MS Data Science pour l'Ingénierie

Thèse Académique

Optimisation Combinatoire : Problème de logistique

Problème de livraison des ordres par les drones
Google Hash Code

Nisrine HAMMOUT

Tuteur : Nicolas GAYTON

Janvier - Juillet 2021

Introduction

Internet a profondément changé la façon dont nous achetons des choses, mais les achats en ligne d'aujourd'hui ne sont probablement pas la fin de ce changement ; après chaque achat, nous devons encore attendre plusieurs jours pour que les biens physiques soient transportés à notre porte.

C'est là que les drones entrent dans les véhicules électriques autonomes livrant des achats en ligne. Voler, donc jamais coincé dans le trafic. Alors que la technologie des drones s'améliore chaque année, il reste un problème majeur : **comment gérer et coordonner tous ces drones ?**

Dans le cadre de mon projet académique intitulé **Optimisation combinatoire appliquée pour la logistique**, j'ai décidé de mettre en pratique mes connaissances théoriques et les adopter pour un problème industriel qui est d'optimiser la route de livraison des drones.

Le problème de ce projet était le sujet dans la compétition internationale Google Hash Code dans l'année 2016.

Lors de ce projet, j'ai utilisé les méthodes suivantes :
— Les méta-heuristiques basées sur l'individu
— Les méta-heuristiques basées sur la population.

Table des matières

Chapitre 1

Contexte générale du projet

1.1 Google Hash Code

Hash Code est un concours de programmation en équipe, organisé par Google, pour les étudiants et les professionnels du monde entier.

Les objectifs de cette compétition est de :

— Résoudre les vrais problèmes d'ingénierie de Google.
— Essayer, essayez et réessayez : La beauté de Hash Code est qu'il n'y a pas de bonne réponse à aucune de nos questions. Les problèmes de chaque tour sont des problèmes d'optimisation, ce qui signifie que vous et votre équipe pouvez soumettre une solution, l'optimiser et la soumettre à nouveau. Ce processus itératif est exactement la façon dont nos ingénieurs travaillent au quotidien chez Google.
— Rencontrer des Hash Coders du monde entier sont partout, participez à la conversation maintenant : rencontrez d'autres Hash Coders.
— S'amuser un peu : L'une des meilleures parties de Hash Code est de se connecter avec d'autres membres de la communauté tout en apprenant et en développant vos compétences en programmation.

1.2 Description du problème

L'objectif est de programmer les opérations de drones afin que les ordres soient terminées dans les meilleurs délais, en prenant en compte :

— Une flotte de drones ;
— Une liste des ordres des clients ;
— Disponibilité des produits dans les entrepôts.

```
 ----  Information general ----
Ligne de la grille:  400
Colonnes de la grille:  600
Le nombre de drones:  30
Le nombre de tours:  112993
La capacité maximum d'une drone en (u):  200

  ----Information Produit ----
Different types de produit:  400

  ---- Information entrepots----
Nombre d'entrepots:  10

  ---- Information Ordres ----
 Nombre d'ordres:  1250
```

FIGURE 1.1 — Information sur les éléments du problème

Dans le graphe suivant on peut voir la distribution des entrepôts, les ordres et le point de départ des drones :

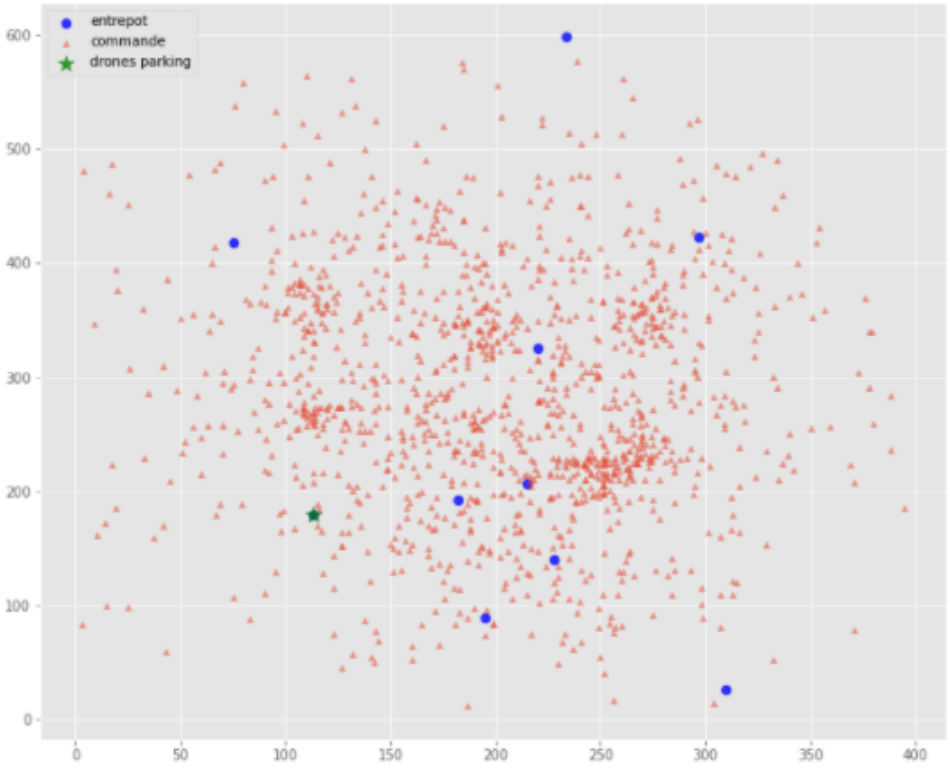

FIGURE 1.2 — Distribution des entrepôts, les ordres et le parking des drones dans la grille

Dans les sections suivantes, je vais décrire en détails les éléments du problème :

1.3 Éléments du problème

1.3.1 La Carte/Map

La simulation se déroule sur une grille à deux dimensions. La grille n'est pas cyclique et un drone ne peut pas voler à l'extérieur de la grille. Les drones peuvent survoler toutes les cellules de la grille.
Chaque cellule est identifiée par une paire de coordonnées entières : $[r, c]$ tel que : $0 \leq r \leq$ Nombre de ligne et $0 \leq c \leq$ Nombre de colonne.
Avec :
— Lignes de la grille : 400
— Colonne de la grille : 600

1.3.2 Entrepôts

Les articles de produits sont stockés dans plusieurs entrepôts. Chaque entrepôt est situé dans une cellule particulière de la grille, différente pour chaque entrepôt. Chaque entrepôt stocke initialement un nombre connu d'articles de chaque type de produit. Aucun nouveau produit au-delà de la disponibilité initiale ne sera stocké dans les entrepôts pendant la simulation, mais les drones peuvent transporter des produits entre les entrepôts.
Tout entrepôt n'a pas nécessairement besoin d'avoir tous les types de produits disponibles.

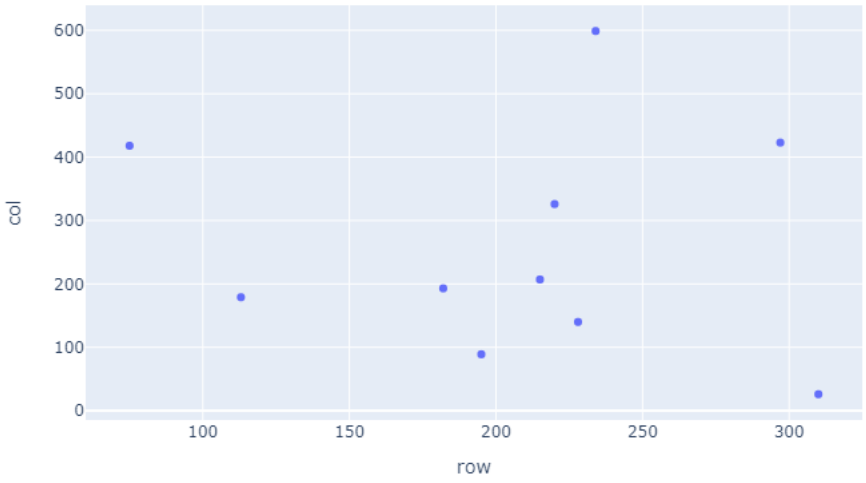

FIGURE 1.3 — Les 10 entrepôts dans la grille

1.3.3 Les Produits

Plusieurs types de produits sont disponibles à l'ordre. Chaque type de produit a un ou plusieurs articles disponibles dans les entrepôts. Chaque type de produit a un poids de produit fixe, identique pour tous les articles du produit. Chaque poids de produit est garanti inférieur ou égal à la charge utile maximale qu'un drone peut transporter.

$$P_{poidsdeproduits} \leq C_{Chargemaxdudrone}$$

Dans la graphe suivant, on peut voir la distribution des produits par chaque entrepôt par couleur :

FIGURE 1.4 — Distribution des produits dans les entrepôts dans la grille

1.3.4 Les ordres

Chaque ordre précise les articles du produit achetés par le client. Les articles d'une ordre peuvent être d'un ou plusieurs types de produits et peuvent contenir plusieurs articles du même type de produit.
Chaque ordre précise la cellule de la grille où les articles du produit doivent être livrés. Il est possible d'avoir plusieurs ordres avec la même cellule de livraison. Aucune ordre n'a la cellule de livraison qui est un emplacement d'un entrepôt.

La ordre est considérée comme exécutée lorsque tous les articles du produit commandé sont livrés. Les produits peuvent être livrés en plusieurs étapes, dans n'importe quel ordre. Il est valable de livrer les articles d'une ordre en utilisant plusieurs drones, y compris en utilisant différents drones en même temps.
Il est garanti que pour chaque type de produit, le nombre total d'articles de produit dans toutes les ordres est pas plus grand que la disponibilité totale des articles de ce type de produit dans tous les entrepôts.
Il n'est pas obligatoire de livrer toutes les ordres.
Dans le graphe suivant, on peut voir la distribution des ordres dans la grille et la couleur nous montre le nombre de produits dans chaque ordre.

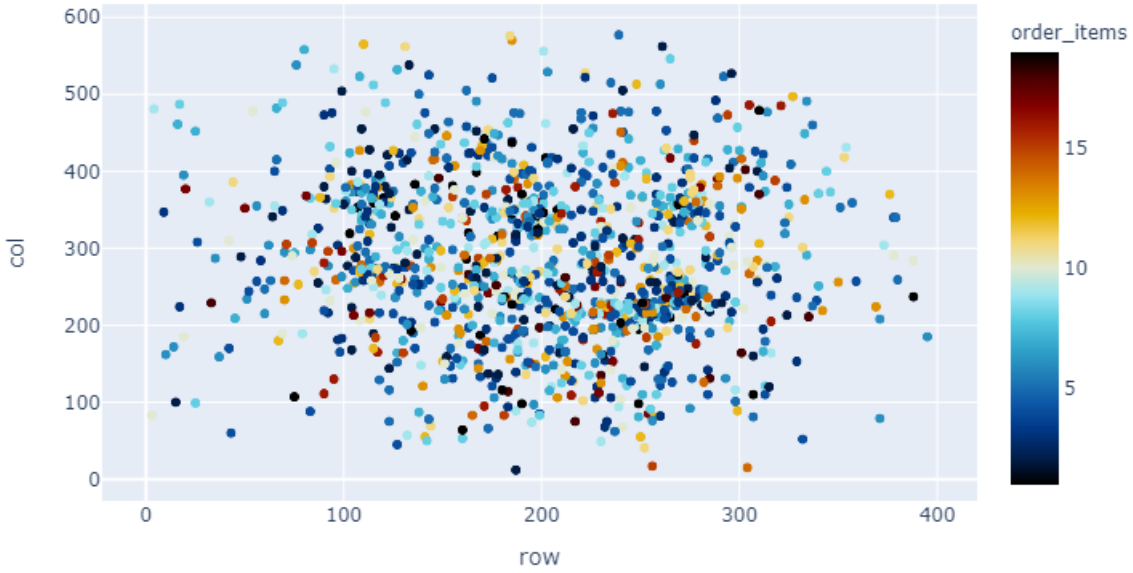

FIGURE 1.5 — Distribution des ordres dans la grille

1.3.5 Les Drones

Les drones transportent les articles des entrepôts aux clients et entre les entrepôts. Les drones utilisent toujours le chemin le plus court pour voler d'une cellule de la grille à une autre. La distance de la cellule $[r_a, c_a]$ et $[r_b, c_b]$ est calculé comme : $\sqrt{|r_a - r_b|^2 + |c_a - c_b|^2}$ distance euclidienne dimensionnelle.
Un vol de drone comme un mouvement unique du drone qui se produit entre deux instructions ultérieures pour le drone donné. Chaque vol de drone prend un tour par unité de distance entre la cellule de départ et la cellule de destination, arrondi à l'entier suivant.

Par exemple, si la distance à parcourir est de 2,9 unités de distance, la durée du vol est de 3 tours. Et si la distance à parcourir est exactement de 4 unités de distance, la durée du vol est de 4 tours.
Plusieurs drones peuvent se trouver dans la même cellule à tout moment sans jamais entrer en collision, car ils peuvent voler à différentes altitudes.
Au début de la simulation, tous les drones se trouvent dans le premier entrepôt (entrepôt avec l'identifiant 0).

1.3.6 Instructions des drones

Chaque drone peut recevoir les instructions de base suivantes :

— Charge : déplace le nombre spécifié d'articles du type de produit spécifié d'un entrepôt vers l'inventaire du drone. Si le drone n'est pas à l'entrepôt, il y volera en utilisant le chemin le plus court avant de charger les articles du produit. Le nombre demandé d'articles du type de produit spécifié doit être disponible dans l'entrepôt. Le poids total des articles dans l'inventaire du drone après le chargement ne peut pas être supérieur à la charge maximale du drone.
— Livrer : fournit le nombre spécifié d'articles du type de produit spécifié à un client. Si le drone n'est pas à destination, il y volera en utilisant le chemin le plus court avant de livrer les articles du produit. Le drone doit avoir le nombre demandé d'articles du type de produit spécifié dans son inventaire.

Chaque drone peut également recevoir les instructions :

— Décharger : déplace le nombre spécifié d'articles du type de produit spécifié de l'inventaire du drone vers un entrepôt. Si le drone n'est pas à l'entrepôt, il y volera en utilisant le chemin le plus court avant de décharger les articles du produit. Le drone doit avoir le nombre demandé d'articles du type de produit spécifié dans son inventaire.
— Attendez : attend le nombre de tours spécifié dans l'emplacement actuel du drone.

1.3.7 La simulation

La simulation se déroule en T tourne, de 0 à T – 1 . Un drone exécute les instructions qui lui sont délivrées dans le ordre dans lequel ils sont spécifiés, un par un. La première instruction émise au drone démarre au tour 0. La durée d'une instruction dépend de son type :

— Chaque instruction de charger / livrer / Décharger prend d+1 tour avec d est la distance parcourue par le drone pour effectuer l'action demandée (d peut être 0 si le drone est déjà à l'emplacement requis). Lorsque la instruction

démarre, le drone vole vers l'emplacement requis dans d tours. Puis l'action réelle Charge/Livrer /Décharger prend 1 tour.

— Chaque Attendez instruction prend w tourne, où w est le nombre de tours spécifié.

Par exemple, supposons qu'au début de la simulation (au début du tour 0) un drone est en l'entrepôt 0 à la cellule [1,1], l'entrepôt 1 est dans la cellule [1, 3], la instruction client 0 doit être livrée à [1,4], et p et q sont des types de produits, dont les articles sont partie de l'ordre 0.

Considérons les instructions suivantes :

— Chargez un article du type de produit p dans l'entrepôt 0. Cela prendra 1 tour : 0 tour pour arriver au entrepôt et 1 tour pour charger l'article .

— Chargez cinq articles du type de produit q dans l'entrepôt 1. Cela prendra 3 tours : 2 tours pour obtenir de [1,1] à l'entrepôt à [1,3] et 1 tour pour charger les articles.

— Livrer un article de type de produit p pour la instruction client 0. Cela prendra 2 tours : 1 tour pour obtenir de [1, 3] à la cellule de livraison à [1, 4] et 1 tour pour livrer l'article.

S'il y a plusieurs actions réelles qui se déroulent dans le même tour dans le même entrepôt, toutes Les instructions de déchargement sont traitées avant toutes les instructions de chargement.

Par exemple, supposons qu'en un tour, trois drones différents se trouvent tous dans l'entrepôt 0 et ordonné d'effectuer les actions suivantes :

— Charger un article de type de produit p à l'entrepôt 0. (premier drone)

— Charger un article de type de produit p à l'entrepôt 0. (deuxième drone)

— Décharger deux articles de type p à l'entrepôt 0. (troisième drone)

Étant donné que le déchargement a priorité sur le chargement, toutes les instructions réussiront même s'il n'y avait aucun article du type de produit p dans l'entrepôt avant ce tour.

1.3.8 Données d'entrée

Les données d'entrée sont fournies sous la forme d'un fichier d'ensemble de données, un fichier en texte brut contenant exclusivement des caractères ASCII avec des lignes terminées par un seul caractère à la fin de chaque ligne (UNIX fins de ligne de style).

Les types de produits, les entrepôts et les ordres sont référencés par des ID entiers. Il y a P types de produits numérotés de 0 à P – 1 , W entrepôts numérotés de 0 à W – 1 et C ordres numérotées de 0 à C – 1 .

La première section du fichier décrit les paramètres de la simulation. Cette section contient une seule ligne contenant les nombres naturels suivants séparés par des espaces simples :

— nombre de lignes dans la zone de simulation (1 ≤ Nombre de lignes ≤ 10000)
— nombre de colonnes dans la zone de simulation (1 ≤ le nombre de colonnes ≤ 10000)
— D nombre de drones disponibles (1 ≤ D ≤ 1000)
— Délai de la simulation (1 ≤ date limite de la simulation ≤ 1000000)
— Charge maximale d'un drone (1 ≤ charge maximale d'un drone ≤ 10000)

La section suivante du fichier décrit les poids des produits disponibles pour les ordres. Cette section contient :

— Une ligne contenant l'entier naturel suivant : P le nombre de types de produits différents disponibles dans les entrepôts (1 ≤ P ≤ 10000).
— Une ligne contenant P nombres naturels séparés par des espaces simples désignant les poids des types de produits, du type de produit 0 au type de produit P - 1. Pour chaque poids, 1 ≤ poids ≤ charge maximale d'un drone.

La section suivante du fichier décrit les entrepôts et disponibilité des types de produits individuels à chaque entrepôt. Cette rubrique contient :

— Une ligne contenant l'entier naturel suivant : W le nombre d'entrepôts (1 ≤ W ≤ 10000)
— Deux lignes pour chaque entrepôt, chacune de deux lignes décrivant les entrepôts suivants de l'entrepôt 0 à l'entrepôt W – 1 :
  — une ligne contenant deux nombres naturels séparés par un seul espace : la ligne et la colonne dans laquelle se trouve l'entrepôt (0 ≤ ligne ≤ nombre de lignes ; 0 ≤ colonne ≤ nombre de colonnes)
  — une ligne contenant P nombres naturels séparés par des espaces simples : nombre d'éléments du types de produits suivants disponibles à l'entrepôt, du type de produit 0 au type de produit P – 1 . Pour chaque type de produit, 0 ≤ nombre d'objets ≤ 10000 .

Chapitre 2

Construction du simulateur de livraison

2.1 Class Drone

La class drone simule le mouvement du drone de l'entrepôt vers les clients et des clients vers les entrepôts.
Le drone reçois un ordre pour la livraison et chaque drone est responsable d'une liste de ordres. Elle livre chaque type de produit à un client.
L'algorithme de simulation est comme suit :

1. **Initialisation** de la drone avec un contenu particulier pour la simulation.Le paramètre 'delivery_order' porte les informations des commandes que ce drone particulier doit livrer. On note que les bons de livraison sont fournis comme suit :'[(order_id, [item_1, ..., item_n], [delivery_x, delivery_y]), ...]' = [("ordre", "produits", "position")]
2. **Fonction start** : est chargée de démarrer le drone, et de donner la première tâche pour chaque existant drone. On construit la liste de livraison en une liste de produits. Le drone livrera des produits à chaque emplacement client. Cette fonction reçoit le paramètre 'wh_logger' qui est l'information de journalisation des entrepôts et des produits disponible pour chaque entrepôt. Ceci est nécessaire car le drone recherchera toujours le produit disponible le plus proche, puis ce drone retirera ce produit de l'entrepôt. Par conséquent, il supprimera le produit des journaux de l'entrepôt.
   - Paramètre tuple wh_logger : Un tuple avec le dataframe des produits et des entrepôts, respectivement.
   - Sortie tuple L'enregistreur modifié des produits et des entrepôts

3. **Fonction Step** : Cette fonction fera marcher le drone une fois vers le futur. Si le drone transporte un produit ou se rend en entrepôt pour obtenir un produit, la fonction fera en sorte que le drone se rapprochera un peu plus de sa cible. Si le drone est au client ou à l'entrepôt, la fonction livrera le produit au client ou il obtiendra le produit de l'entrepôt en particulier, ensuite, il guidera le drone vers l'entrepôt ou le client si le drone est dans le client ou l'entrepôt, respectivement.
   — Paramètre tuple wh_logger : Un tuple avec le dataframe produits et entrepôts, respectivement.
   — Sortie : L'enregistreur modifié des produits et des entrepôts.
4. **Fonction move_step** :Cette méthode est utilisée lorsque le drone transporte un produit. Cette méthode du pas de mouvement est responsable de rapprocher le drone d'un pas plus près de sa cible à chaque fois.
5. **Fonction delivery_action** : Cette méthode est utilisée lorsque le drone arrive chez le client ou l'entrepôt. La méthode d'action de livraison fera l'action de livraison à l'entrepôt ou chez le client, il dépend de l'endroit où se trouve le drone. Conditions :
   — Si le drone est chez le client, l'action de livraison livrera le produit au client, puis retirer le produit de la liste des produits que le drone a livré, puis guider le drone à l'entrepôt le plus proche qui a le prochain produit à livrer. Cela change aussi le journalisation de l'entrepôt, en supprimant le produit que le drone obtiendra de la journalisation de l'entrepôt, ce qui rend le produit «réservation» pour ce drone particulier. Ensuite, il calcule la trajectoire à l'entrepôt et entrez dans le mode de transport pour rendre à l'entrepôt.
   — Si le drone est dans l'entrepôt, le drone calcule la trajectoire au client et récupère le produit qu'il doit livrer au client en définissant l'attribut 'at_product'. Puis il entre dans le transport mode pour aller au client.

   Paramètre tuple wh_logger :Un tuple avec le dataframe produits et entrepôts, respectivement.

   Sortie : L'enregistreur modifié des produits et des entrepôts.
6. **Fonction find_next_product** : C'est la méthode qui est utilisée par l'action de livraison, lorsqu'elle est chez le client, pour trouver l'entrepôt avec le produit souhaité le plus proche. Cette méthode examine toutes les informations de journalisation des entrepôts pour rechercher Le prochain produit qui sera livré et calcule la distance jusqu'à cet entrepôt. Après elle sélectionne L'entrepôt le plus proche pour que le drone aille chercher le produit.

— Paramètre : tuple wh_logger :Un tuple avec le dataframe produits et entrepôts, respectivement.
— Sortie(tuple) : L'enregistreur modifié des produits et des entrepôts.

7. **Fonction write_summary** : Méthode pour créer un résumé de l'état du drone pour le débogage.

2.2 Création des ordres

Le but de cette étape est de créer la liste des ordres initiales à livrer pour tous les drones au format de (ordre, liste des produits, coordonnées de livraison).
Les étapes sont comme suit :
— Avoir les spécifiques de chaque ordre : le nombre de type de produits, codes produits et les coordonnées.
— création du contenu de chaque ordre sous forme de tuple.

2.3 Création de l'enregistreur d'événements

le but de créer l'enregistreur pour contrôler les informations des produits de l'entrepôt et les coordonnées des entrepôts qui seront fournies aux drones.
Fonction build_logger qui crée la journalisation de la disposition de l'entrepôt, de produits et des informations d'emplacement de l'entrepôt.
— Paramètre : DataFrame products_df : dataframe avec les informations de sur les produits.
— DataFrame warehouse_df : dataframe avec les informations particulières de l'entrepôt.
— Sortie sous forme tuple : l'enregistreur d'événements en tant que copie des deux paramètres fournis.

2.4 Création de drones

Le but de cette étape est de créer tous les drones avec un nombre spécifique de ordres à livrer. **La Fonction create_drones** responsable de créer le nombre nécessaire de drones pour livrer toutes les commandes fournies en considérant que le nombre moyen de livraisons par drone est fourni par 'opd'. Et que nous avons un total de 'num_drones' à utiliser.
— Paramètre entier num_drones : Le nombre de drones.
— Paramètre liste orders : La liste des ordres à fournir aux drones.
— Paramètre entier opd : Le nombre moyen de ordres par drone.

— Paramètre bool verbose : Paramètre pour afficher ou non les informations de débogage lors de la simulation.
— Sortie Une liste de drones créés.

2.5 Création du Simulateur

Le simulateur est une fonction qui simulera toutes les étapes de livraison pour toutes les courses.
Cette fonction simulera tous les drones pour un certain nombre d'étapes dans le futur et renverra la liste de tous les drones à cette étape particulière à l'avenir.
— Paramètre liste drones : La liste des drones créés.
— Paramètre entier steps : Le nombre de pas dans le futur pour simuler chaque drone.
— Paramètre tuple logger : Les informations d'état initial des produits dans les entrepôts et les informations particulières des entrepôts.
— Sortie Une liste avec les drones simulés après le nombre d'étapes fourni.

2.6 Test du Simulateur

Nous créons d'abord l'enregistreur pour la simulation : le tableau avec le nombre de produits fournis dans chaque entrepôt.
Deuxièmement, nous créons tous les drones, en fonction de la liste des ordres que nous voulons livrer.
Troisièmement, nous simulons les pas des drones dans le futur.

1. Construction de l'entrepôt et de l'enregistreur de produits
2. Création des drones
3. Simuler les déplacements des drones aux livraisons

Chapitre 3

Optimisation du problème

3.1 Introduction

Lors de ce chapitre on va testé quelques algorithmes d'optimisation tout en expliquant leur fonctionnement. Pour les tests on va se contenter de 10000 tours (turns) avec 15 livraison par le nombre de drones.

3.2 Fonction de coût

L'algorithme est basé sur la liste des commandes à livrer, simulera tous les drones et renverra le pourcentage de la distance parcourue par ce drone particulier et le pourcentage de commandes livrées pour tous les Drones.
Les étapes de construction de fonction de coût est comme suit :

1. Avoir des ordres uniques sous forme d'entiers
2. Construire l'enregistreur de l'entrepôt et de produits
3. Créer des drones pour chaque liste créée
4. Simuler les déplacements des drones aux livraisons
5. Calculer le coût de livraison :
   - Pourcentage du nombre de tours parcouru par tous les drones par rapport au nombre total des tours
   - Pourcentage de ordres livrées par tous les Drone $\frac{NbOrdresLivres}{NbOrdres}$
6. Retourner nb tours(%) + (100- ordres livrés(%)) cette quantité quand chercher à minimiser.
7. Mesurer le temps d'exécution de chaque méthode d'optimisation

3.3 Les Méta-heuristiques

Les méta-heuristiques sont généralement introduites en faisant une distinction entre deux grandes catégories de techniques : les méthodes basés sur les individus et méthodes basées sur la population.
Les techniques basées sur les individus, parfois appelées méthodes de trajectoire, visent à se déplacer dans l'espace de recherche en s'appuyant sur un système de voisinage N qui permet de construire un ensemble de solutions voisines à partir d'une solution actuelle x.
Les techniques basées sur la population tentent de mettre en commun les informations possédés par les individus qui composent la population afin de diriger la recherche selon des mécanismes propres à chaque méta-heuristique.

3.4 Méta-heuristique à base d'individu

3.4.1 Recuit Simulé

Le recuit simulé introduit un paramètre T température. L'idée générale du recuit simulé consiste à pouvoir accepter aussi des voisins non-améliorants, selon une probabilité $p(\Delta H, T)$ qui dépend de la dégradation des coûts $\Delta H = H(y) - H(x)$.
Une fonction RND ( ) est utilisé pour générer un nombre réel dans l'intervalle $[0; 1]$. Nous pouvons identifier ici une des caractéristiques propres aux méta-heuristiques : l'arrêt Critère.
Dans le cadre du recuit, les critères les plus utilisés sont le maximum nombre d'itérations, le nombre maximum d'itérations sans amélioration, l'acquisition d'une solution jugée qualitativement satisfaisante, atteignant une température finale préétablie.
L'algorithme de cette méthode est comme suit :

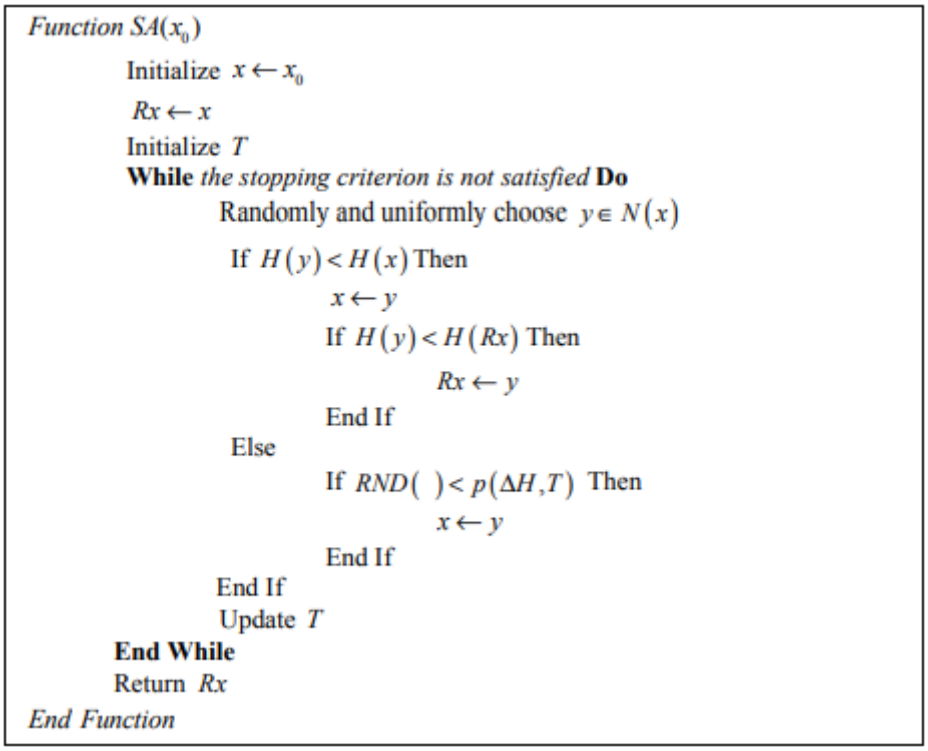

FIGURE 3.1 — Algorithme Recuit simulé (*Source : Livre Metaheuristics for Logistics - Laurent DEROUSSI*)

Application à notre problème :

```
la méthode de recuit simué nous donne un cout de : 32.82666666666667
Le nombre d'itération est de :  500
Nombre d'évaluations de la fonction de cout et de son Jacobien et Hessien :  15545
```

FIGURE 3.2 — Resultat Rcuit simulé

La simulation a un temps de convergence de :

```
Le temps de convergence 5199.941289186478 secondes.
```

FIGURE 3.3 — Recuit simulé temps de convergence

Les ordres accompli durant la simulation est comme suit :

FIGURE 3.4 — Les ordres accomplis

3.5 Méta-heuristique à base de population

L'objectif des méthodes basées sur la population est d'initier une interaction entre les individus qui composent une population afin de générer de nouvelles solutions. Nous nous concentrons sur trois méthodes qui représentent de manière tout à fait appropriée les principes qui peuvent être utilisés : les algorithmes évolutifs, l'algorithme des colonies de fourmis et l'optimisation des essaims de particules.

3.5.1 Algorithmes évolutionnaires

Ces procédures méta-heuristiques reposent sur la théorie de l'évolution de Darwin. Ils tentent de faire évoluer une population, génération par génération, afin que ses individus optimisent une fonction de coût qui représente leur degré d'adaptation au milieu naturel.
Les étapes d'un algorithme évolutif :

— la phase d'initialisation consiste à générer la population initiale. Le Les premiers individus sont souvent créés au hasard afin qu'il puisse avoir une grande diversité génétique ;
— La phase de sélection consiste à choisir les parents qui vont se reproduire, tout en améliorant les chances des meilleurs d'entre eux.

— La phase de reproduction comprend les opérateurs clés : Croisement et Mutation. Croisement est un opérateur généralement binaire qui essaie de préserver les caractéristiques partagées par deux parents afin de générer un ou Deux enfants qui, espérons-le, iront mieux. Cet opérateur est donc un facteur d'exploitation. Selon un taux de mutation établi, le Chromosome peut subir une mutation génétique. Cet opérateur consiste en général, de modifier aléatoirement les informations portées sur un locus. La mutation est un facteur d'exploration ;
— La phase de remplacement consiste simplement à créer la génération par Choisir parmi les parents et les enfants qui en font partie, ceux qui survivre. Le nombre d'individus qui composent chaque génération peut être Constant. Il est conseillé de préserver les meilleurs individus pour assurer une meilleure Convergence de l'algorithme (stratégie élitiste). Il est également souhaitable de Garder au hasard quelques individus afin de maintenir la diversité génétique.

L'algorithme de cette méthode est comme suit :

Function Evolutionary _ Algorithm()
Initialization: creation of the initial population G_0
Assess the adaptation degree of each individual
$Rx \leftarrow x / H(x) = \min(H(y), y \in G_0)$
For *k from 1 to MaxGeneration* **Do**
Parents' selection in G_{k-1}
Reproduction Phase (fathering of children agents of crossover and mutation)
Assess the adaptation degree of each child
If necessary, update the record state Rx
Replacement phase: creation of the generation G_k
End For
Return Rx
End Function

FIGURE 3.5 — Algorithmes évolutionnaires (*Source : Livre Metaheuristics for Logistics - Laurent DEROUSSI*)

Application à notre problème :

```
la méthode de differentiel évolutionnaire  nous donne un cout de : 126.66666666666667
Le nombre d'itération est de :  19
Nombre d'évaluations de la fonction de cout et de son Jacobien et Hessien :  4516
```

FIGURE 3.6 — Résultat

La simulation a un temps de convergence de :

```
Temps de convergence 1407.7667458057404 seconds.
```

FIGURE 3.7 — Temps de convergence

Les ordres accompli durant la simulation est comme suit :

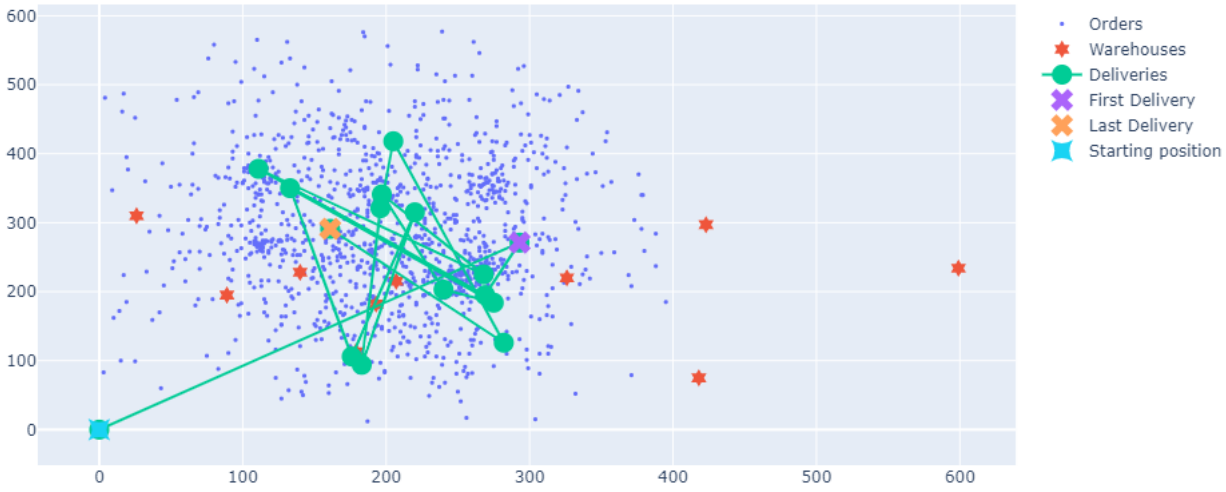

FIGURE 3.8 — Les ordres accomplis

3.6 Conclusion

On conclut de ces des résultats que la méthode basé sur les individus nous donne un résultat meilleur. Néanmoins, pour les tests on a pris un exemple de 15 ordres et cela revient à l'incapacité de mon ordinateur de simuler tous les 1250 ordres.

Chapitre 4

Conclusion

Ce projet a deux coté :

1. Construire petit à petit une simulation en python d'un simple fichier où les données sont sépares par des tabulations.
2. Implémenter des méthodes de méta-heuristiques pour résoudre un problème de logistique en incluant les commandes de drones.

Lors de ce projet, j'ai développer et solidifier plusieurs compétences à savoir :

— La capacité de commencer un projet brute d'une étape o à délivrer un produit exploitable et cela en adoptant une méthode agile qui m'a permet de bien organiser le travail et délivrer le projet à temps.
— Développer mes compétences en recherche opérationnelle et optimisation dans le domaine du logistique.
— Développer mes compétences en développement, simulation et traduire un problème de logistique en python

Je tiens à remercier le professeur Nicolas GAYTON de m'avoir affecter et de m'avoir guider lors de ce sujet très intéressant pour ma carrière. A travers ce dernier, j'ai pu découvrir et manipuler des méthodes d'optimisation combinatoire.

Je remercie aussi le professeur Laurent DEROUSSI d'avoir m'éclaircir et guider sur les différents méthodes d'optimisation dans le domaine de logistique à travers de son livre Metaheuristics for Logistics.

Chapitre 5

Bibliographie

— Le livre de Laurent DEROUSSI Metaheurstics for Logistics : https://www.wiley.com/en-am/Metaheuristics+for+Logistics-p-9781848218086
— Description de la compétition : https://storage.googleapis.com/coding-competitions.appspot.com/HC/2016/hashcode2016_qualification_task.pdf
— Les données d'entrée : https://www.kaggle.com/c/hashcode-drone-delivery