université de BORDEAUX

Collège Sciences et Technologies
UF Mathématiques et Interactions

Livraisons de colis assistées par drones

Paul Archipczuk
Kevin Bonhomme
Martin Debouté
Alan Ferrara
Lin Hirwa Shema

Projet professionnel M2 ROAD / OPTIM

M2 ROAD / CMI OPTIM
2022–2023

Table des matières

Engagement de non plagiat

Nous, Paul Archipczuk, Kevin Bonhomme, Martin Debouté, Alan Ferrara, Lin Hirwa Shema, déclarons être pleinement conscients que le plagiat de documents ou d'une partie d'un document publiés sur toutes formes de support, y compris l'internet, constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.

En conséquence, nous nous engageons à citer toutes les sources que nous avons utilisées pour produire et écrire ce rapport.

Fait à Talence le 25 février 2023.

Signatures

Paul Archipczuk
Kevin Bonhomme
Martin Debouté
Alan Ferrara
Lin Hirwa Shema

1 Introduction

1.1 Contexte

Le groupe français La Poste est un opérateur de services postaux fondé à la fin du XIXème siècle. Cette entreprise assure la livraison de colis et de courrier. Dans le cadre de sa stratégie d'innovation et d'évolution, La Poste souhaite investir dans la mise en place de technologies de pointe applicables sur des problématiques de *Supply Chain*[1]. Face à la transformation et aux problématiques croissantes de l'approvisionnement des villes et des campagnes, le groupe La Poste se trouve confronté à plusieurs problèmes grandissants :

— diminution du nombre de chauffeurs livreurs ;

— diminution des flux de courrier, devenant déficitaire ;

— augmentation des flux de livraison de colis, devenant majoritaire ;

— exigence des clients de plus en plus forte des clients face à un marché dynamique en pleine transformation et émergence de nouveaux services de livraison (livraison en J, H+3/6/9/12).

Fort de la plus grande flotte de véhicules de livraison électriques au monde, le groupe La Poste souhaite repenser son réseau logistique rural de proximité en modernisant sa flotte de véhicules et étudiant des propositions de livraison autonomes et volantes.

La Poste souhaite étudier les gains en temps relatifs à la mise en place de deux drones de livraisons connectés à un véhicule de livraison traditionnel par rapport à l'utilisation simple d'un véhicule de livraison en zone rurale. La Poste, dans un souci d'efficacité, différenciera ses tournées de livraison de colis de ses tournées de livraison de courrier. Nous nous concentrerons ici uniquement sur la livraison de colis.

En nous fondant sur nos connaissances ainsi que sur des articles cités dans la bibliographie nous avons mis en place une démarche scientifique qui consiste à comparer le temps nécessaire pour réaliser une tournée avec un seul véhicule et une tournée d'un véhicule assisté par deux drones. L'ensemble des expérimentations ont été réalisées sur des instances communes, fournies par La Poste. Afin de définir l'itinéraire du véhicule et des drones, nous avons développé diverses heuristiques, en commençant par des cas simplifiés avant d'aborder le problème complet.

A l'issue des expérimentations numériques, nous avons conclu que l'utilisation de drones (sans contraintes d'autonomie, sans prendre en compte le relief, etc...) pour suppléer la tournée du véhicule, représentait un gain de temps significatif. En effet, pour l'ensemble des instances nous avons observé une diminution du temps total pour effectuer la tournée.

1. La *Supply Chain* ou chaîne d'approvisionnement est le système d'organisation et de gestion des flux de produits, des informations et des finances de la production des matières premières à la livraison des produits finis au client final. Elle comprend tous les acteurs impliqués, des fournisseurs de matières premières aux distributeurs en passant par les transporteurs, les usines et les entrepôts. Le but est d'optimiser l'efficacité et la qualité tout en minimisant les coûts pour satisfaire les demandes des consommateurs.

2 Formalisation du problème

2.1 Les 4 cas à étudier

Afin d'appréhender au mieux ce problème, nous allons différencier 4 cas de plus en plus complexes :

— Cas 0 : tournée de livraison simple avec le camion.

— Cas 1 : points de largage et récupération des drones statiques. Le véhicule de livraison est doté de 2 drones de livraison autonomes pouvant amener les colis en ligne droite et un par un aux points de livraison. Ces drones devront être lâchés et récupérés à l'arrêt en un même point (correspondant aux nœuds du graphe routier).

— Cas 2 : points de largage et récupération des drones différenciés. Le véhicule de livraison est doté de 2 drones de livraison autonomes pouvant amener les colis en ligne droite et un par un aux points livraison. Ces drones devront être lâchés et récupérés à l'arrêt mais pourront l'être en deux points différents (correspondant aux nœuds du graphe routier).

— Cas 3 : points de largage et récupération dynamiques. Le véhicule de livraison est doté de 2 drones de livraison autonomes pouvant amener les colis en ligne droite et un par un aux points de livraison. Ces drones devront être lâchés et récupérés en route, à n'importe quel point du parcours.

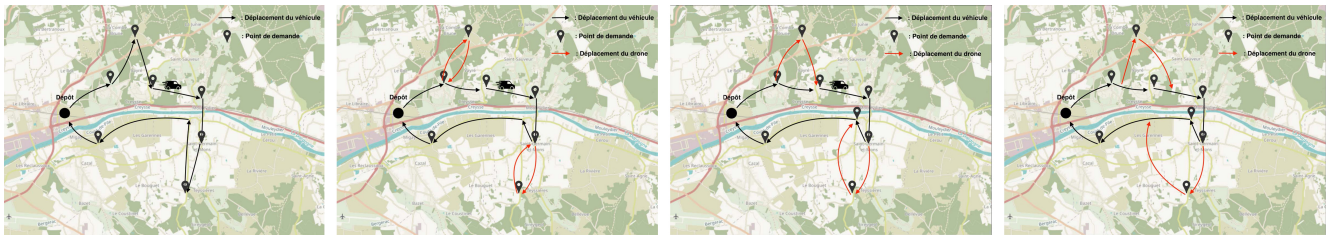

FIGURE 2.1 – Illustration des cas 0, 1, 2 et 3

2.2 Notations

2.2.1 Données

Avant toute chose, nous allons introduire $\mathcal{I}$ qui correspondra à l'ensemble des nœuds du graphe routier et $\mathcal{I}_d \subset \mathcal{I}$ l'ensemble des nœuds possédant une demande strictement positive (le dépôt est élément de cet ensemble). En outre, $\mathcal{S}_d$ est défini comme l'ensemble des sous-tours du graphe induit par les nœuds appartenant à $\mathcal{I}_d$ et des arcs de coût $\hat{c}$.

Pour l'ensemble des cas, et pour l'ensemble des instances, nous avons la connaissance du nombre de colis à livrer au nœud $i \in \mathcal{I}$, que l'on notera d_i avec $d_i \in \mathbb{N}^+$. De plus, nous connaissons les routes qui existent ainsi que leur type et leur longueur. Grâce à la distance et au type de route (qui aura une influence sur la vitesse du véhicule) nous avons calculé une matrice de coûts, qui correspond au temps mis par le véhicule pour se déplacer d'un point i à un point j du graphe routier. Ce coût est noté c_{ij} avec $c_{ij} \in \mathbb{R}^+$ si la route $(i, j) \in \mathcal{I} \times \mathcal{I}$ existe (et $+\infty$ sinon). Le coût du plus court chemin entre $i \in \mathcal{I}_d$ et $j \in \mathcal{I}_d$ est représenté par $\hat{c}_{ij} \in \mathbb{R}^+$. Pour les drones, il nous a suffi de calculer la distance euclidienne entre chaque point du graphe. Les coûts associés pour les drones sont représentés par e_{ij} où $e_{ij} \in \mathbb{R}^+$.

2.2.2 Solutions

On cherche à minimiser le temps pour effectuer la tournée de livraisons. Une solution du problème correspondra à un dictionnaire composé de trois listes, une pour le camion et une pour chaque drone. Chaque liste sera composée d'une succession de `tuples` représentant un évènement. Les différents évènements sont le mouvement, et l'attente qui ont pour `tuple` commun *(nœud source, nœud destination, temps de trajet)*, à la seule différence que pour l'attente le nœud source est identique au nœud de destination. Ces évènements peuvent se trouver dans la liste de n'importe quel véhicule. Enfin, les évènements « lancement de drone » *(nœud de lancement, nœud de lancement, temps de préparation du drone, indice du drone lancé)* et « livraison d'un colis » *(nœud de livraison, nœud de livraison, durée de livraison, nombre de colis livrés)* sont uniquement dans la liste du camion. Le camion étant le véhicule qui guide la tournée et les temps de trajet/d'attente se trouvant toujours en index 2 du `tuple`, la valeur de la solution sera donc la somme des indexes 2 des `tuples` de la liste du camion.

Cette représentation nous permet d'avoir un unique format de solutions pour tous les cas et facilite ainsi leur interopérabilité, leur affichage et leur écriture dans le format texte final.

2.3 Modélisations

2.3.1 Cas 0

Le cas 0 correspond à un problème de voyageur de commerce (*Travelling Salesman Problem*, TSP) dont le véhicule se déplace de nœuds de demande à nœuds de demande en se permettant de repasser par des nœuds intermédiaires.

La principale différence entre ce problème au cas 0 et un TSP correspond au fait que nous ne sommes pas contraints de visiter chaque nœud du graphe routier et que chaque nœud peut être visité plusieurs fois si nécessaire. Cependant, on peut se ramener au problème du TSP en ne considérant que les nœuds de demande et le dépôt. Il suffit alors de calculer le plus court chemin entre chaque nœud de demande ainsi que le dépôt et d'utiliser les temps totaux de ces chemins comme les coûts des arcs pour résoudre un TSP.

Avec cette représentation, nous devons visiter chacun des nœuds considérés et nous ne visitons un nœud qu'une seule fois afin de livrer sa demande. Toute autre visite de ce nœud ne peut s'effectuer que lors d'un parcours d'un plus court chemin entre deux autres nœuds. Une fois ce TSP résolu, on reconstruit une solution du initial en rajoutant les nœuds du plus court chemin entre chaque couple de nœuds consécutifs de la tournée. Il suffit alors d'ajouter le temps de livraison de la demande de chaque nœud à la durée totale de la tournée.

Pour la modélisation, nous avons crée la variable de décision binaire x_{ij} qui vaut 1, si le véhicule livre le nœud $j \in \mathcal{I}_d$ directement après le nœud $i \in \mathcal{I}_d$, et 0 sinon.

Modèle :

$$\sum_{i \in \mathcal{I}_d} 60 \cdot d_i + \min \sum_{i \in \mathcal{I}_d} \sum_{j \in \mathcal{I}_d} \hat{c}_{ij} x_{ij}$$

$$s.c. \sum_{k \in \mathcal{I}_d} x_{ki} = \sum_{j \in \mathcal{I}_d} x_{ij}, \qquad \forall i \in \mathcal{I}_d \tag{2.1}$$

$$\sum_{j \in \mathcal{I}_d} x_{ij} = 1, \qquad \forall i \in \mathcal{I}_d \tag{2.2}$$

$$\sum_{(i,j) \in s} x_{ij} \leqslant |s| - 1, \qquad \forall s \in \mathcal{S}_d \tag{2.3}$$

$$x_{ij} \in \{0, 1\}, \qquad \forall i \in \mathcal{I}_d, \ \forall j \in \mathcal{I}_d \tag{2.4}$$

L'objectif ici est de minimiser le temps total pris par le véhicule pour parcourir sa tournée et livrer les colis. Puisque la durée de livraison des colis est constante, on peut la retirer de la minimisation. La contrainte (2.1) est la contrainte de flot qui stipule que si on entre dans un nœud on doit en sortir. La contrainte (2.2) impose que chaque nœud doit être visité. La contrainte (2.3), élimine les sous-tours en imposant le fait qu'on ne peut retourner à un nœud que si tous les autres nœuds ont été visités. Cette contrainte est générée dynamiquement dans notre implémentation grâce à l'utilisation de *call-backs*.

3 Revue de littérature

3.1 Optimization Approaches for the Traveling Salesman Problem with Drone - N. AGATZ, P. BOUMAN, M. SCHMIDT (2018)

Une première approche pour le cas 3 se fondait sur les travaux de Agatz, Bouman et Schmidt, 2018 dont l'idée principale, en partant d'une solution de TSP, correspond au partitionnement des nœuds de visite en trois états : *Combined*, *Truck* et *Drone*. Dans l'état *Truck* le nœud est visité par le véhicule sans que les drones ne soient dans le véhicule. Pour l'état *Drone* c'est ce dernier qui effectue la livraison et enfin l'état *Combined* signifie que le véhicule effectue la livraison avec au moins un drone présent dans le camion.

Cependant, notre structure de données traite les plus courts chemins et non les nœuds individuellement, il serait ainsi trop compliqué d'adapter cette heuristique.

À titre indicatif, ils en concluent qu'avec leur heuristique, ils obtiennent une réduction du coût de 30% à 38% entre le TSP et le TSP-D (avec drones).

3.2 Vehicle Routing Problem with drones - Z. WANG, J-B SHEU (2019)

Dans cet article de Z. WANG, 2019, les auteurs considèrent le problème comme étant un *Vehicle Routing Problem* (VRP) et non pas un TSP comme vu précédemment. Cependant leur problème initial est différent du notre sur certains points notamment le fait que les drones puissent partir/arriver d'un camion mais aussi d'un hub permettant leur rechargement. De plus, un paramètre est pris en compte, la portée des drones. L'inconvénient principal de l'algorithme de résolution réside dans la limite de nœuds de demande maximal à traiter (qui est de 15) ce qui est trop peu pour notre utilisation quand bien même l'algorithme nous donnerait un résultat exact. Par ailleurs le nombre de camion et de drone proposés au sein de l'article sont variables, leur conclusion étant que le cas le plus intéressant serait d'avoir par camion un nombre de 1 ou 2 drones à disposition, ce qui est cohérent avec la décision d'apporter 2 drones par véhicule. Cependant ils constatent aussi que la différence entre le VRP avec et sans drone est d'autant plus intéressante lorsque la vitesse du drone est 1.5x supérieur à celle du camion, ce dernier paramètre permettrait de réduire le temps de 75%.

3.3 Optimal delivery routing with wider drone-delivery areas along a shorter truck-route - YS. CHANG, HJ. LEE (2018)

La particularité du problème présenté dans cet article par YS. CHANG, 2018 est le fait que le camion soit considéré comme un dépôt itinérant. De ce fait, seul les drones sont aptes à délivrer les colis auprès des clients. Ainsi l'objectif dans ce problème serait de faire un TSP qui chercherait les localisations les plus intéressantes pour permettre aux drones de décoller et ainsi livrer le plus de clients possibles. C'est une vision du problème de livraison de colis qui est totalement différente de celle énoncé de la part de La Poste, cela reste une approche intéressante dont nous prenons en compte de son efficacité qui permettrait un gain de temps de 15% entre un TSP avec ou sans drone.

3.4 Coordinated Logistics with a Truck and a Drone - JG CARLSSON, S. SONG (2018)

JG CARLSSON, 2018 se sont basés principalement sur les travaux effectués par Murray et Chu en 2015 mais aussi ceux de Agatz et al. de 2018. L'approche qu'ils ont développé et qu'ils prénomment *HorseFly Routing Problem* est que, comme dans notre cas 3, le drone a la liberté de partir du camion et d'être recupéré par ce dernier lors qu'il est en mouvement. Dans leur modèle, Carlsson et Song considèrent que le camion se déplace de 2 manières différentes, en zigzag ou en spirale dans le but principal est de couvrir au mieux les nœuds de demande.

4 Algorithmes de résolution du problème

4.1 Approche exacte

Nous avons utilisé une approche exacte pour la résolution du cas 0 grâce au modèle décrit plus haut (2.3.1). Pour le cas 1, nous avons utilisé un MIP relâché afin de faire une matheuristique.

4.2 Approche heuristique

4.2.1 Cas 0

Nous avons choisi une heuristique gloutonne en nous fondant sur notre expérience. Ce type d'algorithme fournit de bons résultats (environ 10% de l'optimale) rapidement et est adaptable, on pourra donc le modifier pour traiter d'autres cas.

L'heuristique *greedy* ou gloutonne consiste à initialiser une solution en choisissant le dépôt comme le premier nœud visité, puis à trouver à chaque étape le nœud de demande non visité le plus proche du dernier nœud visité. La solution est donc construite en ajoutant successivement les nœuds les plus proches jusqu'à ce qu'il n'y ait plus de nœud de demande non visité. Enfin, la solution est complétée en ajoutant le trajet retour au dépôt donc son coût associé. On calcule ensuite la solution finale pour notre problème grâce à une fonction permettant de calculer les chemins les plus courts entre chaque nœud de demande.

4.2.2 Cas 1

Dans une modélisation exacte du cas 1, ainsi que tous les cas ultérieurs, nous ne pouvons plus faire le même choix qu'au cas 0, c'est à dire se restreindre aux nœuds de demande car les drones peuvent être déployés à partir de n'importe quel nœud du graphe routier. Pourtant, sur les instances données, même un simple modèle de voyageur de commerce sur le graphe routier entier ne tourne pas dans un temps raisonnable. Alors, prenant en compte les drones et les contraintes qu'ils imposent, le problème se complexifie et est encore moins solvable en un temps raisonnable. Ainsi, des méthodes approchées sont nécessaires pour traiter ce cas.

Reduced MIP Heuristic

Pour cette heuristique, nous avons fait le même choix que pour le cas 0, *i.e.* se restreindre aux nœuds de demande et au dépôt. Ainsi on ne déploie les drones qu'à partir de ces nœuds. On représente ce problème réduit par un modèle mathématique fondé sur le TSP mais adapté à l'utilisation des drones.

Dans notre modélisation, la notation $d_{\mathcal{A}}$ pour tout ensemble de nœuds $\mathcal{A} \in \mathcal{I}_d$ désignera la somme des demandes des nœuds de $\mathcal{A}$.

Pour se faire, on utilise à l'instar du cas 0 la variable de décision binaire x_{ij} qui vaut 1, si le véhicule livre le nœud $j \in \mathcal{I}_d$ directement après le nœud $i \in \mathcal{I}_d$, et 0 sinon. De plus, on crée la variable de décision $y_{ij}^d \in \mathbb{N}^+$ qui correspond au nombre de colis livrés par le drone $d \in \{1, 2\}$ au lieu $j \in \mathcal{I}$ à partir du véhicule stationné au lieu $i \in \mathcal{I}$.

Enfin, on pose $T_i \in \mathbb{R}^+$, la variable de coût associée à la durée de stationnement du véhicule au lieu $i \in \mathcal{I}$

$$\min \sum_{i\in\mathcal{I}_d}\sum_{j\in\mathcal{I}_d} \hat{c}_{ij}x_{ij} + \sum_{i\in\mathcal{I}_d} T_i$$

$$s.c. \sum_{k\in\mathcal{I}_d} x_{ki} = \sum_{j\in\mathcal{I}_d} x_{ij}, \qquad \forall i \in \mathcal{I}_d \qquad (4.1)$$

$$\sum_{j\in\mathcal{I}_d} x_{ij} \leqslant 1, \qquad \forall i \in \mathcal{I}_d \qquad (4.2)$$

$$\sum_{j\in\mathcal{I}_d} x_{0j} = 1, \qquad (4.3)$$

$$d_j \cdot \sum_{i\in\mathcal{I}_d} x_{ij} + \sum_{i\in\mathcal{I}_d} (y^1_{ij} + y^2_{ij}) = d_j, \qquad \forall j \in \mathcal{I}_d \qquad (4.4)$$

$$y^o_{ij} \leqslant d_j \cdot \sum_{k\in\mathcal{I}_d} x_{ki}, \qquad \forall i \in \mathcal{I}_d,\ \forall j \in \mathcal{I}_d,\ \forall o \in \{1,2\} \qquad (4.5)$$

$$d_{\mathcal{I}_d\setminus s} \cdot x_{ab} \leqslant d_{\mathcal{I}_d\setminus s} \cdot \sum_{i\in s}\sum_{j\in\mathcal{I}_d\setminus s} x_{ij} + \sum_{o\in\{1,2\}}\sum_{i\in s}\sum_{j\in\mathcal{I}_d\setminus s} y^o_{ij}, \qquad \forall a \in s,\ \forall b \in s,\ \forall s \in \mathcal{S}_d \qquad (4.6)$$

$$T_i \geqslant 30 \cdot \sum_{j\in\mathcal{I}_d}\sum_{o\in\{1,2\}} y^o_{i,j} + 60 \cdot x_{ki}, \qquad \forall i \in \mathcal{I}_d \qquad (4.7)$$

$$T_i \geqslant 30 \cdot \sum_{j\in\mathcal{I}_d} y^o_{i,j} + 2 \cdot \sum_{j\in\mathcal{J}} e_{ij}y^o_{ij}, \qquad \forall i \in \mathcal{I}_d,\ \forall o \in \{1,2\} \qquad (4.8)$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in \mathcal{I}_d,\ \forall j \in \mathcal{I}_d \qquad (4.9)$$

$$y^o_{ij} \in \mathbb{N}^+, \qquad \forall i \in \mathcal{I}_d,\ \forall j \in \mathcal{I}_d,\ \forall o \in \{1,2\} \qquad (4.10)$$

$$T_i \in \mathbb{R}^+, \qquad \forall i \in \mathcal{I}_d \qquad (4.11)$$

L'objectif est de minimiser le temps passé pour compléter toute la livraison, c'est-à-dire minimiser la durée totale du transit du véhicule ainsi que la durée (relâchée) de stationnement du véhicule.

La contrainte (4.1) dit que si le camion visite un nœud, il doit aussi le quitter. La contrainte (4.2) dit que chaque nœud ne peut être visité par le camion qu'au plus une fois. La contrainte (4.3) dit que le dépôt doit être visité par le camion. La contrainte (4.4) dit que le nombre total de colis livrés par le véhicule et les drones à un nœud donné doit être égal à sa demande. La contrainte (4.5) dit qu'un drone ne peut être déployé d'un nœud que si ce nœud est visité par le camion.

La contrainte (4.6) caractérise la contrainte de sous-tours. Nos sous-tours sont gérés par des *callbacks* qui identifient un sous-tour s et ajoute cette contrainte pour l'éliminer. Pour le cas 1 on considère un tour comme étant l'ensemble de nœuds accessibles au dépôt selon les chemins des véhicules sélectionnés. Si cet ensemble ne contient pas tous les nœuds de demande, il s'agit d'un sous-tour qui doit être éliminé. Notre contrainte dit que soit toute la demande du sous-tour doit être couvert par des drones, soit le camion doit sort du sous-tour, ou soit toute la demande en dehors du sous-tour est couvert par des drones.

Les contraintes (4.7) et (4.8) stipulent que le temps passé par le camion sur un nœud doit être à la fois supérieur à la durée des livraisons qu'il fait à ce nœud, mais également supérieur à la durée nécessaire pour que les drones complètent leurs livraisons à partir de ce nœud. À ces temps de livraison s'ajoute le temps chargement des drones avant chaque livraison. Cette contrainte est une relaxation de la contrainte actuelle car la nôtre ne prend pas en compte le temps d'attente d'un véhicule lorsque celui-ci doit interagir avec un autre véhicule qui réalise une tâche ne pouvant être interrompue. Ces temps d'attente sont ajoutés lors de la création de la solution finale après l'optimisation. Les contraintes (4.9), (4.10) et (4.11) définissent les domaines de nos variables.

Greedy Path Heuristic

Pour la résolution du cas 1, nous avons créé une fonction *compute_time_savings* qui permet de calculer les gains de temps en utilisant les drones plutôt que le camion pour servir les nœuds de demande. Cette fonction parcourt les nœuds de demande et calcule la différence entre le temps nécessaire pour le camion seul et celui pour les drones et le camion. Si la différence est positive, elle l'ajoute à un dictionnaire *time_savings* associant les économies de temps à une route particulière. Finalement, le dictionnaire est trié selon les économies de temps les plus élevées. Ensuite il suffit de créer la solution finale en changeant l'itinéraire du camion et en affectant les nouveaux déplacements aux drones. Notons que dans l'algorithme (1) nous entendons par « temps de déplacement » le temps total, *i.e.* comprenant la livraison du colis.

Nous avions fait l'hypothèse que cette heuristique ne donne pas de « bons » résultats dans la mesure où l'on part d'une solution initiale du cas 0 dans lequel on fait un TSP sur tous les nœuds de demandes. Ainsi le gain de temps en utilisant les drones sera faible puisque ces derniers auront peu de distance à parcourir (du même ordre de grandeur que celles du camion), ne profitant pas du potentiel géodésique plus avantageux pour les drones.

Dès lors, une stratégie tirant profit de cet avantage consisterait à identifier des *clusters* de nœuds de demande isolés. Il suffirait alors de résoudre un TSP sur un sous-ensemble de nœuds de demande « regroupés » (proches les uns des autres) et ensuite de sous-traiter les nœuds de demande restants plus éloignés par les drones.

Cependant suite à l'implémentation et aux tests de cette approche heuristique par *clusters*, il s'est avéré que cette dernière méthode ne donnait pas de meilleurs résultats, si ce n'est des résultats moins bons en comparaison à l'approche retenue.

4.2.3 Cas 2

Super Node Heuristic

Nous avons implémenté une heuristique fondée sur de la recherche locale afin d'accélérer la tournée. Nous avions donc besoin d'une solution initiale et d'un processus d'amélioration de cette solution. Nous voulons partir d'une « bonne » solution initiale en terme de valeur objectif.

Nous partons de l'idée que la tournée (*i.e.* l'ordre de visites des nœuds de demandes) d'une bonne solution pour le problème de livraison avec drones ressemblerait à la tournée optimale du TSP avec quelques « déformations » (notamment à cause de contraintes du graphe routier). Par conséquent, la solution initiale est construite à partir de la solution optimale du TSP.

Il reste maintenant à affecter un véhicule (camion ou drone) à chaque nœud de demande. Nous décidons que les nœuds dont la demande est supérieure à 1 seront affectés au camion car quel que soit le nombre de colis, la durée de livraison sera la même. Nous appellerons ces nœuds des « super nœuds ». Les autres nœuds seront affectés dans la mesure du possible en suivant l'ordre : *truck*, *drone*, *drone* etc... ou sinon *truck*, *drone*, *truck*.

Le processus d'amélioration se passe sur au niveau de l'affectation nœud-véhicule. Nous améliorons l'affectation nœud-véhicule notamment en inversant les drones qui s'occupent de 2 nœuds consécutifs (afin d'être sur que le premier drone lancé se dirige vers le nœud de demande le plus éloigné, sinon le camion attendra ce drone plus longtemps inutilement).

En outre, nous avons essayé une autre approche provenant de l'article *Algorithms for Solving the Vehicle Routing Problem with Drones* de Schermer, Moeini et Wendt, 2018. Cette approche comprend deux phases (TPH pour *Two Phases Heuristic*) : une phase d'initialisation qui construit une solution de VRP fondée sur du *clustering initialisation*, l'autre est une phase d'optimisation dans laquelle on incorpore les drones. L'idée était donc d'appliquer cet algorithme à notre cas où seul le

camion livre. Cependant, comme vu précédemment, le *clustering* n'est pas une solution adéquate pour ce problème. Donc la solution initiale sans drone de la phase 1 ne sera pas optimale pour le problème VRPD. De plus, le TPH construit des solutions en prenant en compte des contraintes qui ne sont pas imposées dans notre cas, comme l'autonomie d'un drone.

En conclusion, on peut en déduire que pour de grandes instances, cette heuristique sera moins performante que celle que nous allons utiliser. En effet, l'étape qui consiste à scinder les nœuds en *clusters* puis revenir à un véhicule est superflue dans notre cas et complexifie notre algorithme.

Improving Path Heuristic

Une autre méthode que nous avons utilisée pour résoudre le cas 2 est de partir d'une solution de TSP et trouver les livraisons du camion où on peut gagner du temps en utilisant plutôt des drones. L'idée ici est que si dans sa tournée le camion doit aller livrer la demande d'un nœud k entre ses livraisons aux nœuds i et j, il est peut-être plus intéressant que à l'arrivée au nœud i, le camion largue ses drones pour livrer au nœud k, puis continue son trajet au nœud j où il récupère les drones une fois leurs livraisons complètées. Il est clair que la demande du nœud k doit être inférieure ou égale au nombre de drones disponibles, sinon plusieurs allers-retours avec les drones seront nécessaire pour satisfaire d_k, ce qui risque d'augmenter le temps total plutôt que le diminuer.

Plus généralement, cette heuristique prend une solution optimale au TSP et calcule l'ensemble des chemins de la tournée bornés par deux livraisons du camion (ou bornés par une livraison du camion et une visite du dépôt) de manière à ce que la demande totale livrée au sein d'un chemin ne dépasse pas le nombre de drones disponibles. Pour chacun de ces chemins, on évalue le gain de temps qu'on aurait si, au premier nœud du chemin, le camion larguait des drones pour livrer toute la demande au sein du chemin avant de faire sa livraison à ce 1^{er} nœud et se déplacer directement au dernier nœud du chemin où il récupère les drones avant de faire sa livraison à ce nœud. L'algorithme qui calcule ces chemins est décrit dans l'annexe (*c.f.* 2). Tous les chemins qui ont un gain de temps positif sont rentrés dans un modèle mathématique qui va sélectionner les chemins qui maximisent le gain de temps sous la contrainte que les chemins sélectionnés doivent être disjoints.

Afin de modéliser ce cas, nous définissons $\mathcal{C}$ l'ensemble de chemins considérés. Nous posons également $g_i \in \mathbb{R}^+$ qui correspond au temps gagné si on sélectionne le chemin $i \in \mathcal{C}$ pour qu'il soit effectué par des drones. On a aussi $o_{i,j} \in \{0,1\}$, qui est égal à 1 si les chemins $i \in \mathcal{C}$ et $j \in \mathcal{C}$ se chevauchent (ils ont des nœuds en commun en dehors des nœuds à leurs bornes) et 0 sinon. $a_{i,j} \in \mathbb{R}^+$ représente le temps supplémentaire gagné si les chemins $i \in \mathcal{C}$ et $j \in \mathcal{C}$ ne sont pas consécutifs (les drones peuvent ainsi être récupérés un peu plus tard car ils ne sont pas immédiatement utilisés). Posons $z_i \in \mathbb{R}^+$ la variable de coût correspondant au temps supplémentaire gagné en sélectionnant le chemin $i \in \mathcal{C}$.

Les variables de décisions de ce modèle sont x_i qui prend pour valeur 1, si le chemin $i \in \mathcal{C}$ est sélectionné pour être satisfait par des drones, et y_{ij} qui vaut 1 si le chemin $i \in \mathcal{C}$ et $j \in \mathcal{C}$ sont sélectionnés en même temps.

$$\max \sum_{i \in \mathcal{C}} g_i x_i + \sum_{i \in \mathcal{C}} z_i$$

$$\begin{aligned}
s.c.\ & x_i + x_j \leqslant 1, && \forall i \in \mathcal{C},\ \forall j \in \mathcal{C},\ \text{si } o_{i,j} = 1 && (4.12)\\
& x_i + x_j \leqslant y_{ij} + 1, && \forall i \in \mathcal{C},\ \forall j \in \mathcal{C} && (4.13)\\
& 2 \cdot y_{ij} \leqslant x_i + x_j, && \forall i \in \mathcal{C},\ \forall j \in \mathcal{C} && (4.14)\\
& z_j \leqslant a_{i,j} \cdot y_{ij} + M_j(1 - y_{i,j}), && \forall i \in \mathcal{C},\ \forall j \in \mathcal{C} && (4.15)\\
& x_i \in \{0,1\}, && \forall i \in \mathcal{C} && (4.16)\\
& y_{ij} \in \{0,1\}, && \forall i \in \mathcal{C},\ \forall j \in \mathcal{C} && (4.17)\\
& z_i \in \mathbb{R}^+, && \forall i \in \mathcal{C} && (4.18)
\end{aligned}$$

Ici M_j est une borne supérieure à z_j et correspond à la différence entre le coût en temps du chemin $j \in \mathcal{C}$ si celui-ci est satisfait par des drones et le coût en temps du camion dans ce cas. Il s'agit du temps passé par le camion à attendre que tous les drones le rejoignent à la fin du chemin j.

Notre objectif est de maximiser le temps gagné sur les chemins sélectionnés. La contrainte (4.12) dit que deux chemins sélectionnés ne peuvent pas se chevaucher et les contraintes (4.13) et (4.14) définissent nos variables y. La contrainte (4.15) dit que le temps supplémentaire gagné en sélectionnant le chemin $j \in \mathcal{C}$ est égal au temps supplémentaire minimal gagné selon les autres chemins sélectionnés. Les contraintes ((4.16) - (4.17) définissent les domaines de nos variables.

4.2.4 Cas 3

Le cas 3 est particulièrement difficile à mettre en oeuvre car il vient casser le caractère discret de notre problème en permettant aux drones de retourner au camion au cours de son transit entre deux nœuds. Il a notamment fallu mettre en place de nouvelles structures de données pour ce cas que nous n'avons finalement pas pu résoudre en son entièreté dans le temps qui nous était impartie.

Une approche envisagée pour ce dernier cas fut d'utiliser la solution obtenue grâce au *Path Heuristic* et d'améliorer et changer la solution en réduisant le temps d'attente du camion grâce à de nouveaux nœuds artificiels, malheureusement il s'est avéré que la majorité du temps « perdu » se trouvait être du temps de déplacement du camion. Ainsi le rapport coût-bénéfice d'un tel changement n'en valait pas la peine.

Une autre approche fut de « clusteriser » les nœuds de demandes par proximité et de sélectionner celui qui minimisait la demande totale à livrer (étant donné que les drones ne peuvent livrer qu'un colis à la fois). Une fois ce *cluster* identifié, on allait résoudre le problème avec la *Path Heuristic* sur un sous-ensemble des nœuds de demandes, puis il suffisait d'identifier le centroïde du *cluster* assigné aux drones et de créer de nouveaux nœuds artificiels (nouvelles coordonnées GPS) minimisant la distance à ce dernier depuis le sous-tours du camion. Malheureusement nous n'avons pas pu terminer l'implémentation de cette méthode et nous n'avons donc pas de résultats à présenter pour ce dernier cas.

5 Expérimentation et résultats

L'ensemble du code réalisé et utilisé pour effectuer ces tests comparatifs est disponible en libre accès sur ce dépôt Github : https://github.com/mdeboute/VRPWD. L'implémentation a été faite en Python et les tests ont étés effectués sur le serveur `infini2` du CREMI [1] possédant deux processeurs Intel Xeon X5570 64 bits de quatre coeurs chacun, cadencés à 3.2 GHz pour 55 Go de mémoire vive. La totalité des 16 coeurs ont étés utilisés pour effectuer nos tests. Nous avons décidé de fixer un temps limite de 3600 secondes soit 1 heure pour chaque MIP. Enfin, notre modèle de programmation linéaire en nombres entiers fut implémenté avec l'API [2] gurobipy [3] et le solveur que nous avons utilisé pour le résoudre était Gurobi [4] 9.1.0.

Pour la représentation graphique des résultats, le point vert correspond au dépôt, les points rouges correspondent aux nœuds possédant une demande non nulle (il suffit de cliquer sur le nœud pour en voir la demande et le passage de la souris sur un nœud révèle aussi son index), tandis que les points bleus ont une demande nulle. Les arcs noirs représentent le trajet du camion, les arcs verts celui du premier drone et les arcs jaunes celui du second drone. Des images types de nos programmes développés ainsi que leurs sorties sont disponibles en annexe (*c.f.* A).

En ce qui est des nos attentes et prédictions sur les résultats, les spécifications des différents cas à traiter nous en indiquent beaucoup sur ce sujet. En effet, plus le numéro du cas augmente, plus on a de liberté sur les déplacements faits, on s'attend donc à ce que le temps total d'une tournée se réduit lorsque le numéros du cas augmente. De plus, en considérant des méthodes de résolution relativement similaires, on s'attend à ce que le temps de résolution augmente avec le numéro de cas, puisque que le problème se complexifie.

Nous avons pris la décision de noter pour les différents cas, pour chaque algorithme et pour chaque instance, la valeur de la fonction objectif, le temps de résolution, le gap (lorsque cela était possible), et la gain par rapport au cas 0. Ces informations nous semblent être les plus pertinentes à évaluation de la qualité des algorithmes, pour ce problème.

1. Centre de Ressources pour l'Enseignement des Mathématiques et de l'Informatique
2. *application programming interface* ou « interface de programmation d'application »
3. https://pypi.org/project/gurobipy/
4. https://www.gurobi.com/solutions/gurobi-optimizer/

5.1 Cas 0

Instance	Valeur de l'objectif (sec)	Gap (%)	Temps de résolution (sec)
Instance 1	4202.00	0.00	0.06
Instance 2	4952.00	0.00	0.07
Instance 3	7717.00	0.00	0.53
Instance 4	8805.00	0.00	3.21
Instance 5	9956.00	0.00	9.35

Table 5.1 – Résultats du MIP pour le cas 0

Les résultats obtenus sont conformes à nos attentes pour ce cas. Le problème du TSP se résout assez facilement avec un bon modèle mathématique et sur un nombre de nœuds de demande limité. Les résultats du modèle MIP montrent bien qu'on obtient des solutions exactes presque instantanément sauf pour les instances 4 et 5 où le temps de résolution tend drastiquement à augmenter.

Instance	Valeur de l'objectif (sec)	Gap (%)	Temps de résolution (sec)
Instance 1	4328.00	3.00	0.00
Instance 2	5426.00	9.57	0.01
Instance 3	9158.00	18.67	0.01
Instance 4	9854.00	11.91	0.01
Instance 5	10964.00	10.12	0.01

Table 5.2 – Résultats de l'heuristique du plus proche voisin pour le cas 0

En revanche, l'heuristique gloutonne trouve une solution instantanément sur chacune des instances avec un Gap d'environ 10% sur les plus grandes. Avec des instances plus grandes et plus complexes, l'heuristique gloutonne pourrait être intéressante mais sur les instances tels qu'elles sont, seul le résultat du MIP nous intéressera dans la suite et servira de base pour mesurer la pertinence des nos méthodes dans les cas suivants.

5.2 Cas 1

Instance	Valeur de l'objectif (sec)	Gap (%)	Temps de résolution (sec)	Gain (%) par rapport au cas 0
Instance 1	2718.00	0.00	299.69	35.32
Instance 2	3451.00	6.36	3600.00	30.31
Instance 3	6341.00	44.11	3600.00	17.83
Instance 4	x	x	3600.00	x
Instance 5	x	x	3600.00	x

Table 5.3 – Résultats du MIP relâché pour le cas 1

Parmi les méthodes utilisées pour le cas 1, nous remarquons que le MIP relâché fournit les meilleurs gains de temps sur la tournée, allant de 17% à 35% sur les trois premières instances où cette méthode tourne à complétions. Ces résultats sont certainement impressionnants, même en comparaison à ceux du cas 2, le temps de résolution de cette méthode est très long, prenant 5 minutes à résoudre la première instance à l'optimal, et n'arrivant pas à résoudre les autres instances à l'optimale en 1 heure. Ceci est due à notre fonction de sous-tours qui est moins performante que celle du TSP à cause de la prise en compte des drones. Une amélioration sur cette fonction ferait du MIP réduit notre meilleure méthode pour ce cas.

Instance	Valeur de l'objectif (sec)	Gap (%)	Temps de résolution (sec)	Gain (%) par rapport au cas 0
Instance 1	3720.18	36.87	0.09	11.47
Instance 2	4843.76	46.72	0.08	2.19
Instance 3	6772.70	50.92	0.43	12.24
Instance 4	8314.15	x	2.60	5.57
Instance 5	9370.86	x	7.00	5.88

Table 5.4 – Résultats de l'heuristique *Greedy Path* pour le cas 1

La méthode *Greedy Path*, elle, nous donne un gain moins significatif allant de 2% à 12%, mais son temps de résolution est beaucoup plus raisonnable avec un maximum 7 secondes sur l'instance 5.

5.3 Cas 2

Instance	Valeur de l'objectif (sec)	Temps de résolution (sec)	Gain (%) par rapport au cas 0
Instance 1	3066.56	0.02	27.02
Instance 2	3784.28	0.05	23.58
Instance 3	5350.64	0.11	30.66
Instance 4	6312.27	0.24	28.31
Instance 5	8014.47	0.50	19.50

TABLE 5.5 – Résultats de l'heuristique *Super Node* pour le cas 2

Le gain apporté par *Super Node* va de 19 à 30% par rapport au cas 0, ce qui est non négligeable. L'hypothèse de livrer les super nœuds uniquement avec le camion est peut être un peu trop restrictive ou bien la définition du super nœud pourrait être restreinte aux nœuds dont la demande est strictement supérieure à 2. En effet, un super nœud très éloigné de la direction de la tournée avec une demande de 2 forcerait le camion à prendre un chemin plus long alors que les 2 drones pourraient le livrer et gagner du temps, et c'est ce que nous observons sur certaines instances.

De plus nous pourrions améliorer la solution du point de vue de l'ordre de visite des nœuds afin de déformer légèrement l'ordre de tournée (en utilisant le voisinage 2-opt par exemple) tout en gardant par la suite l'étape d'amélioration par l'affectation nœud-véhicule.

Aussi, une autre idée consisterait à faire des aller-retour entre l'amélioration de l'affectation nœud-véhicule et l'amélioration de l'ordre de la tournée afin de parcourir plus l'espace de recherche et de converger vers une meilleure solution. Dans un premier temps, nous améliorerons l'affectation, ensuite, à partir de cette affectation, nous trouverons une autre ordre de visite des nœuds améliorant, puis à partir de cet ordre trouvé une meilleure affectation etc...

Instance	Valeur de l'objectif (sec)	Temps de résolution (sec)	Gain (%) par rapport au cas 0
Instance 1	2807.00	0.08	33.20
Instance 2	3504.00	0.09	29.24
Instance 3	5117.00	0.45	33.69
Instance 4	5978.00	2.71	32.11
Instance 5	6787.00	6.49	31.83

TABLE 5.6 – Résultats de l'heuristique *Path* pour le cas 2

En analysant ces résultats, on observe qu'hormis les instances 1 et 2 dans lesquelles notre MIP relâché du cas 1 trouve des meilleures solutions, on remarque que dans le reste des instances, les heuristiques du cas 2 surpassent largement celles des cas précédents avec des gains de temps entre 19% et 30%. L'heuristique *Path* en particulier trouve des gains de temps d'environ 30% de manière récurrente sur toutes les instances avec un temps de calcul faible de moins de 7 secondes.

6 Conclusion

Dans ce projet, nous avons étudié 4 cas (2.1) du problème de livraisons de colis assistées par drones et nous avons implémenté des algorithmes pour comparer la valeur de la fonction objectif du cas 0 (TSP) avec les 3 autres cas afin d'étudier l'impact d'un ajout de 2 drones sur le temps de livraison. À l'issue d'expérimentations numériques sur des instances fournies par le groupe La Poste, nous en sommes arrivés à la conclusion que dans les conditions fixées pour ce projet, il apparaît que l'ajout de drones pour suppléer le camion améliore de manière significative le temps total de la tournée(5). En particulier, pour l'heuristique *Path* (cas 2), on observe un gain de temps d'environ 30% pour l'ensemble des 5 instances, par rapport au cas 0 dans lequel seul le camion livre. Ainsi, l'hypothèse selon laquelle les drones peuvent partir d'un point du graphe routier et être récupérés à un point éventuellement différent (cas 2) présente les meilleurs résultats. La possibilité de lâcher les drones en route en n'importe quel point du parcours, donc non nécessairement sur les nœuds du graphe routier (cas 3) apporterait possiblement un gain de temps dans la tournée lorsque le camion a une grande distance à parcourir entre deux nœuds de demande, cependant nous n'avons pas de résultats numériques pour confirmer ou rejeter cette hypothèse car nous n'avons pas eu le temps de terminer l'implémentation notre heuristique traitant ce cas.

A Annexe

Algorithme 1 : Heuristique gloutonne du cas 1 - *compute_time_savings*

Data : Instance VRPWDData
Result : Dictionnaire des nœuds de demande à servir par drones associés à leurs gains de temps par rapport à la solution du cas 0

```
1  Créer la solution initiale à l'aide du modèle MIP ou de l'algorithme greedy du cas 0;
2  Créer le dictionnaire vide time_savings;
3  for chaque nœud présent dans la solution initiale do
4      if le nœud est un nœud de demande then
5          On enregistre T1, le temps de déplacement du camion en partant du nœud précédent
           jusqu'à ce nœud;
6          On enregistre T2, le temps de déplacement du camion en partant de ce nœud jusqu'au
           prochain;
7          On enregistre T3, le temps de déplacement du drone en partant du nœud précédent
           jusqu'à ce nœud;
8          On enregistre T4, le temps du chemin le plus court en camion allant du nœud
           précédent jusqu'au nœud suivant;
9          On calcule delta comme la différence entre (T1 + T2) et (T3 + T4);
10         if delta > 0 then
11             On ajoute (nœud précédent, nœud, nœud suivant) comme clé et delta comme valeur
               dans le dictionnaire time_savings;
12         end
13     end
14 end
15 Trier time_savings par delta décroissant;
16 return time_savings;
```

Algorithme 2 : Algorithme de calcul des chemins améliorants

Data : Instance VRPWDData, Solution au cas 0
Result : Liste des chemins qui améliorent la solution lorsqu'on satisfait leur demande par des drones

Récupérer la tournée du camion dans la solution au cas 0;
Créer la liste vide *chemins_complets*;
Créer la liste vide *chemins_incomplets*;
for *chaque étape de la tournée du camion* **do**
    for *chaque chemin de la liste des chemins_incomplets* **do**
        ajouter l'*étape* au *chemin*;
    end
    if *l'étape est celle de livraison* ***ou*** *le début de la tournée* ***ou*** *la fin de la tournée* **then**
        for *chaque chemin de la liste des chemins_incomplets* **do**
            if *la demande du chemin dépasse 2* ***ou*** *on est à la fin de la tournée* **then**
                enlever ce *chemin* des *chemins_incomplets*;
                if *satisfaire la demande sur ce chemin réduit le coût de la solution* **then**
                    ajouter ce *chemin* aux *chemins_complets*;
                end
            end
        end
        créer un nouveau *chemin* avec cette *étape* comme premier élément;
        ajouter ce *chemin* aux *chemins_incomplets*;
    end
end
return *chemins_complets*;

```
❯ python3 src/vrpwdSolver.py -h
Usage: python3 vrpwdSolver.py <instance_directory> <case> <method>
Where:
<instance_directory> is the directory containing the instance files
<case> is the case of the problem to solve, can be a number in (0, 1, 2, 3)
<method> is the algorithm to use to solve the case, can be:
        For case 0:
                heuristic (h for short)
                mip
        For case 1:
                heuristic (h for short)
                mip
        For case 2:
                heuristic (h for short)
                pathheuristic (ph for short)
        For case 3:
                heuristic (h for short)
-v or --verbose is an optional argument to print all the information of the execution
-g or --graphic is an optional argument to plot the solution graph
Example: python3 vrpwdSolver.py data/instance_1/ 0 mip -v -g
```

FIGURE A.1 – Programme principale

```
❯ ./benchmark.sh
Usage: benchmark.sh -d <data_dir> -c <case> -m <method>
Where <case> is one of the following: 0, 1, 2, 3
Where <method> can be:
Case 0:
        mip
        heuristic
        h
Case 1:
        mip
        heuristic
        h
Case 2:
        heuristic
        h
        pathheuristic
        ph
Case 3:
        heuristic
        h
Example: ./benchmark.sh -d data/ -c 0 -m mip
❯ ./benchmark.sh -d data -c 0 -m mip
Experimental Campaign:
Data directory: data
Output directory: log/
Case: 0
Method: mip
Solving data/instance_1
Solving data/instance_2
Solving data/instance_3
Solving data/instance_4
Solving data/instance_5
Done!
```

FIGURE A.2 – *Benchmark Script*

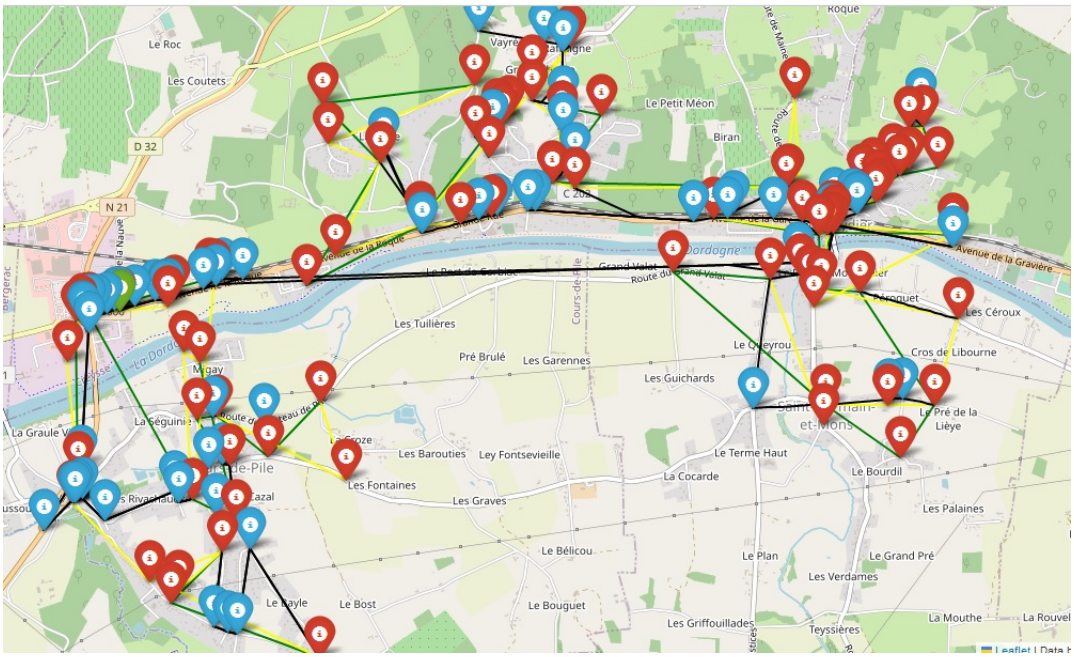

FIGURE A.3 – Solution du MIP pour l'instance 5 sur le cas 2

Implication des membres du groupe :

Kevin :

— Slides et texte pour la présentation AI4 Industry : 5/5
— Organisation du travail de l'équipe avec des outils collaboratifs (Notion) : 5/5
— Modélisation cas 0 : 5/5
— Cas 2 - Heuristique TPH (Article Schermer, Moeini et Wendt, 2018) : 3/5
— Cas 3 - Heuristique (Article Agatz, Bouman et Schmidt, 2018) : 2/5
— Rapport (Structure du rapport, Introduction, Formalisation, Algorithme de résolution du problème, Conclusion) : 5/5
— Slides et texte pour la présentation finale : 5/5

Martin :

— Recherche/Lecture articles scientifique : 2/5
— Génie logiciel et structure de données : 5/5
— Conception/Implémentation cas 0 : 4/5
— Conception/Implémentation cas 1 : 5/5
— Conception/Implémentation cas 2 : 2/5
— Conception/Implémentation cas 3 : 3/5
— Rédaction du rapport V1 : 5/5
— Rédaction du rapport V2 : 4/5

Lin :

— Implémentation du cas 0 : 4/5
— Modélisation du cas 1 : 5/5
— Implémentation du format des solutions : 5/5
— Codage du writer : 4/5
— Cas 1 - Heuristique MIP Réduit (modélisation et implémentation) : 5/5
— Cas 2 - Heuristique Chemins améliorants (modélisation et implémentation) : 5/5
— Conception cas 3 : 3/5
— Rapport (Notations, Modélisation, Reduced MIP Heuristic, Improving Path Heuristic, Résultats) : 5/5

Alan :

— Slides et texte pour la présentation finale : 5/5
— Implémentation du MIP cas 0 : 4/5
— Cas 3 - Heuristique (Article Agatz, Bouman et Schmidt, 2018) : 3/5
— Revue de littérature : 5/5
— Rapport : 2/5

Paul :

— Recherche / Lecture articles scientifiques : 2/5
— Parser / Structure des données / Affichage : 5/5
— Cas 2 - Heuristique Super Node : 5/5
— Rapport (Solutions, Heuristique Super Node) : 5/5

Bibliographie

Agatz, Niels, Paul Bouman et Marie Schmidt (2018). « Optimization approaches for the traveling salesman problem with drone ». In : *Transportation Science* 52.4, p. 965-981.

Carlsson, John Gunnar et Siyuan Song (2018). « Coordinated logistics with a truck and a drone ». In : *Management Science* 64.9, p. 4052-4069.

JG CARLSSON, S. SONG (2018). « Coordinated Logistics with a Truck and a Drone ». In.

Schermer, Daniel, Mahdi Moeini et Oliver Wendt (2018). « Algorithms for solving the vehicle routing problem with drones ». In : *Intelligent Information and Database Systems : 10th Asian Conference, ACIIDS 2018, Dong Hoi City, Vietnam, March 19-21, 2018, Proceedings, Part I 10*. Springer, p. 352-361.

WEELEN, N. (2022). « Drones de livraison : les impacts en termes de coûts de livraison et d'environnement ». In.

YS. CHANG, HJ. LEE (2018). « Optimal delivery routing with wider drone-delivery areas along a shorter truck-route ». In.

Z. WANG, J-B SHEU (2019). « Vehicle Routing Problem with drone ». In.