OPTIMISATION DE LA RÉPARTITION DES CLIENTS ENTRE LES DRONES DE LIVRAISONS : COMPARAISON D'ALGORITHMES ÉVOLUTIONNAIRES

Thibaut MESLIN
thibaut.meslin@hotmail.com – Lien du portfolio

CESI École d'ingénieur, Saint-Étienne-du-Rouvray, 76800, France

Résumé

Dans cet article, nous allons aborder la notion d'optimisation algorithmique de trajets de drones de livraisons. Ce problème repose sur plusieurs domaines d'études dont les véhicules autonomes, routage de véhicules ou encore optimisation algorithmique. Le but est de voir s'il est possible d'optimiser la répartition des clients entre ces drones pour économiser leur batterie et leur temps de trajet / utilisation.

Nous allons utiliser des algorithmes métaheuristiques et heuristiques. Ces algorithmes sont utilisés pour résoudre des problèmes d'optimisation difficile et pour lesquels il faudrait énormément de temps et de ressources pour obtenir une réponse. Nous axerons notre étude sur les algorithmes évolutionnaires. Ces algorithmes dérivent de la théorie de l'évolution de Darwin et répliquent donc le comportement d'espèces humaines dans leurs milieux naturels. On retrouve par exemple les mutations ou croisements entre individus et la notion d'élite. Nous utiliserons trois algorithmes : génétique, colonie d'abeilles artificielles (ABC) et optimisation par essaim de particules (PSO).

Nous réaliserons une expérimentation visant à comparer ces trois algorithmes pour évaluer leurs performances. Le but est de voir si ces algorithmes sont capables de trouver une solution acceptable en temps convenable. Les principales valeurs mesurées seront : le nombre d'itération, le score de fitness de la solution et la vitesse de l'algorithme en nombre d'itérations par seconde.

Abstract

In this paper, we will address the notion of algorithmic optimization of drone delivery routes. This problem is based on several fields of study including autonomous vehicles, vehicle routing and algorithmic optimization. The goal is to see if it is possible to optimize the distribution of customers between these drones to save their battery and their travel time / use.

We will use metaheuristic and heuristic algorithms. These algorithms are used to solve difficult optimization problems for which it would take a lot of time and resources to get an answer. We will focus our study on evolutionary algorithms. These algorithms derive from Darwin's theory of evolution and thus replicate the behavior of human species in their natural environment. We find for example mutations or crossbreeding between individuals and the notion of elite. We will use three algorithms: genetic, artificial bee colony (ABC) and particle swarm optimization (PSO).

We will perform an experiment to compare these three algorithms to evaluate their performance. The goal is to see if these algorithms are able to find an acceptable solution in a suitable time. The main values measured will be the number of iterations, the fitness score of the solution and the speed of the algorithm in number of iterations per second.

Mots-clés

Drones, Optimisation, Algorithmique, Problème de tournée de véhicules (VRP), Métaheuristique, Heuristique, Comparaison

I. INTRODUCTION

Dans un monde où la transition écologique s'accélère et les enjeux environnementaux prennent de plus en plus de place, il pourrait être intéressant de remplacer certains moyens de livraisons par de nouveaux moyens plus écologiques. Il serait possible de remplacer certains camions ou camionnettes servant à délivrer des colis par des drones autonomes. On remplacera donc l'essence ou le gasoil utilisé par une énergie purement électrique pouvant être générée de manière verte. Cette énergie pourrait par exemple être produite par des panneaux solaires ou de la récupération d'énergie ou de chaleur [13].

Également, dans un but d'optimisation temporel, certaines entreprises dans le secteur de la livraison comme Amazon investissent dans la recherche et la construction de drones de livraisons. Ces drones sont des appareils sans pilotes capables d'effectuer des trajets programmés à l'avance sur de plus au moins longues distances en fonction de leurs configurations.

Un grand nombre de configurations sont disponibles sur ces appareils et peuvent influer sur leur performances et autonomies. On peut retrouver les différentes composantes influentes suivantes :

- Les moteurs
- La batterie
- L'aérodynamisme
- Les matériaux
- Le poids global
- Le colis

Ces composantes sont toutes physiques et influencent directement le drone et ses performances. Nous n'évaluerons pas dans cet article les effets de ces composantes sur le temps de trajets des drones.

Nous allons nous concentrer sur la partie logicielle et algorithmique de ces drones. Le drone doit se déplacer depuis le ou les entrepôt(s) jusqu'au domicile du client puis se retourner à son entrepôt d'origine où à l'entrepôt suivant.

Le reste de ce papier se fera de la manière suivante. La section 2 décriera la situation actuelle des drones autonomes et leur place dans le monde actuel. Le pant légal de la solution sera décrit en partie 3. Un état de l'art sur l'existant sera réalisé en partie 4. L'expérimentation et les résultats de celle-ci seront présenté en partie 5 et 6.

II. EXISTANT

Les drones autonomes sont un sujet d'actualité et nous avons pu les voir se démocratiser dans divers domaines pendants ces dernières années. Ces domaines peuvent être : la livraison, la surveillance, l'inspection ou la cartographie par exemple. De nombreuses entreprises commencent à voir l'intérêt de ces technologies pour automatiser ou simplifier leur travail mais également un moyen de limiter leurs émissions polluantes en remplaçant des camions fonctionnant à l'essence ou au gasoil par des drones entièrement électrique. Cela pourrait permettre de limiter les contacts pendant les périodes de pandémies comme lors de la COVID-19 entre 2019 et 2022.

Concernant la recherche dans ces technologies, les travaux sur les drones autonomes se concentrent sur différents sujets comme :

- Planification de trajectoire
- Cartographie simultanée (SLAM)
- Perception de l'environnement
- Coordination multi-drone
- Surveillance de l'état de batterie
- Réduction de bruit

Des plateformes open-source de drones autonomes ont été développées, on peut citer : PX4, ArduPilot ou Dronecode. Ceux-ci ont rendu la conception et le développement de drones plus accessibles pour les chercheurs et développeurs. Ces plateformes ont offert des fonctionnalités de base pour la navigation, le contrôle, la communication mais également des bibliothèques de capteurs et de contrôleurs de vol [16].

Les drones autonomes ont trouvé des applications pratiques pour différentes entreprises. Les entreprises de livraison ou de commerce en ligne sont attirées par le fait de pouvoir livrer sans passer par une société de livraison tierce. De même dans le secteur agricole ou de surveillance, des drones autonomes peuvent être utilisés pour surveiller des champs de culture, prévenir des feux ou encore surveiller des zones à risques ou animaux sauvages [14][15]. Dans le secteur du bâtiment, ceux-ci peuvent être utilisés pour suivre l'avancement ou détecter des fissures ou instabilités structurelles [11][12].

Malgré toutes ces applications, il reste de nombreux défis à relever. Les drones autonomes doivent être capables d'opérer dans tous les milieux en détectant les obstacles et de les éviter, de voler peu importe les conditions météorologiques (pluie, vent, etc…) et enfin être capables de maintenir une communication fiable dans des environnements encombrés par de multiples ondes.

III. LÉGLISATION

III.1. France

En France, la réglementation relative aux drones autonomes est définie par la Direction Générale de l'Aviation Civile (DGAC) et régie par le code des transports [17][18].

Un drone est considéré comme un aéronef et est soumis à des règles strictes en matière de sécurité. L'utilisation des drones autonomes doit être autorisée et déclarée auprès de la DGAC. Les utilisateurs doivent être titulaires d'un certificat de pilote de drone et respecter les règles relatives aux distances de vol, hauteurs de vol et zones de vol interdites.

Chaque drone doit être équipé d'un système de géolocalisation et d'un système de signalisation lumineuse. Si un drone est équipé de caméra ou autres appareils de surveillance, il doit également respecter les règles de confidentialités et de protections de données éditées par la Commission Internationale de l'Informatique et des Libertés (CNIL).

Enfin, chaque utilisateur de drones autonomes doit souscrire à une assurance responsabilité civile pour couvrir d'éventuels dommages causés à des tiers par l'intermédiaire du drone.

Enfin, le sujet de véhicule autonome étant relativement nouveau, les lois évoluent régulièrement dues aux avancées technologiques et des enjeux de la sécurité publique. Il est donc plus que nécessaire de se tenir informé de l'évolution des règlementations en vigueur avant l'utilisation de drone autonomes.

Une chose importante à noter est que la législation concernant les drones autonomes est différente de celles d'autres véhicules autonomes comme les voitures autonomes. Ceux-ci étant par exemple régis par le code de la route et les lois relatives à la sécurité routière.

III.2. Europe

Les règlementations concernant les drones autonomes varient d'un pays à l'autre mais il existe néanmoins une règlementation européenne commune. Depuis 2019, la législation européenne sur les drones est harmonisée et applicable à tout État de l'Union Européenne.

Ce règlement sur les drones (UE 2019/947) est entré en vigueur en décembre 2020, établit des règles communes pour l'exploitation des drones, y compris drones sans pilote (autonome) [19]. Les principaux points de la règlementation sont les suivants :

- Classification en fonction du poids
- Enregistrement et formation obligatoires
- Équipements obligatoires (parachutes de secours par exemple)
- Interdiction de vol au-dessus des zones sensibles (aéroports, zones militaires, etc…)

La règlementation UE 2019/947 ne traite pas des questions de vie privée et de protection de données. Celles-ci sont régies par les lois en vigueur dans l'État d'application [19].

Les règles concernant les drones autonomes n'étant pas encore définies concrètement, ces drones sont soumis aux mêmes réglementations que les drones classiques. Toutefois, l'Agence Européenne de la Sécurité Aérienne (EASA) travaille sur un cadre réglementaire spécifique pour les drones autonomes, celui-ci étant mis à jour régulièrement pour prendre en compte les développements technologiques [19].

III.3. Hors Union Européenne

Les législations concernant les drones autonomes varient considérablement d'un pays à l'autre dans le monde mais voici quelques exemples sur d'autres continents.

Aux États-Unis, la Federal Aviation Administration (FAA) exige que tout drone volant dans l'espace aérien américain soit enregistré auprès d'une agence. Les drones autonomes sont soumis aux mêmes règles que les autres drones en ce qui concerne les exigences d'enregistrement et de vol [20].

Au Canada a mis en place une règlementation incluant des règles spécifiques pour les drones autonomes. Tout drone pesant plus de 250 grammes doit être enregistré et les opérateurs doivent obtenir une certification spéciale s'ils prévoient d'utiliser un drone autonome.

La Chine a mis en place une règlementation stricte pour les drones autonomes. Les opérateurs doivent s'inscrire auprès des autorités et les drones doivent être équipés de systèmes de géolocalisation.

Le Japon a des règles de drones strictes, y compris des drones indépendants. Les opérateurs doivent obtenir une pré-autorisation avant de conduire des drones et avoir des restrictions de vol dans des zones sensibles telles que les aéroports et les centrales nucléaires.

En Australie, les drones sont règlementés par l'Autorité de Sécurité de l'Aviation Civile (CASA). Les opérateurs de drones doivent détenir une licence et suivre des règles strictes en matière de sécurité. On retrouve également des restrictions de zones de vol [21].

Pour conclure, on pourrait dire que les pays hors de l'Europe n'ont pas encore de règlementation pour les drones autonomes et donc appliquent les mêmes règles que pour les drones classiques. Cependant, il existe des organisations internationales comme l'Organisation de l'Aviation Civile Internationale (OACI) ou encore l'Agence de l'Aviation Civile Internationale (ACI). Celles-ci travaillent à l'élaboration de normes et de règles pour les drones autonomes. Certains pays peuvent également établir des accords bilatéraux pour harmoniser les règlementations des drones.

IV. PROBLÈME

Ce problème peut être transposé à un problème de tournées de véhicules avec collecte et livraison ou VRPPD (Vehicule Routing Problem with Pickup and Delivery) [1][2].

Dans ce problème, on a un ou plusieurs entrepôts pour un ou plusieurs véhicules. Chaque véhicule doit passer dans un entrepôt pour récupérer un colis / marchandise et la livrer dans les dépôts concernés par ces colis. On peut transposer ce problème à notre situation : Nous avons un / plusieurs entrepôt(s) et un / plusieurs drone(s), chaque drone doit passer dans l'entrepôt prendre un et un seul colis et le livrer devant la porte du client. La capacité du drone est donc limitée à un seul colis (on fera ici abstraction de sa taille et de son poids) destiné à un et un seul client.

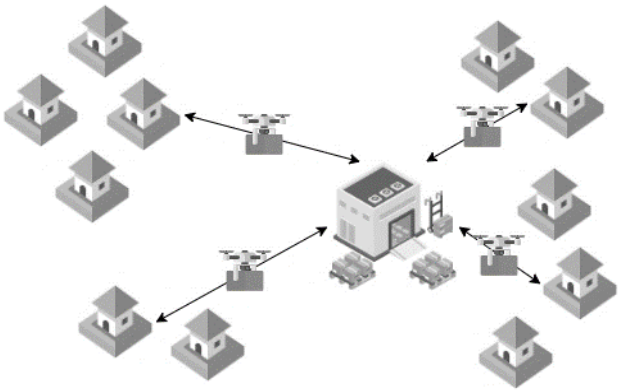

Fig. 1. Exemple problème VRPPD.

Le problème du VRPPD est un problème NP-Complet, c'est-à-dire qu'il ne peut pas être résolu exactement dans des temps acceptables. Comme on ne peut résoudre ce problème en temps acceptable, on peut essayer de tendre vers une solution optimale dans des temps acceptables

[9]. Ces problèmes sont souvent traités en des utilisant algorithmes heuristiques ou métaheuristiques.

IV.1. Théorie des graphes

Une façon de représenter le problème du VRPPD est de le formuler en utilisant la théorie des graphes. Celle-ci est une branche des mathématiques étudiant les relations entre les objets.

Un graphe est un ensemble de nœuds (appelés sommets) reliés par des lignes appelées arêtes ou arc. Le therme « arête » est plus souvent utilisé dans le cas d'un graphe non orienté alors que le mot « arc » est lui plus utilisé dans le cadre d'un graphe orienté. Il existe de nombreux types de graphes :

- Non orienté : Les arêtes peuvent être parcourues dans les deux sens.
- Orienté : Les arêtes ont un sens de parcours (représenté par une flèche), c'est-à-dire qu'elle ne peut être parcourue que dans un sens défini.
- Pondéré : Chaque arête possède un poids qui peut représenter une distance, un coût ou un temps par exemple.
- Connexe : Il existe un chemin entre chaque paire de nœuds, chaque nœud peut relier n'importe quel autre nœud.
- Complet : Chaque nœud est relié à tous les autres nœuds du graphe par une arête. Cela signifie qu'il y a $n(n-1)/2$ arêtes dans un graphe complet à n nœuds.

Une façon courante de représenter un problème du VRPPD est d'utiliser un graphe orienté pondéré où chaque sommet représente un client ou un entrepôt et chaque arc est pondéré par un temps ou une distance ou encore un coût.

Il est possible d'utiliser des algorithmes plus classiques pour résoudre ces problèmes comme celui de Dijkstra pour trouver le plus court chemin entre deux sommets ou encore celui de Bellman-Ford pour gérer les graphes avec des points négatifs ou encore celui de Floyd-Warshall pour calculer le plus court chemin entre tous les sommets du graphe.

IV.2. Algorithme génétique

Nous utiliserons ici un algorithme génétique [3][4]. Cet algorithme est une méthode de résolution de problème d'optimisation s'inspirant de la théorie darwinienne. Il a été créé par John Holland entre 1960 et 1970 puis fût popularisé par David Goldberg vers la fin des années 1980.

L'algorithme génétique est une méthode heuristique reposant sur une approche itérative et stochastique ou chaque itération correspond à une génération d'individu. Nous allons créer des solutions candidates sous formes de chaînes de caractères ou de vecteurs qui sont évalués par une fonction d'adaptation. Cet algorithme se base sur trois opérations principales :

- Sélection
- Croisement
- Mutation

La sélection consiste à choisir les individus les plus adaptés à la résolution du problème. Une fonction d'évaluation est définie et on compare chaque individu à l'aide de cette fonction pour trouver les individus les plus aptes et performants de la génération. Ces individus ont une chance plus grande d'être sélectionnés pour faire partie de la prochaine génération [3][4].

Le croisement consiste à mélanger les informations génétiques de deux individus pour créer une nouvelle génération. On peut faire un parallèle entre cette étape et l'étape de reproduction dans la nature. Le croisement est effectué en choisissant deux parents aléatoires et en mélangeant des parties de leur code génétique dans le but de créer une descendance dont le code génétique serait un mélange de celui des parents.

La mutation permet d'introduire de la diversité dans la population d'individus. Cette opération consiste à modifier aléatoirement le code génétique de certains individus. Cette opération est très importante car elle permet de ne pas tomber dans ce qui pourrait s'apparenter à un optimum local, c'est-à-dire une solution locale pouvant être très loin de l'optimum global.

On répète ces étapes plusieurs fois de suite jusqu'à ce qu'une solution acceptable soit trouvé ou qu'une condition d'arrêt ne soit atteinte. Une solution acceptable est une solution qui correspond à la fonction objective définie au préalable. Une condition d'arrêt pourrait être un nombre d'itération ou encore un temps d'exécution.

En résumé, un algorithme génétique est un algorithme s'inspirant de la sélection naturelle et de la génétique pour résoudre des problèmes complexes. Cet algorithme utilise des méthodes heuristique et stochastique pour trouver une solution optimale en utilisant des processus de croisement, mutation et sélectionnant des individus jusqu'à l'obtention d'une solution optimale.

IV.3. Colonie d'abeilles artificielles

Un algorithme de colonie d'abeilles artificielles ou Artificial Bee Colony Algorithm (ABC) est une métaheuristique inspirée du comportement des abeilles dans la nature et plus particulièrement de la manière dont elles cherchent leur nourriture [5][6]. Cet algorithme a été inventé par Dervis Karaboga et présenté pour la première fois en 2005 dans l'article « *An Idea Based on Honey Bee Swarm for Numerical Optimization* ».

Cet algorithme va rechercher un optimum global dans un espace de recherche multidimensionnel. On distingue trois types d'abeilles :

- Abeilles ouvrières
- Abeilles éclaireuses
- Abeilles butineuses

Les ouvrières sont responsables de la recherche locale de nourriture. Elles sont donc responsables des recherches dans la zone proche de la ruche en utilisant une recherche locale. Elles ont également la capacité d'échanger des informations avec les autres abeilles ouvrières.

Les abeilles éclaireuses sont responsables de la recherche globale de nourriture. Elles explorent des zones éloignées de la ruche, cherchant de potentielles sources de nourritures éloignées. Lorsqu'une nouvelle source de nourriture est découverte, l'abeille rentre à la ruche et communique la position de celle-ci aux autres abeilles.

Les butineuses, quant à elles, sont responsables de l'exploitation de la source de nourriture. Elles se rendent à la source de nourriture, collecte la nourriture présente sur place et la rapporte à la ruche.

On initialise une population de solutions contenant les trois types d'abeilles. Chaque solution est représentée par une position dans l'espace de recherche. À chaque itération :

- Les ouvrières explorent localement les positions de la population de solutions.
- Les éclaireuses explorent globalement l'espace de recherche afin de s'assurer qu'on ne tombe pas dans un optimum local avec la recherche de nouvelles solutions.
- Les butineuses exploitent les solutions les plus prometteuses

Les butineuses évaluent les solutions qu'elles ont exploitées et mettent à jour la population des solutions par un mécanisme de sélection et de remplacement. À la manière d'un algorithme génétique, les solutions les plus performantes sont sélectionnées pour la reproduction tandis que les moins performantes sont remplacées par de nouvelles solutions générées aléatoirement.

On ne stoppe l'algorithme que lorsqu'un critère d'arrêt est atteint (ce peut être un temps d'exécution ou un nombre d'itération par exemple).

Pour résumer, l'algorithme de colonie d'abeilles artificielles simule le comportement des abeilles lors de la recherche de nourriture. Il utilise une population composée de trois sortes d'individus pour explorer et exploiter l'espace de recherche afin de trouver les solutions les plus performantes.

IV.4. Essaim de particules

L'optimisation par essaim de particules ou PSO (pour Particle Swarm Optimization) est un algorithme d'optimisation stochastique simulant le comportement d'un essaim de poissons ou d'oiseaux [7][8].

Cet algorithme d'optimisation utilise une population de particules se déplaçant à travers un espace de recherche dans le but de trouver l'optimum global. Ces particules sont guidées par leur propre meilleure position trouvée jusqu'à présent et la meilleure position trouvée par l'ensemble des particules de la population [7][8].

L'algorithme d'optimisation par essaim de particule se compose de cinq étapes :

- Initialisation
- Évaluation
- Mise à jour de la meilleure position personnelle
- Mise à jour de la meilleure position globale
- Mise à jour de la vitesse et de la position

Lors de l'initialisation, on va créer un ensemble de particules à des positions aléatoire dans l'espace de recherche. Chacune des particules représente une solution potentielle au problème actuelle.

On évalue ensuite chaque particule en calculant le résultat de sa fonction objectif. Cette fonction représente la qualité de la solution associée à la position de la particule dans l'espace de recherche. Cette fonction peut être définie en fonction à minimiser ou maximiser une certaine quantité, selon la nature de problème d'optimisation.

Par la suite, on va comparer la position de chaque particule avec sa meilleure position personnelle. Si la position actuelle est meilleure, on remplace la valeur stockée dans la meilleure position personnelle par sa valeur actuelle.

Ensuite, on va comparer la meilleure position personnelle de chaque particule avec la meilleure position globale actuelle. Si la meilleure position personnelle d'une particule est meilleure que la position globale actuelle alors on remplace cette valeur par la position de la particule en question.

Enfin, on met à jour la vitesse et la position de chaque particule. On utilise les équations suivantes :

- $V(t+1) = wV(t) + c1r1\big(P_{best} - X(t)\big) + c2r2(G_{best} - X(t))$
- $X(t+1) = X(t) + V(t+1)$

Avec :

- $V(t)$: vitesse actuelle
- $X(t)$: Position actuelle
- w : coefficient d'inertie
- P_{best} : meilleure position personnelle
- G_{best} : meilleure position globale
- $c1, c2$: coefficients d'apprentissages
- $r1, r2$: nombres aléatoires entre 0 et 1

On répète les étapes de l'évaluation jusqu'à la mise à jour de la vitesse et de la position jusqu'au déclenchement d'une condition d'arrêt.

Le choix des coefficients d'apprentissage et du coefficient d'inertie peut avoir un impact important sur les performances de l'algorithme. Il existe de nombreuses variantes du PSO qui modifient ces paramètres ou introduisent des modifications supplémentaires pour améliorer la convergence ou la diversité de l'ensemble des particules.

L'idée derrière cet algorithme est que chaque particule suit les autres particules en fonction de leur meilleure position personnelle et de la meilleure position globale. De ce fait, l'ensemble des particules se déplace coopérativement dans l'espace de recherche, explorant de nouvelles régions à la recherche de la meilleure solution possible.

Si nous devions résumer, l'algorithme d'optimisation par essaim de particules basée sur la simulation du comportement social des oiseaux. Cet algorithme utilise une approche itérative pour explorer l'espace de recherche afin de trouver la meilleure solution possible. Une population de particules représente une solution potentielle au problème, et chaque particule suit une trajectoire dans l'espace de recherche en fonction de sa position et de sa vitesse. Les particules se déplacent vers les meilleures solutions trouvées par elles-mêmes et par les autres particules de l'essaim.

V. EXPÉRIMENTATION

Nous allons réaliser une comparaison entre les trois algorithmes présentés ci-dessus. Pour permettre de comparer ceux-ci, nous allons établir un contexte et un périmètre d'étude bien défini ainsi que des paramètres communs. Ce tableau permettra par la suite de conclure sur l'algorithme le plus optimal à utiliser dans un contexte d'optimisation de trajet de drone autonome.

L'étude sera réalisée sur un notebook Jupyter à l'aide de l'IDE Visual Studio Code tournant sur un DELL INSPIRION 5502 (modèle sorti fin 2020) doté des caractéristiques suivantes :

- Processeur Intel Core i7-1165G7 (Quad-Core 1.2 GHz - 2.8 GHz / 4.7 GHz Turbo - 8 Threads - Cache 12 Mo)
- 16Go (2x8Go) de RAM DDR4 cadencés à 3600Mhz
- Puce graphique Intel Iris Xe Graphics (96 EU) cadencée à 1300Mhz
- SSD NVMe PCIe 1T
- Windows 11 Famille (version actuelle 22H2)

V.1. Contexte de l'étude

Pour cette étude, nous allons prendre un exemple relativement simple et n'impliquant pas de devoir utiliser une machine de haute puissance pour pouvoir être traité. Notre problème pouvant être assimilé à un problème du VRPPD, nous aurons au minimum besoin d'un drone, un entrepôt et un client à livrer.

Nous considèrerons que la fenêtre de temps s'étend sur toute la journée et que le service se fait sans interruption donc tout au long de la journée et sans pause. Cela ne parait pas infaisable car le(s) drone(s) n'ayant pas besoin de pilotes peuvent fonctionner sans action humaine et en dehors des plages horaires ouvrées.

Nous admettrons également que le(s) drone(s) à (ont) une capacité limitée à un seul colis peu importe sa taille ou son poids. Cela permettra de se rendre compte de la charge de travail et de la distance parcourue par le drone.

Il sera également posé que le(s) drone(s) à (ont) une autonomie illimitée. Cela permettant de ne pas avoir de contraintes liées à la batterie et d'éviter la contrainte de distance maximale à parcourir (secteur géographique).

Le schéma ci-dessous résume la situation sous forme d'un graphe pondéré par la distance entre l'entrepôt et la maison du client.

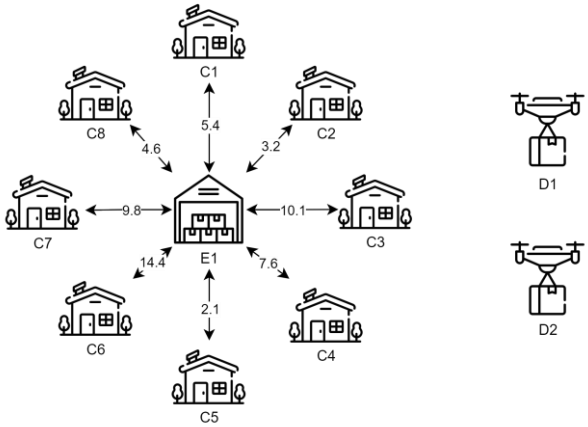

Fig. 2. Schéma du contexte.

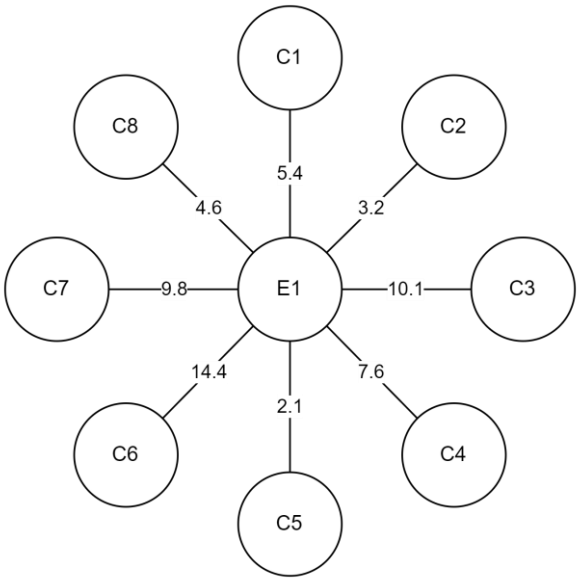

Fig. 3. Graphe non-orienté pondéré du contexte.

V.2. Objectif

L'objectif de cette comparaison est de voir quel algorithme nous donneras les meilleures performances pour résoudre le problème du VRPPD décrit dans le contexte ci-dessus. On cherchera à minimiser la distance parcourue par chaque drone, par exemple si la distance totale à parcourir est de 100 kilomètres avec deux drones, chaque drone devra faire environ 50 kilomètres.

Le but est d'aider ingénieur, chercheur et entreprises dans le secteur de la logistique à sélectionner l'algorithme le plus pertinent à utiliser dans leur cas. On va donc chercher à voir quel algorithme peux permettre d'économiser temps, argent et trajets.

V.3. Paramètres d'entrés

Pour cette étude, nous allons définir plusieurs paramètres d'entrés qui vont permettre de rendre l'expérience reproductible par quiconque souhaiterai la reproduire. Mais cela est également important pour pouvoir avoir des résultats vraiment comparables. Ces paramètres d'entrés sont divers, on pourrait citer par exemple :

- Nombre de drones
- Nombre d'entrepôts
- Vitesse des drones

Premièrement, nous prendrons en compte qu'il n'y a qu'un seul entrepôt dans notre situation. Nous pouvons supposer cela car les entrepôts sont en général construits non loin des villes et sont donc aisément accessible par des drones de livraisons.

Deuxièmement, nous supposerons que l'entreprise disposera de deux drones de livraison fonctionnant en simultané et que ceux-ci ont une autonomie illimitée.

Troisièmement, il sera admis que huit colis seront à livrer à huit clients différents répartis tout autour de l'entrepôt. La distance entre ces clients et l'entrepôt sera connue via la pondération des axes par la distance en kilomètre entre l'entrepôt et le domicile du client.

Quatrièmement, il sera dit qu'un drone ne peut transporter qu'un seul colis à la fois. Il sera donc nécessaire pour le drone de retourner à l'entrepôt après avoir déposé le colis chez le client pour prendre en charge un nouveau colis. Le poids et la taille du colis seront négligeables.

Dernièrement, nous supposerons que les drones voyageront à vitesse constante tout au long de leur parcours et qu'aucun temps d'arrêts ne sera fait et qu'aucune condition météorologique comme du vent ou de la pluie ne viendra réduire ou augmenter la vitesse.

Les tests se dérouleront sous un nombre d'itérations illimité en un temps maximal d'exécution d'une minute. L'algorithme pourra faire autant d'itération possible en une minute puis s'arrêtera une fois le temps dépassé. Il est également important de préciser que chaque algorithme ne sera pas lancé sur un thread mais que chaque algorithme sera lancé une fois l'exécution du précédent terminé.

V.4. Métriques d'évaluations

a - Fitness de la solution finale

Le but du score de fitness est de noter la solution produite dans le but de permettre une comparaison entre les différentes solutions produites. Cette valeur sera notre principal critère d'évaluation donc plus la valeur produite sera proche de 1, plus la qualité de la solution sera bonne et inversement plus la valeur sera proche de 0 ou négative, plus la solution sera mauvaise.

Notre but étant de minimiser la différence de distance parcourue entre les drones, plus la valeur de fitness sera proche de 1, plus cette différence sera minime et donc plus la solution sera correcte.

Cette valeur sera mesurée à partir d'une fonction qui sera décrite plus tard dans ce papier (cf. Fonctions communes), cette fonction sera commune à tous les algorithmes.

b - Nombre d'itération

Nous allons dans un premier temps évaluer les algorithmes en fonction du nombre d'itération qu'il va falloir à l'algorithme pour atteindre une solution convenable.

Nous comparerons en fonction du nombre d'itération total pour un lancement puis pour 5 lancements. Cela nous permettra de ne pas avoir de « coup de chance » sur un bon lancement et de faire une moyenne sur plusieurs lancés. Cela devrait permettre d'éviter ce problème et de réellement comparer les algorithmes.

On calculera également le temps pour réaliser une itération en moyenne et également le temps de l'itération la plus longue et la plus courte. Cela permet de savoir si l'algorithme peut bloquer sur une itération ou sur une condition particulière qui n'aurai pas encore été identifié.

c - Robustesse de l'algorithme

La robustesse d'une métaheuristique est une mesure importante et consiste à modifier le nombre d'entrées ou de facteurs simultanés (dans notre cas le nombre de drone par exemple). Tester cette robustesse permet de savoir si l'algorithme est capable de maintenir des performances élevées et stable sur des problèmes nécessitant plus de ressources que le cas simple.

Ceci va nous permettre de vérifier si notre algorithme sera capable de s'adapter à des problèmes nécessitant plus de ressources de calculs et plus de temps de traitement. Nous allons donc faire varier le nombre de clients et le nombre de drones utilisés pour effectuer les livraisons.

On s'assurera donc de la capacité de l'algorithme à fonctionner dans des conditions réelles. De plus, on saura si notre solution sera rapidement limitée ou non.

VI. ANALYSE DES RÉSULTATS

VI.1. Algorithmes et complexité

Tous les algorithmes utilisés ont été développés par mes soins et sont regroupés dans un notebook Jupyter disponible sur mon GitHub et disponible à l'adresse suivante : GitHub.

a - Fonctions communes

Il existe deux fonctions communes à tous les algorithmes cités plus tôt :

- *Generate_random_solution*
- *Calculate_fitness_vrppd*

Pour la première, elle est utile pour générer une solution aléatoire pouvant répondre à notre demande. Elle ne prend pas de paramètre mais utilise des variables globales relatives aux nombres de drones et la matrice de distance contenant les distances entre les sommets du graphe. Elle produit une solution sous forme de liste de liste ou chaque sous-liste correspond au trajet d'un drone. On rappelle que chaque sommet est séparé par un retour à l'entrepôt. Exemple de sortie :

[[0, 1, 0, 3, 0, 6, 0], [0, 2, 0, 4, 0, 5, 0, 7, 0, 8, 0]]

Elle utilise principalement des opérations $\mathbf{O(n)}$. Comme on crée k sous-liste(s) en fonction du nombre de drone, la complexité de cette fonction est donc $\mathbf{O(n^2)}$.

Pour le calcul du score de fitness d'une solution, cette fonction calcule la distance totale parcourue par chaque drone et y attribue un score de fitness en fonction de la différence de distance entre les distances totales parcourues par les drones. Pour calculer ce score de fitness, on utilise la formule suivante :

$$\boldsymbol{fitness} = \frac{\mathbf{1} - (\boldsymbol{distance_{max}} - \boldsymbol{distance_{min}})}{\frac{\boldsymbol{distance\_totale}}{\boldsymbol{k}}}{\mathbf{2}}$$

La complexité de cette fonction est égale à la complexité de la fonction précédente : $\mathbf{O(n^2)}$ avec n la taille de la matrice et k le nombre de drone.

b - Algorithme génétique

L'algorithme génétique utilise une fonction externe appelée *'mutation'* qui prend en paramètre une solution et échange un ou plusieurs clients entre les drones. La complexité de cette fonction est plutôt simple et possède donc une complexité de $\mathbf{O(1)}$.

La fonction principale se compose d'une boucle principale itérant jusqu'à une condition d'arrêt. Elle évalue le score de fitness pour chaque solution de la population dans la population. On retrouve une boucle *for* pour calculer la génération de la population et une boucle *while*, permettant de muter les individus dans le but de créer la nouvelle population. Il y a également d'autres boucles permettant de trier et sélectionner les individus élites par exemple. La complexité de l'algorithme est donc de :

$$\mathbf{O(p \times n^2)}$$

Avec p : la taille de la population.

c - Colonie d'abeilles artificielles (ABC)

Cet algorithme utilise une fonction de mutation utilisé pour effectuer une permutation simple entre deux index différents du dépôt. Cette fonction est appelée *'swap'* et est appelée lors de la phase de recherche par les abeilles employées, elle est de complexité $\mathbf{O(1)}$.

La complexité de la boucle principale est, à l'instar de celle de l'algorithme génétique expliqué plus tôt, dépendante de la taille des paramètres d'entrés. Cette fonction se compose de trois phases différentes correspondant chacune à un type d'abeille (cf. Colonie d'abeilles artificielles). Chaque partie simule le comportement de l'abeille en question avant de passer à la prochaine abeille. La complexité totale de cet algorithme est :

$$\mathbf{O(b \times n^2)}$$

Avec b : le nombre d'abeille.

d - Essaim de particules (PSO)

L'algorithme PSO utilisé ici se compose d'une seule boucle. Cette boucle est elle-même composée de trois boucles, une boucle d'analyse du score de fitness de la solution contenue par la particule, une autre pour la mise à jour de la vélocité des particules et la dernière pour mettre à jour la position des particules. Les boucles de mise à jour sont plus complexes que la boucle d'analyse du score de fitness car elles utilisent elles-mêmes une boucle afin d'actualiser la position en x puis en y. La complexité de cet algorithme est donc :

$$\mathbf{O(s \times n^2)}$$

Avec s : la taille de l'essaim de particules.

VI.2. Comparaison

a - Paramétrage

Chaque algorithme sera exécuté pendant une minute exactement et aurait une population de solution initiale de 50. Avec la matrice de distance correspondant au graphe disponible en figure 3 :

$$\begin{bmatrix} 0 & 5.4 & 3.2 & 10.1 & 7.6 & 2.1 & 14.4 & 9.8 & 4.6 \\ 5.4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 14.4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Les paramètres uniques aux algorithmes sont décrits ci-dessous :

- Génétique
  - Taille de l'élite : 5
- Colonie d'abeilles artificielles
  - Nombre d'abeille : 5
  - Nombre d'itération sans amélioration : 10
- Essaim de particules
  - w : 0.5
  - $c1, c2$: 1

b - Analyse des résultats

X	Fitness	Itération	Vitesse
G	0.972028	65302	1089 it/s
ABC	0.986014	185903	4813 it/s
PSO	0.972028	45590	957 it/s

Fig. 4. Tableau de comparaison avec 1 itération.

X	Fitness	Itération	Vitesse
G	0.986014	66098	1129 it/s
ABC	0.986014	194839	4785 it/s
PSO	0.986014	59127	986 it/s

Fig. 5. Tableau de comparaison avec 5 itérations.

En termes de score de fitness, on peut constater que l'algorithme ABC est plus performant pour trouver une solution optimale. Cela peut être dû au fait que celui-ci faire beaucoup plus d'itérations que les autres algorithmes donc peut explorer plus de solutions.

Au regard des itérations et de la vitesse, encore une fois l'algorithme ABC est supérieur et cette fois-ci de très loin aux autres algorithmes. Il est probable que cela soit à cause du fait que les abeilles ont chacune une tâche bien définie et que l'une d'elle est dédié à une recherche locale en permutant des éléments d'une solution.

En conclusion, on peut dire que chacun des algorithmes cités est utilisable pour résoudre des problèmes d'optimisations complexes de la taille de notre exemple. L'algorithme à privilégier serait celui ayant les meilleures performances globales donc la colonie d'abeilles artificielles (ABC). Néanmoins, les algorithmes génétique et essaim de particules (PSO) arrivent non-loin derrière avec des performances équivalentes mais un nombre d'itérations moindre.

Enfin, d'autres tests ont été effectués avec une matrice générée aléatoirement (dimensions et pondérations aléatoires). Le code est également présent dans le notebook Jupyter présent sur le GitHub. Le tableau est le suivant :

X	Fitness	Itération	Vitesse
GG	0.996567	50661	845 it/s
ABC	0.996567	141150	2354 it/s
PSO	0.996567	56859	948 it/s

Fig. 6. Tableau de comparaison avec matrice 15 par 15 et 5 drones.

X	Fitness	Itération	Vitesse
G	0.963589	24006	400 it/s
ABC	0.963457	78711	1312 it/s
PSO	0.994798	26355	439 it/s

Fig. 7. Tableau de comparaison avec matrice 30 par 30 et 5 drones.

On constate que l'algorithme d'optimisation par essaim de particules (PSO) semble plus performant pour des problèmes de plus grandes tailles que notre problème de base. Néanmoins, il reste inférieur en termes d'itérations et de vitesse de calcul à l'algorithme ABC.

VI.3. Conclusion

Dans cette étude, nous avons comparé trois métaheuristiques appartenant à la famille des algorithmes évolutionnaires : Algorithme génétique, colonie d'abeilles artificielles (ABC) et optimisation par essaim de particule (PSO).

Nous avons comparé les performances de ces algorithmes en termes de qualité de solution, temps d'exécution et stabilité de solution. Nous avons effectué des tests à l'aide d'un problème défini s'apparentant à un problème de la tournée de véhicules avec collecte et livraison et étudiés les résultats de nos algorithmes sur ce problème.

Les informations qui sont ressorties permettent de conclure que tous les algorithmes sont capables de résoudre le problème donné dans des temps plus qu'acceptables. Nous avons pu constater une légère différence dans les performances classant l'algorithme de colonie d'abeilles artificielles en tête des trois algorithmes comparés. Néanmoins, l'algorithme de l'optimisation par essaim de particule se révèle plus performant pour les problèmes plus complexes en taille

de données avec un score de fitness plus élevé en moyenne que les autres algorithmes testés.

Ces résultats montrent qu'il est important de bien choisir l'algorithme correspondant à son problème. De plus, on peut déduire que ces algorithmes étant capable de résoudre approximativement des problèmes NP-complets avec contraintes tel que le VRPPD, ils peuvent être utiliser pour résoudre d'autres problèmes plus simple ou au moins aussi complexe.

Il pourrait être intéressant de pousser cette étude plus loin en essayant d'optimiser ces algorithmes et d'ajouter de la complexité en greffant de nouvelles contraintes. Par exemple, le problème de la tournée de véhicules n'est pas en manque de contraintes, nous avons ici par exemple fait notre expérience sur un problème de la tournée de véhicule avec collecte et livraison mais on pourrait compléter celui-ci en ajoutant une fenêtre de temps pour les livraisons, un ordre de livraison ou encore des contraintes de coûts. Cela permettrait d'étudier plus amplement la robustesse au changement de ces algorithmes.

GLOSSAIRE

Drone : Petit avion sans équipage embarqué, télécommandé ou programmé [Le Robert].

Heuristique : Méthode de calcul qui fournit rapidement une solution réalisable, pas nécessairement optimale ou exacte, mais qui est suffisante pour avancer ou pour tirer des leçons dans la résolution d'un problème d'optimisation difficile [Datafranca].

Métaheuristique : Une métaheuristique est un algorithme d'optimisation visant à résoudre des problèmes d'optimisation difficile (souvent issus des domaines de la recherche opérationnelle, de l'ingénierie ou de l'intelligence artificielle) pour lesquels on ne connaît pas de méthode classique plus efficace [Datafranca].

Optimisation : Donner les meilleures conditions de fonctionnement à *[quelque chose]* [Le Robert].

Stochastique : Qui se produit par l'effet du hasard [Le Robert].

Optimum : État, degré de développement de quelque chose jugé le plus favorable au regard de circonstances données [Larousse].

Robustesse (algorithme) : La robustesse d'une métaheuristique est sa capacité à maintenir des performances de haute qualité sur une large gamme de problèmes d'optimisation, sans être trop affectée par des changements dans les paramètres ou les conditions du problème. Une métaheuristique robuste peut trouver des solutions de haute qualité sur différents types de problèmes sans avoir besoin d'être finement réglée pour chaque problème individuel [10].

Thread : Un thread est l'unité de base à laquelle un système d'exploitation alloue du temps processeur. Chaque thread a une priorité de planification et maintient un ensemble de structures utilisé par le système pour enregistrer le contexte du thread quand l'exécution du thread est en pause. Le contexte du thread comprend toutes les informations dont le thread a besoin pour reprendre l'exécution sans interruption, notamment son ensemble de registres de CPU et de pile [Microsoft].

Matrice de distance : En mathématiques, une matrice de distance euclidienne est une matrice de taille $n \times n$ représentant l'espacement d'un ensemble de n points dans un espace euclidien [Wikipédia].

Algorithme : Ensemble des règles opératoires propres à un calcul ; suite de règles formelles [Le Robert].

Fitness : La fitness mesure,la qualité de l'individu exprimée sous forme d'un nombre ou d'un vecteur. On dit qu'un individu i est meilleur que l'individu j quand i est plus proche de la solution que j [CADCOM].

Complexité (algorithmique) : On appelle complexité d'un algorithme la mesure de la longueur de ses traces d'exécution en fonction de ses paramètres d'entrée [CNRS].

REFERENCES

[1] Guenoune, Hani & El-Ksouri, Mohamed & Boukra, Abdelmadjid & Berghida, Meryem. (2014). Résolution du problème de tournées de véhicules avec collecte et livraison simultanées avec une approche coopérative de métaheuristiques. 10.13140/RG.2.2.33527.78245.

[2] Radhoui, H. (2020). Problème de gestion de distribution : Cas du problème de tournées de véhicules avec collecte et livraison de marchandises dans un réseau multimodal (Doctoral dissertation, Normandie Université).

[3] Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 80, 8091-8126.

[4] Mitchell, M. (1995, September). Genetic algorithms: An overview. In Complex. (Vol. 1, No. 1, pp. 31-39).

[5] Gu, Z., Zhu, Y., Wang, Y., Du, X., Guizani, M., & Tian, Z. (2022). Applying artificial bee colony algorithm to the multidepot vehicle routing problem. Software : Practice and Experience, 52(3), 756-771.

[6] Utama, D. M., Fitria, T. A., & Garside, A. K. (2021, June). Artificial bee colony algorithm for solving green vehicle routing problems with time windows. In Journal of Physics: Conference Series (Vol. 1933, No. 1, p. 012043). IOP Publishing.

[7] Kunnapapdeelert, S., Johnson, J. V., & Phalitnonkiat, P. (2022). Green last-mile route planning for efficient e-commerce distribution.

Engineering Management in Production and Services, 14(1), 1-12.

[8] Chen, J., & Shi, J. (2019). A multi-compartment vehicle routing problem with time windows for urban distribution–A comparison study on particle swarm optimization algorithms. Computers & Industrial Engineering, 133, 95-106.

[9] Elshaer, R., & Awad, H. (2020). A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. Computers & Industrial Engineering, 140, 106242.

[10] Raidl, G. R. (2010). Metaheuristic optimization. In The Springer Handbook of Computational Intelligence (pp. 931-954).

[11] Rankohi, S., & Waugh, L. (2013). Review and analysis of augmented reality literature for construction industry. Visualization in Engineering, 1(1), 1-18.

[12] Derkx, F., Dumoulin, J., Sorin, J. L., & Legeay, V. (2002, September). Concept de Plate-forme Mobile Instrumentée (PMI) pour l'inspection des ouvrages d'art. In Colloque Micro-Drones (Vol. 2002).fi

[13] Güven, M., Bedir, H., & Anlaş, G. (2019). Optimization and application of Stirling engine for waste heat recovery from a heavy-duty truck engine. Energy Conversion and Management, 180, 411-424.

[14] Mogili, U. R., & Deepak, B. B. V. L. (2018). Review on application of drone systems in precision agriculture. Procedia computer science, 133, 502-509.

[15] Ahirwar, S., Swarnkar, R., Bhukya, S., & Namwade, G. (2019). Application of drone in agriculture. International Journal of Current Microbiology and Applied Sciences, 8(01), 2500-2505.

[16] Allouch, A., Cheikhrouhou, O., Koubâa, A., Khalgui, M., & Abbes, T. (2019, June). MAVSec: Securing the MAVLink protocol for ardupilot/PX4 unmanned aerial systems. In 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC) (pp. 621-628). IEEE.

[17] Tsiamis, N., Efthymiou, L., & Tsagarakis, K. P. (2019). A comparative analysis of the legislation evolution for drone use in OECD countries. Drones, 3(4), 75.

[18] Voss, W. G. (2013). Privacy law implications of the use of drones for security and justice purposes. International Journal of Liability and Scientific Enquiry, 6(4), 171-192.

[19] European Union Aviation Safety Agency, EASA (2022). Easy Access Rules for Unmanned Aircraft Systems.

[20] Federal Aviation Administration (2023). https://www.faa.gov/guidance.

[21] Civil Aviation Safety Authority, CASA (2022). Civil Aviation Safety Regulations.