusqu'à ce que dernier était plein. Cette solution nous permettait bien de remplir les camions aléatoires mais nous avions toujours très peu de véhicules et cela ne correspondait pas totalement à de l'aléatoire.
2. C'est pour cela que nous avons mis une seconde solution en place dans `randomGeneration` qui consiste à choisir aléatoirement entre *Créer un nouveau véhicule ou Prendre un véhicule existant.* Le fait de randomiser cette étape de plus nous a permis d'avoir des graph avec beaucoup plus de véhicules et un affichage qui correspondait beaucoup plus à de l'aléatoire.

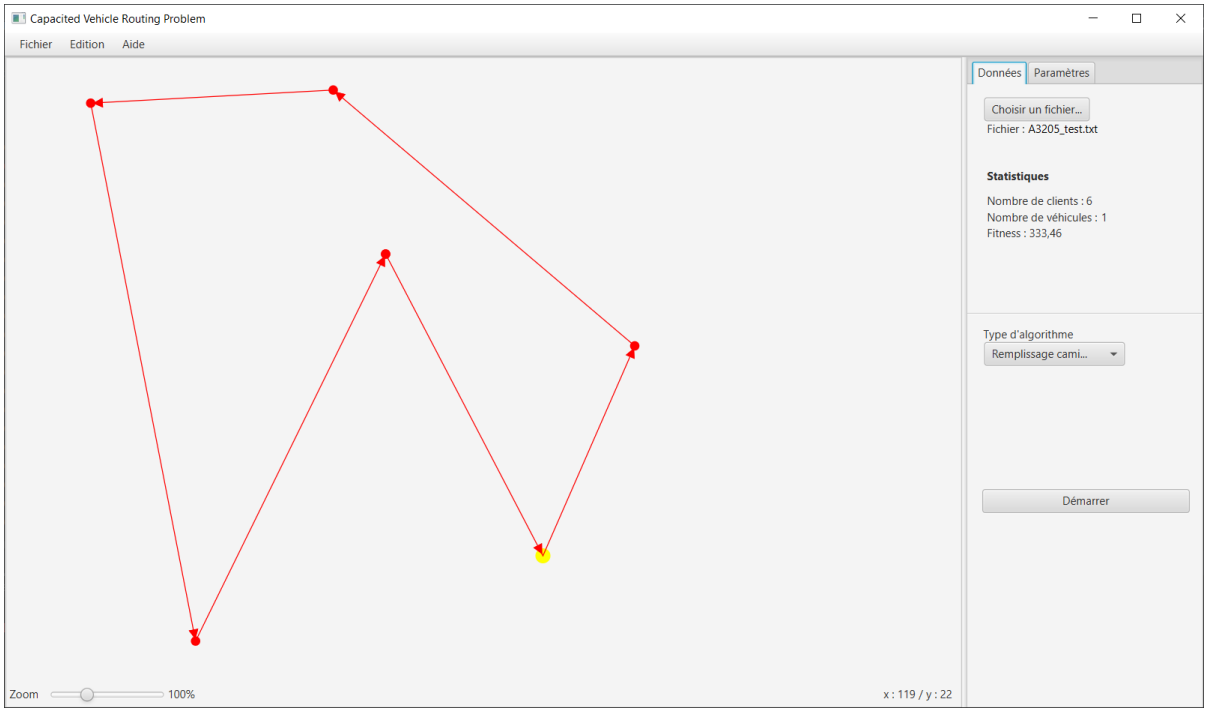
4. Création d'opérateurs de voisinage pour les méthodes à base de voisinages. 🤝

Afin d'implémenter les méthodes à base de voisinages, nous devions au préalable créer des méthodes qui permettait d'effectuer des transformations élémentaires sur le graph.

Parmi ces transformations élémentaires que nous avons vu en cours. Nous avons choisi d'en certaines qui nous on permis par la suite de générer des voisins.

Pour expliquer nous avons créer un graph de test afin de montrer l'effet des transformations élémentaires sur des véhicules et leur client.

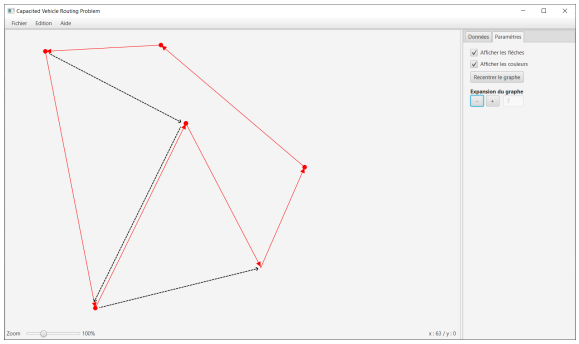
Chaque point représente un client et le point jaune est le dépôt

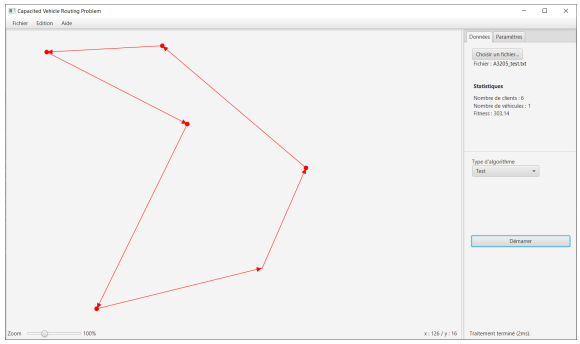


RELOCATE

La première transformation élémentaire que nous avons implémenté est le *relocate intra*. Cette transformation élémentaire permet de modifier l'emplacement d'un client dans sa tournée.

Dans notre exemple nous avons appliqué `relocate` sur le point 5 afin de le relocaliser en 4ème position.

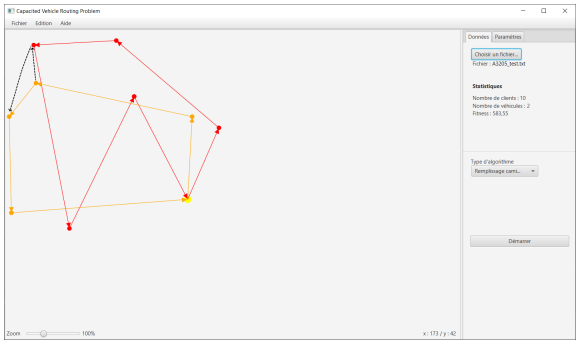




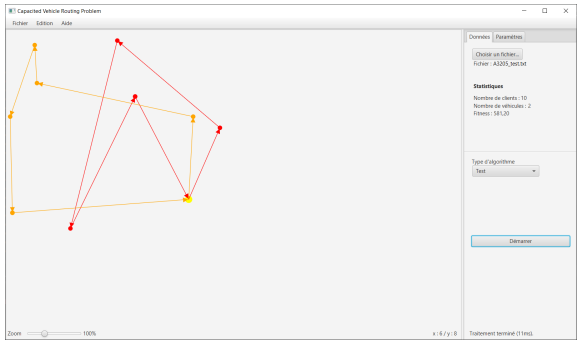
On peut voir sur la figure de droite le résultat de la tournée du véhicule après l'application de cette méthode.

Une variante de cette transformation élémentaire est le *relocate extra* qui va nous permet aussi de changer la position d'un point mais cette fois-ci afin de le déplacer dans une autre tournée.

Dans l'exemple ci-dessous nous avons appliquer un *relocate* du 3ème point du véhicule rouge pour le déplacer sur la tournée jaune en 4ème position.



Relocate extra

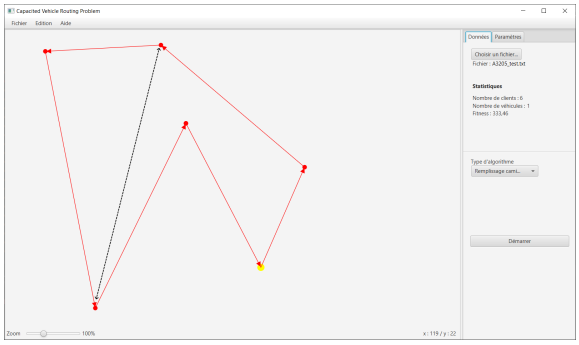


Résultat

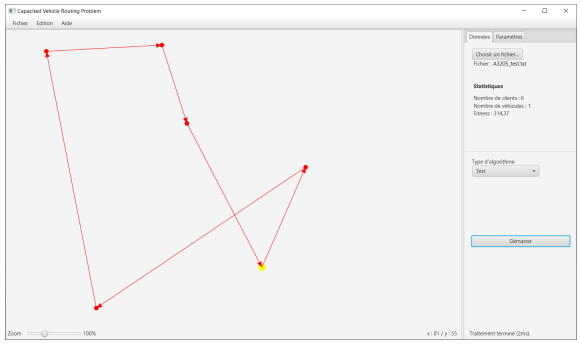
EXCHANGE

Nous avons ensuite fait la même chose pour la transformation élémentaire exchange qui permet d'échanger deux points au sein d'une même route ou deux points de deux routes différentes.

Toujours avec le même graph de départ nous avons cette fois-ci échangé la position du point 2 avec le point 4.



Exchange entre le point 2 et 4



Résultat

2-OPT

L'implémentation de Two Opt consiste à prendre au sein d'un véhicule, une arête du graph et de l'échanger avec une autre du même véhicule.

Exemple: On inverse l'arête 2-3 avec l'arête 6-7

1	2	3	4	5	6

Pour cela on change les positions des deux arêtes de sorte à les inverser

1	2	6	4	5	3

Et enfin il faut inverser l'ordre des clients entre ces deux arêtes afin de conserver un chemin correct.

1	2	6	5	4	3

5. Implémentation de deux métaheuristiques ⚙️

Après avoir défini des transformations élémentaires nous avons pu implémenter les deux métaheuristiques : Recuit simulé et la méthode Tabou.

a. Recuit simulé (Simulated Annealing) 🌡️

Le principe de cette algorithme est tel, qu'à chaque itération, nous allons réaliser aléatoirement une transformation élémentaire du Graph qui va nous permettre de calculer la différence Δf entre la fitness de notre graph et la fitness de son voisin que l'on vient de générer.

Si Δf est négatif, alors c'est que la fitness à été amélioré avec cette transformation, alors on va pouvoir conserver cet état du graph

Ce même voisin peut également être accepté avec une probabilité $e^{-\frac{\Delta_f}{T}}$

T représente la *température* que nous avons fait varier dans nos test. La température initial peut également être calculé à l'aide de `getInitialTemperature()` qui reprend l'équation $-\Delta f / ln(0.8)$.

A chaque itération, cette variation décroit de façon géométrique et de raison μ :

$$T = T \times \mu$$

A l'aide de l'algorithme nous avons extrait plusieurs fichiers CSV contenant des informations issues de l'exécution de Recuit simulé sur les données fournies.

Nous avons décidé de lancé l'exécution de recuit simulé sur la totalité des fichiers en variant certains paramètres :

- La variation μ
- La température

En croisant ces paramètres on obtient le tableau suivant des fichiers que nous avons générés.

μ = 0.5 / T = 10	μ = 0.5 / T = 50	μ = 0.5 / T = 250
μ = 0.7 / T = 10	μ = 0.7 / T = 50	μ = 0.7 / T = 250
μ = 0.9 / T = 10	μ = 0.9 / T = 50	μ = 0.9 / T = 250

b. Tabou (Tabu) 🙊

La méthode tabou a pour même objectif que Recuit de minimiser la fitness. Pour cela, l'algorithme va explorer le voisinage du graph afin de choisir la meilleure solution.

A chaque itération nous allons générer tout les voisins du graph à l'aide des différentes transformations élémentaires. Ensuite nous allons sélectionner le minimum des ces voisins en vérifiant que cette solution n'appartient pas à la liste Tabou.

Tabou peut fournir par moment des solutions moins bonnes dans le but de sortir d'un minimum local. Afin de ne pas retomber dans le même minimum local il est essentiel de conserver en mémoire la dernière solution. C'est à cela que la liste tabou va nous servir.

Nous allons donc conserver les solutions dans une `Queue` Java de taille ajustable.

Pour Tabou nous avons également fait varier un paramètre lors de différentes exécution. Nous avons choisi de faire varier la taille de liste tabou

1	10	20	30

6. Comparaison des deux métaheuristiques ⚔️

Dans cette partie, nous allons analyser plusieurs facteurs entre les deux métaheuristiques utilisées :

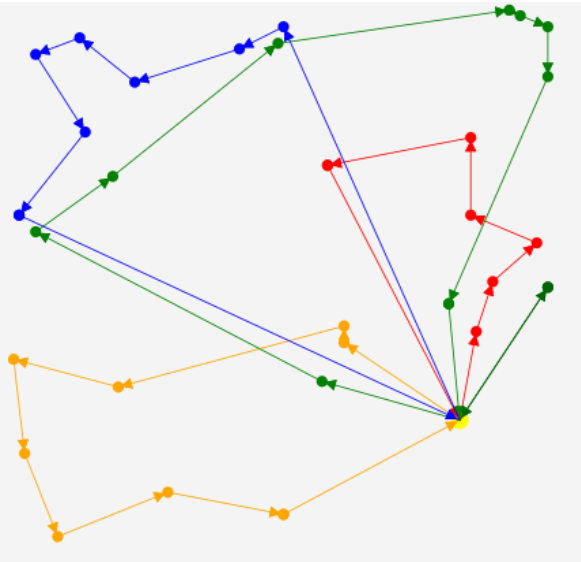
- Le temps d'exécution
- La qualité des solutions obtenues (amélioration de la fitness)
- L'évolution du nombre de véhicules minimum
- Le nombre de solutions générées en fonction des structures de voisinages utilisées
- La valeur des paramètres et leur influence $(\text{nb itérations}, \text{taille liste taboue}, \mu, t)$

Nous allons comparer les données des fichiers CSV exportés. L'analyse complète avec les données de toutes les simulations peut être retrouvée en annexe.

RECUIT SIMULÉ

Voici un échantillon de tests utilisant Recuit simulé avec :

- Nombre d'itérations = 10 000
- Variation μ = 0.9
- Température = 50



Exemple de recuit simulé sur A3205,
fitness finale = 919.4

Echantillon de tests avec Recuit simulé, μ = 0.9, t = 50

Nom fichier	Nb clients	Fitness de base	Nb vehicules min	Metaheuristique	Fitness resultat	Vehicules resultat	Nombre iterations	Temps d'execution
A3205.txt	32	2828,63	5	Recuit simule	939,43	5	10000	3062
A3305.txt	33	2105,73	5	Recuit simule	846,9	5	10000	2923
A3306.txt	33	2500,98	6	Recuit simule	921,95	6	10000	2979
A3405.txt	34	2858,45	5	Recuit simule	863,98	5	10000	2967
A3605.txt	36	2183,67	5	Recuit simule	762	5	10000	3160
A3705.txt	37	2927,17	5	Recuit simule	875,78	5	10000	3069
A3706.txt	37	2829,32	6	Recuit simule	952,75	6	10000	3233

Nom fichier	Nb clients	Fitness de base	Nb vehicules min	Metaheuristique	Fitness resultat	Vehicules resultat	Nombre iterations	Temps d'execution
A3805.txt	38	2596,03	5	Recuit simule	864,36	6	10000	3097
A3905.txt	39	2445,66	5	Recuit simule	787,39	5	10000	3214
A3906.txt	39	2569,31	6	Recuit simule	889,49	7	10000	3307
A4406.txt	44	2552,61	6	Recuit simule	935,48	6	10000	3536
A4506.txt	45	3801,41	6	Recuit simule	1048,89	7	10000	3907
A4507.txt	45	3877,82	7	Recuit simule	1312,46	8	10000	3833
A4607.txt	46	3073,31	7	Recuit simule	1116,08	7	10000	3871
A5307.txt	53	3718,02	7	Recuit simule	1191,75	7	10000	4084
A5407.txt	54	3883,51	7	Recuit simule	1315,13	9	10000	4280
A5509.txt	55	4985,81	9	Recuit simule	1582,77	10	10000	4403
A6009.txt	60	5430,43	9	Recuit simule	1597,43	9	10000	4759
A6109.txt	61	4219,37	9	Recuit simule	1234,77	11	10000	4817
A6208.txt	62	4854,83	8	Recuit simule	1366,98	8	10000	5064
A6310.txt	63	4955,3	10	Recuit simule	1831,61	10	10000	5420
A6409.txt	64	4320,98	9	Recuit simule	1410,38	9	10000	5496
A6509.txt	65	4272,53	9	Recuit simule	1288,43	9	10000	5492
A6909.txt	69	5084,48	9	Recuit simule	1424,74	10	10000	5618
A8010.txt	80	7635,77	10	Recuit simule	2032,2	10	10000	6231
c101.txt	101	5503,34	10	Recuit simule	1146,51	10	10000	6916
c201.txt	101	4987,64	10	Recuit simule	1138,89	9	10000	6862
r101.txt	101	4303,37	8	Recuit simule	1031,06	10	10000	6752

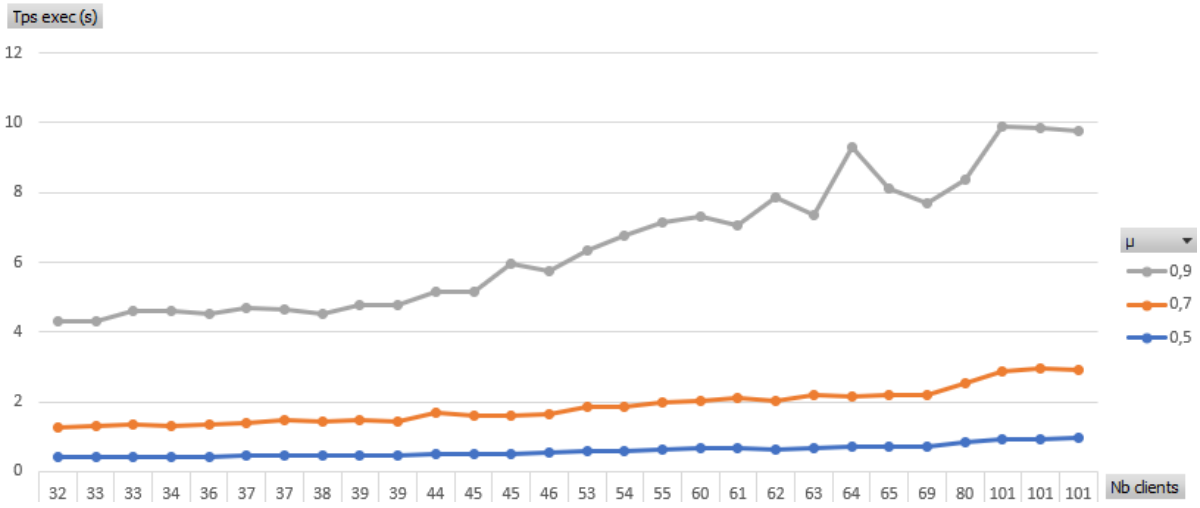
Temps d’exécution total = 122352ms soit 2 minutes et 02 secondes.

👉 Temps d’exécution

Simulation pour 10 000 itérations, température = 50

Tps exec (s)	Variation µ			
fichier	0,5	0,7	0,9	Total général
A3205.txt	0,4	0,9	3,1	4,3
A3305.txt	0,4	0,9	3,0	4,3
A3306.txt	0,4	0,9	3,2	4,6
A3405.txt	0,4	0,9	3,3	4,6
A3605.txt	0,4	0,9	3,2	4,5
A3705.txt	0,5	0,9	3,3	4,7
A3706.txt	0,4	1,0	3,2	4,6
A3805.txt	0,4	1,0	3,1	4,5
A3905.txt	0,5	1,0	3,3	4,8
A3906.txt	0,5	1,0	3,4	4,8
A4406.txt	0,5	1,2	3,5	5,2
A4506.txt	0,5	1,1	3,5	5,2
A4507.txt	0,5	1,1	4,3	6,0
A4607.txt	0,5	1,1	4,1	5,8
A5307.txt	0,6	1,3	4,5	6,4
A5407.txt	0,6	1,2	4,9	6,7
A5509.txt	0,6	1,3	5,2	7,1
A6009.txt	0,7	1,4	5,3	7,3
A6109.txt	0,7	1,4	5,0	7,1
A6208.txt	0,6	1,4	5,8	7,9
A6310.txt	0,7	1,5	5,1	7,4
A6409.txt	0,7	1,4	7,2	9,3
A6509.txt	0,7	1,5	5,9	8,1
A6909.txt	0,7	1,5	5,5	7,7
A8010.txt	0,8	1,7	5,8	8,4
c101.txt	0,9	1,9	7,0	9,9
c201.txt	0,9	2,0	6,9	9,8
r101.txt	1,0	1,9	6,8	9,7
Total général	**16,7**	**35,4**	**128,4**	**180,6**

Temps d’exécution avec recuit simulé pour 1 000 itérations et t = 50



Comparaison des temps d’exécution en fonction de la variation µ et du nombre de clients

- Temps total pour les simulations avec t = 10 : 180,58s
- Temps total pour les simulations avec t = 50 : 180,56s
- Temps total pour les simulations avec t = 250 : 180,73s
- **Somme = 541.87s soit 9 minutes et 2 secondes**

Nous n'avons pas croisé les données dans cette analyse avec la température car elle n'influe pas sur le temps d'exécution de la simulation.

👉 Nombre de véhicules — évolution

On va comparer l'évolution du nombre véhicules entre le début et la fin de la simulation.

Au départ, le nombre de véhicule minimum est défini au niveau du graphe de base. Après simulation, on récupère le nombre de véhicules du graphe. Nous avons rajouté une colonne "différence" qui représente combien de véhicules ont été ajouté par rapport au nombre minimal.

Nous avons lancé les simulations suivantes avec 10 000 itérations :

$\mu = 0.5$

Fichier	Nb vehicule min	Nb vehicule résultat	Difference	t
A3305.txt	5	5	0	10
A3306.txt	6	6	0	10
A3605.txt	5	5	0	10
A3705.txt	5	5	0	10
A3805.txt	5	5	0	10
A4406.txt	6	6	0	10
A4507.txt	7	7	0	10
A4607.txt	7	7	0	10
A5307.txt	7	7	0	10
A5509.txt	9	9	0	10
A6009.txt	9	9	0	10
A6310.txt	10	10	0	10
A6409.txt	9	9	0	10
c101.txt	10	10	0	10
c201.txt	10	10	0	10
A3205.txt	5	6	1	10
A3405.txt	5	6	1	10
A3706.txt	6	7	1	10
A3905.txt	5	6	1	10
A3906.txt	6	7	1	10
A4506.txt	6	7	1	10
A5407.txt	7	8	1	10
A6208.txt	8	9	1	10
A6109.txt	9	11	2	10
A6509.txt	9	11	2	10
A6909.txt	9	11	2	10
A8010.txt	10	12	2	10
r101.txt	8	10	2	10
A3205.txt	5	5	0	50
A3305.txt	5	5	0	50
A3405.txt	5	5	0	50
A3605.txt	5	5	0	50
A3705.txt	5	5	0	50
A3805.txt	5	5	0	50
A3905.txt	5	5	0	50
A3906.txt	6	6	0	50
A4406.txt	6	6	0	50
A4507.txt	7	7	0	50
A4607.txt	7	7	0	50
A5509.txt	9	9	0	50
A6009.txt	9	9	0	50
A6208.txt	8	8	0	50

$\mu = 0.7$

Fichier	Nb vehicule min	Nb vehicule résultat	Difference	t
A3305.txt	5	5	0	10
A3306.txt	6	6	0	10
A3405.txt	5	5	0	10
A3605.txt	5	5	0	10
A3705.txt	5	5	0	10
A3906.txt	6	6	0	10
A4406.txt	6	6	0	10
A4607.txt	7	7	0	10
A5509.txt	9	9	0	10
A6310.txt	10	10	0	10
A6509.txt	9	9	0	10
A6909.txt	9	9	0	10
c101.txt	10	10	0	10
r101.txt	8	8	0	10
A3205.txt	5	6	1	10
A3706.txt	6	7	1	10
A3805.txt	5	6	1	10
A3905.txt	5	6	1	10
A4506.txt	6	7	1	10
A4507.txt	7	8	1	10
A5307.txt	7	8	1	10
A6009.txt	9	10	1	10
A6208.txt	8	9	1	10
A6409.txt	9	10	1	10
A8010.txt	10	11	1	10
c201.txt	10	11	1	10
A5407.txt	7	9	2	10
A6109.txt	9	11	2	10
A3205.txt	5	5	0	50
A3305.txt	5	5	0	50
A3306.txt	6	6	0	50
A3605.txt	5	5	0	50
A3805.txt	5	5	0	50
A3905.txt	5	5	0	50
A3906.txt	6	6	0	50
A4406.txt	6	6	0	50
A4506.txt	6	6	0	50
A4507.txt	7	7	0	50
A4607.txt	7	7	0	50
A5307.txt	7	7	0	50
A5509.txt	9	9	0	50
A6009.txt	9	9	0	50

$\mu = 0.9$

Fichier	Nb vehicule min	Nb vehicule résultat	Difference	t
A3205.txt	5	5	0	10
A3305.txt	5	5	0	10
A3306.txt	6	6	0	10
A3405.txt	5	5	0	10
A3605.txt	5	5	0	10
A3705.txt	5	5	0	10
A3805.txt	5	5	0	10
A3906.txt	6	6	0	10
A4406.txt	6	6	0	10
A4507.txt	7	7	0	10
A4607.txt	7	7	0	10
A5509.txt	9	9	0	10
A6109.txt	9	9	0	10
A6208.txt	8	8	0	10
A6409.txt	9	9	0	10
A8010.txt	10	10	0	10
c201.txt	10	10	0	10
A3706.txt	6	7	1	10
A3905.txt	5	6	1	10
A4506.txt	6	7	1	10
A5307.txt	7	8	1	10
A5407.txt	7	8	1	10
A6009.txt	9	10	1	10
A6310.txt	10	11	1	10
A6509.txt	9	10	1	10
A6909.txt	9	10	1	10
c101.txt	10	11	1	10
r101.txt	8	10	2	10
A3205.txt	5	5	0	50
A3305.txt	5	5	0	50
A3306.txt	6	6	0	50
A3405.txt	5	5	0	50
A3605.txt	5	5	0	50
A3705.txt	5	5	0	50
A3706.txt	6	6	0	50
A3905.txt	5	5	0	50
A4406.txt	6	6	0	50
A4607.txt	7	7	0	50
A5307.txt	7	7	0	50
A6009.txt	9	9	0	50
A6208.txt	8	8	0	50
A6310.txt	10	10	0	50

A6310.txt	10	10	0	50
A6409.txt	9	9	0	50
A6909.txt	9	9	0	50
c101.txt	10	10	0	50
c201.txt	10	10	0	50
r101.txt	8	8	0	50
A3306.txt	6	7	1	50
A3706.txt	6	7	1	50
A4506.txt	6	7	1	50
A5307.txt	7	8	1	50
A5407.txt	7	8	1	50
A6109.txt	9	10	1	50
A6509.txt	9	10	1	50
A8010.txt	10	11	1	50
A3205.txt	5	5	0	250
A3305.txt	5	5	0	250
A3306.txt	6	6	0	250
A3405.txt	5	5	0	250
A3605.txt	5	5	0	250
A3705.txt	5	5	0	250
A3706.txt	6	6	0	250
A3805.txt	5	5	0	250
A3905.txt	5	5	0	250
A4507.txt	7	7	0	250
A4607.txt	7	7	0	250
A5307.txt	7	7	0	250
A6009.txt	9	9	0	250
A6208.txt	8	8	0	250
A6909.txt	9	9	0	250
A8010.txt	10	10	0	250
c201.txt	10	10	0	250
A3906.txt	6	7	1	250
A4406.txt	6	7	1	250
A4506.txt	6	7	1	250
A5407.txt	7	8	1	250
A5509.txt	9	10	1	250
A6109.txt	9	10	1	250
A6310.txt	10	11	1	250
A6409.txt	9	10	1	250
A6509.txt	9	10	1	250
c101.txt	10	11	1	250
r101.txt	8	9	1	250

A6509.txt	9	9	0	50
A3405.txt	5	6	1	50
A3705.txt	5	6	1	50
A3706.txt	6	7	1	50
A5407.txt	7	8	1	50
A6109.txt	9	10	1	50
A6310.txt	10	11	1	50
A6409.txt	9	10	1	50
A6909.txt	9	10	1	50
A8010.txt	10	11	1	50
c101.txt	10	11	1	50
c201.txt	10	11	1	50
r101.txt	8	9	1	50
A6208.txt	8	10	2	50
A3205.txt	5	5	0	250
A3305.txt	5	5	0	250
A3306.txt	6	6	0	250
A3405.txt	5	5	0	250
A3605.txt	5	5	0	250
A3705.txt	5	5	0	250
A3805.txt	5	5	0	250
A3906.txt	6	6	0	250
A4507.txt	7	7	0	250
A4607.txt	7	7	0	250
A6409.txt	9	9	0	250
A8010.txt	10	10	0	250
c101.txt	10	10	0	250
c201.txt	10	10	0	250
A3706.txt	6	7	1	250
A3905.txt	5	6	1	250
A4406.txt	6	7	1	250
A4506.txt	6	7	1	250
A5307.txt	7	8	1	250
A5407.txt	7	8	1	250
A5509.txt	9	10	1	250
A6009.txt	9	10	1	250
A6109.txt	9	10	1	250
A6208.txt	8	9	1	250
A6310.txt	10	11	1	250
A6509.txt	9	10	1	250
A6909.txt	9	10	1	250
r101.txt	8	11	3	250

A6409.txt	9	9	0	50
A6509.txt	9	9	0	50
A8010.txt	10	10	0	50
c101.txt	10	10	0	50
c201.txt	10	10	0	50
A3805.txt	5	6	1	50
A3906.txt	6	7	1	50
A4506.txt	6	7	1	50
A4507.txt	7	8	1	50
A5509.txt	9	10	1	50
A6909.txt	9	10	1	50
A5407.txt	7	9	2	50
A6109.txt	9	11	2	50
r101.txt	8	10	2	50
A3205.txt	5	5	0	250
A3305.txt	5	5	0	250
A3306.txt	6	6	0	250
A3405.txt	5	5	0	250
A3605.txt	5	5	0	250
A3705.txt	5	5	0	250
A3905.txt	5	5	0	250
A3906.txt	6	6	0	250
A4406.txt	6	6	0	250
A4506.txt	6	6	0	250
A4507.txt	7	7	0	250
A4607.txt	7	7	0	250
A5509.txt	9	9	0	250
A6009.txt	9	9	0	250
A6208.txt	8	8	0	250
A6310.txt	10	10	0	250
A6909.txt	9	9	0	250
A8010.txt	10	10	0	250
c101.txt	10	10	0	250
c201.txt	10	10	0	250
r101.txt	8	8	0	250
A3706.txt	6	7	1	250
A3805.txt	5	6	1	250
A5307.txt	7	8	1	250
A5407.txt	7	8	1	250
A6109.txt	9	10	1	250
A6409.txt	9	10	1	250
A6509.txt	9	10	1	250

Variation μ → Température t ↓	0.5	0.7	0.9	
10	0,643	0,593	0,429	0,555
50	0,286	0,500	0,429	0,405
250	0,393	0,571	0,250	0,405
Moyenne	0,440	0,548	0,369	-

On remarque que les paramètres influent peu sur l'évolution du nombre de véhicules. Voyons dans la prochaine partie si ces paramètres influent sur la fitness.

👉 Comparaison de la fitness

Comparaison fitness Recuit pour 10 000 itérations et t = 10

Aa	Nom fichier	Fitness de base	Fitness résultat	Amélioration fitness	Variation	Température
	c101.txt	5466,47	1096,74	4369,73	0.5	10
	c101.txt	6980,19	1488,14	5492,05	0.7	10
	c101.txt	5991,27	1353,04	4638,23	0.9	10
	c201.txt	6521,97	1433,26	5088,7	0.5	10
	c201.txt	5194,22	1230,57	3963,64	0.7	10
	c201.txt	7163,57	1430,94	5732,62	0.9	10
	r101.txt	5121,46	1041,88	4079,57	0.5	10
	r101.txt	4750,03	951,42	3798,61	0.7	10
	r101.txt	5381,61	1015,93	4365,68	0.9	10

	Fitness de base	Fitness résultat	Amélioration fitness
Moyenne	5841,20	1226,88	4614,31
Ecart-type	817,23	194,35	644,89

Variation µ	Moyenne amélioration fitness
0.5	4512,67
0.7	4418,100
0.9	4912,18

Comparaison fitness Recuit pour 10 000 itérations et t = 50

Aa	Fichier	Fitness de base	Fitness résultat	Amélioration fitness	Variation	Température
	c101.txt	7331,18	1503,74	5827,44	0.5	50
	c101.txt	5459,01	1095,23	4363,77	0.7	50
	c101.txt	5503,34	1146,51	4356,83	0.9	50
	c201.txt	5774,6	1253,42	4521,18	0.5	50

Aa	Fichier	Fitness de base	Fitness résultat	Amélioration fitness	Variation	Température
	c201.txt	7467,89	1426,3	6041,59	0.7	50
	c201.txt	4987,64	1138,89	3848,75	0.9	50
	r101.txt	5812,13	1202,66	4609,47	0.5	50
	r101.txt	5200,78	1160,21	4040,57	0.7	50
	r101.txt	4303,37	1031,06	3272,31	0.9	50

	Fitness de base	Fitness résultat	Amélioration fitness
Moyenne	5759,99	1217,56	4542,43
Ecart-type	976,16	145,77	837,74

Variation µ	Moyenne amélioration fitness
0.5	4986,03
0.7	4815,31
0.9	3825,96

Comparaison fitness Recuit pour 10 000 itérations et t = 250

Aa	Fichier	Fitness de base	Fitness résultat	Amélioration fitness	Variation	Température
	c101.txt	4960,42	1080,3	3880,12	0.5	250
	c101.txt	6520,49	1293,73	5226,76	0.7	250
	c101.txt	5435,54	1162,68	4272,86	0.9	250
	c201.txt	7454,7	1560,17	5894,53	0.5	250
	c201.txt	5343,4	1175,71	4167,68	0.7	250
	c201.txt	6787,18	1312,04	5475,14	0.9	250
	r101.txt	5476,67	1012,4	4464,27	0.5	250
	r101.txt	3943,48	891,53	3051,95	0.7	250
	r101.txt	4686,54	956	3730,53	0.9	250

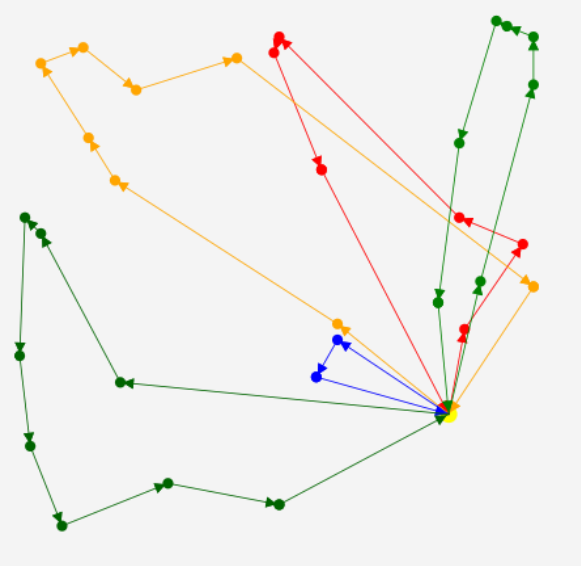
	Fitness de base	Fitness résultat	Amélioration fitness
Moyenne	5623,16	1160,51	4462,65
Ecart-type	1042,83	195,28	859,44

Variation μ	Moyenne amélioration fitness
0.5	4746,31
0.7	4148,80
0.9	4492,84

TABOU

Voici un échantillon de tests utilisant Tabou avec :

- Nombre d'itérations = 10 000
- Taille de la liste tabou = 1



Exemple de Tabou sur A3205, fitness finale = 871.4

Echantillon de tests avec Tabou, nb.itér = 10 000, taille liste = 1

Nom fichier	Nb clients	Fitness de base	Nb vehicules min	Metaheuristique	Fitness resultat	Vehicules resultat	Nombre iterations	Temps d'execution
A3205.txt	32	2660.91	5	Tabou	827.84	7	10000	10114
A3305.txt	33	2467.34	5	Tabou	758.13	6	10000	8887
A3306.txt	33	1935.48	6	Tabou	724.33	6	10000	10529
A3405.txt	34	2356.09	5	Tabou	874.12	6	10000	11785
A3605.txt	36	2343.4	5	Tabou	875.54	7	10000	11342
A3705.txt	37	2461.65	5	Tabou	778.75	5	10000	20589
A3706.txt	37	2465.11	6	Tabou	1071.21	7	10000	15340
A3805.txt	38	3042.28	5	Tabou	928.68	6	10000	16825
A3905.txt	39	2890.27	5	Tabou	844.89	5	10000	13107
A3906.txt	39	2720.48	6	Tabou	857.52	6	10000	23313

Nom fichier	Nb clients	Fitness de base	Nb vehicules min	Metaheuristique	Fitness resultat	Vehicules resultat	Nombre iterations	Temps d'execution
A4406.txt	44	3028.64	6	Tabou	1038.34	6	10000	19344
A4506.txt	45	3382.5	6	Tabou	1070.63	8	10000	26598
A4507.txt	45	4008.46	7	Tabou	1341.45	8	10000	18009
A4607.txt	46	3301.48	7	Tabou	1204.16	7	10000	20310
A5307.txt	53	3604.99	7	Tabou	980	8	10000	36661
A5407.txt	54	4635.46	7	Tabou	1322.54	8	10000	27515
A5509.txt	55	3710.26	9	Tabou	1315.86	10	10000	29189
A6009.txt	60	3961.2	9	Tabou	1317.25	9	10000	107172
A6109.txt	61	4147.71	9	Tabou	1319.84	13	10000	68284
A6208.txt	62	5467.44	8	Tabou	1498.62	10	10000	96772
A6310.txt	63	4756.89	10	Tabou	1791.68	10	10000	61322
A6409.txt	64	3664.89	9	Tabou	1212.61	10	10000	60478
A6509.txt	65	4307.97	9	Tabou	1358.08	10	10000	43421
A6909.txt	69	5407.65	9	Tabou	1508.23	9	10000	67733
A8010.txt	80	6396.56	10	Tabou	1649.28	11	10000	99426
c101.txt	101	7724.61	10	Tabou	1630.22	11	10000	259570
c201.txt	101	5778.26	10	Tabou	1200.45	10	10000	161251
r101.txt	101	5656.51	8	Tabou	1051.33	9	10000	261736

Temps d’exécution total = 1606622ms soit 26 minutes 46 secondes.

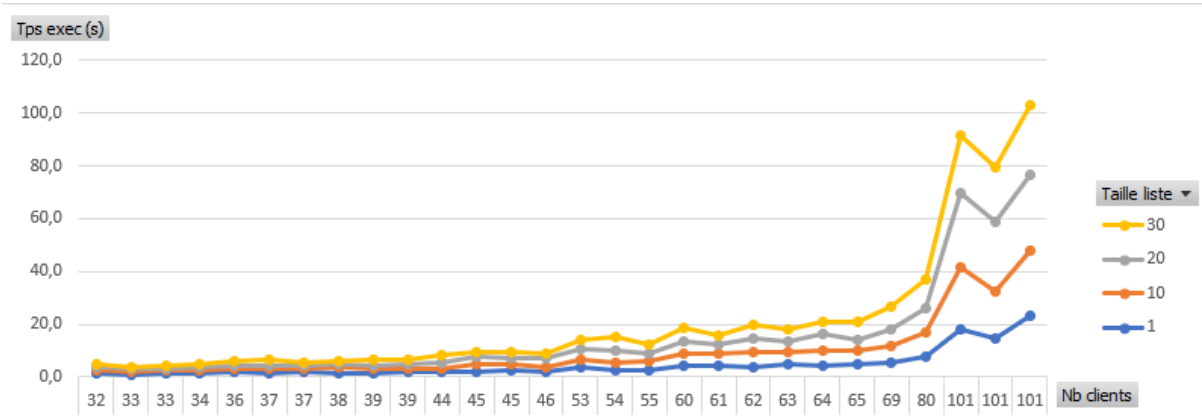
👉 Temps d’exécution

Tps exec (s)	Taille liste				
Fichier	1	10	20	30	Total général
A3205.txt	1,3	1,0	1,3	1,1	4,7
A3305.txt	1,1	0,9	0,9	0,9	3,7
A3306.txt	1,1	1,2	1,0	0,9	4,2
A3405.txt	1,3	1,1	1,1	1,4	4,9
A3605.txt	1,8	1,3	1,4	1,6	6,1
A3705.txt	1,5	1,5	1,4	2,1	6,4
A3706.txt	1,7	1,1	1,2	1,5	5,6
A3805.txt	1,7	1,8	1,3	1,3	6,0
A3905.txt	1,6	1,5	1,4	2,1	6,6
A3906.txt	1,7	1,4	1,7	1,6	6,4
A4406.txt	1,7	1,5	2,5	2,4	8,1
A4506.txt	2,1	2,4	3,0	2,1	9,7
A4507.txt	2,6	2,4	2,1	2,2	9,3
A4607.txt	2,1	1,7	3,2	1,9	9,0
A5307.txt	3,7	3,0	3,8	3,8	14,2
A5407.txt	2,7	3,0	4,3	5,3	15,3
A5509.txt	2,6	3,3	3,0	3,2	12,2
A6009.txt	4,4	4,2	4,6	5,3	18,5
A6109.txt	4,4	4,2	3,6	3,7	15,9
A6208.txt	4,0	5,7	4,8	5,5	19,9
A6310.txt	4,8	4,4	4,4	4,5	18,1
A6409.txt	4,4	5,7	6,5	4,3	20,8
A6509.txt	4,7	5,3	4,1	7,1	21,2
A6909.txt	5,3	6,6	6,1	8,4	26,5
A8010.txt	7,9	8,7	9,5	10,9	37,1
c101.txt	17,8	23,9	28,3	21,3	91,3
c201.txt	14,7	17,9	26,2	20,5	79,4
r101.txt	23,2	25,0	28,6	26,2	103,0
Total général	**127,7**	**141,7**	**161,4**	**153,1**	**583,9**

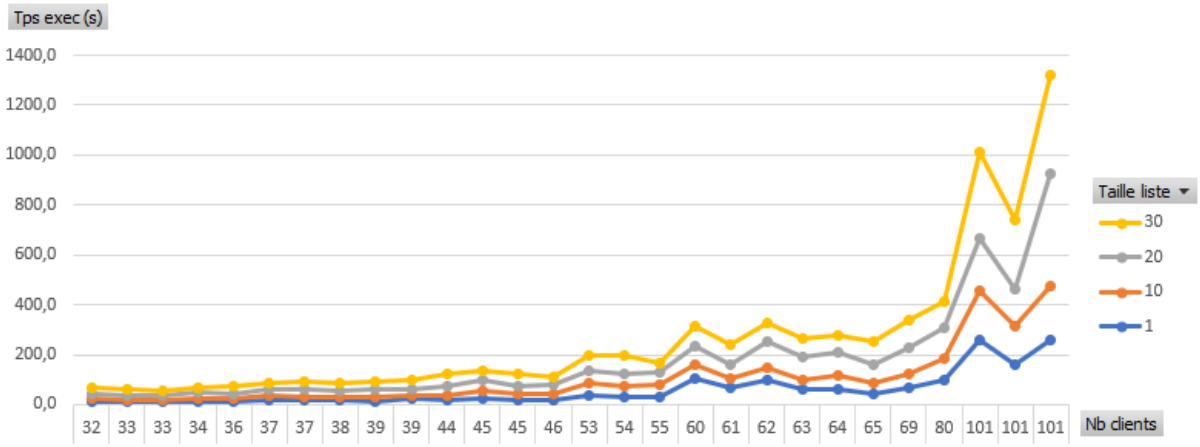
Temps d’exécution avec Tabou pour 1 000 itérations

Tps exec (s)	Taille liste				
Fichier	1	10	20	30	Total général
A3205.txt	10,1	11,3	19,6	25,0	66,1
A3305.txt	8,9	10,6	14,7	24,5	58,7
A3306.txt	10,5	8,9	15,7	18,3	53,4
A3405.txt	11,8	12,3	22,4	19,3	65,7
A3605.txt	11,3	11,6	18,5	30,0	71,5
A3705.txt	20,6	13,6	27,5	23,4	85,1
A3706.txt	15,3	17,0	27,4	29,8	89,5
A3805.txt	16,8	12,2	25,6	28,4	83,1
A3905.txt	13,1	16,9	28,8	36,1	94,9
A3906.txt	23,3	13,9	25,7	34,7	97,6
A4406.txt	19,3	18,1	38,6	44,2	120,3
A4506.txt	26,6	29,7	39,7	41,8	137,8
A4507.txt	18,0	22,0	36,3	49,3	125,6
A4607.txt	20,3	23,9	38,3	29,9	112,4
A5307.txt	36,7	48,4	49,9	62,0	197,0
A5407.txt	27,5	47,5	48,3	71,8	195,1
A5509.txt	29,2	48,6	50,5	35,1	163,4
A6009.txt	107,2	52,3	73,6	84,2	317,3
A6109.txt	68,3	35,6	57,9	75,7	237,5
A6208.txt	96,8	50,1	105,0	77,0	328,9
A6310.txt	61,3	38,5	91,8	75,1	266,7
A6409.txt	60,5	59,0	91,0	64,1	274,5
A6509.txt	43,4	43,0	71,1	92,6	250,2
A6909.txt	67,7	52,9	110,2	105,5	336,3
A8010.txt	99,4	88,3	122,0	101,8	411,5
c101.txt	259,6	195,0	211,7	346,5	1012,8
c201.txt	161,3	155,2	144,3	279,0	739,8
r101.txt	261,7	215,4	449,8	393,0	1319,9
Total général	**1606,6**	**1351,7**	**2056,3**	**2297,9**	**7312,5**

Temps d’exécution avec Tabou pour 10 000 itérations



Comparaison des temps d’exéc. pour 1 000 itérations en fonction de la taille de la liste et du nombre de clients



Comparaison des temps d'exéc. pour 10 000 itérations en fonction de la taille de la liste et du nombre de clients

- Temps total pour 1 000 itérations : 583.9
- Temps total pour 10 000 itérations : 7312.5
- **Somme = 7896.4s soit 2 heures, 11 minutes et 36 secondes**

Le nombre d'itérations influe **énormément** sur le temps d'exécution, ce qui est logique.

👉 Nombre de véhicules — évolution

On va comparer l'évolution du nombre véhicules entre le début et la fin de la simulation.

Au départ, le nombre de véhicule minimum est défini au niveau du graphe de base. Après simulation, on récupère le nombre de véhicules du graphe. Nous avons rajouté une colonne "différence" qui représente combien de véhicules ont été ajouté par rapport au nombre minimal.

Nous avons lancé les simulations suivantes :

1 000 itérations

Fichier	Nb vehicule min	Nb vehicule résultat	Difference	Taille liste
A3306.txt	6	6	0	1
A3705.txt	5	5	0	1
A3905.txt	5	5	0	1
A3906.txt	6	6	0	1
A4406.txt	6	6	0	1
A4607.txt	7	7	0	1
A6009.txt	9	9	0	1
A6310.txt	10	10	0	1
A6909.txt	9	9	0	1
c201.txt	10	10	0	1
A3205.txt	5	5	0	10
A3306.txt	6	6	0	10
A3605.txt	5	5	0	10
A3905.txt	5	5	0	10
A3906.txt	6	6	0	10
A6109.txt	9	9	0	10
c201.txt	10	10	0	10
r101.txt	8	8	0	10
A3305.txt	5	5	0	20
A3306.txt	6	6	0	20
A3705.txt	5	5	0	20
A3906.txt	6	6	0	20
A5307.txt	7	7	0	20
A5509.txt	9	9	0	20
c101.txt	10	10	0	20
A3205.txt	5	5	0	30
A3306.txt	6	6	0	30
A3605.txt	5	5	0	30
A3906.txt	6	6	0	30
A4406.txt	6	6	0	30
A4607.txt	7	7	0	30
A5509.txt	9	9	0	30
A6109.txt	9	9	0	30
A6310.txt	10	10	0	30
A6909.txt	9	9	0	30
A3305.txt	5	6	1	1
A3405.txt	5	6	1	1
A3706.txt	6	7	1	1
A3805.txt	5	6	1	1
A4507.txt	7	8	1	1
A5307.txt	7	8	1	1
A5407.txt	7	8	1	1

10 000 itérations

Fichier	Nb vehicule min	Nb vehicule résultat	Difference	Taille liste
A3405.txt	5	5	0	1
A3605.txt	5	5	0	1
A3705.txt	5	5	0	1
A3905.txt	5	5	0	1
A6009.txt	9	9	0	1
c101.txt	10	10	0	1
c201.txt	10	10	0	1
A3306.txt	6	6	0	10
A3705.txt	5	5	0	10
A4607.txt	7	7	0	10
A5307.txt	7	7	0	10
A5509.txt	9	9	0	10
c201.txt	10	10	0	10
A3605.txt	5	5	0	20
A3705.txt	5	5	0	20
A3805.txt	5	5	0	20
A3905.txt	5	5	0	20
A5307.txt	7	7	0	20
A5509.txt	9	9	0	20
A6909.txt	9	9	0	20
c201.txt	10	10	0	20
A3705.txt	5	5	0	30
A5407.txt	7	7	0	30
A3205.txt	5	6	1	1
A3305.txt	5	6	1	1
A3306.txt	6	7	1	1
A3706.txt	6	7	1	1
A3906.txt	6	7	1	1
A4406.txt	6	7	1	1
A4506.txt	6	7	1	1
A4607.txt	7	8	1	1
A6109.txt	9	10	1	1
A6208.txt	8	9	1	1
A6310.txt	10	11	1	1
A6409.txt	9	10	1	1
A6509.txt	9	10	1	1
r101.txt	8	9	1	1
A3205.txt	5	6	1	10
A3305.txt	5	6	1	10
A3605.txt	5	6	1	10
A3805.txt	5	6	1	10
A3906.txt	6	7	1	10

A5509.txt	9	10	1	1
A6409.txt	9	10	1	1
A6509.txt	9	10	1	1
A8010.txt	10	11	1	1
c101.txt	10	11	1	1
r101.txt	8	9	1	1
A3305.txt	5	6	1	10
A3706.txt	6	7	1	10
A3805.txt	5	6	1	10
A4607.txt	7	8	1	10
A5407.txt	7	8	1	10
A5509.txt	9	10	1	10
A6509.txt	9	10	1	10
A6909.txt	9	10	1	10
A3205.txt	5	6	1	20
A3405.txt	5	6	1	20
A3605.txt	5	6	1	20
A3706.txt	6	7	1	20
A3805.txt	5	6	1	20
A4507.txt	7	8	1	20
A4607.txt	7	8	1	20
A6009.txt	9	10	1	20
A6109.txt	9	10	1	20
A6208.txt	8	9	1	20
A6509.txt	9	10	1	20
c201.txt	10	11	1	20
r101.txt	8	9	1	20
A3305.txt	5	6	1	30
A3705.txt	5	6	1	30
A3706.txt	6	7	1	30
A3805.txt	5	6	1	30
A5307.txt	7	8	1	30
A5407.txt	7	8	1	30
A6009.txt	9	10	1	30
A6208.txt	8	9	1	30
A6409.txt	9	10	1	30
c201.txt	10	11	1	30
r101.txt	8	9	1	30
A3205.txt	5	7	2	1
A3605.txt	5	7	2	1
A4506.txt	6	8	2	1
A6208.txt	8	10	2	1
A3405.txt	5	7	2	10
A3705.txt	5	7	2	10
A4406.txt	6	8	2	10
A4507.txt	7	9	2	10
A6009.txt	9	11	2	10
A6208.txt	8	10	2	10
A6409.txt	9	11	2	10
A8010.txt	10	12	2	10

A4406.txt	6	7	1	10
A4506.txt	6	7	1	10
A5407.txt	7	8	1	10
A6009.txt	9	10	1	10
A6310.txt	10	11	1	10
A6409.txt	9	10	1	10
A6509.txt	9	10	1	10
A6909.txt	9	10	1	10
A3205.txt	5	6	1	20
A3305.txt	5	6	1	20
A3306.txt	6	7	1	20
A3706.txt	6	7	1	20
A3906.txt	6	7	1	20
A4507.txt	7	8	1	20
A4607.txt	7	8	1	20
A6109.txt	9	10	1	20
A6208.txt	8	9	1	20
A6310.txt	10	11	1	20
A6509.txt	9	10	1	20
c101.txt	10	11	1	20
A3205.txt	5	6	1	30
A3305.txt	5	6	1	30
A3306.txt	6	7	1	30
A3405.txt	5	6	1	30
A3605.txt	5	6	1	30
A3905.txt	5	6	1	30
A3906.txt	6	7	1	30
A4406.txt	6	7	1	30
A4506.txt	6	7	1	30
A4507.txt	7	8	1	30
A4607.txt	7	8	1	30
A5509.txt	9	10	1	30
A6009.txt	9	10	1	30
A6208.txt	8	9	1	30
A6310.txt	10	11	1	30
A6909.txt	9	10	1	30
c201.txt	10	11	1	30
A3805.txt	5	7	2	1
A4507.txt	7	9	2	1
A5307.txt	7	9	2	1
A5407.txt	7	9	2	1
A5509.txt	9	11	2	1
A8010.txt	10	12	2	1
A3405.txt	5	7	2	10
A3706.txt	6	8	2	10
A3905.txt	5	7	2	10
A4507.txt	7	9	2	10
A6109.txt	9	11	2	10
A6208.txt	8	10	2	10
A8010.txt	10	12	2	10

c101.txt	10	12	2	10
A3905.txt	5	7	2	20
A4406.txt	6	8	2	20
A4506.txt	6	8	2	20
A5407.txt	7	9	2	20
A6409.txt	9	11	2	20
A8010.txt	10	12	2	20
A3405.txt	5	7	2	30
A3905.txt	5	7	2	30
A4506.txt	6	8	2	30
A4507.txt	7	9	2	30
A6509.txt	9	11	2	30
c101.txt	10	12	2	30
A4506.txt	6	9	3	10
A5307.txt	7	10	3	10
A6310.txt	10	13	3	10
A6310.txt	10	13	3	20
A6909.txt	9	12	3	20
A8010.txt	10	13	3	30
A6109.txt	9	13	4	1

c101.txt	10	12	2	10
r101.txt	8	10	2	10
A3405.txt	5	7	2	20
A4406.txt	6	8	2	20
A4506.txt	6	8	2	20
A6009.txt	9	11	2	20
A6409.txt	9	11	2	20
r101.txt	8	10	2	20
A3706.txt	6	8	2	30
A5307.txt	7	9	2	30
A6409.txt	9	11	2	30
A6509.txt	9	11	2	30
A8010.txt	10	12	2	30
c101.txt	10	12	2	30
A6909.txt	9	12	3	1
A5407.txt	7	10	3	20
A8010.txt	10	13	3	20
A3805.txt	5	8	3	30
A6109.txt	9	12	3	30
r101.txt	8	11	3	30

Itérations → Taille liste ↓	1 000	10 000	Moyenne
1	0.893	1.036	0.965
10	1.250	1.107	1.179
20	1.107	1.071	1.089
30	0.929	1.357	1.1430
Moyenne	1.0448	1.1427	-

On remarque que le nombre d'itérations influe sur le résultat du nombre de véhicule ajouté, contrairement à la taille de la liste tabou qui n'affecte pas nécessairement ce nombre (du moins, on ne peut rien en conclure ici.)

On ajoute plus de véhicule par rapport au nombre de départ avec un plus grand nombre d'itérations. Voyons dans la prochaine partie si elle influe sur la fitness.

👉 Comparaison taille liste tabou ↔ fitness

Comparaison fitness Tabou pour 1 000 itérations

Aa Title	Fichier	# Fitness de base	# Fitness résultat	# Amélioration fitness	# Taille liste tabou
	c101.txt	5044.08	1175.87	3868.21	1
	c101.txt	4868.78	1113.95	3754.84	10
	c101.txt	7043.04	1582.95	5460.09	20

Aa Title	Fichier	Fitness de base	Fitness résultat	Amélioration fitness	Taille liste tabou
	c101.txt	4338.54	1203.94	3134.59	30
	c201.txt	5252.56	1107.22	4145.34	1
	c201.txt	6509.65	1519.96	4989.69	10
	c201.txt	5136.67	1150.03	3986.64	20
	c201.txt	5524.57	1258.59	4265.98	30
	r101.txt	5252.56	1107.22	4145.34	1
	r101.txt	6509.65	1519.96	4989.69	10
	r101.txt	5136.67	1150.03	3986.64	20
	r101.txt	5524.57	1258.59	4265.98	30

	Fitness de base	Fitness résultat	Amélioration fitness
Moyenne	5511.78	1262.36	4249.42
Ecart-type	750,61	168,77	602,41

Taille liste tabou	Moyenne amélioration fitness
1	4052,96
10	4578,07
20	4477,79
30	3888,85

Comparaison fitness Tabou pour 10 000 itérations

Aa	Fichier	Fitness base	Fitness résultat	Amélioration fitness	Taille liste tabou
	c101.txt	7724.61	1630.22	6094.4	1
	c101.txt	6759.96	1564.01	5195.95	10
	c101.txt	5950.77	1243.8	4706.97	20
	c101.txt	6844.72	1702.48	5142.24	30
	c201.txt	5778.26	1200.45	4577.81	1
	c201.txt	7181.93	1499.37	5682.56	10
	c201.txt	6685.66	1357.81	5327.85	20

Aa	Fichier	Fitness base	Fitness résultat	Amélioration fitness	Taille liste tabou
	c201.txt	6058.55	1362.53	4696.02	30
	r101.txt	5778.26	1200.45	4577.81	1
	r101.txt	7181.93	1499.37	5682.56	10
	r101.txt	6685.66	1357.81	5327.85	20
	r101.txt	6058.55	1362.53	4696.02	30

	Fitness de base	Fitness résultat	Amélioration fitness
Moyenne	6557.41	1415.07	5142.34
Ecart-type	604.07	157.80	481.93

Taille liste tabou	Moyenne amélioration fitness
1	5083,34
10	5520,36
20	5120,89
30	4844,76

On peut remarquer sans surprise que pour 10 000 itérations par rapport à 1 000 itérations :

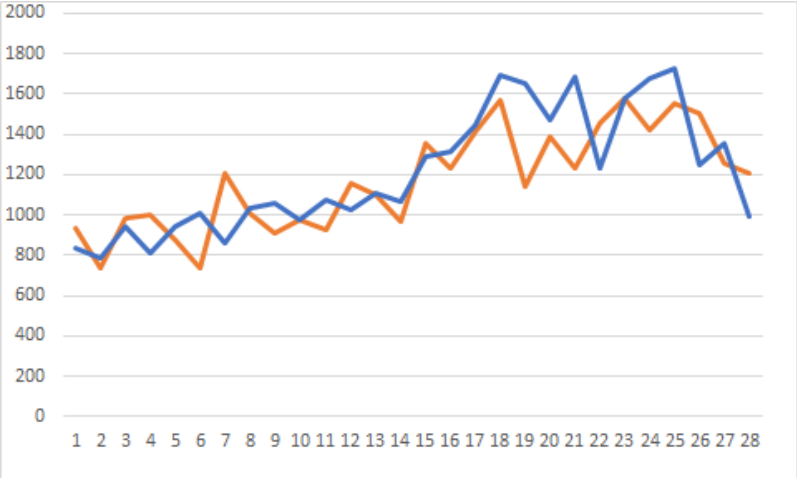
- L'amélioration de la fitness est plus élevée
- L'écart-type est moins grand : il y a moins de disparités entre les résultats

On remarque aussi que pour une taille de liste Tabou de 10, on obtient de meilleurs résultats.

On peut aussi conclure que le nombre de véhicules ajouté par rapport au nombre minimum n'influe pas forcément sur le choix de la meilleure solution.

TABOU & RECUIT SIMULÉ

Sur ce graph nous avons choisi de comparer les résultats que nous avons entre Tabou_10000_20 et Recuit_simulé_10000_0,9_50



TABOU
Liste de la taille = 20

RECUIT
Variation = 0.9
Température = 50

Fitness finale pour chaque graphe (de 1 à 28) et de l'algorithme choisi

Nous pouvons remarquer certaines fois nous n'arrivons pas toujours à obtenir les meilleurs résultats car les deux valeurs sont rarement égales cependant l'écart entre les deux algorithmes et souvent moindres ce qui signifie que les résultats sont tout de mêmes très significatif.
Sur ces deux lancements il n'y a pas un algorithme qui se distingue particulièrement. Cependant on remarque pour les fichiers de grande taille (n°>18) que Recuit a tendance à donner de meilleurs résultats.

Pour comparer les temps d'exécutions entre ces deux algorithmes, il ne sont pas vraiment comparables car on a des ordres de grandeurs totalement différents.

Pour lancer toutes les simulations de Tabou cela a pris plus de 2 heures contre 9 minutes pour recuit simulé.

Lancement du programme ▶️

Nous avons réalisé ce projet en utilisant l'IDE IntelliJ IDEA Ultimate 2021.2.3.

La version de Java utilisée est la 18 : **openjdk-18.0.1**

Le projet a été initialisé avec Maven pour la gestion des dépendances : voir fichier `pom.xml`

Dépendances :

- JavaFX 17.0.2 pour l'interface graphique, compatible avec Mac
- OpenCSV 5.6 pour réaliser plus simplement les exports en fichier CSV

Interface graphique 💻

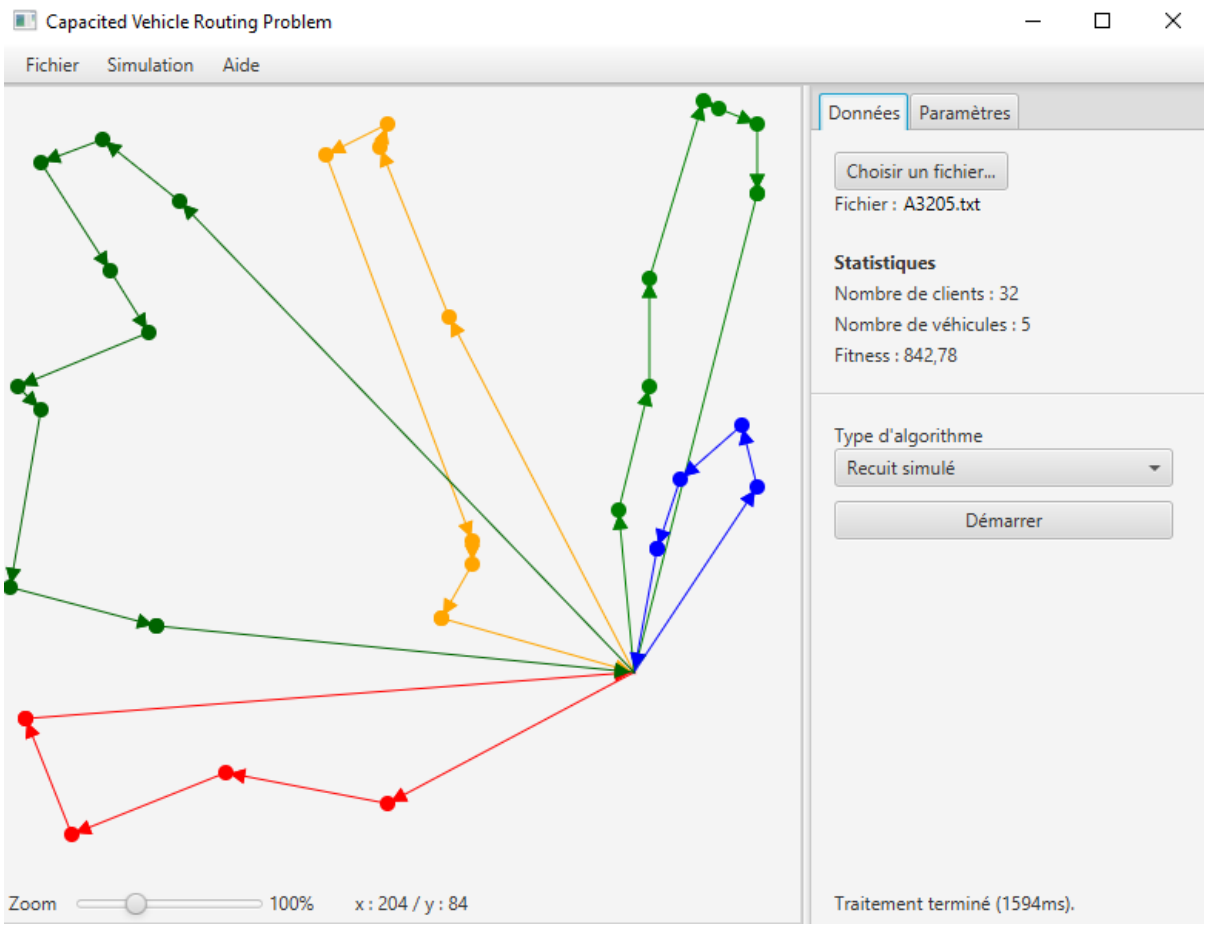
Nous avons construit l'interface graphique avec JavaFX et SceneBuilder.

Pour tester l'interface graphique, vous pouvez passer la variable suivante à `false` dans le fichier principal `App.java` :

```
private static final boolean AUTOSTART_SIMULATIONS = true;
```

Cela fera en sorte de ne pas lancer les simulations vers les fichiers CSV au lancement de l'application et l'interface graphique se lancera.

Cette interface nous a permis d'illustrer nos graphes afin pouvoir les inspecter autrement que par des lignes dans une console.



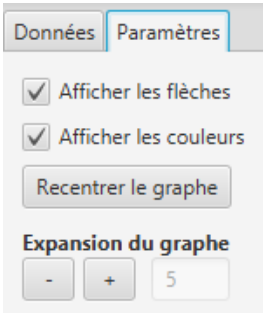
Fonctionnement :

1. Importer un graphe avec le bouton Choisir un fichier
2. Choisir un type d'algorithme dans la liste déroulante
3. Démarrer la simulation

Autres fonctionnalités :

- Vous pouvez zoomer / dézoomer sur le graphe avec la molette de la souris ou avec la barre de zoom en bas à gauche
- Vous pouvez vous déplacer librement dans le graphe avec la souris
- Les coordonnées de la souris dans le graphe sont écrites en bas à gauche
- Il est possible de survoler un client pour voir ses coordonnées :

- Dans l'onglet paramètres vous avez plusieurs choix d'options :



- Il est possible d'afficher / masquer les flèches du graphe
- Il est possible d'afficher le graphe en noir et blanc ou alors les couleurs
- Utilisez le bouton Recentrer le graphe si vous l'avez déplacé trop loin par erreur
- L'expansion du graphe permet de "l'écarter" pour une meilleure lisibilité

Export en fichier CSV 📤

Lorsque que le programme est lancé avec `AUTOSTART_SIMULATIONS = true;` toutes les simulations dans la fonctions `main` vont se lancer consécutivement.

Fichier exporté

Pendant le lancement de ces simulations, un fichier CSV est créé sous le répertoire `/files/exports`.

Il respecte la nomenclature suivante :

`Nom algorithme_nbIterations_paramètre1_paramètre2.csv`

Les paramètres sont facultatifs et varient selon l'algorithme utilisé.

Le séparateur utilisé est la virgule (`,`) et les décimaux sont écrits sous la forme `#.##`

Exemples :

- Tabou lancé avec 10 000 itérations et une taille de liste tabou à 1 se nommera :
  Tabou_10000_1.csv

- Recuit simulé lancé avec 10 000 itérations, une variation à 0.9 et une température de 10 se nommera :
  Recuit simule_10000_0.9_10.csv

Colonnes du fichier CSV :

- Nom fichier = Le nom du graphe importé (A3205.txt)
- Nb clients = Le nombre de clients (nœuds) du graphe
- Fitness de base = La fitness (distance totale de tous les véhicules) avant simulation
- Nb vehicules min = Le nombre de véhicule minimum avant simulation
- Metaheuristique = Algorithme utilisé (ici Tabou ou Recuit simulé)
- Fitness resultat = La fitness après simulation
- Vehicules resultat = Le nombre de véhicule minimum après simulation
- Nombre iterations = Le nombre d'itérations à exécuter
- Temps execution = Temps d'exécution en utilisant l'algorithme choisi avec ce graphe (en ms)
- Amélioration fitness = différence entre la fitness de base et la fitness résultat ($f_{base} - f_{resultat}$)
- Liste tabou (Tabou uniquement) = Taille de la liste tabou passée en paramètre
- Variation μ (Recuit simulé uniquement) = Variation du recuit passée en paramètre
- Température (Recuit simulé) = Température du recuit passée en paramètre

Fonctionnement

La classe `CsvExporter` gère la création du fichier. Il stocke l'algorithme utilisé et tous ses paramètres.

- Sa méthode `createCSV()` doit être appelé en premier, il s'occupe de créer le fichier CSV vierge en remplissant uniquement les colonnes (première ligne du fichier). Il va retourner un `CSVWriter` (qui provient de OpenCSV). Ce writer va être utilisé après pour pointer vers la ligne suivante du fichier à écrire.
- Sa méthode `writeCSV(csvData, writer)` permet de remplir la ligne actuelle du fichier CSV avec les données passées en paramètre et de pointer sur celle d'après.

La classe `CsvData` gère les données qui peuvent être passées au `CsvExporter` afin de les écrire dans le fichier CSV.

Quand on lance une simulation, on peut voir apparaître dans la console le temps d'exécution et où en est la simulation. Ceci était très pratique pour voir le temps d'exécution de chaque graphe, qui est ensuite reporté dans le fichier CSV.

Exemple de sortie console d'une simulation :

```
[SIMULATION] Lancement de tous les fichiers avec les parametres CsvExporter{algorithm=
Tabou,iterationCount=10000, tabuListSize=20}
[SIMULATION] (1/28) Lecture fichier A3205.txt - simulation terminee (19621ms).
[SIMULATION] (2/28) Lecture fichier A3305.txt - simulation terminee (14729ms).
[SIMULATION] (3/28) Lecture fichier A3306.txt - simulation terminee (15698ms).
[SIMULATION] (4/28) Lecture fichier A3405.txt - simulation terminee (22410ms).
[SIMULATION] (5/28) Lecture fichier A3605.txt - simulation terminee (18548ms).
[SIMULATION] (6/28) Lecture fichier A3705.txt - simulation terminee (27538ms).
[SIMULATION] (7/28) Lecture fichier A3706.txt - simulation terminee (27421ms).
[SIMULATION] (8/28) Lecture fichier A3805.txt - simulation terminee (25643ms).
[SIMULATION] (9/28) Lecture fichier A3905.txt - simulation terminee (28806ms).
[SIMULATION] (10/28) Lecture fichier A3906.txt - simulation terminee (25720ms).
[SIMULATION] (11/28) Lecture fichier A4406.txt - simulation terminee (38600ms).
[SIMULATION] (12/28) Lecture fichier A4506.txt - simulation terminee (39650ms).
[SIMULATION] (13/28) Lecture fichier A4507.txt - simulation terminee (36344ms).
[SIMULATION] (14/28) Lecture fichier A4607.txt - simulation terminee (38274ms).
[SIMULATION] (15/28) Lecture fichier A5307.txt - simulation terminee (49931ms).
[SIMULATION] (16/28) Lecture fichier A5407.txt - simulation terminee (48323ms).
[SIMULATION] (17/28) Lecture fichier A5509.txt - simulation terminee (50496ms).
[SIMULATION] (18/28) Lecture fichier A6009.txt - simulation terminee (73637ms).
[SIMULATION] (19/28) Lecture fichier A6109.txt - simulation terminee (57884ms).
[SIMULATION] (20/28) Lecture fichier A6208.txt - simulation terminee (105019ms).
[SIMULATION] (21/28) Lecture fichier A6310.txt - simulation terminee (91837ms).
[SIMULATION] (22/28) Lecture fichier A6409.txt - simulation terminee (90984ms).
[SIMULATION] (23/28) Lecture fichier A6509.txt - simulation terminee (71135ms).
[SIMULATION] (24/28) Lecture fichier A6909.txt - simulation terminee (110199ms).
[SIMULATION] (25/28) Lecture fichier A8010.txt - simulation terminee (122043ms).
[SIMULATION] (26/28) Lecture fichier c101.txt - simulation terminee (211749ms).
[SIMULATION] (27/28) Lecture fichier c201.txt - simulation terminee (144298ms).
[SIMULATION] (28/28) Lecture fichier r101.txt - simulation terminee (449753ms).
[SIMULATION] Simulation terminee (temps total = 2058,54s).
```

Conclusion

Ce projet à été très enrichissant car il nous a permis de mieux apprivoiser les algorithmes à base de voisinage que nous avions vu en cours, notamment celui du recuit simulé et tabou. Il nous a également permis lors du développement de travailler l'optimisation de notre code car cela était obligatoire si nous voulions obtenir des temps de calculs raisonnables.

Ensuite, il a été très intéressant d'analyser ces deux métaheuristiques en lançant différentes simulations sur de nombreux graphes. En effet, l'analyse de nos résultats nous a permis de constater l'évolution des temps d'exécution ainsi que des résultats obtenus en fonction de nos différents paramètres d'entrée.

En conclusion, bien que ce projet était un peu long, il n'en demeurait pas moins intéressant.

Annexes 📁

Sources :

- https://en.wikipedia.org/wiki/Simulated_annealing
- https://en.wikipedia.org/wiki/Tabu_search
- Illustrations : https://unsplash.com/
- Rédaction de ce rapport réalisé sur https://www.notion.so/
- Librairie OpenCSV : http://opencsv.sourceforge.net/

Fichier Excel d'analyse complet

Voir dans l'archive rendue 'Analyse données CVRP.xlsx' ou bien sur le repo Github

Notre repo Github

https://github.com/remi-martinez/cvrp

Rémi Martinez

Baptiste Aubert

Année 2021 - 2022

POLYTECH LYON