École Nationale Supérieure d'Informatique
pour l'Industrie et l'Entreprise

Projet recherche

Amazon Last-Mile Routing Research Project

Sié KAMBIRE & Mohamed DOUMBIA

Encadrant : Massinissa MERABET

Mai 2025

Filière : Formation initiale, 2e année ENSIIE

Année scolaire 2024-2025

Contents

1 Introduction

Dans le cadre de notre cours **Projet Recherche**, nous avons travaillé sur le sujet *Amazon Last Mile Routing Research*. Ce projet s'inscrit dans une démarche d'initiation à la recherche, visant à développer des compétences fondamentales telles que la rigueur scientifique, l'analyse critique, la résolution de problèmes complexes et la capacité à explorer des approches innovantes.

Le sujet s'appuie sur le challenge lancé par Amazon, qui a pour objectif de concevoir des approches novatrices — reposant notamment sur l'intelligence artificielle, l'apprentissage automatique ou d'autres techniques avancées — pour résoudre le problème du séquencement d'itinéraires dans le cadre de la livraison du dernier kilomètre. L'enjeu est de dépasser, ou du moins de compléter, les méthodes d'optimisation classiques qui, bien que performantes, présentent des coûts computationnels élevés, en particulier à grande échelle.

C'est dans cette optique que nous avons exploré différentes stratégies algorithmiques, dans le but de concevoir des solutions plus efficaces, plus adaptables et mieux alignées avec les contraintes et les aléas du monde réel.

Il est important de préciser que, dans un premier temps, nous avons choisi de mettre de côté les contraintes liées à la performance d'exécution, afin de nous concentrer sur le développement d'un programme fonctionnel, fiable et correct. Néanmoins, si le temps le permet, nous envisageons d'optimiser notre code afin d'en améliorer l'efficacité et les performances globales.

NB : Notre travail s'est appuyé sur un projet réalisé lors du challenge. Afin de gagner du temps, nous allons volontairement omettre la présentation détaillée du problème, des données et de l'algorithme général, pour nous concentrer directement sur notre propre contribution, les difficultés rencontrées, ainsi que les solutions que nous avons pu proposer.
Nous vous recommandons vivement de lire le rapport de ce groupe avant de poursuivre la lecture du présent document, afin de bénéficier d'un aperçu complet du contexte.

Ce rapport est disponible sur notre dépôt github dans lequel se trouve notre projet. Nous vous invitons à lire attentivement le README

2 Problèmes rencontrés et solutions proposées

2.1 Manipulation des données

Le premier problème auquel nous avons été confrontés concerne la manipulation des données. Celles-ci nous ont été fournies dans des fichiers *JSON* au format relativement désorganisé, rendant leur exploitation initiale difficile. Les jointures entre les différentes sources de données étaient complexes à mettre en œuvre, ce qui a ralenti la préparation des données pour les adapter au format requis par notre programme.

Un autre point important à souligner est la taille de ces fichiers : leur volume important a significativement affecté les temps d'exécution, notamment parce qu'il était nécessaire de les relire à chaque fois pour en extraire les informations dont on a besoin (par exemple les informations d'une route particulière).

En résumé, cette étape nous a pris plusieurs jours, le temps de comprendre quelles données se trouvaient dans quels fichiers, sous quelle forme, et comment les extraire efficacement. Bien que cette phase ait été fastidieuse, elle nous a permis de nous familiariser avec la gestion de données complexes, une compétence essentielle dans tout projet impliquant l'exploitation de données à grande échelle.

2.2 Amélioration du programme linéaire

Comme l'équipe ayant participé au challenge, nous avons modélisé notre problème à l'aide du *Vehicle Routing Problem with Time Windows* (VRPTW), dont la formulation mathématique est la suivante :

$$\text{Minimiser} \quad \sum_{(i,j)\in E} x_{i,j} \cdot p_{i,j} \tag{B1}$$

$$\begin{cases} \displaystyle\sum_{(i,j)\in E} x_{i,j} = 1 & \forall i \in V \setminus \{v_0\} \\ \displaystyle\sum_{(i,j)\in E} x_{i,j} = 1 & \forall j \in V \setminus \{v_f\} \\ \displaystyle\sum_{(i,j)\in E} x_{i,j} = \sum_{(i,j)\in E} x_{j,i} & \forall i \in V \setminus \{v_0, v_f\} \\ \displaystyle\sum_{(i,j)\in E} x_{i,j} - \sum_{(i,j)\in E} x_{j,i} = 1 & i = v_0 \\ \displaystyle\sum_{(i,j)\in E} x_{i,j} - \sum_{(i,j)\in E} x_{j,i} = -1 & i = v_f \\ x_{i,i} = 0 & \forall i \in V \\ t_i + p_{i,j} - t_j \leq M \cdot (1 - x_{i,j}) & \forall (i,j) \in E \\ a_i \leq t_i \leq b_i & \forall i \in V \\ x_{i,j} \in \{0,1\} & \forall (i,j) \in E \end{cases} \tag{1}$$

Nous avons remarqué que ce programme linéaire pouvait être simplifié afin d'améliorer le temps d'exécution. On obtient ainsi la version suivante :

$$\text{Minimiser} \quad \sum_{(i,j)\in E} x_{i,j} \cdot p_{i,j} \tag{B1}$$

$$
\begin{cases}
\displaystyle\sum_{(i,j)\in E} x_{i,j} = 1 & \forall i \in V \setminus \{v_0\} \\
\displaystyle\sum_{(i,j)\in E} x_{i,j} = 1 & \forall j \in V \setminus \{v_f\} \\
\displaystyle\sum_{(i,j)\in E} x_{j,i} = 0 & i = v_0 \\
\displaystyle\sum_{(i,j)\in E} x_{i,j} = 0 & i = v_f \\
x_{i,i} = 0 & \forall i \in V \\
t_i + p_{i,j} - t_j \leq M \cdot (1 - x_{i,j}) & \forall (i,j) \in E \\
a_i \leq t_i \leq b_i & \forall i \in V \\
x_{i,j} \in \{0,1\} & \forall (i,j) \in E \\
x_{i,j} + x_{j,i} \leq 1 & \forall (i,j) \in E \\
t_j - \dfrac{b_j - t_i}{\text{seuil}} \leq M \cdot x_{i,j} & \forall (i,j) \in E
\end{cases}
\tag{2}
$$

Nous avons retiré la contrainte " du programme initial, car elle découle directement des deux contraintes précédentes. L'avant-dernière contrainte (du PL2) a été ajoutée après avoir observé des solutions contenant des arêtes aller-retour, ce qui nous a semblé incohérent puisque la contrainte 7 du PL1 interdit déjà les cycles. Quoi qu'il en soit, cette contrainte supplémentaire permet d'éliminer ces cas.

Enfin, la dernière contrainte du PL2 a été ajoutée pour limiter les cas où la variable t_i est très proche de la borne supérieure b_i. En effet, nous avons remarqué que dans certaines résolutions locales (notamment dans les zones ID), les valeurs de t_i atteignaient quasiment b_i, alors que dans la plupart des cas, b_i représente la fin de la journée ce qui rend les résolutions locales dans les zones ID suivantes impossibles puisqu'il ne reste plus de temps. Cette contrainte permet donc de limiter t_i à une valeur raisonnablement inférieure à b_i.

2.3 Résolution des problèmes

Après avoir préparé les données ainsi que le programme linéaire permettant de trouver le trajet optimal entre les zones ID, puis la solution globale en résolvant les problèmes locaux dans chaque zone ID, il ne restait plus qu'à lancer notre programme sur les différentes routes.

Comme expliqué précédemment, la résolution globale peut échouer lorsqu'une résolution locale dans une zone ID donnée — qui n'est pas la dernière à parcourir — absorbe la totalité du temps restant pour la livraison. Pour pallier ce problème, nous avons décidé d'introduire un seuil permettant de contrôler le temps passé dans les zones ID.

Cependant, même avec cette contrainte, nous ne parvenions toujours pas à obtenir une solution globale. Nous avons donc mis en place une stratégie de décrément du seuil : à chaque échec de résolution, le seuil est réduit d'un certain pas jusqu'à devenir inférieur à zéro, ce qui entraîne l'arrêt du programme. C'est cette mise à jour qui nous a permis d'obtenir certaines solutions viables.

Toutefois, les valeurs optimales du seuil et du pas varient d'une route à une autre, mais également selon le regroupement effectué. Nous présenterons notre algorithme de regroupement dans la section suivante.

Nous avons passé de nombreuses heures à tester différentes combinaisons de seuils et de pas pour obtenir certaines résolutions globales. Finalement, nous avons pu obtenir des solutions pour les deux premières routes. Malheureusement, nous n'avons pas eu suffisamment de temps pour résoudre les autres. Bien que nous ayons tenté d'optimiser les routes 3 à 5, notre programme tournait pendant des dizaines de minutes sans aboutir, ce qui nous a conduits à interrompre les calculs.

La machine ou le solveur que nous avons utilisé ne sont probablement pas assez puissants pour traiter efficacement ces cas complexes.

3 Axes d'amélioration

3.1 Algorithme de classification des zones ID

Nous avons observé que les stops d'une route étaient déjà rattachés à des zones ID préexistantes. Nous avons donc, dans un premier temps, tenté de travailler avec cette classification fournie. Cependant, nous nous sommes rapidement rendu compte qu'elle n'était pas pertinente dans le cadre de notre optimisation.

Par exemple, pour la première route, on dénombrait environ 130 stops répartis dans 31 zones ID. Or, plusieurs de ces zones ne contenaient qu'un seul stop, ce qui compliquait la structuration du problème et réduisait l'intérêt d'un traitement localisé. Cette granularité excessive augmentait la complexité du programme sans véritable gain sur la qualité des solutions.

Face à ce constat, nous avons décidé de mettre en place notre propre méthode de regroupement des stops, en nous appuyant sur un algorithme de classification hiérarchique. Pour cela, nous avons utilisé la matrice des distances entre stops comme base de calcul. Cette approche permet de regrouper les stops géographiquement proches les uns des autres, ce qui rend le découpage plus cohérent pour une résolution locale du problème.

Malgré des résultats nettement meilleurs que ceux obtenus avec la classification initiale, cette nouvelle méthode ne garantissait pas toujours la réussite de la résolution globale. En effet, certains regroupements donnaient lieu à des solutions optimales pour les routes 1 et 2, tandis que d'autres faisaient échouer le programme : ce dernier tournait parfois indéfiniment, ce qui nous obligeait à l'arrêter manuellement.

Ce comportement s'explique par le fait que la qualité du regroupement impacte directement la répartition du temps disponible entre les zones. Un mauvais découpage peut entraîner une saturation du temps dans une zone intermédiaire, rendant la poursuite du trajet impossible.

Afin de maximiser nos chances de succès, nous avons expérimenté différents seuils de distance pour générer les regroupements, en ajustant le niveau de granularité (nombre de clusters). Toutefois, le choix optimal de ces paramètres reste délicat, car il dépend à la fois de la topologie de la route, de la densité des stops et des contraintes temporelles associées.

Pour améliorer encore cette étape, il serait pertinent d'explorer des méthodes de regroupement alternatives, telles que le clustering par densité (DBSCAN) ou le k-means avec pondération temporelle, qui pourraient prendre en compte à la fois la proximité géographique et les fenêtres de temps associées aux stops. L'intégration d'un critère d'équilibrage entre les clusters (en nombre de stops ou en durée moyenne de visite) pourrait également permettre une meilleure gestion du temps global lors de la résolution.

3.2 Pistes à explorer

Malheureusement, nous n'avons pas disposé de suffisamment de temps pour corriger certains bugs et approfondir de nouvelles pistes d'optimisation. Néanmoins, plusieurs axes d'amélioration nous

paraissent prometteurs et mériteraient d'être explorés dans le cadre de travaux futurs :

- **Utilisation d'un modèle de machine learning pour le regroupement des stops** : Nous aurions pu entraîner un modèle supervisé à prédire des regroupements efficaces à partir de ceux ayant déjà mené à des solutions viables. Cela permettrait d'automatiser et potentiellement d'optimiser l'étape de classification en tenant compte de critères géographiques et temporels.
- **Exploitation des bornes inférieure et supérieure fournies par l'arbre couvrant de poids minimum (MST)** : Ces bornes, classiques en optimisation combinatoire, auraient pu être utilisées pour évaluer la qualité d'une solution locale, filtrer certains sous-problèmes non prometteurs, ou guider la recherche vers les zones les plus critiques à optimiser.
- **Amélioration du programme linéaire** : Bien que notre modèle ait été fonctionnel, nous pensons qu'il peut encore être allégé ou renforcé par des contraintes supplémentaires mieux adaptées à la structure du problème. Une étude approfondie des formulations issues de la littérature sur le TSP-TW ou le VRP pourrait inspirer des modifications efficaces.
- **Optimisation du code et des paramètres internes** : Le code actuel est encore assez brut, avec peu de mécanismes d'optimisation ou de modularisation. Par exemple, la valeur de la constante M (grande constante utilisée dans certaines contraintes) reste fixe, alors qu'elle pourrait être adaptée dynamiquement selon les caractéristiques de chaque sous-problème local. Cela permettrait d'améliorer la précision numérique et la vitesse de convergence du solveur.

4 Conclusion

Ce projet nous a permis de nous confronter à un problème réel de logistique urbaine : le séquencement d'itinéraires de livraison dans le dernier kilomètre. À travers l'exploration du modèle VRPTW et la manipulation de données complexes, nous avons développé un programme capable de proposer des solutions 'optimales' dans certains cas.

Malgré certaines difficultés techniques, notamment liées à la classification en zone ID et aux performances du solveur, notre démarche nous a permis d'aboutir à des résultats prometteurs sur plusieurs routes. L'introduction d'un seuil adaptatif et le développement d'une méthode de regroupement hiérarchique des stops ont joué un rôle central dans l'amélioration des performances.

Cependant, de nombreuses pistes restent à explorer, notamment l'intégration de techniques d'apprentissage automatique pour le regroupement, l'optimisation fine des paramètres du modèle, ou encore l'étude de nouvelles formulations inspirées de la littérature sur les problèmes de tournées.

En somme, cette expérience a été une opportunité enrichissante d'appliquer nos compétences en algorithmique et en recherche opérationnelle, tout en mettant en lumière les défis concrets que pose l'optimisation en contexte réel.