République Algérienne démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 08 Mai 1945 – Guelma

Faculté des Mathématiques, d'Informatique et des Sciences de la Matière

Département d'Informatique

Mémoire de Fin d'étude Master

Filière : Informatique

Option : Sciences et Technologies de l'information et de la Communication

Thème :

Modélisation et évaluation d'un système de covoiturage dynamique optimisé

Présenté par : Bousena Amina

Membres du jury :

N°	Nom et Prénom	Qualité
1	Dr. Mohamed Chaoui	Président
2	Dr. Abdelmoumène Hiba	Superviseur
3	Dr. Said Brahimi	Examinateur

Juin 2023

Remerciement

Tout d'abord, je tiens à exprimer ma gratitude envers Dieu pour m'avoir donné la force, la connaissance, la capacité et l'opportunité de réaliser ce travail de manière satisfaisante.

J'aimerais exprimer mes remerciements les plus sincères à mon encadrante « Dr. Abdelmoumène Hiba », pour sa patience, sa disponibilité, ses conseils et sa capacité à me guider. Son soutien constant a grandement contribué à mon épanouissement personnel et académique. Ce fut un honneur pour moi d'être encadrée par une enseignante aussi incroyable.

Je tiens à remercier sincèrement les membres du jury d'avoir consacré leur temps précieux et leurs efforts pour lire et évaluer mon travail.

Je souhaite exprimer ma reconnaissance envers tous les enseignants du département d'informatique de l'Université du 8 mai 1945 de Guelma pour la qualité de leur formation, à tous nos chers collègues.

Je tient également à remercier tous ceux qui ont participé de près ou de loin à la réalisation de ce travail.

Dédicace

Je dédie ce travail

À mes très chers parents

dont l'amour, le soutien indéfectible, les sacrifices consentis et les précieux conseils ont été les piliers de ma réussite.

À mes sœurs Radia et Souad et mes frères Mohamed et Amer

pour vos encouragements et votre soutien inconditionnel, qui ont été une source constante de motivation.

A ma sœurs Rania dieu ait son âme

A mes petits anges

Adem, Rawane, Rana, Oussama et Arwa

À tous amis

et tous ceux qui m'ont soutenu et encouragé durant toute la période de mon travail. . .

AMINA

Résumé

Le covoiturage dynamique est devenu une solution prometteuse pour faciliter la mise en relation des passagers et des conducteurs, optimisant ainsi l'utilisation des véhicules et réduisant l'impact environnemental. Cependant, afin de garantir le succès de ces systèmes, il est essentiel d'établir des appariements efficaces entre les passagers et les conducteurs.

C'est dans ce contexte que s'inscrit notre travail. Notre objectif principal est l'utilisation de l'algorithme métaheuristique "Firefly" pour résoudre le problème dynamique d'appariement entre les conducteurs et les passagers.

La solution que nous proposons, dans ce mémoire, prend en considération des contraintes spatio-temporelles, des contraintes de capacité et des contraintes de temps d'attente. Notre modèle vise à optimiser les appariements en minimisant à la fois les temps d'attente des passagers et les distances totales parcourues.

Pour évaluer l'efficacité de notre approche, nous avons mené une étude en utilisant des données générées à partir d'une zone géographique réelle située dans la wilaya de Guelma. Cette évaluation rigoureuse nous a permis d'analyser en détail les performances de notre algorithme et de mettre en évidence ses avantages et ses limites.

Les résultats obtenus démontrent que l'algorithme Firefly a permis d'obtenir des appariements plus efficaces, réduisant à la fois les temps d'attente des passagers et les distances parcourues. Ces résultats soulignent l'efficacité de notre approche pour améliorer la qualité et l'efficience des systèmes de covoiturage.

Mots clés

Covoiturage dynamique, Appariement dynamique, Optimisation, Algorithme Firefly, Contraintes spatio-temporelles, Contraintes de capacité.

Abstract

Dynamic ridesharing has become a promising solution to facilitate the matching of passengers and drivers, optimizing vehicle use and reducing environmental impact. However, to ensure the success of these systems, it is essential to establish effective matches between passengers and drivers.

This is the context of our work. Our main objective is to use the "Firefly" metaheuristic algorithm to solve the dynamic matching problem between drivers and passengers.

The solution we propose in this thesis takes into account spatio-temporal constraints, capacity constraints and waiting time constraints. Our model aims to optimize matching by minimizing both passenger waiting times and total distance traveled.

To evaluate the effectiveness of our approach, we conducted a study using data generated from a real geographical area located in the wilaya of Guelma. This rigorous evaluation enabled us to analyze the performance of our algorithm in detail, and to highlight its advantages and limitations.

The results obtained show that the Firefly algorithm achieved more efficient matches, reducing both passenger waiting times and distances travelled. These results underline the effectiveness of our approach in improving the quality and efficiency of ridesharing systems.

Keywords

Dynamic ridesharing, Dynamic matching, Optimization, Metaheuristic, Firefly algorithm, Spatio-tempral constraints, Capacity constraints.

ملخص

ظهرت مشاركة السيارات كحل واعد لتسهيل الاتصال بين الركاب والسائقين، مما يتيح استخدامًا أكثر كفاءة للسيارات ويقلل من التأثير البيئي. ومع ذلك، فإن ضمان نجاح هذه الأنظمة يعتمد على إيجاد مطابقات فعالة بين الركاب والسائقين.

في هذا السياق، يركز عملنا على تطوير وتطبيق خوارزمية تسمى "Firefly" مستوحات من سلوك اليرعات لحل مشكلة المطابقة الديناميكية بين السائقين والركاب. الهدف الرئيسي لمشروعنا هو تحسين التطابقات من خلال تقليل أوقات انتظار الركاب ومسافات السفر، مع مراعاة القيود المكانية والزمانية، سعة السيارة و وقت انتظار الركاب.

لتقييم فعالية نهجنا، أجرينا دراسة باستخدام البيانات التي تم الحصول عليها من منطقة جغرافية حقيقية تقع في ولاية قالمة. سمح لنا هذا التقييم الدقيق بتحليل أداء خوارزميتنا بالتفصيل وإبراز مزاياها وقيودها.

أظهرت نتائجنا أن خوارزمية Firefly حققت نتائج أكثر فاعلية، مما قلل من أوقات انتظار الركاب ومسافات السفر.

كلمات مفتاحية

استخدام ديناميكي للسيارات، مطابقة ديناميكية، تحسين، خوارزمية Firefly، القيود المكانية والزمانية ، قيود القدرة.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

La mobilité est devenue une nécessité incontournable dans notre société moderne, mais elle est également à l'origine de nombreux défis environnementaux. Alors que de nouveaux moyens de transport continuent de se développer, le transport public, en particulier en milieu rural, peine souvent à répondre aux besoins de mobilité de manière compétitive. Parallèlement, l'utilisation intensive des voitures particulières a des conséquences néfastes sur l'environnement, telles que la congestion routière et les émissions de gaz à effet de serre, qui contribuent au réchauffement climatique.

Face à ces enjeux, le concept de covoiturage a émergé comme une solution innovante pour concilier mobilité et durabilité. Le covoiturage offre une alternative attrayante en permettant le partage des trajets entre différentes personnes se rendant dans la même direction. En optimisant l'utilisation des véhicules, le covoiturage présente des avantages potentiels environnementales et économiques.

Avec le temps, le covoiturage a évolué vers une forme plus flexible et adaptable, connue sous le nom de covoiturage dynamique. Contrairement au covoiturage statique, où toutes les demandes et offres sont préalablement connues, le covoiturage dynamique permet aux passagers et aux conducteurs de soumettre leurs demandes et offres en temps réel, et les réponses à ces demandes sont également fournies en temps réel. Cette transition vers le covoiturage dynamique vise à améliorer les appariements entre conducteurs et passagers, offrant ainsi une solution plus souple et mieux adaptée aux besoins de mobilité changeants.

Cependant, la gestion efficace du covoiturage dynamique représente un défi majeur en raison des contraintes auxquelles ce système est soumis. En effet, afin d’optimiser les appariements entre conducteurs et passagers, il est nécessaire de prendre en compte des contraintes complexes telles que les capacités des véhicules, les contraintes de temps à travers des fenêtres temporelles, les contraintes de lieu et les préférences des utilisateurs.

Dans le cadre de ce projet, notre objectif principal est de mettre en place un système d'appariement dynamique optimisé pour le covoiturage, visant à améliorer les correspondances

entre conducteurs et passagers (un conducteur – plusieurs passagers), à réduire le temps d'attente des passagers et à minimiser la distance parcourue, tout en respectant les contraintes spatiales, les contraintes temporelles et les contraintes de capacité.

Afin de réaliser cet objectif, nous nous penchons sur l'utilisation de l'algorithme métaheuristique Firefly, reconnu pour sa capacité à résoudre efficacement des problèmes d'optimisation complexes. L'algorithme Firefly tire son inspiration de la communication lumineuse entre les lucioles pour trouver des solutions optimales. L'un des avantages clés de l'algorithme Firefly réside dans sa capacité à explorer l'espace de recherche de manière intelligente et à converger rapidement vers des solutions de haute qualité. Les lucioles virtuelles représentent les individus de notre système, à savoir des solutions d'appariements entre les conducteurs et les passagers, et leur comportement est guidé par des règles d'attraction et de mouvement similaires à celles des lucioles réelles.

Ce mémoire est organisé comme suit :

Chapitre 1 : dans ce chapitre, nous allons donner un aperçu sur le covoiturage, ses composants et ses contraintes. Nous étudions et analysons, par la suite une liste des travaux connexes.

Chapitre 2 : ce chapitre est consacré à notre contribution. Nous commençons par motiver et justifier nos choix, par la suite, nous présentons les concepts de base de l'algorithme Firefly et ses composants. Ensuite, nous décrivons la modélisation proposée et détaillons chaque phase de réalisation de notre travail.

Chapitre 3 : les détails de l'implémentation de notre système seront discutés dans ce chapitre. Les résultats obtenus et les expérimentations effectuées seront aussi présentés et interpréter.

Chapitre 1 : Etat de l’art

1. 1 Introduction

La croissance de la population a exacerbé de nombreux problèmes de mobilité urbaine, tels que les embouteillages, la pollution de l'air et la consommation de carburant. Devant ces enjeux, le covoiturage est apparu comme une solution prometteuse pour une utilisation rationnelle et judicieuse des moyens de transport.

Dans ce chapitre, nous allons présenter les différents concepts liés au covoiturage, nous allons détailler le covoiturage dynamique, ainsi que ses composants et ses caractéristiques. Nous allons traiter le covoiturage dynamique comme étant un problème d’optimisation et nous allons discuter les différents critères à optimiser dans ce cadre. Enfin, nous présentons une synthèse des travaux qui ont exploré les métaheuristiques pour résoudre ce type de problème.

1. 2 Covoiturage

Le covoiturage, est une forme de transport où un conducteur non professionnel propose de partager son véhicule avec d'autres passagers qui ont des itinéraires et des horaires similaires. L'objectif principal du covoiturage est de réaliser tout ou une partie d'un trajet commun, ce qui permet aux voyageurs individuels de réduire les coûts de transport en partageant les frais tels que l'essence, les péages et le stationnement avec les autres participants (Ballet, et al., 2007 ; Furuhata et al., 2013 ; Martins et al. ; 2021).

D'après Silwal et al. (2019), un système de covoiturage est caractérisé par l'existence de plusieurs demandes de trajet, chacune ayant des points de départ et d'arrivée spécifiques, ainsi qu'un nombre défini de véhicules disponibles. L'objectif principal est d'optimiser les itinéraires afin de répondre au mieux à ces demandes.

Furuhata et al., (2013) indiquent que le covoiturage a trois gagnants. Le premier gagnant est les participants, ils bénéficient d'une économie de coûts liés aux déplacements, tels que les frais d'essence, les péages et les frais de stationnement. Le deuxième gagnant est l'environnement avec moins d'émissions, ce qui signifie une réduction de la pollution de l'air, et une consommation moindre du carburant. Le troisième gagnant est la société, grâce à la

réduction des embouteillages et à une utilisation plus efficace des ressources disponibles. Les nombreux avantages de ce système ont attiré l'attention du public, ce qui a conduit à la diffusion de son idée dans le monde.

Les évolutions technologiques de ces dernières années ; tels que le développement de l'Internet, des smartphones, des plateformes de développement d'applications mobiles, du système de paiement en ligne et du système de positionnement global (GPS), ont fait évoluer le covoiturage, du covoiturage traditionnel (statique) vers le covoiturage dynamique (Agatz et al., 2012 ; Ting et al., 2021).

- ***Le covoiturage statique*** est généralement considéré comme un service de covoiturage dans lequel un ensemble de passagers voyagent dans un véhicule privé d'un point de départ commun à un point d'arrivée commun unique. Le système organise les conducteurs et les passagers avant le départ du trajet, et le nombre de passagers ainsi que l'itinéraire sont connus à l'avance. Il est facile et simple à développer et donne des résultats optimaux pour les petites instances, mais il ne peut être appliqué à la réalisation du covoiturage en temps réel (Agatz et al., 2012 ; Silwal et al., 2019).
- ***Le covoiturage dynamique*** est un service qui permet à rassembler les conducteurs et les passagers en temps réel. Il s'agit de pouvoir trouver un appariement entre les demandes et les offres dans un délai rapide pour un trajet donné et en fonction de la position des véhicules des conducteurs potentiels. Il est plus utile et offre une plus grande flexibilité aux passagers et aux conducteurs. Dans le système de covoiturage dynamique, aucune planification préalable n'est nécessaire, en raison du temps de réponse plus court, le développement du système sera plus compliqué (Shen et al., 2016 ; Silwal et al., 2019).

Nous allons présenter plus en détails le covoiturage dynamique dans ce qui suit.

1. 3 Covoiturage dynamique

1.3.1. Définition

Le covoiturage dynamique, également connu sous le nom de covoiturage en temps réel, est un système de transport automatisé qui permet la mise en relation des conducteurs et des passagers en temps réel. Son objectif est de regrouper les voyageurs qui ont des itinéraires et des horaires similaires dans un délai court (Agatz et al., 2012).

En effet, la difficulté de ce système réside dans la mise en relation des individus qui doivent respecter des contraintes spatio-temporelles (point de départ, point d'arrivée, temps de départ et temps d'arrivée), contrainte des sièges vacants et contrainte de détour, etc.

Ces contraintes doivent être spécifiées à l'avance par les conducteurs et les passagers avant que le trajet souhaité soit défini et effectué. Il offre plus de flexibilité que le covoiturage statique, et cette flexibilité est démontrée lors des détours, puisque des déviations peuvent être nécessaires à différents moments du trajet pour prendre et déposer des passagers, tant que ces distances de déviation ne dépassent pas la distance de tolérance du conducteur (Silwal et al., 2019 ; Martins et al., 2021).

1.3.1.1. Composants d'un système de covoiturage dynamique

Un système de covoiturage est composé de cinq éléments nécessaires à sa réalisation (Jeribi, 2012).

- ***Conducteurs :*** sont des individus qui proposent de partager leur véhicule avec un ou plusieurs passagers lors de leurs trajets.
- ***Passagers :*** sont des individus qui cherchent à partager un trajet avec un conducteur, ils réservent une place dans son véhicule pour effectuer le même trajet ou un trajet similaire.
- ***Offre de covoiturage :*** elle contient les paramètres de déplacement du conducteur. Ces paramètres déterminent les éléments spécifiques du voyage à effectuer : le point de départ, le point d'arrivée, date, le temps de départ, le temps d'arrivée, le nombre de sièges disponibles, etc.
- ***Demande de covoiturage :*** le covoiturage s'effectue uniquement s'il y a une demande. Il s'agit d'une demande d'un passager exprimant son besoin de se déplacer en voiture d'un lieu à un autre. Cette demande contient les éléments spécifiques du voyage souhaité, comme le point de départ, le point d'arrivée, l'heure de départ, l'heure d'arrivée, etc.
- ***Contraintes d'appariement :*** pour assurer l'appariement entre l'offre et la demande, il est important de prendre en compte certaines contraintes spécifiques. Ces contraintes peuvent porter sur :
  - ***Le temps*** : les dates et heures des trajets proposés doivent correspondre aux demandes.
  - ***L'itinéraire*** : les points de départ et d'arrivée fournis par les passagers doivent être inclus dans l'itinéraire du conducteur ou sur son chemin.
  - ***Le nombre de sièges disponibles*** dans un véhicule, doit être suffisant pour accueillir les passagers demandant le même trajet.

En respectant et en satisfaisant ces contraintes, il est possible de réaliser un appariement efficace entre les conducteurs et les passagers.

La figure 1.1 représente la structure d'un système de covoiturage dynamique en indiquant ses composants. Cette figure a été inspiré du travail de (Shen et al. ,2016).

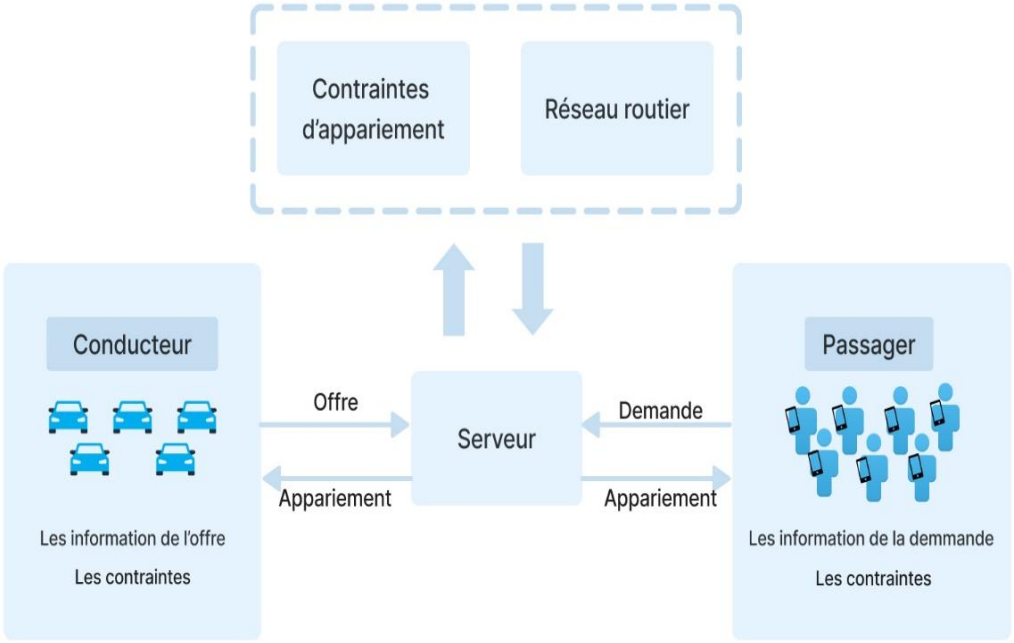


Figure 1.1: *Structure d'un système de covoiturage dynamique.*

1.3.3. Caractéristiques du covoiturage dynamique

Lors de la création d'un service de covoiturage dynamique, un ensemble de caractéristiques et de contraintes doivent être pris en compte. Chaque contrainte peut avoir un rôle important dans la conception de l'algorithme, en simplifiant les calculs et permettant de réduire les espaces de recherche, mais elle peut aussi rendre l'appariement plus complexe et avoir un impact sur les résultats finaux.

Dans ce qui suit, nous allons présenter des différents types de caractéristiques et de contraintes.

- ***Contraintes de trajet (CTr)***

  Dans un système de covoiturage, chaque demande doit être transportée de son point de départ à son point d'arrivée, et le point de départ doit être visité en premier.

- ***Contraintes de temps (CT)***

  Cette contrainte est liée au temps de trajet, en associant une fenêtre temporelle à chaque demande de trajet. Dans les systèmes de covoiturage, cette fenêtre temporelle est généralement donnée par chaque passager qui indique le temps de départ au plus tôt de son point de départ et le temps d'arrivée au plus tard à son point d'arrivée. Ainsi, pour qu'un passager participe à un trajet partagé, il doit être récupéré à son point de départ et déposé à son point d'arrivée dans le cadre de la fenêtre temporelle qu'il a spécifiée.

- ***Contrainte de capacité (CC)***

  Elle limite le nombre de passagers partageant le même véhicule au nombre de sièges disponibles dans ce véhicule.

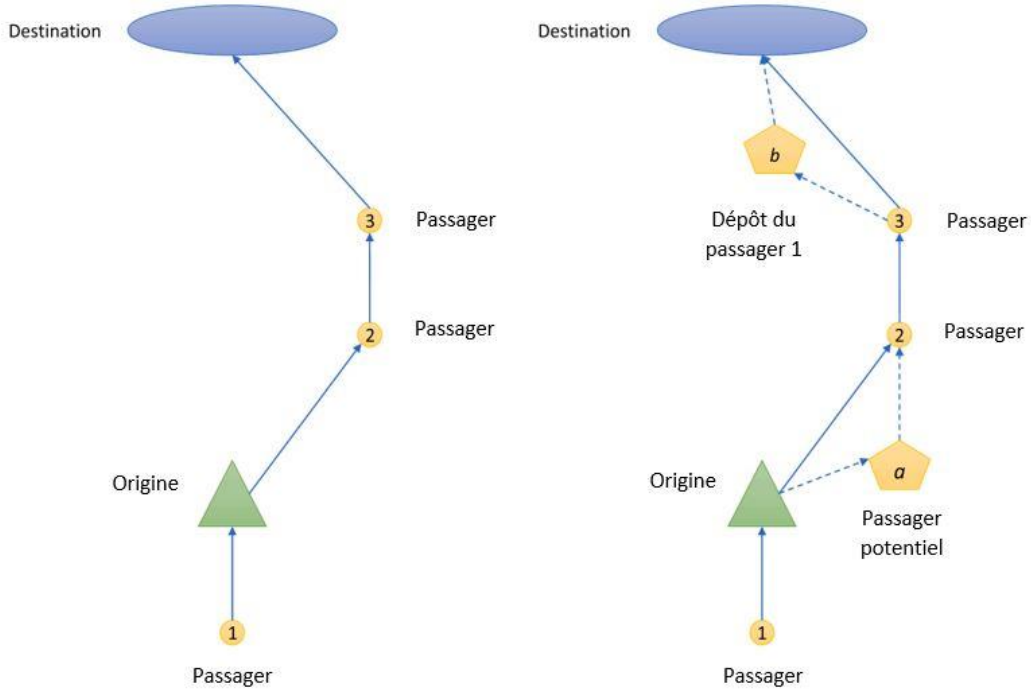
- ***Contrainte de coût (CCo)***

  Le passager du covoiturage peut spécifier le coût du trajet maximum qu'il est prêt à payer, et doit donc être affecté à des offre qui ne dépassent pas le coût maximum spécifié.

- ***Contrainte de détour (CD)***

  Dans certains systèmes de covoiturage, le conducteur est permis de faire des détours pour prendre et/ou déposer des passagers. Ce choix est accepté afin de répondre à un maximum de demandes tout en réduisant le nombre de véhicules utilisés.

  La figure 1.2 présente deux scénarios de covoiturage avec et sans détour.



(a)Trajet du conducteur sans détours pour récupérer et déposer des passagers.

(b)Trajet du conducteur avec possibilité de détours pour récupérer et déposer des passagers.

Figure 1.2: *Représentation de deux scénarios de covoiturage : sans détours et avec détours (Martins et al., 2021).*

- ***Contrainte de temps d'attente (CTA)***

  En plus de leur fenêtre temporelle spécifique, à la fois les conducteurs et les passagers ont la possibilité de définir un temps d'attente maximal. Ce temps d'attente maximum représente leur flexibilité en termes de temps. Pour les passagers, il indique qu'ils peuvent attendre jusqu'à cette limite de temps avant le départ. Du côté des conducteurs, ce temps d'attente est lié à leur flexibilité pour attendre des passagers (Mourad et al. 2019 ; Plate, 2019).

1.3.4. Critères d'optimisation

Le covoiturage permet aux utilisateurs de partager des trajets en véhicule, ce qui présente plusieurs avantages. Les conducteurs vont être aidés sur les frais de déplacement en partageant les coûts du trajet et améliorer la capacité de déplacement des passagers à leur disposition. Les plateformes de covoiturage agissent en tant que serveur de services, facilitant l'appariement automatique des conducteurs et des passagers pour des trajets à court terme. En tenant compte de ces objectifs généraux, la plupart des études sur le covoiturage examinent un

ou plusieurs objectifs spécifiques suivants, lors de la détermination des correspondances de covoiturage :

- **Minimiser la distance du trajet (Min_D)**

  La distance totale du trajet inclut tous les kilomètres parcourus par les participants, que ce soit en tant que conducteurs ou passagers, pour atteindre leur destination. Cet objectif est important pour la société, car il contribue à la réduction de la pollution et la congestion (Agatz et al., 2012).

- **Minimiser le temps de trajet (Min_T)**

  Le temps de trajet est le temps passé dans le véhicule lors du déplacement entre le point de départ et d'arrivée. Il joue un rôle important d'un point de vue sociétal car il est lié à la fois aux émissions de gaz des véhicules et à la vitesse de déplacement. En effet, des trajets plus rapides peuvent contribuer à réduire les émissions globales. De plus, il est un élément essentiel en termes de commodité pour les participants, car il affecte leur expérience de déplacement et leur permet d'optimiser leur emploi du temps (Agatz et al., 2012).

- **Maximiser le nombre de participants (Min_P)**

  Cet objectif maximise le nombre de conducteurs et de passagers satisfaits dans le système. En réalisant cet objectif, le système peut offrir une disponibilité des conducteurs et des passagers, ce qui contribue à créer un système de covoiturage efficace bénéfique pour tout le monde (Agatz et al., 2012).

- **Minimiser le nombre de véhicules (Min_V)**

  Cette mesure représente le nombre de véhicules utilisés dans le cadre du covoiturage dynamique. La réduction de ce nombre présente des avantages pour la société et l'environnement (Mourad et al., 2019).

- **Minimiser le temps d'attente (Min_TA)**

Ce critère est également important pour assurer une expérience de covoiturage satisfaisante pour tous les participants. En effet, l'optimisation de ce facteur permet d'améliorer les services en considérant la satisfaction des passagers desservis (Mourad et al., 2019).

1. 4 Travaux connexes

Le problème de covoiturage dynamique est un problème NP-hard (Herbawi & Weber, 2012). Ainsi, plusieurs approches ont été proposées pour résoudre ce problème, qu'il s'agisse d'approches exactes, heuristiques ou métaheuristiques.

La méthode de séparation et évaluation (branch and bound) et *la programmation en nombres mixtes* ont été utilisé par Huang et al. (2013) pour minimiser le temps de trajet. Arnaud et Kenneth (2014) ont opté pour une approche basée sur *les techniques de linéarisation et de brisure de symétrie* pour résoudre le problème de covoiturage dynamique avec un problème de changement de rôle, afin de minimiser la distance de trajet.

Wang et al. (2018) ont proposé une approche d'optimisation avec la notion de stabilité et plusieurs formulations mathématiques pour minimiser le nombre de véhicules et le coût de trajet. Tafreshian et Masoud (2020) ont utilisé *l'algorithme de relaxation lagrangienne* pour obtenir une solution stable et maximiser le confort social, etc.

Les approches heuristiques et métaheuristiques ont été largement adoptées dans le contexte du covoiturage dynamique.

Herbawi et Weber (2012) ont proposé un algorithme génétique pour modéliser le problème d'appariement des trajets avec des fenêtres temporelles. Leur objectif est de minimiser la distance et le temps de trajet et de maximiser le nombre d'appariement. Les résultats indiquent que l'idée du covoiturage dynamique est réalisable et que l'algorithme proposé est capable de résoudre le problème de covoiturage avec des fenêtres temporelles en un temps raisonnable.

Ben Cheikh et al. (2015) ont développé une approche distribuée originale basée sur des algorithmes coopératifs évolutionnaires pilotés par la communication entre agents (ABECoA: Agent-based Evolutionary Cooperative Algorithme). Dans le but de minimiser le temps d'attente des conducteurs et des passagers, le temps de retard et le temps total de trajet des véhicules.

Ben Cheikh et Hammadi (2016) ont proposé une nouvelle approche évolutionnaire dans laquelle les chromosomes sont définis comme des agents autonomes et intelligents (E2AIA : Evolutionary Algorithm Autonomous and Intelligent Agents). Grâce à une négociation précise du protocole, les agents chromosomes peuvent contrôler les opérateurs génétiques et les guider

pour trouver des solutions optimales dans un temps raisonnable. Les résultats des expériences indiquent que E2AIA est performant en termes d'optimalité et de temps de calcul.

Ghilas et al. (2016) ont proposé un algorithme heuristique basé sur la recherche de voisinage large adaptative (ALNS : Adaptive Large Neighborhood Search) pour résoudre le problème de prise en charge et de dépôt avec fenêtres de temps et lignes planifiées. Des aspects complexes tels que les horaires des trajets, la synchronisation et les contraintes de fenêtres temporelles sont efficacement pris en compte dans l'algorithme proposé. Les résultats des expériences montrent que l'ALNS est très efficace pour trouver des solutions de haute qualité pour les instances générées du problème de prise en charge et de dépôt avec fenêtres de temps et lignes de transport prévues.

Ben Cheikh et al. (2017) ont proposé une nouvelle approche, appelée approche métaheuristique basée sur des opérateurs génétiques contrôlés (MACGeO : Metaheuristics Approach Based on Controlled Genetic Operators), qui repose sur un nouveau codage dynamique, et est développée pour résoudre le problème de la correspondance de covoiturage multi-sauts. Les résultats des tests montrent que la correspondance de covoiturage multi-sauts pourrait augmenter de manière significative le nombre de demandes appariées tout en minimisant le nombre de véhicules nécessaires.

Huang and al. (2018) ont proposé une approche d'allocation de covoiturage orientée par les chemins des fourmis (APCA : Ant Path-Oriented Carpooling Allocation) pour résoudre le problème du service de covoiturage sous contrainte temporelle. Ils ont utilisé trois approches comparées, y compris l'optimisation par colonies de fourmis basée sur l'allocation, l'algorithme génétique et le recuit simulé. Les auteurs se sont intéressés à la comparaison de la performance de ces approches avec des représentations basées sur le chemin et l'affectation. L'objectif est de maximiser le nombre total des passagers correspondants et les taux d'utilisation des sièges, de minimiser les distances des utilisateurs. Le résultat montre que cette approche atteint une performance notable par rapport aux autres approches.

Ben Cheikh et al. (2020) ont développé une nouvelle métaheuristique basée sur la recherche tabou, intégrée dans le système d'optimisation de covoiturage dynamique (DyCOS : Dynamic Carpooling Optimization System), un système qui prend en charge le processus de placement automatique et optimal des passagers dans un délai très court ou même en cours de route.

Smet (2021) a proposé la montée de colline à acceptation tardive (LAHC : Late Acceptance Hill Climbing) pour décider quels utilisateurs agissent en tant que conducteurs pour construire leurs itinéraires.

Jadivi et al. (2021) ont proposé un algorithme qui utilise l'optimisation basée sur la biogéographie (BBO : Biogeography Based Optimization) pour résoudre un problème d'optimisation multi objectif pour le covoiturage en ligne.

Zhan et al. (2021) ont proposé un algorithme de colonie d'abeilles artificielles modifié (MABC : Modified Artificial Bee Colony) avec "path relinking" pour résoudre le problème de partage de trajets en temps réel. L'objectif est de maximiser le nombre de participants, de minimiser le coût et le temps de trajet, et de prendre en compte les contraintes de capacité, de fenêtre temporelle et de coût de déplacement.

Carvalho et Golpayegani (2022) ont proposé le modèle (MaMoP : Multi-agent, Multi-objective Preference-based ridesharing model). Ils ont intégré les algorithmes évolutionnaires pour trouver une solution optimisée qui maximise les bénéfices, réduit le temps de trajet et les coûts.

Gao et al. (2022) ont proposé un mécanisme d'appariement basée sur le vote (VOMA : VOting based MAtching) pour calculer des solutions d'appariement quasi-optimales pour les conducteurs et les passagers, en respectant leur vie privée.

Hsieh (2022) a intégré un algorithme méta-heuristique hybride appelé algorithme hybride basé sur l'évolution différentielle et l'algorithme Firefly (Firefly-DE : Firefly Algorithm and Differential Evolution) pour faire correspondre conducteurs et passagers. Dans le but de trouver les appariements permettant de minimiser la distance de déplacement globale et de répartir les économies de coûts entre les participants.

Nous allons classer ci-dessous les travaux analysés précédemment par les approches métaheuristiques, en fonction de l'ensemble des contraintes et des critères d'optimisation du covoiturage dynamique présentés précédemment.

Le tableau 1.1 présente la synthèse de ces travaux.

Référence	**Configuration**	**Contraintes**						**Objectifs**						**Métaheuristiques**
		CTr	**CT**	**CCo**	**CC**	**CD**	***CTA***	**Min_D**	**Min_T**	**Max_P**	**Min_V**	**Min_TA**	**Min_C**	
Herbawi et Weber (2012)	M	✓	✓		✓	✓	✓	✓	✓					GA: Genetic Algorithm
Ben Cheikh et al. (2015)	M	✓	✓		✓	✓	✓	✓	✓	✓		✓		ABECoA: Agent-based Evolutionary Cooperative Algorithme
Ben Cheikh et Hammadi (2016)	M	✓	✓		✓	✓	✓	✓	✓	✓		✓		E2AIA: Evolutionary Algorithm Autonomous and Intelligent Agents
Ben Cheikh et al. (2017)	M	✓	✓		✓	✓	✓	✓	✓	✓		✓		MACGeO: Metaheuristics Approach Based on Controlled Genetic Operators
Huang and al., (2018)	S-M	✓	✓		✓	✓		✓		✓				APCA: Ant Path-Oriented Carpooling Allocation
Ben Cheikh-Graiet(2020)	M	✓	✓	✓	✓	✓		✓	✓	✓	✓			RT, DyCOS: Tabu Search and Dynamic Carpooling Optimization System
Smet (2021)	M	✓	✓	✓	✓	✓	✓			✓		✓		LAHC: Late Acceptance Hill Climbing
Jadivi et al. (2021)	M	✓	✓		✓		✓	✓		✓				BBO: Biogeography Based Optimization

Zhan et al. (2021)	M	✓	✓	✓	✓				✓	✓			✓	MABC: Modified Artificial Bee Colony
Carvalho et Golpayegani (2022)	M	✓	✓	✓	✓		✓		✓				✓	MaMoP: Multi-agent, Multi-objective Preference-based ridesharing model
Gao et al. (2022)		✓	✓					✓			✓	✓		VOMA: VOting based MAtching
Hsieh (2022)	M	✓	✓	✓	✓		✓	✓					✓	Firefly-DE: Firefly Algorithm and Differential Evolution

Contraintes : CTr : contrainte de trajet, **CT :** contrainte de temps, **CCo :** contrainte de cout, **CC :** contrainte de capacité, **CD :** contrainte de détour, **CTA :** contrainte de temps d'attente.

Objectifs : Min_D : minimiser la distance, **Min_T :** minimiser le temps, **Max_P :** maximiser le nombre de participants, **Min_V :** minimiser le nombre de véhicule, **Min_TA :** minimiser le temps d'attente, **Min_C :** minimiser le coût.

Configuration : S : un seul passager par trajet**, M :** plusieurs passagers par trajet.

Tableau 1.1: *Classification des travaux connexes.*

1. 5 Conclusion

Dans ce chapitre nous avons présenté brièvement le contexte de notre travail, à savoir le covoiturage dynamique, ses composants, ainsi que les contraintes et les critères. Pour terminer, nous avons étudié quelques travaux existant dans le cadre de covoiturage dynamique et les métaheuristique. Nous avons procédé à une classification de ces travaux, ce qui nous a permis de bien comprendre le principe de fonctionnement de ce domaine.

Dans le chapitre suivant, nous allons expliquer en détail notre approche de modélisation et de formalisation pour résoudre le problème de covoiturage dynamique. Nous cherchons à trouver une solution optimale qui comprend des affectations bien optimisées afin de répondre aux demandes émises de la manière la plus efficace possible.

Chapitre 2 : Contribution

2. 1 Introduction

Au cours des deux dernières décennies, les chercheurs se sont de plus en plus intéressés à l'étude des méthodes utilisées par la nature pour résoudre les problèmes d'optimisation. Cette approche a conduit à l'émergence régulière de nouvelles métaheuristiques, offrant des algorithmes efficaces capables de produire des solutions de qualité proches de l'optimum dans la plupart des cas. Les métaheuristiques bio-inspirées sont spécifiquement conçues pour trouver des solutions relativement bonnes dans un temps de calcul raisonnable.

Dans le cadre de notre étude, nous proposons d'utiliser l'algorithme Firefly (Yang, 2009), pour résoudre notre problème de covoiturage dynamique. Le problème que nous étudions implique des contraintes temporelles : temps de départ et d'arrivée des conducteurs et des passagers ainsi que le temps d'attente des passagers. En outre, nous prenons également en compte des contraintes spatiales : points de départ et d'arrivée et des contraintes de capacité

Notre objectif est de minimiser à la fois le temps d'attente des passagers, et la distance totale parcourue par les conducteurs et les passagers. Ce chapitre présente les différentes étapes réalisées afin de modéliser et de résoudre notre problème par l'algorithme des lucioles.

2. 2 Problématique et objectifs

De nos jours, l'augmentation de la population et la dispersion des habitants ont conduit à une augmentation de la demande de transport public et à des difficultés de répondre efficacement aux différents besoins des clients. Par conséquent, un nombre croissant de personnes optent pour des véhicules privés dans leurs déplacements quotidiens. Cependant, cette utilisation intensive des voitures particulières, associée à une mobilité accrue, aggrave les problèmes environnementaux et soulève des défis tels que la congestion, les difficultés de stationnement et la lenteur des déplacements.

Cette demande croissante en matière de transport, conjuguée à des attentes élevées quant à la qualité des services, a donné lieu à de nouveaux modes de transport visant à satisfaire les

besoins des passagers. Ces derniers cherchent souvent à se déplacer rapidement et confortablement, tout en contribuant à réduire la congestion, la pollution et en favorisant le développement économique.

Le covoiturage est un des nouveaux modes de transport qui s'est imposé ces dernières années (tel qu'Uber, Lyft, DiDi) comme une solution bien établie et approuvée en raison de sa contribution à la résolution des problèmes environnementaux et économiques.

Les progrès technologiques ont donné lieu à une forme dynamique du covoiturage qui consiste à mettre en relation des conducteurs et des passagers en temps réel en tenant compte du facteur temporel et spatial. Ce mode de transport est caractérisé par une flexibilité et une souplesse d'utilisation (Silwal et al., 2019).

Le succès du covoiturage du point de vue de la plateforme, du conducteur et du passager nécessite une optimisation sophistiquée de tous les composants intégrés qui fournissent collectivement des services de manière flexible et instantanée. Néanmoins, l'amélioration de l'efficacité opérationnelle de ces systèmes est un défi majeur pour les plateformes de covoiturage. Notre travail s'inscrit dans cette problématique en cherchant à relever ce défi.

L'objectif principal de ce projet est de résoudre le problème du covoiturage dynamique en optimisant les appariements entre les demandes des passagers et les offres des conducteurs. Notre étude se concentre spécifiquement sur le cas où un conducteur peut accepter plusieurs *passagers (un conducteur - plusieurs passagers)*, avec respect des contraintes spatio-temporelles, des contraintes de capacité et du temps d'attente des passagers et avec des objectifs à deux niveaux distincts :

- ***Au niveau local ;*** optimiser la récupération et le dépôt des passagers à l'intérieur de chaque véhicule. Cela implique de trouver la meilleure répartition temporelle pour *minimiser le temps d'attente des passagers*.
- ***Au niveau global ;*** répartir de manière optimale les passagers entre les différents véhicules disponibles. L'objectif est d'optimiser la répartition des passagers afin de *minimiser la distance totale parcourue par les conducteurs et les passagers*.

Dans le but de réaliser notre objectif, nous avons décidé d'explorer les approches métaheuristiques, en particulier celles basées sur l'intelligence en essaim. Nous avons identifié l'algorithme Firefly (luciole) comme une solution prometteuse pour résoudre notre problème.

Selon Fister et al. (2014), l'algorithme Firefly a démontré son efficacité dans différentes applications, notamment dans le domaine du covoiturage dynamique. Cette approche présente trois avantages principaux qui en font un choix pertinent pour résoudre l'appariement des conducteurs et des passagers dans le contexte du covoiturage dynamique :

- ***Répartition automatique de la population :*** L'algorithme Firefly permet une répartition automatique de l'ensemble des individus (conducteurs et passagers) en sous-groupes, favorisant ainsi la recherche dans des régions d'espace de recherche spécifiques. Cette capacité de subdivision facilite la recherche de solutions optimales en permettant à chaque sous-groupe de se déplacer autour d'un mode local.
- ***Mécanisme d'attraction innovant :*** L'algorithme Firefly utilise un mécanisme d'attraction entre les individus qui est novateur et contribue à accélérer la convergence vers des solutions optimales. Cette caractéristique est particulièrement bénéfique dans le contexte du covoiturage dynamique, où il est crucial de trouver rapidement des appariements optimaux entre les conducteurs et les passagers.
- ***Adaptabilité à différents problèmes d'optimisation :*** L'algorithme Firefly a démontré son efficacité pour résoudre divers types de problèmes d'optimisation. Dans le cas du covoiturage dynamique, cet algorithme peut être adapté pour prendre en compte des contraintes spécifiques. Il offre ainsi une flexibilité pour répondre aux exigences propres au covoiturage dynamique et permettre la recherche de solutions optimales.

Bien que l'algorithme Firefly soit moins utilisé que d'autres méthodes d'optimisation, il a été appliqué avec succès dans les travaux de (Hsieh,2020 ; Hsieh,2022a ; Hsieh,2022b) pour optimiser le partage des coûts dans un système de covoiturage dynamique. Ces études se concentraient principalement sur l'aspect spécifique du partage des coûts dans le covoiturage et ne couvraient pas tous les aspects du système.

Par conséquent, il est essentiel d'explorer davantage l'utilisation de l'algorithme Firefly dans d'autres aspects du covoiturage dynamique, principalement l'appariement dynamique sous contraintes.

2. 3 Description du problème

Le système de covoiturage que nous voulons mettre en place est composé d'un ensemble d'offres/demandes, un mécanisme d'appariement et un ensemble de critères à optimiser.

1. **Offres et demandes de trajet :** Les conducteurs et les passagers ont la possibilité de soumettre leurs offres et demandes de trajet à une échéance proche de l'heure de départ

souhaitée. Les conducteurs précisent leur point de départ, leur destination, les horaires de départ et d'arrivée désirés, ainsi que le nombre de places disponibles dans leur véhicule. De leur côté, les passagers indiquent leur point de départ, leur destination, les horaires de départ et d'arrivée souhaités, ainsi que la durée maximale d'attente qu'ils peuvent tolérer.

2. Mécanisme d'appariement : notre service doit exécuter régulièrement un mécanisme d'appariement en respectant plusieurs contraintes, notamment :

- **Contrainte temporelle :** les appariements sont basés sur le temps de départ et d'arrivée souhaités des conducteurs et des passagers.
- **Contrainte spatiale :** en prenant en compte les points de départ et d'arrivée pour optimiser les itinéraires et la proximité géographique.
- **Contrainte de siège :** les appariements sont faits en fonction du nombre de places disponibles dans les véhicules des conducteurs, en favorisant les trajets avec un seul conducteur et plusieurs passagers pour maximiser l'utilisation des ressources.

3. Optimisation des critères : L'appariement est effectué en tenant compte des critères à optimiser, à savoir :

- **Distance totale parcourue :** L'objectif est de minimiser la distance totale parcourue par les conducteurs et les passagers en trouvant des correspondances efficaces.
- **Temps d'attente des passagers :** Nous cherchons à réduire le temps d'attente des passagers en les appariant rapidement avec des conducteurs ayant des trajets compatibles et qui leur font gagner du temps.

4. Communication des résultats : Une fois les appariements établis, les résultats sont communiqués aux conducteurs et aux passagers concernés. Ceux qui ont été mis en correspondance avec succès sont retirés de la liste d'attente. La liste d'attente sera mise à jour en temps réel lorsque de nouvelles offres sont reçues.

2.3.1. Formalisation du problème

Afin de formuler de manière plus précise ce problème, il est nécessaire de représenter de manière concrète les détails des offres de voyage soumises par les conducteurs ainsi que les demandes de voyage des passagers.

2.3.1.1. Offre de trajet

Soit *C* un ensemble de conducteurs qui soumettent leurs offres de voyage. Une offre de trajet du conducteur, notée $c_i, avec\ i\epsilon\{1,2,3,\cdots C\}$ est définie par $O(c_i) = \left(Id_{c_i}, Pd_{c_i}, Pa_{c_i}, Td_{c_i}, Ta_{c_i}, S_{c_i}\right)$, où :

- Id_{c_i}, est l'identifiant du conducteur.
- Pd_{c_i}, est le point de départ du conducteur.
- Pa_{c_i}, est le point de destination du conducteur.
- Td_{c_i}, indique le temps de départ au plus tard du conducteur.
- Ta_{c_i}, indique le temps d'arrivée au plus tard du conducteur.
- S_{c_i}, représente le nombre de sièges disponibles.

2.3.1.2. Demande d trajet

Soit P un ensemble de passagers qui déposent leurs demandes. Une demande d'un passager, notée $p_i\ avec\ i\ \epsilon\{1,2,3,\cdots P\}$ est définie par $D(p_i) = \left(Id_{p_i}, Pd_{p_i}, Pa_{p_i}, Td_{p_i}, Ta_{p_i}, Wt_{p_i}\right)$, où :

- Id_{p_i}, est l'identifiant du passager.
- Pd_{p_i}, est le point de départ du passager.
- Pa_{p_i}, est le point d'arrivée du passager.
- Td_{p_i}, indique le temps de départ au plus tard du passager.
- Ta_{p_i}, indique le temps d'arrivée au plus tard du passager.
- MWT_{p_i}, représente le temps d'attente maximum que le passager peut attendre avant son départ.

A partir des offres et des demandes de trajets soumises par les conducteurs et les passagers, la solution finale est une liste d'appariements entre les conducteurs et les passagers qui respectent les contraintes d'appariements que nous allons présenter dans ce qui suit.

2.3.1.3. Contrainte de temps

- Le temps de départ du passager apparié avec le conducteur ne doit pas être avant le temps de départ du conducteur.
- Le temps d'arrivée réel d'un conducteur à sa destination après avoir déposé un passager ne doit pas dépasser son temps d'arrivée au plus tard.
- Le temps d'attente maximal d'un passager MWT_{p_i}doit être supérieur ou égale au temps d'attente actuel que nous notons CWT_{p_i}, i.e., $MWT_{p_i} \geq CWT_{p_i}$.

2.3.1.4. Contrainte de l'aspect spatiale

Le point de départ Pd_{p_i} et le point de destination Pa_{p_i} du passager doivent être inclus dans l'itinéraire du conducteur. Dans notre travail, nous n'avons considéré que le cas des trajets inclusifs, i.e., $[Pd_{p_i}, Pa_{p_i}] \subset [Pd_{c_i}, Pa_{c_i}]$

2.3.1.5. Contrainte de capacité

Les conducteurs doivent respecter la capacité maximale de leurs véhicules lors de l'appariement.

Exemple illstratif

Dans l'exemple que nous présentons à la figure 2.1, nous considérons un conducteur (C1) et cinq passagers {P1, P2, P3, P4, P5}. Le conducteur C1 propose une offre de trajet avec les informations suivantes : départ du point 1 et arrivée au point 5, un temps de départ au plus tard fixée à 8h00 et un temps d'arrivée au plus tard fixée à 9h00, offrant trois places disponibles.

Parmi les cinq passagers, P1 souhaite effectuer un trajet du point 1 au point 4, avec un temps de départ au plus tard à 8h00 et un temps d'arrivée au plus tard à 8h33, et un temps d'attente maximal de 3 minutes. Les demandes des autres passagers sont les suivantes :

- P2 : trajet du point 6 au point 7, avec un temps de départ au plus tard à 8h00 et un temps d'arrivée au plus tard à 8h20, et un temps d'attente maximal de 5 minutes.
- P3 : trajet du point 2 au point 5, avec un temps de départ au plus tard à 8h03 et un temps d'arrivée au plus tard à 9h01, et un temps d'attente maximal de 2 minutes.
- P4 : trajet du point 3 au point 4, avec un temps de départ au plus tard à 8h15 et un temps d'arrivée au plus tard à 8h40, et un temps d'attente maximal de 10 minutes.
- P5 : trajet du point 3 au point 5, avec un temps de départ au plus tard à 8h20 et un temps d'arrivée au plus tard à 9h04, et un temps d'attente maximal de 5 minutes.

Tous les passagers respectent les contraintes spatio-temporelles, à l'exception du passager P2 qui ne respecte pas les contraintes spatio-temporelles avec le conducteur.

Pour résoudre ce problème d'appariement dynamique, et sélectionner les passagers à transporter, nous allons le modéliser et le résoudre en utilisant l'algorithme Firefly.

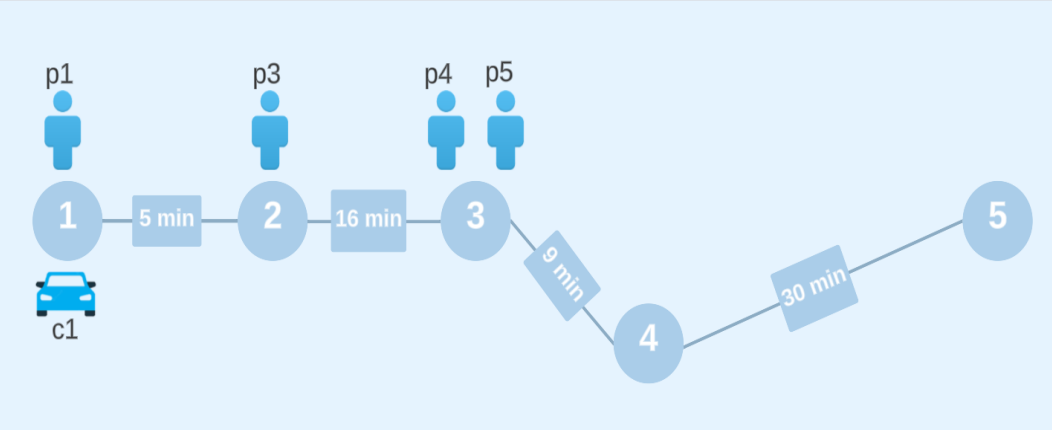


***Figure2.1**: Scénario d'appariement des trajets.*

Dans la section ci-dessous, nous expliquons l'algorithme Firefly et ses différentes phases.

2. 4 Algorithme Firefly

L'algorithme des lucioles est une technique d'optimisation métaheuristique inspirée du comportement des lucioles dans la nature. Il a été développé par Xin-She Yang en 2009 dans le but de résoudre des problèmes d'optimisation en imitant les schémas de clignotement et le comportement d'attraction des lucioles (Yang ,2009).

Le rôle des lumières clignotantes des lucioles est double : attirer les individus en vue de l'accouplement et avertir les prédateurs potentiels (Yang, 2010).

Yang (2010) propose trois règles idéales que l'algorithme Firefly suit en relation avec la lumière clignotante et son intensité :

- Les lucioles sont unisexes, de sorte qu'une luciole sera attirée par d'autres lucioles quel que soit leur sexe.
- L'attraction est proportionnelle à la luminosité et les deux diminuent lorsque la distance augmente. Ainsi, pour deux lucioles clignotantes, la moins brillante se rapprochera de la plus brillante. Si aucune luciole n'est plus brillante qu'une luciole donnée, elle se déplacera au hasard.
- La luminosité d'une luciole est déterminée par la topologie de la fonction objective.

En se basant sur ces trois règles, le principe de fonctionnement de l'algorithme Firefly peut être résumé en plusieurs étapes :

1. **Spécifier la fonction objectif :** définir la fonction objective que l'algorithme cherche à optimiser.

2. **Générer une population initiale de lucioles :** créer une population de lucioles avec des positions aléatoires dans l'espace de recherche. Chaque luciole représente ainsi une solution potentielle du problème.
3. **Déterminer l'intensité lumineuse :** évaluer l'intensité lumineuse de chaque luciole en fonction de sa position et de la fonction objectif.
4. **Calculer l'attractivité des lucioles :** calculer l'attractivité de chaque luciole en fonction de son intensité lumineuse et de la distance par rapport aux autres lucioles,
5. **Mouvement des lucioles moins lumineuses vers une luciole plus lumineuse :** déplacer les lucioles moins lumineuses vers les lucioles plus lumineuses en utilisant l'attractivité et la distance comme guides.
6. **Mise à jour de l'intensité lumineuse :** mettre à jour l'intensité lumineuse des lucioles, classer la population et déterminer la meilleure solution actuelle (Yang ,2010).

En répétant les étapes 3 à 6 de manière itérative, l'algorithme Firefly cherche à converger vers une solution optimale (ou proche de l'optimal) dans l'espace de recherche. Il offre une approche intéressante pour résoudre différents problèmes d'optimisation en utilisant les principes d'attraction et de luminosité des lucioles.

2. 5 Modélisation proposée

L'algorithme Firefly standard a été initialement développé pour résoudre des problèmes d'optimisation continue, ce qui le rend inadapté aux problèmes d'optimisation discrets tels que l'optimisation du covoiturage dynamique. Afin de l'appliquer à un contexte de covoiturage discret, il est nécessaire de l'adapter en modifiant chaque étape et caractéristique de l'algorithme pour prendre en compte les espaces discrets, où les solutions reposent sur des appariements discrets entre les conducteurs et les passagers. (Karthikeyan et al., 2015).

La figure 2.2 présente les différentes étapes de notre proposition de modélisation du problème par l'algorithme Firefly.

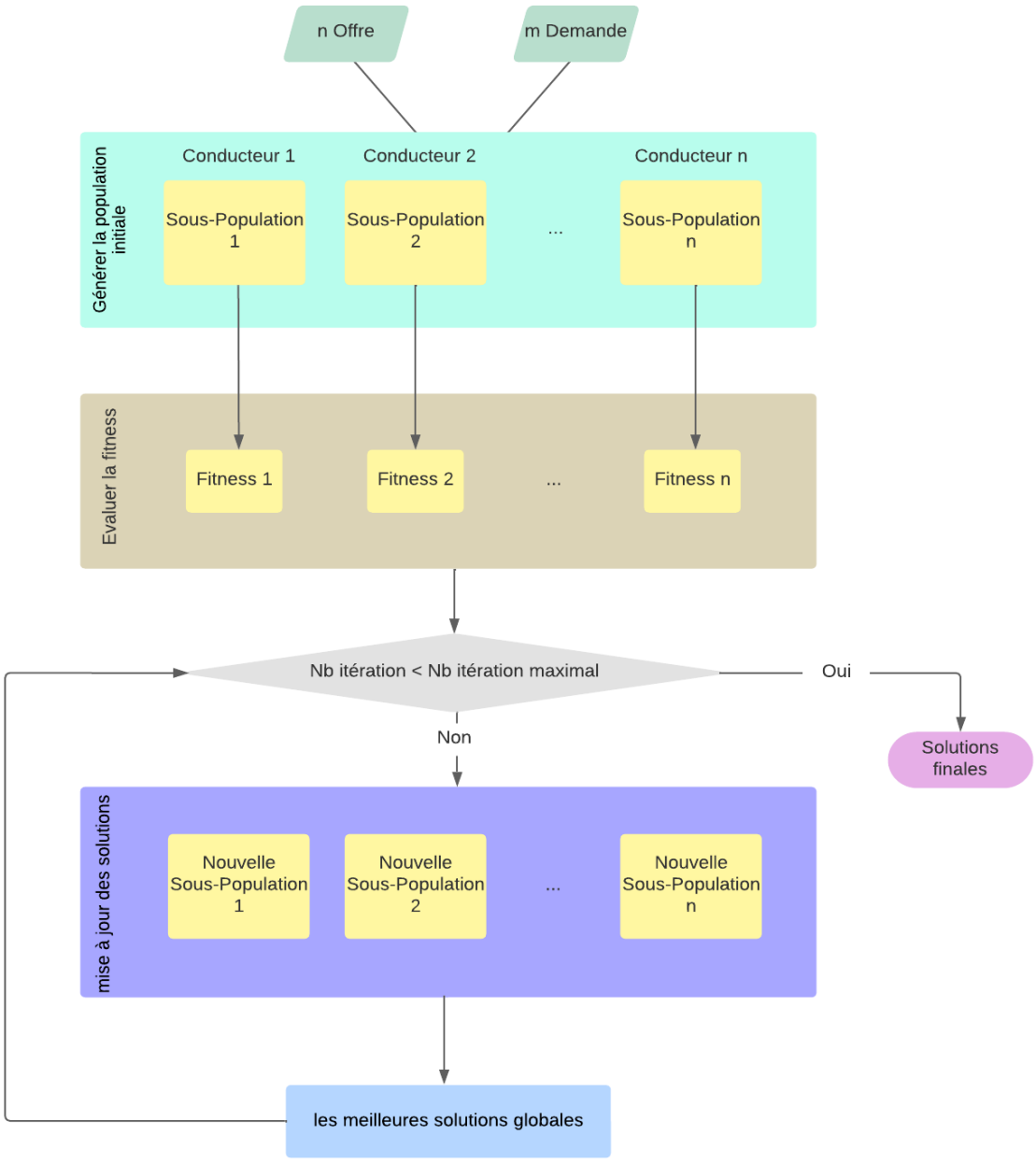


Figure 2.2 : *Différentes étapes de notre modélisation.*

2. 5.1 Génération des solutions initiales discrètes

2.5.1.1. Représentation de la solution

Le problème de covoiturage est une combinaison d'appariement entre les conducteurs et les passagers, de telle sorte de que la solution peut être exprimée par l'affectation des passagers avec des conducteurs. Dans notre approche chaque solution représente par une

luciole. Nous avons représenté chaque luciole par un tableau binaire – contenant des uns et des zéros.

Une valeur de 1 indique que le conducteur prendra le passager correspondant, tandis qu'une valeur de 0 indique que le passager ne sera pas pris par le conducteur. Nous avons utilisé la représentation binaire pour visualiser ces appariements de manière claire et concise. Un exemple de représentation de la luciole est présenté ci-dessous :

	P1	P2	P3	P4	P5
C1	1	0	1	0	1

2.5.1.2. Génération de la population

Nous avons généré en *parallèle* et de manière aléatoire une population de "lucioles" pour *chaque conducteur*. Nous avons pris en considération ces trois contraintes lors de la génération de la population :

- L'itinéraire des passagers est inclus dans l'itinéraire du conducteur,
- Le temps de départ du passager ne doit pas être avant le temps de départ du conducteur,
- Le conducteur ne peut pas prendre plus de passagers que le nombre de sièges disponibles.

Exemple : On va se référer au même exemple illustratif présenté dans la figure 2.1. Nous rappelons qu'on a : ***Conducteur :*** $\{C1\}$, ***Passagers :*** $\{\mathrm{P1}, \mathrm{P2}, \mathrm{P3}, \mathrm{P4}, \mathrm{P5}\}$. Nous fixons la taille de la population à 4.

La figure 2.3 présente un exemple de population générée pour un conducteur et cinq passagers, où le conducteur a une capacité de prendre trois passagers. La génération de la population est effectuée de manière aléatoire, en respectant les contraintes sus-citées.

P1	P2	P3	P4	P5
1	0	1	0	1
0	0	1	1	1
1	0	0	1	1
1	0	1	1	0

Figure 2.3: *exemple de population générée.*

2. 5.2 Évaluation de la luciole

Chaque luciole est évaluée pour déterminer sa fonction objectif, qui est associée à l'intensité lumineuse de la luciole correspondante. Dans le cadre de cette étude, l'objectif est de minimiser à la fois la distance et le temps d'attente des passagers.

Pour ce faire, nous proposons une fonction objectif qui évalue le rapport entre la distance entre les passagers et les conducteurs et le temps d'attente maximal par rapport au temps d'attente actuel des passagers. Cette fonction objectif est définie comme suit :

$$F(x) = \sum_{i=1}^{C} \sum_{j=1}^{P} \sqrt{\left(Pd_{p_j} - Pd_{c_i}\right)^2 + \left(Pa_{p_j} - Pa_{c_i}\right)^2} + (MWT_{p_j} - CWT_{p_j}) * x_j \quad (2.1)$$

Où :

- $\sqrt{\left(Pd_{p_j} - Pd_{c_i}\right)^2 + \left(Pa_{p_j} - Pa_{c_i}\right)^2}$, indique la distance euclidienne entre les points de départ et d'arrivée de l'itinéraire du conducteur et l'itinéraire du passager,
- $(MWT_{p_j} - CWT_{p_j})$, indique la différence entre le temps d'attente maximal d'un passager et son temps d'attente actuel noté CWT_{p_j}.
- x_j , est la variable de décision, qui est égale à 1 si le passager est accepté et 0 sinon.

L'évaluation et l'analyse de la fonction objectif que nous avons proposée nous a permis de faire une constatation importante dans notre problème :

Si la valeur de la fonction objectif est négative, cela signifie que le temps d'attente actuel dépasse le temps d'attente maximal, et par conséquent, la demande est automatiquement rejetée. En effet, la distance étant toujours positive, cela indique un délai trop long pour le passager.

D'autre part, si la valeur de la fonction objectif est supérieure à zéro, cela signifie que la solution est acceptée. Enfin, lorsque la fonction objectif a une valeur nulle, deux cas de figure doivent être considérés : si l'itinéraire du passager est identique à celui du conducteur, la demande est acceptée ; dans le cas contraire, elle est rejetée :

- $F(x) > 0$; dans ce cas la demande est acceptée,
- $F(x) < 0$; dans ce cas la demande est rejetée,
- $F(x) = 0$; dans ce cas nous considérons deux situations :
    - La demande est acceptée si l'itinéraire du passager est le même que le conducteur ; c.-à-d. le passager a les mêmes points de départ et d'arrivée que le conducteur.
    - La demande est rejetée si l'itinéraire de passager est différent que celui du conducteur.

Sur la base de la fonction objectif proposée, les contraintes et de nos critères d'optimisation qui consiste à minimiser la distance parcourue et le temps d'attente, nous avons formulé le problème comme suit :

$$\min F(x) = \sum_{i=1}^{C} \sum_{j=1}^{P} \sqrt{\left(Pd_{p_j} - Pd_{c_i}\right)^2 + \left(Pa_{p_j} - Pa_{c_i}\right)^2} + (MWT_{p_j} - CWT_{p_j}) * x_j \quad (2.2)$$

Exemple – Revenons à l'exemple précédent. Après avoir calculé la fitness, les valeurs obtenues sont les suivantes :

Fitness(s1) = Fitness ([1, 0, 1, 0, 1]) = 11.0

Fitness(s2) = Fitness ([0, 0, 1, 1, 1]) = 11.24

Fitness(s3) = Fitness ([1, 0, 0, 1, 1]) = 10.24

Fitness(s4) = Fitness ([1, 0, 1, 1, 0]) = 13.24

2. 5.3 Mise à jour de la solution

Dans l'algorithme Firefly, le mouvement des lucioles est basé sur l'intensité lumineuse et la comparaison entre deux lucioles. L'attractivité d'une luciole est déterminée par sa luminosité, qui est associée à la fonction objectif codée. Ainsi, dans le cas d'un problème de minimisation,

une luciole plus lumineuse se déplacera vers une luciole moins lumineuse (Yang, 2010). Le processus de mise à jour des solutions est réalisé selon les étapes suivantes :

1. **Distance**

Pour calculer la distance entre les lucioles, nous proposons d'utiliser la distance de Hamming. La distance de Hamming entre deux solutions correspond au nombre d'éléments qui diffèrent dans la séquence (Kuo et al., 2009).

Par exemple, pour calculer la distance entre deux lucioles (s1 et s2) qui sont représentées dans l'exemple de la figure 2.1, nous comparons chaque bit des deux solutions et enregistrons le nombre de bits dont les valeurs ne sont pas égales. Ainsi, la distance de Hamming entre s1 et s2 est de 2.

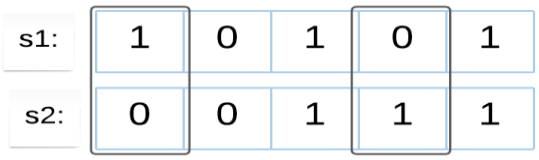


Figure 2.4: *Distance de Hamming entre deux lucioles.*

2. **Attractivité**

Pour calculer l'attractivité, nous avons ajusté l'équation d'attractivité utilisée dans l'algorithme Firefly continu (Yang, 2010), en remplaçant la distance cartésienne par la distance de Hamming, pour répondre aux exigences de notre problème du covoiturage. L'équation est définie comme suit (Singh, 2020) :

$$\beta = \beta_0 e^{-\gamma r^2} \tag{2.3}$$

Où : β_0, est l'intensité lumineuse à la source,

γ, est un coefficient d'absorption de la lumière prédéfinie,

r, est la distance de Hamming entre deux lucioles.

Dans notre travail, nous considérons le paramètre β comme une probabilité qui guide la modification de la valeur d'une variable dans une luciole. Une valeur d'une variable d'une luciole moins performante sera remplacée par la valeur correspondante dans une luciole plus performante. Nous nous sommes référées dans notre choix au travail de Bidar et al. (2018).

Ainsi, plus la valeur de β est élevée, plus la solution moins performante est susceptible d'adopter les valeurs des variables de la meilleure solution.

3. Mouvement

Lors de la transition de l'algorithme Firefly continu à sa version discrète, l'adaptation du mouvement devient une étape cruciale. Il est essentiel de préserver toutes les caractéristiques du mouvement de l'algorithme Firefly, telles que l'exploitation et l'exploration, tout en les adaptant avec soin à la version discrète. L'exploitation fait référence aux déplacements locaux vers des solutions optimales locales, tandis que l'exploration correspond aux déplacements de l'algorithme vers la meilleure solution globale disponible (Bidar et al., 2018).

- ***Déplacement local (exploitation)***

En adaptant la définition du déplacement local des lucioles aux espaces de problèmes discrets, le mouvement d'une luciole vers une luciole plus lumineuse implique le partage de valeurs avec la luciole plus lumineuse par la luciole moins lumineuse. Dans l'algorithme des lucioles continues, le taux de convergence des lucioles est contrôlé par le paramètre β (Bidar et al., 2018). En effet, β encourage l'exploitation de l'algorithme et rapproche les lucioles les unes des autres. Cependant, dans notre cas de problème discret, il est nécessaire de définir un nouveau déplacement local adapté aux espaces de problèmes discrets, afin de rapprocher les lucioles dans ces espaces.

Le mouvement β de la solution 1 vers la solution 2 dans un espace de problème discret peut être réalisé en suivant les étapes suivantes :

- Identifier les variables qui partagent la même valeur dans les solutions. Ces valeurs seront conservées inchangées dans une luciole moins lumineuse.
- Générer un nombre aléatoire ε dans l'intervalle [0, 1].
- En tenant compte de deux cas distincts :
    - Lorsque $\beta > \varepsilon$, nous remplaçons les valeurs des variables correspondantes dans la luciole moins lumineuse par les valeurs correspondantes de la meilleure solution.
    - Lorsque $\beta \leq \varepsilon$, nous conservons les valeurs précédentes des variables dans la luciole moins lumineuse, sans les mettre à jour avec les valeurs de la meilleure solution.

Répéter cette étape jusqu'à ce que toutes les lacunes soient remplies.

Exemple - Appliquons les étapes de déplacement à l'exemple précédent pour mieux illustrer le processus :

La solution s1 est meilleure que la solution s2 car elle a une valeur de fitness inférieure.

Premièrement, nous allons identifier les variables qui partagent la même valeur dans les deux solutions.

s1:	1	0	1	0	1
s2:		0	1		1

Ensuite, nous allons générer une valeur aléatoire ε pour chaque variable différente dans la solution moins lumineuse.

Si $\beta > \varepsilon$:

s2:	1	0	1		1

Si $\beta \leq \varepsilon$:

s2:	0	0	1		1

- ***Déplacement global (exploration)***

Pour éviter les pièges des optima locaux dans les métaheuristiques, nous avons adopté une approche d'exploration sous forme d'un déplacement global. Au cours de cette étape, nous avons testé différentes méthodes de mutation ; la mutation aléatoire de réinitialisation, mutation par inversion, mutation par échange (Srivatsa et al., 2019 ; Bidar et al., 2018).
Après ces tests, nous avons retenu la méthode de mutation par échange (*swap mutation*). Cet opérateur sélectionne aléatoirement deux valeurs dans la luciole et échange leur position (Kumar et al., 2020).

Pour décrire notre algorithme Firefly discret, nous définissons les variables suivantes :

Variable/symbol	Definition
N	La taille de population
Maxt	nombre maximal d'itérations
$O(c_i)$	Offre du conducteur, $i \in \{1,2,\dots,C\}$
$D(p_j)$	Demande du passager, $j \in \{1,2,\dots,P\}$

γ	Coefficient d'absorption de la lumière
β_0	Coefficient d'attraction valeur de base
r_{ij}	La distance de Hamming entre luciole i et j
ε	Nombre aléatoire avec une distribution uniforme dans [0,1]
F(x)	La fonction objectif
S_n	Solution d'appariement entre offre et demande c_i avec p_j, $n \in \{1,2,\dots,N\}$

Tableau 2.2: *Notations des variables/symboles pour l'algorithme Firefly discret.*

L'algorithme proposé est présenté ci-dessous.

Algorithme : Pseudo-code de l'algorithme proposé.

Entrée : O(c_i), D(p_j), N, γ, β_0, Maxt

Sortie : S_g

Générer une population S_n de N lucioles pour chaque conducteur
Calculer la valeur de fitness F(S_n) pour chaque luciole
Tant que (t < Maxt)
Pour i = 1 jusqu'à n
Pour j = i+1 jusqu'à n
Si (F(S_i) < F(S_j))
Calculer la distance de Hamming r_{ij}
Déterminer l'attractivité β
Déplacer la luciole j vers la luciole i
Mutation pour l'exploitation
Fin Si
Fin Pour j
Fin Pour i
Mutation pour l'exploration
Mettre à jour la meilleure solution globale S_g
t = t + 1
Fin Tant que
Retourner la meilleure solution globale S_g

La figure 2.5 présente le déroulement de l'algorithme proposé sur un ensemble de 4 conducteurs et un ensemble de passagers.

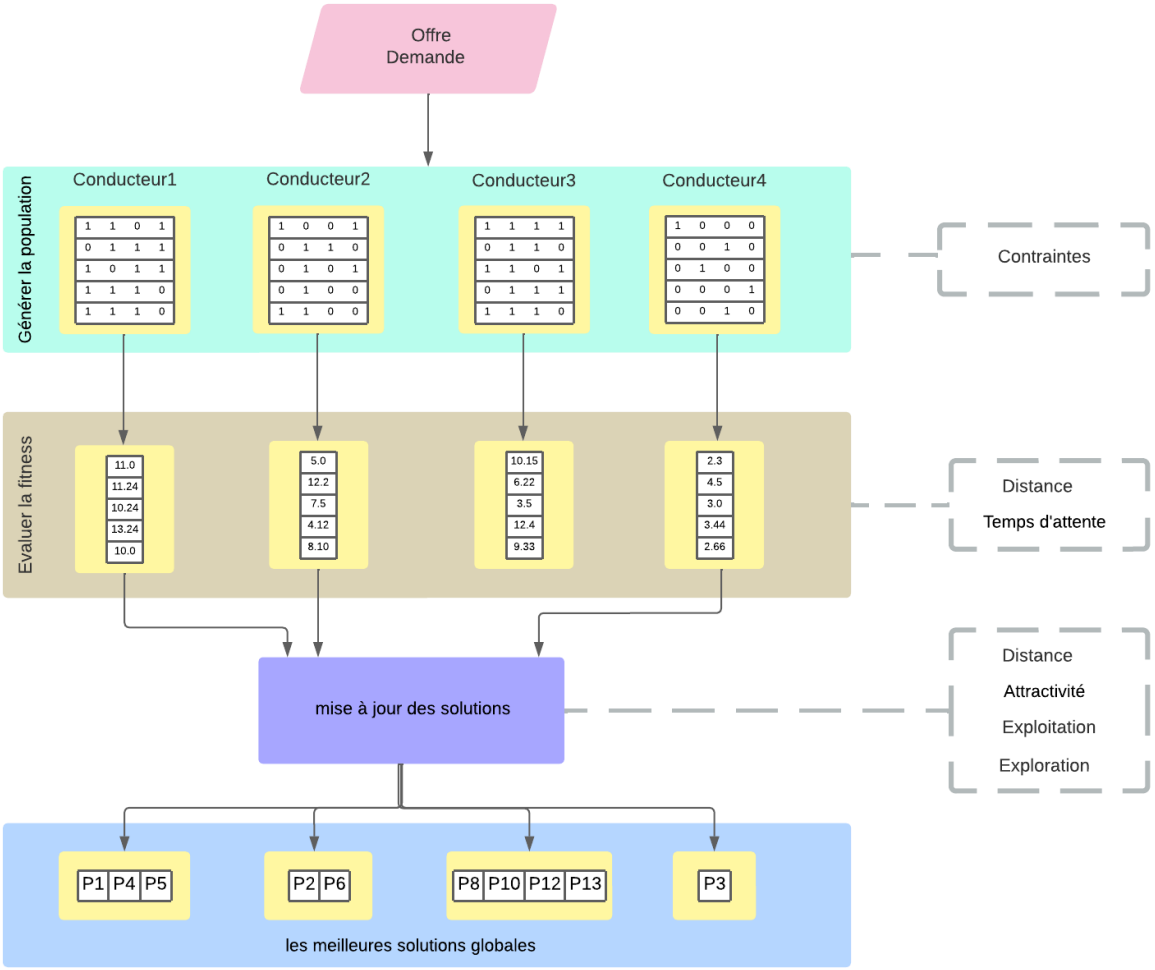


Figure 2.5: *Déroulement de l'algorithme sur un exemple.*

2. 6 Conclusion

Dans ce chapitre, nous avons présenté en détail notre proposition de modélisation et résolution du problème d'appariement dynamique sous contraintes avec l'algorithme Firefly. Nous avons adapté les étapes et les caractéristiques de l'algorithme pour tenir compte des espaces discrets propres au covoiturage que notre système suit afin de construire une solution.

En exploitant l'interaction et l'attraction entre les "lucioles" représentant les individus du système, nous avons pu concevoir une solution à notre problème en cherchant à minimiser le temps d'attente des passagers et la distance totale parcourue.

Dans le chapitre suivant, nous aborderons les aspects pratiques de la mise en œuvre de notre application. Nous fournirons des informations détaillées sur les choix d'implémentation, les technologies utilisées et les étapes concrètes pour mettre en place notre système.

Chapitre 3 : Implémentation

3. 1 Introduction

Une fois notre modèle est construit, nous passons à sa mise en œuvre. Nous allons décrire, dans ce chapitre, les aspects pratiques de notre travail en détaillant les outils et les langages utilisés. Par la suite, nous présentons la base de données utilisée pour valider notre système, ainsi que les étapes d'implémentation. Enfin, nous exposons le simulateur que nous avons développé et nous partageons les résultats obtenus lors de nos expérimentations.

3. 2 Langages et outils de développement

Nous avons utilisé plusieurs outils pour faciliter le développement et la mise en œuvre de notre système. Dans cette section, nous allons présenter ces outils en détail.

- ***Le langage python***

Python est un langage de programmation de haut niveau, interprété et orienté objet, qui possède une sémantique dynamique. Grâce à ses structures de données intégrées et à son typage dynamique, il est largement utilisé pour le développement rapide d'applications et comme langage de script ou de liaison pour connecter des composants existants. La simplicité de sa syntaxe, qui privilégie la lisibilité, permet de réduire les coûts de maintenance des programmes. Python encourage également la modularité et la réutilisation du code grâce à la prise en charge des modules et des packages. L'interpréteur Python et la vaste bibliothèque standard sont disponibles gratuitement, en source ou en binaire, sur toutes les principales plateformes, et peuvent être distribués librement. L'un des principaux avantages de Python est sa productivité accrue, grâce à la rapidité du cycle d'édition-test-débogage, sans nécessiter d'étape de compilation [1].

- ***Visual studio code***

Visual Studio Code (VS Code) est un éditeur de code source léger mais extrêmement puissant, conçu pour fonctionner sur les systèmes d'exploitation Windows, macOS et Linux. Il est largement utilisé par les développeurs grâce à sa grande flexibilité et à sa vaste gamme de fonctionnalités. VS Code se distingue par sa richesse en fonctionnalités. Il propose un support

intégré avancé pour les langages couramment utilisés, tels que JavaScript, TypeScript et Node.js, facilitant ainsi le développement d'applications Web et serveur. De plus, son écosystème d'extensions est très vaste, ce qui permet aux développeurs d'ajouter des fonctionnalités spécifiques à d'autres langages de programmation et environnements d'exécution, tels que C++, C#, Java, Python, PHP, Go et .NET. Grâce à son interface utilisateur intuitive et à sa grande adaptabilité, VS Code est devenu l'un des éditeurs de code les plus populaires et appréciés de la communauté des développeurs [2].

- ***Les bibliothèques utilisées***
  - ***Numpy***

NumPy est une bibliothèque essentielle pour le calcul scientifique en Python. Elle fournit un objet de tableau multidimensionnels, ainsi que des objets dérivés tels que les tableaux masqués et les matrices. De plus, NumPy propose une multitude de routines optimisées permettant d'effectuer rapidement des opérations sur ces tableaux [3].

- ***Pandas***

Pandas est une bibliothèque Python qui fournit des structures de données rapides, flexibles et expressives, conçues pour faciliter le travail avec des données "relationnelles" ou "étiquetées". Son objectif est de simplifier et rendre intuitif le traitement des données dans des tâches d'analyse réelles. De plus, pandas vise à devenir l'outil open source le plus puissant et flexible pour l'analyse et la manipulation de données dans n'importe quel langage [4].

- ***Ast***

Le module ast est une ressource précieuse qui permet aux applications Python à traiter les arbres de la grammaire de syntaxe abstraite de Python. Cela peut être particulièrement utile lorsque l'application travaille avec des fichiers CSV. La grammaire abstraite Python elle-même peut changer à chaque version de Python ; ce module permet de trouver de manière programmative à quoi ressemble la grammaire actuelle [5].

- ***Random***

Ce module propose des implémentations de générateurs de nombres pseudo-aléatoires pour différentes distributions [6].

- ***Math***

Ce module permet d'accéder aux fonctions mathématiques définies par la norme C [7].

- ***Datetime***

Le module datetime fournit des classes permettant de manipuler les dates et les heures. Il prise en charge les opérations arithmétiques sur les dates et les heures [8].

- ***Mapbox Directions API***

L'API Directions de Mapbox permet de fournir aux utilisateurs des itinéraires, des instructions de changement de direction, des durées de trajet et des distances entre deux pairs de coordonnées. [9].

3. 3 Base de données

Afin de valider notre contribution, nous avons généré un ensemble de données réelles. Nous avons sélectionné arbitrairement une zone géographique réelle. Ensuite, nous avons généré aléatoirement des emplacements des conducteurs et des passagers en se basant sur cette zone géographique.

La procédure de sélection des données d'entrée que nous avons adoptée est donc générale et peut être appliquée à d'autres zones géographiques dans le monde réel. Nos cas de test sont générés sur la base *d'une zone géographique réelle située dans la wilaya de Guelma*.

Nous avons utilisé l'API Directions de Mapbox pour obtenir les détails de l'itinéraire. La réponse à une requête est un objet JSON qui contient les propriétés suivantes :

- ***Waypoints :*** collection des points de passage distincts.
- ***Routes :*** Une collection d'objets de trajet organisés, qui contient :
    - **Duration :** La durée du trajet, en secondes.
    - **Distance :** La distance du trajet, en mètres.
    - **Legs :** Un ensemble d'objets de segments de trajet, qui contient :
        - **Steps** : Une collection d'objets représentant les étapes de l'itinéraire, chacun contient :
            - **Manœuvre** : Un objet de manœuvre d'une étape, il contient :
                - **Location :** Les coordonnées de la manœuvre sont présentées par des paires de coordonnées GPS (longitude, latitude).
            - **Distance** : La distance de la manœuvre à la prochaine étape du trajet, en mètres.

- **Duration** : Le temps estimé de la manœuvre à la prochaine étape du trajet, en secondes.

La figure 3.1 représente un schéma de processus de traitement d'une requête API Directions de Mapbox et la structure de la réponse.

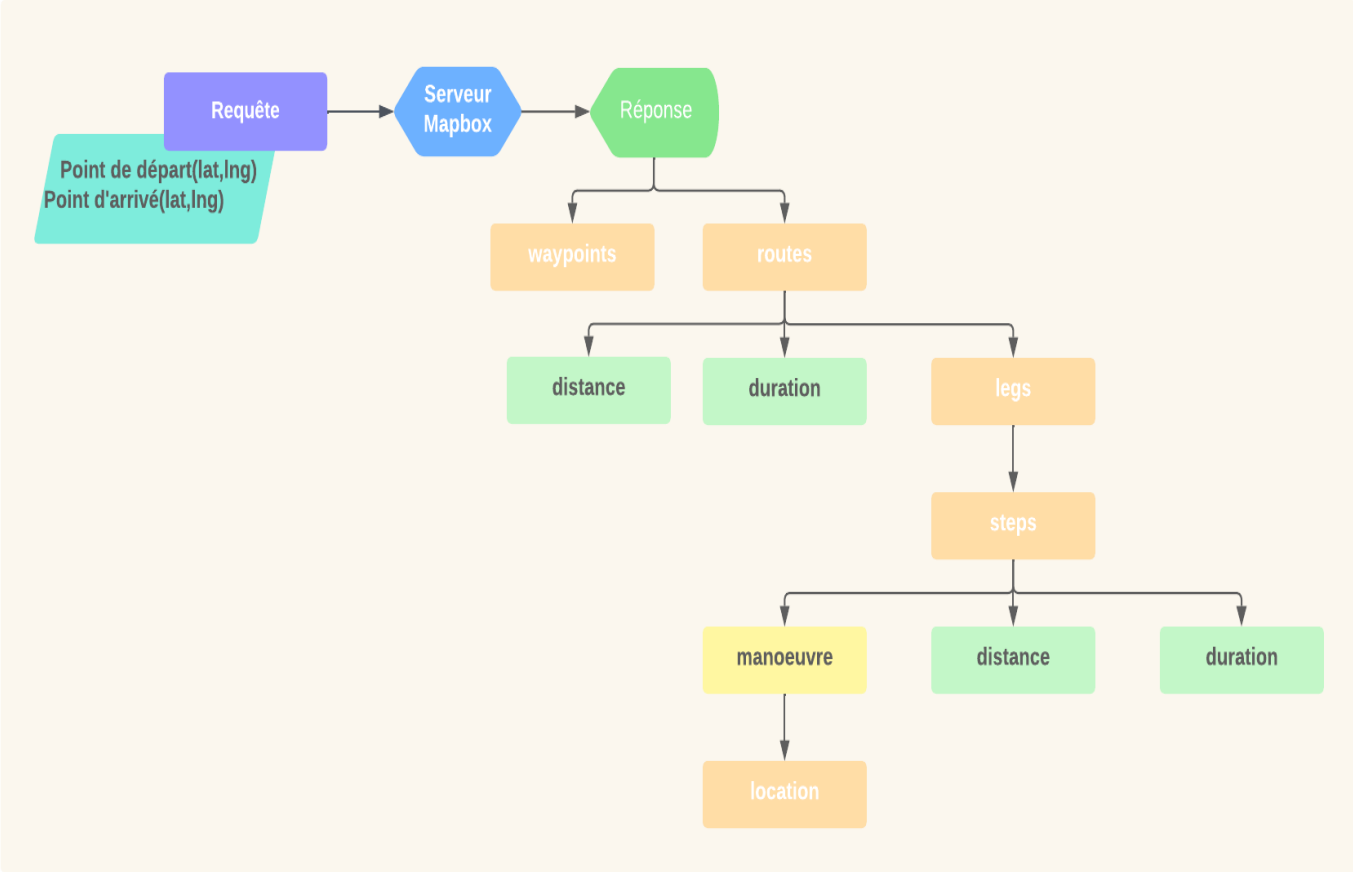


Figure 3.1: *Schéma du processus de traitement d'une requête API Directions de Mapbox.*

3. 4 Résultats

Dans cette section, nous allons présenter l'implémentation de notre adaptation de l'algorithme Firefly dans le contexte du problème de covoiturage dynamique. Nous allons illustrer les détails de notre travail ainsi que les résultats, dans un premier temps, pour un scénario de *deux conducteurs et six passagers.*

3.4.1. Scénario d'exécution

Le tableau 3.1 présente la structure des offres fournies par les conducteurs dans ce scénario.

C_Id	*PD*	*PA*	*TD*	*TA*	*S*
C1	7.141583, 36.157858	7.167036, 36.305	2023-06-17 08:00	2023-06-17 08:40	3
C2	7.165933, 36.309835	7.25168, 36.243316	2023-06-17 08:00	2023-06-17 08:42	2

Tableau 3.1 : *Offres des conducteurs.*

La sortie relative aux offres des conducteurs dans notre application est donnée à la figure 3.2.

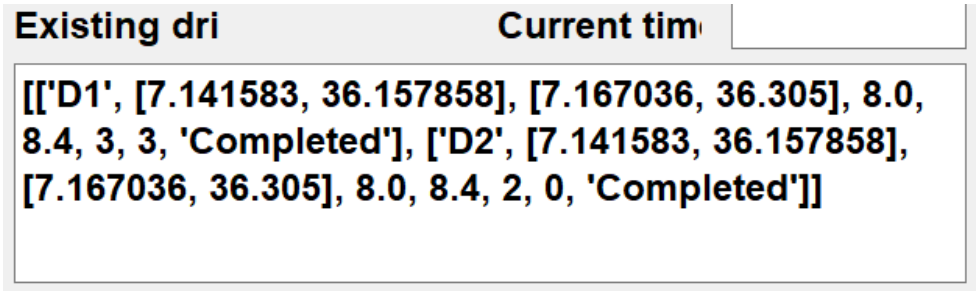


Figure 3.2 : *Sortie des offres.*

Le tableau 3.2 présente les demandes des six passagers.

P_Id	*PD*	*PA*	*TD*	*TA*	*MWT*
P1	7.140029, 36.159306	7.167036, 36.305	2023-06-17 08:00	2023-06-17 08:39	11
P2	7.140852, 36.159523	7.165954, 36.306546	2023-06-17 08:01	2023-06-17 08:38	15
P3	7.165954, 36.306546	7.167036, 36.305	2023-06-17 08:05	2023-06-17 08:09	14
P4	7.251923, 36.246456	7.25168, 36.243316	2023-06-17 08:08	2023-06-17 08:14	0
P5	7.165933, 36.309835	7.251923, 36.246456	2023-06-17 08:20	2023-06-17 09:12	6
P6	7.165933, 36.309835	7.25168, 36.243316	2023-06-17 08:06	2023-06-17 08:52	14

Tableau 3.2 : *Demandes des passagers.*

Où : PD : les coordonnées du point de départ, généré aléatoirement selon la zone géographique située dans la wilaya de Guelma.

PA : les coordonnées du point d'arrivée, généré aléatoirement selon la zone géographique située dans la wilaya de Guelma.

TD : temps de départ, généré dans l'intervalle de 08 :00 à 11 :30 du matin.

TA : temps d'arrivée, généré en fonction de la durée de trajet.

S : capacité maximale du véhicule du conducteur, c'est un nombre aléatoire compris entre 1 et 3.

MWT : le temps d'attente maximal, généré de manière aléatoire entre 0 et 15 minutes.

Nous avons utilisé les paramètres suivants pour l'algorithme Firefly :

β_0 = 1.0 ; facteur d'attractivité initial pour les lucioles,

γ = 0.2 ; coefficient de réduction pour la variation de l'attractivité au fil des itérations,

Nombre d'itérations maximal = 10,

Taille de la population = 10.

La sortie relative aux demandes des passagers dans notre application est donnée à la figure 3.3.

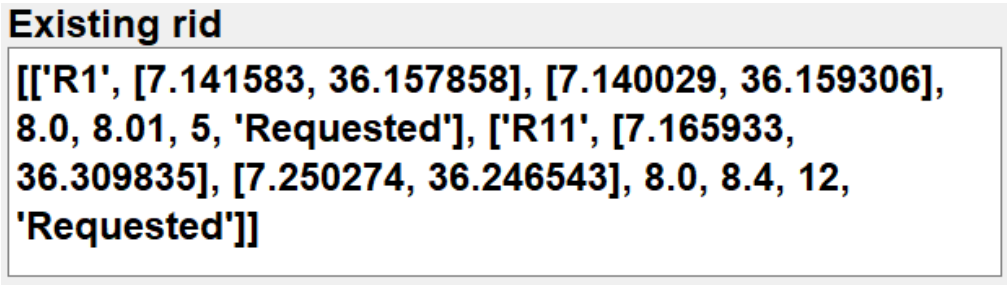


Figure 3.3 : *Sorties des demandes.*

Les offres de covoiturage et les demandes des passagers sont reçues dans une fenêtre de temps spécifique, de 7h30 à 12h00 du matin. Toutes les 30 minutes, de nouvelles offres sont reçues par le système, et toutes les 5 minutes, les demandes des passagers sont reçues. Le tableau 3.3 présente le processus de réception des demandes et des offres du scénario précédent.

Heure	**07 :30 - 07 :35- 07 :40 - 07 :45- 07 :50 - 07 :55**	**07 :55 - 08 :00**	**08 :00 - 08 :05**	**08 :05 - 08 :10**	**08 :10 - 08 :15**	**08 :15 - 08 :20**	**08 :25 - 08 :30**
Offres	C1, C2	-	-	-	-	-	-
Demandes	-	R1	R2, R3	R4, R6	-	R5	-

Tableau 3.3: *Processus de réception des demandes et des offres.*

À 07 :30, le système a reçu les offres des conducteurs 1 et 2, chacune étant déposée au moins 30 minutes avant l'heure de départ respective. Les passagers ont soumis leurs demandes avec une marge de 1 à 5 minutes avant leur temps de départ. Entre 07 :30 et 07 :55, aucune demande n'a été reçue par le système. De 07 :55 à 08 :00, le passager 1 a soumis sa demande, suivis des passagers 2 et 3 entre 08 :00 et 08 :05. Les passagers 4 et 6 ont déposé leurs demandes entre 08 :05 et 08 :10, et enfin, le passager 5 a déposé sa demande entre 08 :15 et 08 :20.

3.4.2. Génération de la population

Nous avons généré simultanément et de manière aléatoire une population pour chaque conducteur et nous les avons traitées en parallèle et en temps réel.

Chaque 5 minutes, cette population est mise à jour avec l'arrivée de nouvelles demandes, en fonction des identifiants des conducteurs et des passagers existants. Les passagers sont sélectionnés en fonction du nombre de sièges disponibles dans le véhicule du conducteur et dans l'ordre de priorité en fonction de leur demande. Lors de la génération, nous avons pris en compte les critères suivants :

- Vérification que l'itinéraire des passagers est inclus dans l'itinéraire du conducteur : Pour cela, nous avons comparé les coordonnées (longitude, latitude) du point de départ des passagers avec les coordonnées de tous les points du trajet du conducteur.
- Vérification du temps de départ des passagers par rapport au temps de départ du conducteur : nous nous sommes assuré que le temps de départ des passagers plus le temps d'attente maximal ne soit pas antérieur au temps de départ du conducteur.

- Les passagers sont pris en fonction du nombre de sièges disponibles dans le véhicule du conducteur. Le tableau 3.4 présente la génération de la population initiale, avec une taille égale à 5.

Conducteur / Heure	C1	C2
08 :00	[1], [1], [1], [1], [1], [1], [1], [1], [1], [1]	-
08 :05	[1, 0], [1, 1], [0, 1], [1, 1], [0, 1], [1, 0], [0, 1], [1, 1], [1, 1], [0, 1]	-
08 :10	[1, 0, 0], [1, 0, 0], [1, 0, 0], [1, 0, 0], [1, 0, 0]	[0, 1, 1], [0, 1, 0], [0, 1, 1], [0, 1, 0], [0, 1, 1]
08 :15	-	-
08 :20	-	[1], [1], [1], [1], [1], [1], [1], [1], [1], [1]
08 :25	-	-
08 :30	-	-

Tableau 3.4: *Génération de la population initiale.*

La figure 3.4 présente une capture de la génération de la population initiale.

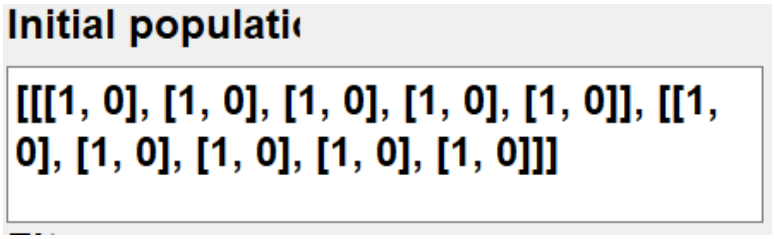


Figure3.4 : *Génération de la population initiale.*

Durant la période de 07 :55 à 08 :00, seule la demande du passager 1 a été reçue. Par conséquent, la population initiale pour l'heure 08 :00 a été générée uniquement en tenant compte du passager 1. Aucune population n'a été générée pour le conducteur 2, car l'itinéraire du passager 2 n'était pas inclus dans l'itinéraire du conducteur 2. Après 5 minutes, une nouvelle population initiale a été générée pour les conducteurs 1 et 2 en incluant les passagers existants et non appariés lors

des périodes précédentes avec les conducteurs existants. Pour chaque plage de temps, nous avons appliqué la même approche pour traiter les demandes des passagers, nous avons pu générer les solutions potentielles correspondantes, en prenant en compte l'itinéraire, le temps et la capacité du véhicule. À 08 :25 et 08 :30, aucun passager existant n'était disponible pour générer la population initiale en fonction de leurs demandes.

3.4.3. Evaluation de la fonction fitness

Notre fonction objectif vise à minimiser à la fois la distance et le temps d'attente. Pour ce faire, nous avons calculé les valeurs de fitness en utilisant l'équation (2.1), qui tient compte à la fois de la distance et du temps d'attente. Ces valeurs ont été utilisées pour évaluer la qualité de chaque solution potentielle et guider le processus d'optimisation vers des solutions plus efficaces.

Pour évaluer la distance entre les conducteurs et les passagers, nous avons utilisé l'API Mapbox Distance. Cette API nous a permis de calculer la distance en fonction des coordonnées des points de départ et d'arrivée des conducteurs et des passagers. Pour le temps, nous avons effectué la soustraction du temps d'attente maximum à le temps actuelle.

Le tableau 3.5 présente les résultats de calcul de la fonction de fitness pour quelque population initiales.

Conducteur / **Heure**	**C1**		**C2**	
	Population initiale	**Fitness**	**Population initiale**	**Fitness**
08 :00	1	983.105	-	-
08 :05	1, 0	1423.592	-	-
	1, 1	19960.717		
	0, 1	18537.124		
08 :10	1, 0, 0	18550.299	0, 1, 0	485.502
08 :20	-	-	1	-381.096

Tableau 3.5: *Résultat de calcul du fitness.*

La figure 3.5 présente la calcul de la fitness.

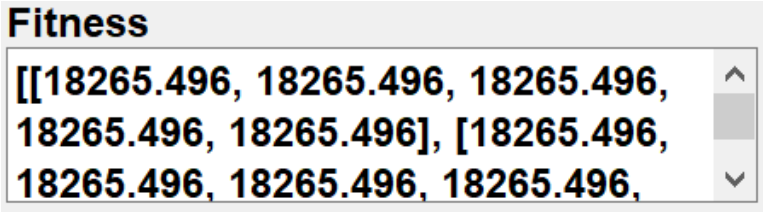


Figure 3.5 : *Calcul de la fitness.*

Après avoir généré les populations initiales à différents moments, nous calculons la fitness de chaque sous-population générée (luciole). Ensuite, nous effectuons des vérifications sur les valeurs de fitness obtenues. Dans le tableau 3.5, à 08 :15, si la valeur de fitness est négative, nous éliminons cette solution. De plus, si la valeur de fitness est égale à zéro, nous éliminons également cette solution, sauf dans le cas où l'itinéraire du conducteur est le même que celui du passager, les temps de départ et d'arrivée sont identiques, et le temps d'attente maximal est également égal à zéro.

3.4.4. Mise à jour des solutions

Nous avons effectué une mise à jour de la population initiale de chaque conducteur à chaque itération en utilisant une combinaison d'exploration et d'exploitation basée sur la valeur de fitness de chaque luciole. Les solutions potentielles ayant une valeur de fitness plus faible ont été identifiées comme faisant partie des meilleures solutions. Les autres solutions ont été progressivement déplacées vers ces meilleures solutions en appliquant l'exploitation et l'exploration à chaque itération, jusqu'à atteindre notre condition d'arrêt, qui est le nombre d'itérations maximal.

Le tableau 3.6 présenté les résultats d'exploitation et exploration sur deux lucioles.

Conducteur / Heure	**C1**		**Mise à jour les solutions**			
			Exploitation		**Exploration**	
	Lucioles	**Fitness**	**Luciole**	**Fitness**	**Luciole**	**Fitness**
08 :05	1, 0	1423.592	1, 0	1423.592	0, 1	18537 .124
	1, 1	19960.717				

Tableau 3.6: *le résultat d'exploitation et exploration sur deux lucioles.*

Nous avons comparé les valeurs de fitness de chaque luciole au sein de la même population. Dans notre exemple, nous avons comparé 19960.717 à 1423.592, et avons déplacé la deuxième luciole (fitness = 19960.717) vers la première (fitness = 1423.592) en utilisant la distance de Hamming et le calcul de l'attractivité β.

Ensuite, nous avons appliqué l'exploitation pour exploiter de nouvelles solutions potentielles. Nous avons identifié les meilleures solutions pour chaque population en utilisant la méthode d'exploitation discutée dans le chapitre précédent.

Nous avons appliqué l'exploration pour explorer de nouvelles solutions et les ajuster afin de les améliorer. Nous avons utilisé la mutation par échange, en échangeant deux indices spécifiques et en calculant la fitness de ces nouvelles solutions.

La solution globale est déterminée par la solution qui présente la fitness minimale. Dans notre cas, la luciole [1, 0] avec une fitness de 1423.592 est la meilleur solution globale.

Cette approche de comparaison, exploitation et exploration nous a permis de trouver des solutions optimales et de continuer à améliorer nos résultats.

3.4.1. Résultat final

Le tableau 3.7 présenté le résultat final de l'appariement de l'échantillon de données.

Heure	**Passagers appariés**	
	C1	**C2**
08 :00	R1	-
08 :05	R2	-
08 :10	R3	R6
08 :20	-	-

Tableau 3.7:*Résultat final d'appariement.*

Le nombre de passagers satisfaits est 4. Cependant, il y a 2 passagers qui ne sont pas satisfaits. La figure 3.6 illustre le résultat d'appriement.

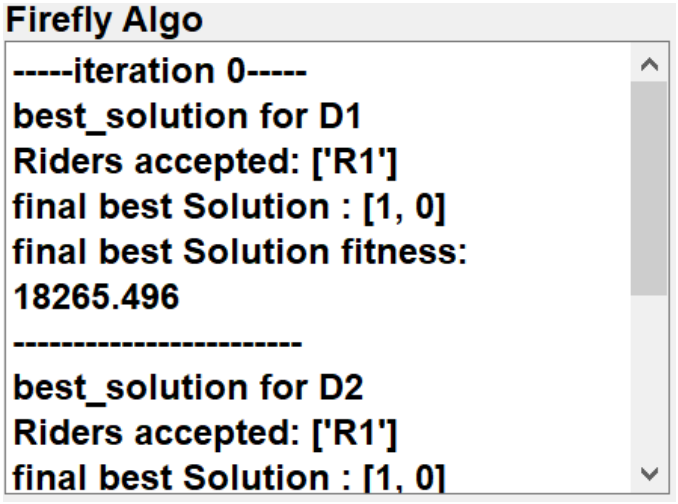


Figure 3.6 : *Résultat d'appariement.*

3. 5 Expérimentation

Pour démontrer l'efficacité et la performance de notre méthode, nous présentons 4 scénarios où on fait varier le nombre de véhicules (conducteur) et le nombre de passagers. Ceci a pour objectif de tester le comportement et la performance de notre approche et tester aussi le passage à l'échelle.

Nous lançons notre simulation de covoiturage de 7h30 à 12h00 et nous imposons des plages horaires rapprochées, permettant ainsi de satisfaire un maximum de demandes de passagers. Les résultats de performance de notre approche sont présentés dans le Tableau 3.8.

Scénario	**C**	**P**	**Siège disponible**	**Nombre de passagers acceptés**	**Nombre de passagers rejetés**	**Temps d'attente gagné**	**Distance parcourue (m)**	**Temps d'exécution (s)**
1	1	5	3	3	2	720	18239.949	173.329
2	5	15	12	11	4	2630.695	53512.739	372.588
3	10	25	23	19	6	4555.99	75257.269	412.8
4	15	35	33	29	8	6729.731	86371.747	579.59

Tableau 3.8: *les résultats de performance.*

Dans le premier scénario, nous avons un conducteur avec 5 passagers, dans ce cas nous avons choisi les points de départ et d'arrivée de manière aléatoire sur les trajets de conducteurs, avec

intervalle de temps similaire au temps de conducteur. Le temps d'attente des passagers était également aléatoire, variant entre 0 et 15 minutes. Le nombre de sièges disponibles dans le véhicule est fixé à 3. Dans ce cas, nous avons constaté que 3 passagers étaient acceptés tandis que 2 passagers étaient rejetés. Avec temps d'exécution 161.5s.

Dans le deuxième scénario, nous avons fixé le nombre de conducteurs à 5 et le nombre de passagers à 15, alors que seuls 12 sièges étaient disponibles. Dans ce scénario, nous avons modifié le nombre d'itérations pour évaluer son impact sur les résultats. Tout d'abord, en fixant le nombre maximal d'itérations à 10, nous avons constaté que 11 passagers étaient acceptés tandis que 4 ne l'étaient pas, avec temps d'exécution égale à 372.588s. En augmentant le nombre d'itérations à 50, nous avons observé une détérioration, avec 10 passagers satisfaits et 5 insatisfaits, avec temps d'exécution égale à 905.317s.

Dans le scénario 3, nous avons fixé le nombre de conducteurs à 10 et le nombre de passagers à 25, alors que 23 sièges étaient disponibles. Nous avons constaté que 19 passagers étaient acceptés et 6 rejetés, avec temps d'exécution égale à 412.8s.

Dans le scénario 4, nous avons 15 conducteurs et 35 passagers, ainsi que 33 sièges disponible. Nous avons obtenu 29 passagers étaient acceptés et 8 rejetés, avec temps d'exécution égale à 579.59s.

L'histogramme ci-dessous (figure 3.7) correspond aux valeurs moyennes des passagers acceptés et rejetés.

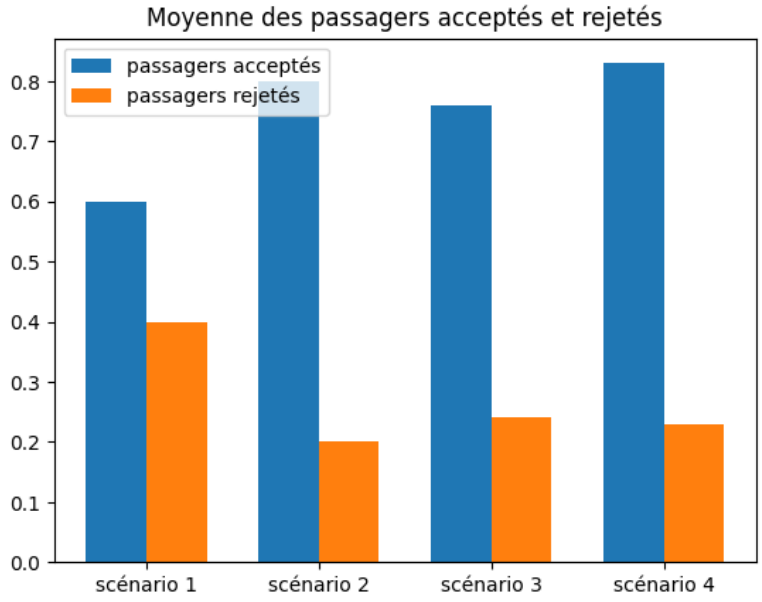


***Figure 3.7** : Moyenne des passagers acceptés et rejetés.*

La figure 3.8 présente un histogramme correspondant aux valeurs de temps d'attente gagné et la distance parcourue pour chaque scénario.

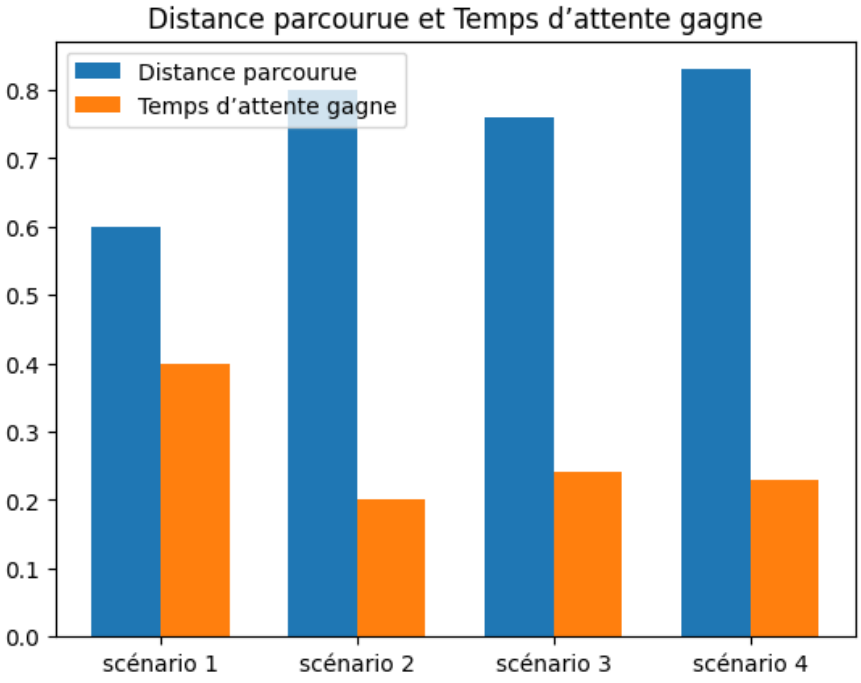


Figure 3.8 : *Temps d'attente gagné et la distance parcourue pour chaque scénario.*

L'utilisation de l'algorithme Firefly dans le contexte du covoiturage dynamique a mis en évidence son efficacité et les résultats prometteurs obtenus en sont témoins.

Les résultats de notre étude ont démontré que l'algorithme Firefly est capable de fournir des appariements efficaces, conduisant à une satisfaction accrue des passagers. En optimisant les itinéraires et en équilibrant la demande et l'offre, nous avons constaté une réduction significative des temps d'attente des passagers. Cela signifie que les passagers sont moins contraints par de longues périodes d'attente et peuvent profiter d'une expérience de voyage plus agréable et pratique.

3. 6 Conclusion

Dans ce chapitre, nous avons présenté l'implémentation de notre approche et présenté en détail les différentes étapes de développement de notre système. Nous avons également exposé les résultats obtenus à chaque étape. Les expérimentations que nous avons réalisées ont permis de tester la fiabilité, l'efficacité et la pertinence de notre solution. En effet, les résultats obtenus sont extrêmement encourageants et viennent confirmer la justesse de nos choix et de

notre modélisation. Ils démontrent l'efficacité de notre approche dans la résolution du problème posé et renforcent notre confiance dans sa capacité à fournir des solutions pertinentes pour la gestion des trajets en covoiturage.

Conclusion générale

Le covoiturage dynamique émerge comme une solution moderne et flexible pour répondre aux besoins de déplacement dans nos sociétés contemporaines. En s'adaptant aux contraintes spatio-temporelles, il offre aux utilisateurs une manière pratique et rapide de trouver des trajets partagés, améliorant ainsi l'efficacité globale des déplacements.

Dans cette étude, nous avons exploré l'utilisation de l'algorithme métaheuristique Firefly pour résoudre le problème d'appariement dynamique dans le covoiturage. Notre objectif était de minimiser la distance parcourue et le temps d'attente des passagers, tout en tenant compte des contraintes de capacité, des contraintes spatio-temporelles et des contraintes de temps d'attente des passagers.

Les résultats obtenus à travers nos expérimentations ont démontré l'efficacité et la performance de notre système basé sur l'algorithme Firefly. Nous avons observé une amélioration significative du nombre de passagers satisfaits, grâce à une réduction notable des temps d'attente. De plus, nous avons réussi à minimiser la distance totale parcourue, contribuant ainsi à une utilisation plus efficace des ressources de transport.

Ces résultats prometteurs renforcent l'importance des algorithmes métaheuristiques dans la résolution des problèmes complexes liés au covoiturage dynamique. L'algorithme Firefly s'est avéré être un outil efficace pour optimiser les appariements entre conducteurs et passagers, offrant une solution avantageuse pour les utilisateurs.

Des perspectives d'amélioration de notre travail restent, toutefois, indispensables.

1. Optimisation des paramètres de l'algorithme : Il est possible de mener des études approfondies pour déterminer les meilleurs paramètres d'optimisation de l'algorithme Firefly spécifiquement pour le problème du covoiturage dynamique.
2. Intégration de préférences spécifiques des utilisateurs : Les utilisateurs du covoiturage dynamique ont souvent des préférences particulières, telles que des critères de confort, d'accompagnement ou de flexibilité des horaires.
3. Gestion de contraintes plus complexes : Actuellement, les contraintes prises en compte dans le covoiturage dynamique se limitent aux contraintes de capacité et aux contraintes spatio-temporelles. Cependant, il existe d'autres contraintes potentielles à prendre en

compte, telles que les contraintes de détour, de partage, des contraintes environnementales, etc.

Références bibliographiques

Agatz, N., Erera, A. L., Savelsbergh, M. W., & Wang, X. (2011). Dynamic ride-sharing: A simulation study in metro Atlanta. Procedia-Social and Behavioral Sciences, 17, 532-550.

Agatz, N., Erera, A., Savelsbergh, M., & Wang, X. (2012). Optimization for dynamic ride-sharing: A review. European Journal of Operational Research, 223(2), 295-303.

Armant, V., & Brown, K. N. (2014, November). Minimizing the driving distance in ride sharing systems. In 2014 IEEE 26th International Conference on Tools with Artificial Intelligence (pp. 568-575). IEEE.

Ballet, J. C., & Clavel, R. (2007). Le covoiturage en France et en Europe: état des lieux et perspectives (Doctoral dissertation, Centre d'études sur les réseaux, les transports, l'urbanisme et les constructions publiques (CERTU)).

Ben Cheikh, S., Tahon, C., & Hammadi, S. (2017). An evolutionary approach to solve the dynamic multihop ridematching problem. Simulation, 93(1), 3-19.

Bidar, M., Mouhoub, M., & Sadaoui, S. (2018, July). Discrete firefly algorithm: A new metaheuristic approach for solving constraint satisfaction problems. In 2018 IEEE Congress on Evolutionary Computation (CEC) (pp. 1-8). IEEE.

Cheikh, S. B., & Hammadi, S. (2016). Multi-Hop Ridematching optimization problem: Intelligent chromosome agent-driven approach. Expert Systems with Applications, 62, 161-176.

Cheikh, S. B., Hammadi, S., & Tahon, C. (2015). Agent-based evolutionary cooperative approach for dynamic multi-hop ridematching problem. IFAC-PapersOnLine, 48(3), 887-892.

Cheikh-Graiet, S. B., Dotoli, M., & Hammadi, S. (2020). A Tabu Search based metaheuristic for dynamic carpooling optimization. Computers & Industrial Engineering, 140, 106217.

de Carvalho, V. R., & Golpayegani, F. (2022). Satisfying user preferences in optimised ridesharing services: A multi-agent multi-objective optimisation approach. Applied Intelligence, 52(10), 11257-11272.

Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M. E., Wang, X., & Koenig, S. (2013). Ridesharing: The state-of-the-art and future directions. Transportation Research Part B: Methodological, 57, 28-46.

Gao, J., Wong, T., Selim, B., & Wang, C. (2022). VOMA: A privacy-preserving matching mechanism design for community ride-sharing. IEEE Transactions on Intelligent Transportation Systems, 23(12), 23963-23975.

Ghilas, V., Demir, E., & Van Woensel, T. (2016). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. Computers & Operations Research, 72, 12-30.

Herbawi, W., & Weber, M. (2012, June). The ridematching problem with time windows in dynamic ridesharing: A model and a genetic algorithm. In 2012 IEEE congress on evolutionary computation (pp. 1-8). IEEE.

Hsieh, F. S. (2020). A comparative study of several metaheuristic algorithms to optimize monetary incentive in ridesharing systems. ISPRS International Journal of Geo-Information, 9(10), 590.

Hsieh, F. S. (2022). Trust-based recommendation for shared mobility systems based on a discrete self-adaptive neighborhood search differential evolution algorithm. Electronics, 11(5), 776.

Hsieh, F. S. (2022, June). A Hybrid Firefly-DE algorithm for Ridesharing Systems with Cost Savings Allocation Schemes. In 2022 IEEE World AI IoT Congress (AIIoT) (pp. 649-653). IEEE.

Huang, S. C., Jiau, M. K., & Liu, Y. P. (2018). An ant path-oriented carpooling allocation approach to optimize the carpool service problem with time windows. IEEE Systems Journal, 13(1), 994-1005.

Huang, Y., Jin, R., Bastani, F., & Wang, X. S. (2013). Large scale real-time ridesharing with service guarantee on road networks. arXiv preprint arXiv:1302.6666.

Javidi, H., Simon, D., Zhu, L., & Wang, Y. (2021, January). A multi-objective optimization framework for online ridesharing systems. In 2021 IEEE International Conference on Big Data and Smart Computing (BigComp) (pp. 252-259). IEEE.

Jeribi, K. (2012). Conception et realisation d'un systeme de gestion de vehicules partages: de la multimodalite vers la co-modalite (Doctoral dissertation, Ecole Centrale de Lille).

Karthikeyan, S., Asokan, P., Nickolas, S., & Page, T. (2015). A hybrid discrete firefly algorithm for solving multi-objective flexible job shop scheduling problems. International Journal of Bio-Inspired Computation, 7(6), 386-401.

Kumar, R., Memoria, M., & Chandel, A. (2020, June). Performance analysis of proposed mutation operator of genetic algorithm under scheduling problem. In 2020 International Conference on Intelligent Engineering and Management (ICIEM) (pp. 193-197). IEEE.

Kuo, I. H., Horng, S. J., Kao, T. W., Lin, T. L., Lee, C. L., Terano, T., & Pan, Y. (2009). An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model. Expert systems with applications, 36(3), 7027-7032.

Martins, L. D. C., de la Torre, R., Corlu, C. G., Juan, A. A., & Masmoudi, M. A. (2021). Optimizing ride-sharing operations in smart sustainable cities: Challenges and the need for agile algorithms. Computers & Industrial Engineering, 153, 107080.

Mourad, A., Puchinger, J., & Chu, C. (2019). A survey of models and algorithms for optimizing shared mobility. Transportation Research Part B: Methodological, 123, 323-346.

Plate, O. (2019). Ridesharing with Multiple Riders (Doctoral dissertation, Karlsruhe Institute of Technology).

Shen, B., Huang, Y., & Zhao, Y. (2016). Dynamic ridesharing. Sigspatial Special, 7(3), 3-10.

Silwal, S., Gani, M. O., & Raychoudhury, V. (2019, June). A survey of taxi ride sharing system architectures. In 2019 IEEE International Conference on Smart Computing (SMARTCOMP) (pp. 144-149). IEEE.

Singh, P., Meena, N. K., Yang, J., & Slowik, A. (2020). Swarm intelligence algorithms: A tutorial. Swarm Intelligence Algorithms, 1-15.

Smet, P. (2021). Ride sharing with flexible participants: a metaheuristic approach for large - scale problems. International Transactions in Operational Research, 28(1), 91-118.

Srivatsa, D., Teja, T. K., Prathyusha, I., & Jeyakumar, G. (2019). An empirical analysis of genetic algorithm with different mutation and crossover operators for solving Sudoku. In Pattern Recognition and Machine Intelligence: 8th International Conference, PReMI 2019, Tezpur, India, December 17-20, 2019, Proceedings, Part I (pp. 356-364). Springer International Publishing.

Tafreshian, A., & Masoud, N. (2020). Using subsidies to stabilize peer-to-peer ridesharing markets with role assignment. Transportation Research Part C: Emerging Technologies, 120, 102770.

Ting, K. H., Lee, L. S., Pickl, S., & Seow, H. V. (2021). Shared mobility problems: a systematic review on types, variants, characteristics, and solution approaches. Applied Sciences, 11(17), 7996.

Wang, X., Agatz, N., & Erera, A. (2018). Stable matching for dynamic ride-sharing systems. Transportation Science, 52(4), 850-867.

Yang, X. S. (2009). Firefly algorithms for multimodal optimization. In Stochastic Algorithms: Foundations and Applications: 5th International Symposium, SAGA 2009, Sapporo, Japan, October 26-28, 2009. Proceedings 5 (pp. 169-178). Springer Berlin Heidelberg.

Yang, X. S. (2010). Nature-inspired metaheuristic algorithms. Luniver press.

Zhan, X., Szeto, W. Y., Shui, C. S., & Chen, X. M. (2021). A modified artificial bee colony algorithm for the dynamic ride-hailing sharing problem. Transportation Research Part E: Logistics and Transportation Review, 150, 102124.

Webographie

[1] *python*. https://www.python.org/doc/essays/blurb, (Consulté le 27 mai 2023)

[2] *visualStudioCode*. https://code.visualstudio.com/docs, (Consulté le 27 mai 2023)

[3] *NumPy*. https://numpy.org/doc/stable/, (Consulté le 27 mai 2023)

[4] *Pandas*. https://pypi.org/project/pandas/, (Consulté le 27 mai 2023)

[5] *Ast*. https://docs.python.org/fr/3.8/library/ast.html, (Consulté le 27 mai 2023)

[6] *Random*.https://docs.python.org/fr/3.8/library/random.html?highlight=random#random.random, (Consulté le 27 mai 2023)

[7] *Math*. https://docs.python.org/fr/3.8/library/math.html?highlight=math#module-math, (Consulté le 27 mai 2023)

[8] *Datetime*. https://docs.python.org/fr/3.8/library/datetime.html?highlight=datetime#module-datetime, (Consulté le 27 mai 2023)

[9] https://docs.mapbox.com/help/tutorials/getting-started-directions-api/ (consulté le 27 mai 2023)