POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Problème d'optimisation d'un système de transport à la demande sur plusieurs gares

ELMEHDI MEHIRI
Département de mathématiques et génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Mathématiques appliquées

Février 2025

© Elmehdi Mehiri, 2024.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Problème d'optimisation d'un système de transport à la demande sur plusieurs gares

présenté par **Elmehdi MEHIRI**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Michel GENDREAU, président
Antoine LEGRAIN, membre et directeur de recherche
Quentin CAPPART, membre et codirecteur de recherche
Guy DESAULNIERS, membre

REMERCIEMENTS

Je tiens tout d'abord à remercier mes directeurs de recherche, Antoine Legrain et Quentin Cappart, pour avoir accepté de m'encadrer et m'avoir permis de travailler sur ce projet. Tous deux m'ont beaucoup épaulé et encouragé tout au long de ce projet, que ce soit pour me suggérer des idées ou relire mon travail. Sans eux, je ne serais certainement pas là où je suis actuellement.

Ensuite, Je souhaite également remercier Michel Gendreau, qui a accepté de présider mon jury, ainsi que Guy Desaulniers, qui a accepté d'en faire partie. Je les remercie chaleureusement l'un et l'autre pour leurs précieuses corrections et suggestions, qui ont contribué de manière significative à l'amélioration de la qualité de ce document.

Je tiens à remercier également la Société nationale des chemins de fer français (SNCF) pour le financement accordé.

Je remercie aussi le GERAD et le CIRRELT pour m'avoir accueilli, ainsi que leur administration pour toute l'aide fournie au quotidien.

Finalement, j'aimerais remercier tous mes collègues de laboratoire. En particulier, je remercie Öykü Naz Attila, qui m'a beaucoup aidé sur ce projet. Je tiens à la remercier pour sa disponibilité chaque fois que j'avais besoin d'aide, ainsi que pour le code Python qu'elle a développé et qui m'a été très utile.

RÉSUMÉ

La SNCF (Société nationale des chemins de fer français) a développé un système de transport combinant les avantages du rail et la flexibilité de la route. Ce système est un transport à la demande destiné à servir les passagers dans les gares, en les transportant chacun vers un lieu spécifique, puis de les ramener de leur emplacement vers les gares, tout en satisfaisant les préférences de chaque passager, telles que les temps de trajet maximum et les temps de disponibilité, et en respectant les ressources disponibles, comme le nombre de véhicules, leur capacité et les heures de départ des trains. L'efficacité de ce système de transport dépend de l'optimisation des itinéraires des véhicules qui circulent sur le réseau routier.

Notre but ici est d'optimiser ce système de transport en fournissant un modèle mathématique valide et une méthode d'optimisation. Nous commençons par situer le problème dans la littérature, où nous avons trouvé que le problème peut être vu comme une intersection entre le problème Dial-a-Ride et le problème de tournées de véhicules avec *Backhauls*. Ensuite, nous formulons le problème comme un programme linéaire en nombres entiers mixtes en le décrivant sur le graphe du réseau routier, tout en respectant les contraintes de *Backhauls*, et où l'objectif est de minimiser les coûts de déplacement ainsi que les coûts des passagers non desservis. Nous proposons par la suite une décomposition du problème en un problème maître et des sous-problèmes, puis nous détaillons la mise en œuvre de la méthode de résolution qui est une heuristique de plongée basée sur la génération de colonnes. Nous modélisons le sous-problème comme un problème de plus court chemin avec contraintes de ressources. Ce dernier est résolu à l'aide d'une méthode de programmation dynamique exacte basée sur un algorithme d'étiquetage adapté pour notre problème. Enfin, nous testons notre méthode en utilisant 40 instances générées synthétiquement pour évaluer l'efficacité de la méthode de résolution proposée et du programme en nombres entiers, ainsi que la qualité des solutions fournies. Les résultats démontrent l'efficacité de la méthode de résolution, qui a permis de résoudre de manière optimale 80% des instances et de fournir un écart d'intégralité moyen de seulement $3,6\%$.

ABSTRACT

The SNCF (French railway company) has developed a transportation system combining the advantages of rail and the flexibility of road. This system is a demand-responsive transport system designed to serve passengers at stations, by transporting each passenger to a specific location and bringing passengers from their locations to the stations. It meets each passenger's preferences, such as maximum travel times and availability times. It also respects available resources, including the number of vehicles, their capacities, and train departure times. The efficiency of this transportation system depends on the optimization of its vehicles routes as they navigate a road network.

To address this, our goal is to optimize the transportation system by providing a valid mathematical model and an optimization method. First, we start by positioning the problem in the literature, where we found that the problem can be seen as an intersection between the Dial-a-Ride Problem and the Vehicle Routing Problem with Backhauls. Then, we formulate the problem as a mixed integer program by describing the problem on the road network graph, that preserves the backhaul constraints, and where the objective is to minimize the costs of traveling and unserved passengers. Afterwards, we propose a decomposition of the problem into a master problem and subproblems, then we detail the implementation of the resolution method which is a Column Generation-based Diving Heuristic. We model the subproblem as a shortest path problem with resource constraints. This subproblem is solved using an exact dynamic programming method based on a labeling algorithm adapted to our problem. Finally, we test our method using a set of synthetically generated instances. This allows us to evaluate the effectiveness of the proposed resolution method, the integer program, and the quality of the provided solutions. The results demonstrate the effectiveness of the resolution method where it was able to solve optimally 80% of total instances and provide an integrality gap average of only 3.6%.

TABLE DES MATIÈRES

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT vi

TABLE DES MATIÈRES vii

LISTE DES TABLEAUX ix

LISTE DES FIGURES x

LISTE DES SIGLES ET ABRÉVIATIONS xi

CHAPITRE 1 Introduction 1
- 1.1 Définitions et concepts de base 1
- 1.2 Objectifs de recherche 6
- 1.3 Plan du mémoire 6

CHAPITRE 2 Revue de littérature 8
- 2.1 Problèmes de collecte et livraison 8
  - 2.1.1 Classification de problèmes de collecte et livraison 11
  - 2.1.2 Contraintes supplémentaires 11
  - 2.1.3 Problème de transport à la demande 13
  - 2.1.4 Problème de collecte et livraison avec *Backhauls* 14
- 2.2 Méthodes de résolution 15
  - 2.2.1 Méthodes exactes 15
  - 2.2.2 Méthodes heuristiques et métaheuristiques 17
  - 2.2.3 Génération de colonnes 18

CHAPITRE 3 Formulation d'un système de transport à la demande sur plusieurs gares 21
- 3.1 Description du problème 21
- 3.2 Réseau associé 22
- 3.3 Formulation mathématique 26

CHAPITRE 4 Méthodologie de résolution 33

4.1 Résolution par génération de colonnes et l'arrondi des valeurs fractionnaires dans un contexte de recherche en profondeur 33
4.1.1 Problème maître 33
4.1.2 Sous-problème 35
4.1.3 Description de l'algorithme général 36
4.2 Résolution du sous-problème 42
4.2.1 Formulation générique du problème de plus court chemin avec contraintes de ressources 43
4.2.2 Formulation spécifique du problème de plus court chemin avec contraintes de ressources pour notre problème 44
4.2.3 Algorithmes d'étiquetage basés sur la programmation dynamique . . 49
4.2.4 Adaptation d'un algorithme d'étiquetage monodirectionnel avant pour résoudre les sous-problèmes 50

CHAPITRE 5 Résultats numériques 56
5.1 Paramètres Généraux 56
5.2 Génération d'instances 57
5.3 Résultats 58
5.4 Analyse de la qualité des solutions 62

CHAPITRE 6 Conclusion 67
6.1 Synthèse des travaux 67
6.2 Limitations de la solution proposée 68
6.3 Perspectives de travaux futurs 68

RÉFÉRENCES 69

LISTE DES TABLEAUX

Tableau 3.1 Notations utilisées pour le modèle mathématique 27
Tableau 4.1 Informations sur les arcs 48
Tableau 4.2 Informations sur les fenêtres de consommation de ressources sur les nœuds 48
Tableau 5.1 Résultats des tests du MIP et $CGDH$ 61
Tableau 5.2 Comparaison des résultats 62
Tableau 5.3 La fonction objectif de chaque modèle 63
Tableau 5.4 Comparaison des qualités des solutions 64
Tableau 5.5 Comparaison des moyennes de performance 65

LISTE DES FIGURES

Figure 1.1 Véhicule utilisé par le service «Ma course SNCF» 4
Figure 1.2 Exemple simplifié d'un réseau de 3 gares, 3 véhicules, 6 passagers, et une solution réalisable 5
Figure 2.1 Exemple de modélisation de VRP à 6 sommets et deux véhicules . . . 9
Figure 2.2 Divers structures de demandes 10
Figure 2.3 Classification des problèmes de collecte et livraison 12
Figure 3.1 Un exemple du graphe associé G d'une instance de 2 gares, chacune avec deux requêtes de livraison et deux requêtes de collecte. 25
Figure 4.1 Illustration d'une heuristique de plongée dans un arbre d'énumération 38
Figure 4.2 Organigramme de la procédure de résolution heuristique de plongée basée sur la génération de colonnes (CGDH) 42
Figure 4.3 Illustration d'un réseau du SPPRC associé à un sous-problème d'un véhicule initialement situé à la gare 1. Consulter les Tableaux 4.1 et 4.2 pour les détails des couleurs utilisées. 45
Figure 4.4 Exemple d'un chemin partiel du véhicule k desservant un passager de type livraison d'une autre gare 52
Figure 5.1 Visualisation d'une solution trouvée après l'exécution du code 57

LISTE DES SIGLES ET ABRÉVIATIONS

Abbrévation	Signification en anglais	Signification en français
CGDH	Column Generation-based Diving Heuristic	Heuristique de plongée basée sur la génération de colonnes
DARP	Dial-a-Ride Problem	Problème de transport à la demande
MIP	Mixed Integer Program	Programme en nombres entiers mixte
MP	Master problem	Problème maître
PDP	Pickup and Delivery Problems	Problèmes de collecte et de livraison
REF	Resource Extension Function	Fonction d'extension des ressources
RMP	Restricted Master Problem	Problème maître restreint
SNCF	National Company of the French Railways	Société nationale des chemins de fer français
SPMIP	Subproblem as a Mathematical Integer Program	Sous-problème en tant que programme mathématique en nombres entiers
SPP	Shortest Path Problem	Problème de plus court chemin
SPPRC	Shortest Path Problem with Resource Constraints	Problème de plus court chemin avec contraintes de ressources
VRP	Vehicle Routing Problem	Problème de tournées de véhicules
VRPB	Vehicle Routing Problem with Backhauls	Problème de tournées de véhicules avec les retours en arrière
VRPPD	Vehicle Routing Problem with Pickup and Delivery	Problème de tournées de véhicules avec collecte et livraison
VRPTW	Vehicle Routing Problem with time windows	Problème de tournées de véhicules avec fenêtres de temps

CHAPITRE 1 Introduction

1.1 Définitions et concepts de base

Les problèmes de tournées de véhicules (« *Vehicle Routing Problem* » - VRP) constituent une catégorie fondamentale de la recherche opérationnelle. Les coûts de transport constituant une part substantielle du coût global d'un produit, la résolution des VRP revêt donc une importance pratique significative dans diverses industries. Pour les entreprises spécialisées dans la livraison de marchandises ou le transport de passagers, comme UPS, FedEx et Postes Canada, la résolution des VRP est essentielle.

Le problème de tournées de véhicules de base consiste à déterminer un ensemble de routes, qui couvrent un ensemble de clients ayant des demandes connues, à l'aide d'une flotte de véhicules identiques localisés initialement à un dépôt central, tout en minimisant les coûts de déplacement totaux. Pour refléter les contraintes des problèmes réels, de nombreuses variantes de ce problème ont également été formulées. Parmi les variantes de base du VRP, citons : le VRP avec fenêtres de temps (« *Vehicle Routing Problem with Time Windows* » - VRPTW), pour lequel le service des clients doit avoir lieu dans un intervalle de temps prédéfini ; le VRP avec contraintes de capacité (« *Capacitated Vehicle Routing Problem* » - CVRP), où les véhicules ont une capacité de transport limitée pour les marchandises/passagers qui doivent être transportés ; le VRP avec collecte et livraison (« *Vehicle Routing Problem with Pickup and Delivery* » - VRPPD), dans lequel une flotte de véhicules hétérogènes, localisée à plusieurs dépôts, doit satisfaire un ensemble de demandes de transport. Chaque demande est définie par un point de collecte et un point de livraison correspondant et une demande à transporter.

Depuis sa première apparition en 1959 [1], plusieurs études ont été réalisées pour explorer la richesse des problèmes des VRP, qui intègrent des contraintes plus complexes ayant des applications directes dans les problèmes de la vie réelle. Ce mémoire s'inscrit dans cette thématique de recherche. Il porte sur un VRPPD dont les éléments à transporter sont des passagers, ce qui implique un ensemble de contraintes de temps appliquées sur ces passagers. Ce problème peut s'énoncer brièvement comme suit. Étant donné un ensemble de dépôts, un ensemble d'éléments séparés en deux parties, une partie distribuée au niveau des dépôts (type *Delivery*), et l'autre partie localisée en dehors des dépôts (type *Pickup*), et une flotte de véhicules non identiques possédant une capacité finie basée au niveau d'un ensemble de dépôts, quel est l'ensemble des routes partant et se terminant à un dépôt qui permet de livrer les éléments de type *Delivery*, et de collecter les éléments type *Pickup*, tout en minimisant la somme totale des coûts de transport ?

L'origine de ce problème vient d'un problème réel rencontré par La Société nationale des chemins de fer français (SNCF). Étant donné que la SNCF dessert plus de 3 000 gares sur le réseau ferroviaire français et gère plus de 15 000 départs de trains transportant 10 millions de voyageurs par jour [2], l'optimisation de son système de transport est une nécessité. Il existe des populations situées dans des zones éloignées des gares existantes, où la construction de nouvelles gares n'est pas une solution efficace, soit pour des raisons économiques, comme les zones à faible densité telles que les zones rurales, où les revenus ne peuvent peut-être pas couvrir les coûts d'installation de nouvelles gares avec tous les coûts associés ; ou pour des raisons géologiques qui peuvent rendre la construction de gares difficile ou impraticable, comme un sol ou un substrat instable, une activité sismique, un terrain rocheux ou des nappes phréatiques souterraines. Cela nécessite de trouver une solution innovante pour servir la population de ce type de zones. Une solution efficace à ce problème est l'utilisation de systèmes de transport à la demande, particulièrement adaptés à ce type de zones.

La SNCF a proposé un système de transport à la demande, combinant un ensemble de navettes partagées situées dans les gares. La tâche de ces navettes est de transporter les voyageurs depuis et vers les gares, mais contrairement aux navettes partagées traditionnelles comme les bus qui ont des arrêts fixes, les itinéraires de ces navettes dépendent des voyageurs choisis à transporter. Les passagers qui descendent d'un train dans une gare doivent réserver à l'avance une place dans une navette située dans cette gare. Chacun de ces passagers a un endroit différent où il doit être déposé. Les navettes commencent alors leur itinéraire depuis une gare et transportent un groupe de passagers chacun vers sa propre destination. Une fois le dernier passager descendu à son lieu de dépose, la navette doit commencer à récupérer (s'il y en a) un autre groupe de passagers qui se trouvent autour d'une gare. Ces passagers doivent être récupérés chacun depuis son propre emplacement puis transportés avec les autres passagers jusqu'à la gare qu'ils souhaitent atteindre.

Comme mentionné, les navettes ne commencent à collecter les passagers à leur emplacement que lorsqu'elles sont complètement vides. Cette restriction de collecte et de livraison est connue sous le nom de VRP avec *Backhauls* (VRPB), introduit par Deif et Bodin (1984) [3]. Dans ce problème, l'ensemble des clients est divisé en clients de transport de ligne (*Line-haul*) et en clients de transport de retour (*Backhaul*). Chaque client de ligne nécessite une livraison/dépôt et chaque client de retour nécessite une collecte/ramassage. Les clients de ligne sont visités en premier, suivis par les clients de retour. Toutes les livraisons doivent être chargées aux dépôts et toutes les collectes doivent être transportées aux dépôts. Ce type de partitionnement est très fréquent dans les problèmes de distribution. L'intérêt de ne pas mélanger les ramassages et les livraisons sur les itinéraires des véhicules est généralement d'éviter de manipuler à nouveau les marchandises le long de l'itinéraire, car les véhicules sont

souvent chargés à l'arrière. Un exemple pratique se présente dans le secteur de l'épicerie où les supermarchés et les magasins sont les clients de ligne et les fournisseurs de l'épicerie sont les clients de retour. Un autre exemple peut être trouvé dans la distribution de boissons, où la livraison de bouteilles pleines est suivie de la collecte des bouteilles vides à la fin de l'itinéraire.

Dans notre cas, comme les éléments à transporter sont des personnes, l'application de la stratégie *Backhauls* permet d'offrir un meilleur service à la clientèle. En effet, mélanger ramassages et livraisons sur le même itinéraire augmenterait le temps de trajet des passagers. En d'autres termes, un passager qui a été récupéré d'une gare (un client de ligne) aimerait arriver à sa destination le plus rapidement possible, sans qu'on permette aux collectes (les clients de retour) d'être servis avant la livraison de ce passager. Le temps passé dans la navette par ce dernier augmenterait nécessairement, ce qui pourrait entraîner l'insatisfaction des clients.

En parlant des systèmes de transport à la demande, et du service à la clientèle lié au temps passé par les passagers dans les navettes, il existe une famille de problèmes de collecte et livraison qui combinent ces aspects, appelée *Dial-a-Ride Problems* (DARPs). Dans ce type de problèmes de collecte et livraison, chaque passager est caractérisé par : un point de collecte/ramassage et un autre point de livraison/dépôt, choisis et fixés par le passager lui-même ; ainsi qu'un temps de trajet maximum dans le véhicule à ne pas dépasser. Par conséquent, le système de transport proposé devrait servir les clients dans le cadre des restrictions mentionnées, y compris le principe des *Backhauls* et les aspects des problèmes DARP, tout en respectant les ressources disponibles comme le nombre de véhicules et leur capacité.

La SNCF a expérimenté ce service de transport sous le nom de «Ma Course SNCF» sur 5 communes de la Sarthe, de février 2021 à juillet 2022. Le service a su trouver sa place dans les habitudes de déplacements en répondant aux besoins de mobilité des habitants, pour lesquels 700 comptes clients ont été créés, 350 courses ont été réalisées par mois, avec 98% de clients satisfaits [4]. Le projet semble donc pratique et prometteur s'il est étendu à d'autres régions. Sur la Figure 1.1, on peut voir le type de véhicules utilisés par le service «Ma course SNCF» fourni par la SNCF [1].

1. Source de l'image :`https://ridewithvia.com/resources/ma-course-sncf`

FIGURE 1.1 Véhicule utilisé par le service «Ma course SNCF»

Dans ce mémoire, nous visons à optimiser ce système de transport en cherchant les itinéraires optimaux, qui respectent les contraintes mentionnées telles que le temps de trajet maximum des passagers, les heures de départ des trains et les capacités des véhicules. L'objectif principal est la minimisation des coûts totaux de déplacement et les coûts des passagers non-servis. D'autres objectifs peuvent également être envisagés, tels que la minimisation des temps d'attente des passagers avant le début du service, et la minimisation des temps d'arrivée des véhicules aux gares.

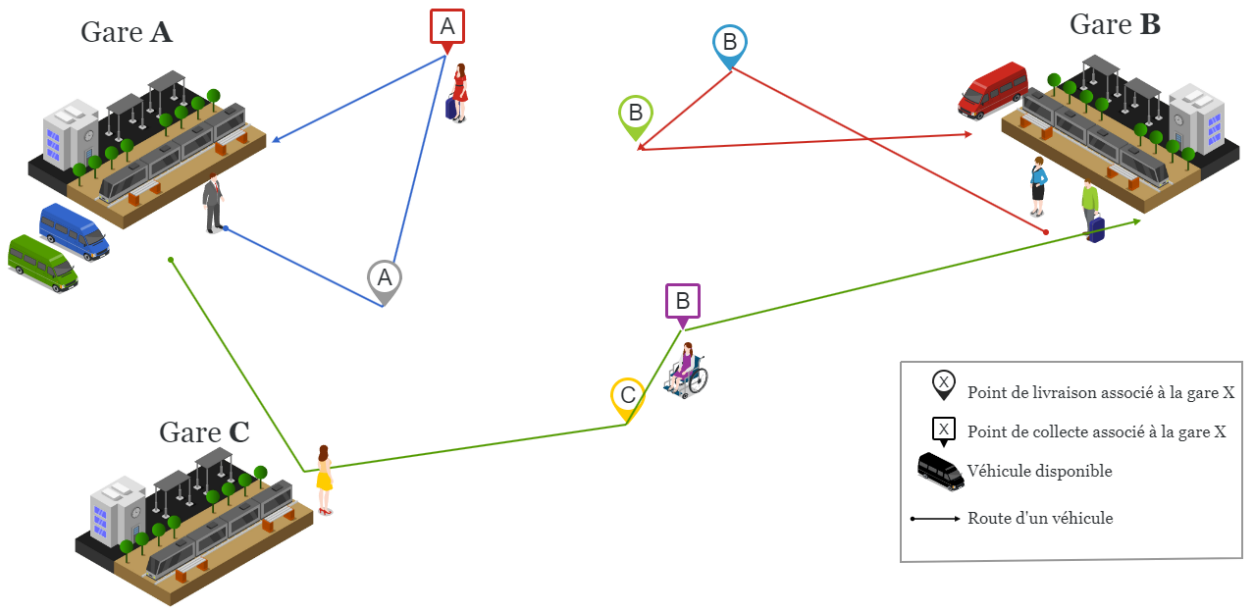


FIGURE 1.2 Exemple simplifié d'un réseau de 3 gares, 3 véhicules, 6 passagers, et une solution réalisable

La Figure 1.2 illustre un exemple simple du problème, contenant un réseau de : 3 gares A, B et C ; 3 véhicules Rouge, Vert et Bleu ; et 6 passagers distingués par leurs couleurs de vêtements, dont 4, le Gris , la Bleue, le Vert et la Jaune, sont des clients de ligne (demandes de livraison), où ces points de livraison sont représentés par les épingles rondes, alors que les deux autres passagers, la Rouge et la Violette, sont des clients de retour (demandes de ramassage), chacun ayant un point de collecte représenté par une épingle carrée. Une solution de trois itinéraires est présentée, chaque itinéraire correspondant à un véhicule de la même couleur. Initialement, les véhicules Bleu et Vert se trouvent à la gare A, et le véhicule Rouge se trouve à la gare B. Ce qui signifie qu'initialement, aucun véhicule ne se trouve à la gare C.

Le véhicule Bleu, qui se trouve initialement à la gare A, commence son trajet en livrant le passager Gris, puis il ramasse la passagère Rouge, et enfin retourne à la gare A. Le véhicule Bleu, initialement situé à la gare B, n'a que des demandes de livraison dans son itinéraire, il commence son itinéraire en livrant la passagère Bleue, puis le passager Vert, et retourne finalement vide à la gare B. D'autre part, comme la gare C n'a pas de véhicule disponible, donc un véhicule d'une autre gare doit desservir les passagers associés à la gare C. Dans ce cas, il s'agit du véhicule Vert, qui commence son itinéraire à la gare A, puis se rend à la gare C pour desservir la passagère Jaune, ce qui consiste en une requête de livraison, puis il passe à l'emplacement de la passagère Violette pour la prendre (demande de collecte) et

la transporter jusqu'à sa destination, qui est la gare B, et qui sera l'emplacement final du véhicule Vert.

Comme on peut le voir, même si le véhicule Vert était initialement situé à la gare A, il a transporté des passagers d'autres gares, ce qui est autorisé dans notre problématique, à condition que, sur le même trajet du véhicule, tous les clients de ligne (demandes de livraison) viennent de la même gare, et que tous les clients de la ligne de retour (demandes de collecte) aillent à la même gare.

À noter que, dans notre cas, nous ne considérons pas un problème de routage de véhicules à trajets multiples (*Multi-trip Vehicle Routing*), où un même véhicule peut effectuer plusieurs trajets commençant et se terminant à une gare. Ici, nous cherchons une solution où chaque véhicule effectue au plus un trajet (une route).

1.2 Objectifs de recherche

L'objectif de ce mémoire est de déterminer si la méthode de génération de colonnes peut être utilisée efficacement pour résoudre le problème qui consiste en une combinaison des problèmes DARP et VRPB, et de fournir une formulation appropriée du problème en utilisant la programmation linéaire en nombres entiers. Pour cela, nous utilisons la formulation basée sur la décomposition de Dantzig-Wolfe et résolvons le sous-problème associé de manière exacte en utilisant la programmation dynamique, en particulier un algorithme d'extension d'étiquettes monodirectionnel (*Monodirectional forward labeling algorithm*) adapté à sous-problème. Ce dernier est une variante du problème de plus court chemin avec contraintes de ressources comme nous l'explorerons dans les prochains chapitres.

1.3 Plan du mémoire

Le chapitre 2 présente tout d'abord une revue de la littérature des travaux réalisés sur les différents types de problèmes de collecte et de livraison. Il se concentre particulièrement sur les articles traitant les problèmes DARP et VRPB. Nous présentons également un aperçu des méthodes de résolution incluant la génération de colonnes.

Le chapitre 3 place ensuite la variante du problème de collecte et livraison traitée dans ce mémoire dans un cadre de programmation linéaire en nombres entiers. Nous décrivons aussi une description complète du problème.

Nous examinons ensuite en détail la méthodologie utilisée pour résoudre notre problème

dans le Chapitre 4. Nous présentons le cadre de l'algorithme général incluant la génération de colonnes et le fonctionnement de l'heuristique de plongée, ainsi que la modélisation du sous-problème comme un problème de plus court chemin avec des contraintes de ressources, et comment le résoudre avec des algorithmes dynamiques d'étiquetage.

Le Chapitre 5 présente des résultats numériques obtenus suite à plusieurs expériences réalisées sur un ensemble d'instances de notre problème. Nous évaluons ensuite la performance de notre algorithme de résolution et la qualité de la formulation mathématique proposée. Nous comparons également la qualité des solutions obtenues à l'aide d'un ensemble d'indicateurs proposés.

Nous terminons ensuite avec le Chapitre 6, où nous résumons les contributions de ce mémoire, et nous mentionnons certaines limites de notre travail, ainsi que des perspectives et pistes de travaux futurs.

CHAPITRE 2 Revue de littérature

Dans ce chapitre, nous décrivons la grande famille des problèmes de collecte et de livraison, incluant les contraintes les plus fréquentes. Nous décrivons également les recherches qui ont déjà été menées sur des problèmes partageant des aspects communs avec le nôtre, tels que le DRPR et le VRPB. De plus, nous présentons les principales méthodes de résolution utilisées pour ce type de problème, incluant des méthodes exactes et approchées, et en particulier, la génération de colonnes.

2.1 Problèmes de collecte et livraison

La famille des problèmes de collecte et livraison («*Pickup and Delivery Problems*» - PDP) a été définie pour la première fois en 1995 par Savelsbergh et al. [5]. En 2007, Berbeglia et al. [6] ont présenté un cadre général permettant de modéliser un grand nombre de problèmes de collecte et livraison. Ils classent ces problèmes en trois grandes catégories : les problèmes *Many-to-Many*, où un seul type d'objet doit être collecté à certains endroits et livré à d'autres ; les problèmes *One-to-Many-to-One*, où un type d'objet est collecté à un dépôt et un autre type doit être livré au même dépôt ou à un dépôt différent ; et les problèmes *One-to-One*, où chaque objet est unique et a des lieux de collecte et de livraison spécifiques. Ils notent que les méthodes pour résoudre ces problèmes varient largement, incluant des approches exactes ainsi que des heuristiques.

Contrairement au VRP, dans cette famille de problèmes, il ne s'agit plus simplement de visiter les différents sommets associés aux clients avec les véhicules, mais il faut également prendre en compte la nature des demandes. Ces demandes concernent un ensemble de produits de types différents. Les demandes peuvent être de deux natures différentes : soit une demande de collecte (*Pickup*), soit une demande de livraison (*Delivery*). La solution optimale d'un problème de tournée de véhicules avec collecte et livraison VRPPD se compose d'un ensemble de tournées, telles que :

— toutes les demandes sont satisfaites ;

— la charge maximale des véhicules est respectée en tout point de la tournée ;

— la somme des coûts de tournées est minimale.

Dans la littérature, les problèmes de tournées de véhicules avec collecte et livraison sont généralement modélisés sur un graphe orienté $G = (V, A)$, avec $V = \{1, \ldots, n\}$ l'ensemble des sommets qui modélisent les demandes. À cet ensemble de sommets V est ajouté un sommet 0

qui correspond au dépôt où est positionnée la flotte de véhicules. Les éléments de l'ensemble $A = \{(i,j) \mid i,j \in V\}$ sont considérés comme des arcs si le problème est asymétrique, ou comme des arêtes s'il est symétrique. À un sommet i est associé un ensemble de valeurs q_i^j correspondant à la demande en produit x_j sur le sommet i. Une demande de collecte est associée sur le graphe à une quantité positive, tandis qu'une demande de livraison est associée à une quantité négative. À un arc $(i,j) \in A$ est associé une valeur $c_{i,j}$ qui correspond au coût pour aller du sommet i au sommet j. La Figure 2.1 illustre une solution composée de deux tournées. Chaque véhicule est affecté à une tournée, c-à-d., une suite ordonnée de clients à desservir. Elle démarre au dépôt, visite une liste ordonnée de clients, avant de revenir au dépôt.

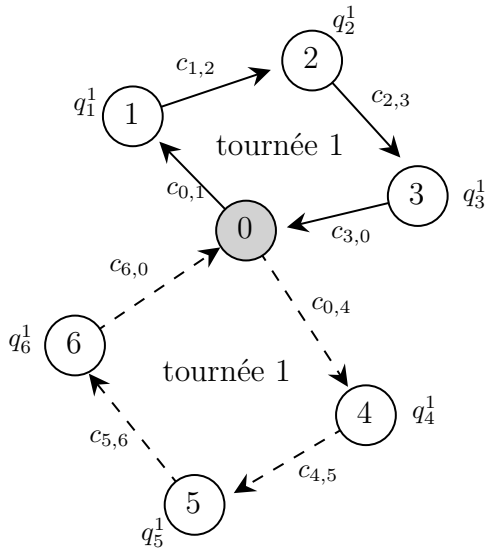
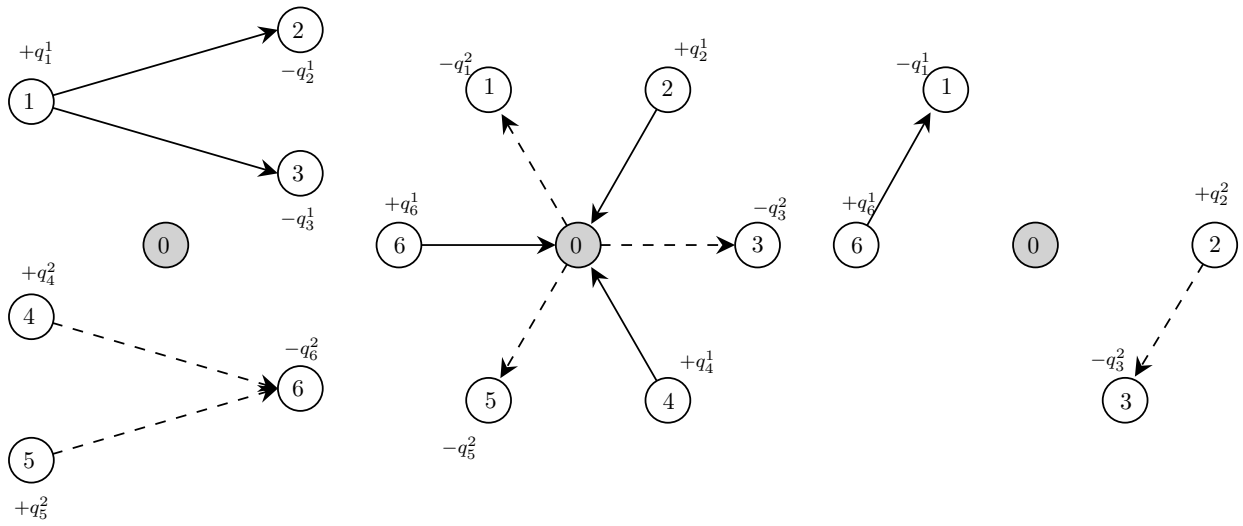


FIGURE 2.1 Exemple de modélisation de VRP à 6 sommets et deux véhicules

Cette famille de problèmes est classée selon deux critères principaux : la structure des demandes et le type de visites à réaliser.

a) **La structure des demandes**

La structure concerne la disposition des points de collecte et de livraison. Elle est généralement divisée en trois types, comme illustré dans la figure 2.2. Dans cette figure, les liens de transport à réaliser entre les points de collecte et de livraison des marchandises sont représentés par des arcs. Chaque type d'arc correspond à un type de marchandise x_i. Les arcs pleins correspondent aux produits x_1 et les arcs en pointillés correspondent aux produits x_2.



(a) Structure *Many-to-Many* (b) Structure *One-to-Many-to-One* (c) Structure *One-to-One*

FIGURE 2.2 Divers structures de demandes

— La première structure, appelée *Many-to-Many* (M-M), s'applique aux problèmes où plusieurs points de collecte et de livraison existent pour chaque type d'objet (Figure 2.2(a)).

— La deuxième structure, appelée *One-to-Many-to-One* (1-M-1), est une structure centralisée. Un sommet spécifique, généralement le dépôt contient tous les produits nécessaires pour effectuer les livraisons aux sommets, tandis qu'un autre sommet spécifique, souvent le même sommet, stocke tous les produits que les véhicules collectent (Figure 2.2(b)).

— La troisième structure, nommée *One-to-One* (1-1), permet de modéliser des problèmes où, à chaque demande de collecte, correspond une seule demande de livraison située sur un autre sommet du graphe (Figure 2.2(c)).

Selon cette classification, notre problème tombe dans la catégorie *One-to-Many-to-One* (1-M-1), puisque tous les passagers à livrer se trouvent dans les dépôts, et tous les passagers à récupérer finissent dans les dépôts.

b) **Types de visites**

Le deuxième critère selon Berbeglia et al. [6], est la nature de la visite effectuée par le véhicule sur les sommets du graphe. Dans les problèmes de type PDP, un sommet peut avoir différentes natures. Un sommet ne peut être associé qu'à des demandes de collecte (resp. de livraison), dans ce cas les sommets sont appelés sommets de collecte (resp. sommets de livraison). Un sommet peut également avoir des demandes de collecte et de livraison. Selon ce fait, les problèmes PDP sont séparés en trois catégories.

— Le problème est noté $\boldsymbol{P/D}$ si chaque sommet est un sommet de collecte ou de livraison ;

— sinon si au moins un sommet a simultanément une demande de livraison et une autre demande de collecte, alors le problème est noté $\boldsymbol{PD}$;

— enfin, il est noté $\boldsymbol{P-D}$ si au moins un sommet nécessite d'effectuer les deux opérations, mais que le véhicule peut revenir plus tard pour effectuer la deuxième opération.

Ainsi, notre problème fait partie des problèmes P/D.

2.1.1 Classification de problèmes de collecte et livraison

La Figure 2.3 inspirée de l'article de synthèse de Berbeglia et al. [6] et de la thèse de doctorat de Chassaing [7], illustre les liens entre quelques problèmes de la famille des problèmes de collecte et livraison PDP les plus étudiés dans la littérature. Dans cette figure, un problème B est un descendant d'un autre problème A si et seulement si A est une généralisation de B, ou, en d'autres termes, B est un cas particulier de A.

Les trois branches principales sont représentées avec les problèmes : *One-to-One* (1-1), *Many-to-Many* (M-M), et *One-to-Many-to-One* (1-M-1). Les arcs entre les problèmes représentent les contraintes à ajouter ou à supprimer pour passer de l'un à l'autre. Les contraintes situées sur les arcs sont celles présentées ci-dessus.

Hormis «*Backhauls*» et «*Mixtes*» qui appartiennent à la branche des *One-to-Many-to-One*, ces contraintes correspondent au cas où le problème est de type P/D, introduit initialement par Goetschalckx et Jacobs-Blecha en 1989 [8]. Les sommets de livraison et de collecte sont donc séparés. Si le problème impose aux solutions de traiter des sommets de livraison avant les sommets de collecte, alors le problème doit respecter la contraintes «*Backhauls*», sinon il est appelé «*Mixtes*».

Le problème du transport à la demande («*Dial-A-Ride Problem*» - DARP) fait partie de la famille des problèmes *One-to-One*.

2.1.2 Contraintes supplémentaires

Des contraintes supplémentaires sont parfois ajoutées aux variantes PDP pour avoir une modélisation plus proche de la réalité. Les contraintes les plus répandues dans la littérature sont :

➤ fenêtres de temps ;

- flotte de véhicules hétérogènes ;
- multi-dépôts ;
- centres de transbordement permettent l'échange de produits entre véhicules ;
- temps de trajet maximal pour les véhicules ;
- temps de trajet maximal pour les objets ;
- visites multiples et traitement partiel des demandes.

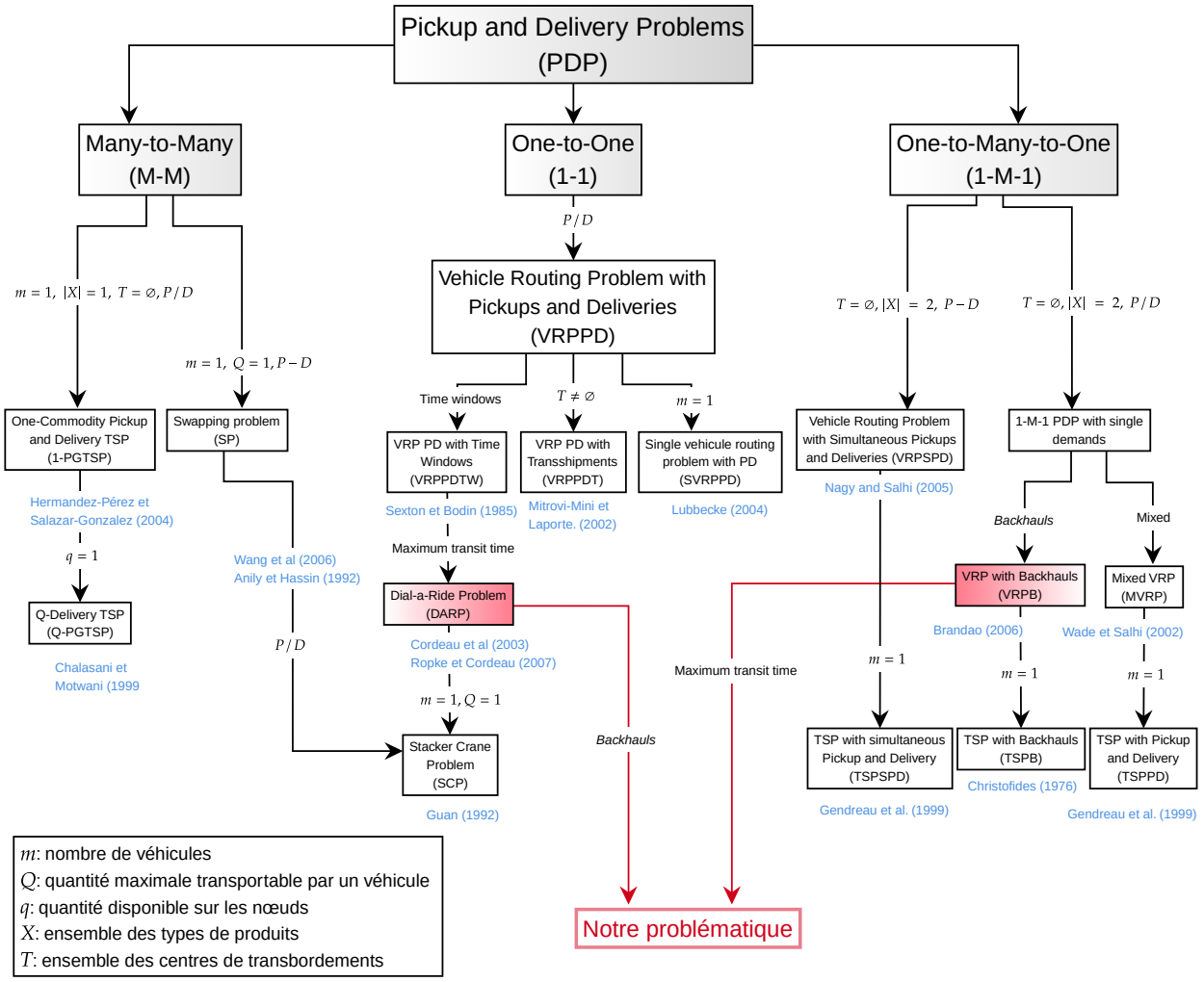


FIGURE 2.3 Classification des problèmes de collecte et livraison

Fenêtres de temps

Une fenêtre de temps (*time window*) exige que certaines requêtes soient gérées en respectant un intervalle spécifique pour le début de traitement, que ce soit pour une livraison ou une

collecte. Cet intervalle est défini par une date de début au plus tôt et une date de fin au plus tard. Cette contrainte peut s'appliquer aux demandes de collecte comme de livraison [9].

Flotte hétérogène

Les problèmes avec flottes hétérogènes sont ceux dans lesquels les véhicules ne sont pas identiques. Les différences résident le plus souvent dans la charge maximale qu'ils peuvent supporter [10]. Mais d'autres contraintes peuvent également être envisagées, comme les frais de déplacement et la vitesse de chaque véhicule.

Multi-dépôts

Dans ce genre de problèmes, le graphe associé comprend plusieurs dépôts d'où les véhicules peuvent partir. Le plus souvent, les véhicules doivent retourner à leur dépôt d'origine à la fin de leur itinéraire. Le nombre de véhicules peut être différent dans chaque dépôt.

Temps de trajet maximal pour les véhicules

Cette contrainte impose que les tournées d'une solution respectent une durée maximale qui est la différence entre la date de sortie du dépôt et la date de retour au dépôt. Cette contrainte permet de modéliser la durée de travail du chauffeur [11].

Temps de trajet maximal pour les objets

La flotte de véhicules peut transporter une variété de produits, incluant des articles périssables qui doivent être livrés dans des délais maximaux spécifiques, comme c'est le cas pour les produits frais. Une fois le produit collecté, le véhicule doit respecter un délai maximal pour parvenir au point de livraison. Cette limite de temps est connue dans la littérature sous le nom de «*maximum transit time*» ou encore «*maximum ride time*». Cette contrainte est courante également dans le transport de passagers et aide à évaluer la qualité de service du point de vue clientèle. Elle est notamment pertinente pour le problème du transport à la demande (DARP), décrit en détail dans l'étude de Cordeau et Laporte [12, 13].

2.1.3 Problème de transport à la demande

Le DARP a été formulé pour la première fois par Stein [14]. Il s'agit d'un cas spécifique de problème de collecte et livraison, avec la présence de fenêtres de temps à prendre en compte

et un temps de transit maximal. Il s'agit d'un problème de type *One-to-One*, où chaque point de ramassage a un unique point de livraison lui correspondant [12].

Le DARP a été largement étudié dans la littérature sur les problèmes de tournées de véhicules. Plusieurs études, notamment celles de Cordeau et Laporte [15], Molenbruch et al. [16], et plus récemment Ho et al. [17], se concentrent sur la classification des études existantes, en abordant les défis associés, et en soulignant les directions de recherche futures. Les travaux de Cordeau et Laporte [15] explorent principalement les définitions des problèmes et font la distinction entre les scénarios du DARP à un seul véhicule et à plusieurs véhicules. Ho et al. [17] classent les différentes variantes de DARP en fonction de leurs caractéristiques, et font une distinction entre le DARP statique et dynamique, ainsi qu'entre formes déterministes et stochastiques. Ces études fournissent également une vue d'ensemble des approches de solutions développées.

La première formulation du DARP utilisant la programmation linéaire en nombres entiers a été introduite par Cordeau en 2006 [18], où l'auteur a également proposé un algorithme *Branch-and-Cut* pour résoudre le problème.

La résolution du DARP à l'aide de méthodes exactes a ses limites lorsqu'il s'agit de grandes instances ; la méthode *Branch-and-Cut* proposée par Cordeau [18] traite des instances comportant jusqu'à 36 clients, tandis que la méthode de Ropke et al. [19] résout des instances comportant jusqu'à 96 clients grâce à l'ajout d'inégalités valides au sein de l'arbre de recherche.

Riedler et Raidl [20] ont présenté deux méthodes de résolution exactes basées sur une méthode appelée le *Branch-and-Check* et la décomposition de Benders basée sur la logique (*Logic Based Benders Decomposition*) où l'objectif est de maximiser le nombre de clients servis pour des instances ayant jusqu'à 50 passagers. Gaul et al. [21] proposent des modèles mathématiques linéaires en nombres entiers basés sur les évènements *(Event-based MILP models*) pour exploiter les contraintes de capacité et de fenêtre de temps afin de réduire la taille du graphe considéré. Cette approche permet de résoudre des instances comprenant jusqu'à 8 véhicules et 96 clients en moins de 461 secondes.

2.1.4 Problème de collecte et livraison avec *Backhauls*

Le VRPB est une variante du problème de collecte et livraison dans lequel tous les clients de livraison (*Linehauls*) doivent être satisfaits avant que le véhicule n'effectue la collecte (*Backhauls*) à partir de n'importe quel point de ramassage sur son itinéraire. Ce problème est particulièrement pertinent dans le cadre des efforts constants de réduction des coûts de distribution par l'utilisation de la capacité inutilisée d'un véhicule vide retournant au dépôt.

Santos et al. en 2020 [22] ont présenté une revue de littérature sur le VRPB, dans laquelle ils analysent la littérature du VRPB dans une perspective de durabilité, qui couvre les objectifs environnementaux et sociaux, les réseaux collaboratifs et la logistique inverse. Les premières études sur les VRPB ont été réalisées par Golden et al. en 1985 [23], et par Goetschalckx et Jacobs-Blecha en 1989 [8]. Récemment, en 2023, une étude autour de l'application du VRPB sur les satellites à deux échelons avec capacité a été présentée par Dumez et al. dans [24]. Aussi, en 2023, Sukhpal et Kumar [25] ont proposé une formulation de programmation linéaire en nombres entiers mixtes pour le VRPB multi-voyages multi-compartiments (*Multi-trip multi-compartment*) qui consiste à trouver les itinéraires les plus efficaces pour une flotte hétérogène de véhicules multi-compartiments afin de transporter simultanément des marchandises hétérogènes. Plus récemment, en 2024, une heuristique à base neuronale utilisant l'apprentissage par renforcement profond (*Deep Reinforcement Learning*) a été proposée par Wang et al. [26] pour résoudre le VRPB.

2.2 Méthodes de résolution

Dans cette section, nous nous concentrons sur les méthodes de résolution qui ont été appliquées aux problèmes DARP et VRPB.

De nombreuses méthodes de résolution ont été utilisées pour résoudre les problèmes DARP et VRPB ainsi que leurs variantes. Dans ce qui suit, nous donnons un aperçu des méthodes de résolution classiques, c-à-d. les méthodes exactes et heuristiques. Nous renvoyons à nouveau le lecteur aux articles de synthèse de Ho et al. [17] et Santos et al. [22] pour une analyse plus approfondie des méthodes de résolution classiques utilisées pour résoudre le DARP et le VRPB respectivement.

2.2.1 Méthodes exactes

Le principal avantage des méthodes exactes est leur capacité à trouver la solution optimale à un problème et à garantir son optimalité. Toutefois, les méthodes exactes sont coûteuses en termes de temps de calcul. Par conséquent, elles ne sont pas pratiques pour résoudre de grandes instances de problèmes combinatoires NP-difficiles tels que les DARP et les VRPB. À notre connaissance, la plus grande instance DARP résolue à l'optimalité compte 849 clients et plus de 70 véhicules disponibles, et la solution optimale a été trouvée en moins de 8,1 heures par Karimi et al. [27].

Les méthodes exactes ont principalement été utilisées pour les versions statiques et déterministes des DARP et des VRPB. Les versions dynamiques nécessitent des décisions rapides

qui sont incompatibles avec les méthodes exactes. À l'inverse, les versions statiques peuvent prévoir un temps de calcul suffisant pour atteindre l'optimalité. Par exemple, les méthodes exactes peuvent être utilisées lorsque les décisions sont prises une fois par jour et que toutes les demandes sont déjà connues la veille. Les trois méthodes exactes les plus utilisées pour résoudre les problèmes DARP et le VRPB sont : *Branch-and-Cut*, *Branch-and-Price* et *Branch-and-Price-and-Cut*. Les trois suivent le principe de l'algorithme *Branch-and-Bound*.

- *Branch-and-Cut* : L'algorithme *Branch-and-Cut* utilise l'algorithme *Branch-and-Bound* en combinaison avec la méthode de plans de coupants (*Cutting plane method*) afin de resserrer la relaxation linéaire du problème de programmation en nombres entiers. Cette combinaison permet d'éliminer les solutions fractionnaires non valides tout en explorant efficacement l'arbre généré par l'algorithme *Branch-and-Bound*. Cela permet une convergence plus rapide vers des solutions entières et donne de meilleures bornes pour vérifier l'optimalité. La méthode *Branch-and-Cut* a été appliquée pour la première fois au DARP par Cordeau en 2006 [18]. L'auteur a adapté les inégalités valides du TSP et du VRP à un modèle de programmation linéaire en nombres entiers mixtes à 3 indices modélisant le DARP.

  Afin de traiter les variantes du DARP, d'autres auteurs ont développé des algorithmes *Branch-and-Cut* avec de nouvelles inégalités valides liées à leurs problèmes particuliers tels que les véhicules et utilisateurs hétérogènes [28], et les problèmes avec multi-dépôts [29]. Récemment, en 2024 [30], plusieurs classes de nouvelles inégalités valides ont été introduites pour améliorer la recherche en se concentrant sur les clients incompatibles, qui est la principale caractéristique de la variante du DARP étudiée dans cette dernière recherche.
- *Branch-and-Price* : La méthode *Branch-and-Price* est une combinaison de l'algorithme *Branch-and-Bound* avec un processus de génération de colonnes *(Column Generation)*. L'avantage de cette méthode exacte est qu'elle permet de traiter des problèmes de grande taille en décomposant le problème en un problème maître et un sous-problème. Plus de détails sur la génération de colonnes seront présentés dans la section 2.2.3.

  La première application de la méthode *Branch-and-Price* sur une variante de DARP a été publiée en 2011 par Garaix et al. [31], où le sous-problème est résolu à l'aide d'une méthode exacte.
- *Branch-Price-and-Cut :* L'approche *Branch-Price-and-Cut* est une autre variante de l'approche *Branch-and-Bound* qui combine les principes des deux méthodes *Branch-and-Price* et *Branch-and-Cut*. Cette méthode a été appliquée pour la première fois pour le DARP en 2015 par Gschwind et Irnich dans [32].

D'autre part, deux algorithmes *Branch-Cut-and-Price* pour le VRPB ont été proposés dans [33]. Le premier suit l'approche traditionnelle avec un seul sous-problème, tandis que le second exploite la répartition des clients entre *Linehauls* et *Backhauls* et définit deux types de sous-problèmes.

2.2.2 Méthodes heuristiques et métaheuristiques

Le DARP et le VRPB sont des problèmes NP-difficiles. Cela signifie qu'il n'existe aucun algorithme à complexité polynomiale pour les résoudre de façon optimale jusqu'à présent. Par conséquent, l'utilisation d'heuristiques est naturellement attrayante.

Une grande variété d'heuristiques a été utilisée pour résoudre les problèmes NP-difficiles. Suivant la taxonomie utilisée par [34], certaines heuristiques sont basées sur la trajectoire et d'autres sur la population. Une heuristique basée sur la trajectoire traite une solution à la fois. Elle part d'une solution initiale qui sera modifiée à chaque itération. Chaque fois qu'une nouvelle meilleure solution est atteinte, cette nouvelle meilleure solution est enregistrée. Une heuristique basée sur la population traite plusieurs solutions simultanément. À chaque itération, la population actuelle de solutions (génération actuelle) est utilisée pour générer une nouvelle population de solutions (génération suivante). Le principe est de donner naissance à la meilleure solution à partir de la sélection de bonnes caractéristiques à travers les générations.

- **Recuit simulé** : Le recuit simulé a été introduit par Metropolis et al. [35] et appliqué pour la première fois aux problèmes d'optimisation en 1983 par Kirkpatrick et al. [36]. Il est basé sur le processus de recuit. Le recuit est un traitement thermique utilisé en métallurgie, au cours duquel un matériau est chauffé à une température donnée, puis refroidi à une vitesse lente et contrôlée. Appliqué aux problèmes d'optimisation, le recuit simulé fonctionne avec une solution unique. À chaque itération, une nouvelle solution est générée. Si elle est meilleure, elle est acceptée. Si elle n'est pas meilleure, la nouvelle solution est acceptée avec une probabilité qui dépend de la température et de la qualité de la nouvelle solution. Les températures élevées donnent des probabilités d'acceptation élevées et les températures basses des probabilités d'acceptation faibles. Au début de la recherche, la température est élevée pour stimuler l'exploration. Ensuite, la température diminue progressivement pour stimuler l'exploitation.

  Le recuit simulé a été appliqué sur le DARP dans de nombreux travaux, tels que [37], [38] et [29]. Il a également été mis en œuvre sur le VRPB, tels que [39] et [40].
- **Recherche avec tabous** : La recherche avec tabous, introduite par Glover et McMillan

en 1986 [41], a été appliquée pour la première fois au DARP par Cordeau et Laporte en 2003 [13], et au VRPB par Duhamel et al. en 1997 [42]. Cette métaheuristique basée sur la trajectoire empêche d'explorer certaines solutions en les plaçant temporairement dans une liste taboue. L'ensemble du voisinage de la solution courante est généré et la meilleure solution non taboue est acceptée. Ainsi, la liste taboue évite d'être piégée dans des cycles.

- **Recherche par grand voisinage** : La recherche par grand voisinage (*«Large Neighborhood Search»* - LNS) a été introduite pour la première fois en 1998 combinée avec une méthode exacte par Shaw [43], puis couplée à une heuristique sous le nom de ruiner-et-recréer (*ruin and recreate*) par Schrimpf et al. [44]. Son principe consiste à détériorer partiellement la solution courante avant de la reconstruire de manière améliorée. A chaque itération de l'algorithme, l'opérateur de destruction dégrade la solution, puis, l'opérateur de réparation reconstruit une nouvelle solution complète à partir de la solution partielle. La LNS possède une variante connue appelée la recherche par grand voisinage adaptative (*«Adaptive Large Neighborhood Search»* - ALNS) introduite en 2006 par Ropke et Pisinger [45]. La principale différence est que l'ALNS sélectionne les opérateurs en fonction de leurs probabilités pondérées. La pondération de chaque opérateur est ajustée de manière dynamique en fonction de ses performances passées.

  Le LNS et ses variantes ont été largement étudiés et ont souvent montré de très bons résultats sur les problèmes de tournées de véhicules, à savoir le DARP [46], et [47] ; et le VRPB [48].

- **Algorithme génétique** : Les algorithmes génétiques ont été introduits pour la première fois par Holland en 1975 [49]. Cet algorithme a été appliqué au DARP et au VRPB dans divers articles, tels que [50] et [51] respectivement. Il s'agit d'une métaheuristique à base populationnelle, inspirée de la théorie de l'évolution darwinienne. Plusieurs parents sont sélectionnés dans une population initiale. Ensuite, les enfants sont générés à partir des parents à l'aide d'opérateurs de croisement et de mutation. Enfin, certains individus (des parents et des enfants) sont sélectionnés pour former la génération suivante. Parmi les premières études sur l'application des algorithmes génétiques au DARP et au VRPB, on peut citer les travaux de Cubillos et al. [52], et ceux de Ganesh et al. [53], respectivement.

2.2.3 Génération de colonnes

La génération de colonnes offre plusieurs avantages par rapport aux autres algorithmes d'optimisation, ce qui en fait une technique puissante pour résoudre des problèmes à grande échelle,

comme ceux de collecte et livraison. Elle permet de diviser le problème en sous-problèmes plus petits et plus faciles à gérer, qui peuvent être résolus indépendamment, puis combinés pour trouver la solution optimale. Il est ainsi possible de résoudre des problèmes à grande échelle qui seraient autrement insolubles avec les techniques d'optimisation traditionnelles. En outre, la génération de colonnes peut être utilisée pour générer dynamiquement de nouvelles variables au fur et à mesure que le problème évolue dans le temps. Enfin, l'approche de génération de colonnes est hautement personnalisable, ce qui permet d'incorporer des contraintes ou des objectifs supplémentaires selon les besoins.

La génération de colonnes est une technique utilisée en optimisation mathématique pour résoudre des problèmes linéaires comportant un grand nombre de variables. L'idée de la génération de colonnes est de ne travailler qu'avec un sous-ensemble de variables suffisamment significatif, formant ce que l'on appelle le problème maître restreint («*Restricted Master Problem*» - RMP). Les variables supplémentaires ne sont incluses qu'en cas de besoin, comme dans la méthode du simplexe, où une variable prometteuse est trouvée à chaque itération pour entrer dans la base. Le processus d'itération dans la génération de colonnes implique l'optimisation du RMP pour déterminer la valeur optimale actuelle de la fonction objectif et les multiplicateurs duaux, puis la recherche d'une variable ayant un coût réduit négatif, ce qui implique l'optimisation d'un sous-problème. Si une variable améliorante est trouvée, elle est ajoutée au RMP et le processus se répète ; sinon, nous en avons fini avec la relaxation linéaire du problème principal. Afin d'obtenir des solutions entières au problème original, la génération de colonnes peut être intégrée dans un cadre de *Branch-and-Bound*, la méthode résultante étant communément appelée *Branch-and-Price* [54].

Le Branch-and-Price a été utilisé pour résoudre les problèmes DARP dans de nombreux travaux ; parmi les premiers nous pouvons citer [55], [56], et [57]. Dumas et al. [55] ont proposé une approche en deux phases pour résoudre le problème en effectuant d'abord un *mini-clustering* puis un routage optimal, une modification innovante de la méthode *« cluster first - route second »*, ce qui permet un regroupement (*clustering*) plus intelligent des clients. Il s'agit d'un algorithme d'insertion séquentielle heuristique qui regroupe les clients proches (*mini-clusters*) qui peuvent être efficacement servis par le même segment de route de véhicule. Un algorithme de génération de colonnes optimales construit ensuite des routes correspondant aux journées de travail des conducteurs en enchaînant ces segments de route de véhicule. Enfin, les segments de route sont rouverts afin de réoptimiser chaque route. Plus tard en 1991, Desrosiers et al. [56], ont utilisé une version améliorée de la même approche à deux phases *« cluster first - route second »* pour résoudre une autre variante du problème DARP. Dans cette approche, la génération de colonnes a été intégrée dans la deuxième phase, pour construire simultanément toutes les routes de véhicules.

D'autres travaux ont suivi, appliquant le Branch-and-Price pour résoudre le DARP et ses variantes, notamment [58], [47], et [59]. D'autre part, une première application de l'approche Branch-and-Price à une variante du VRPB a été réalisée en 1995 par Gélinas et al. [60], où ils ont utilisé des stratégies impliquant un branchement sur les variables de ressources (temps ou capacité).

CHAPITRE 3 Formulation d'un système de transport à la demande sur plusieurs gares

Dans ce chapitre, nous décrivons notre problème de manière formelle et détaillée. Nous commençons par présenter les acteurs principaux de notre problème, à savoir les gares et leurs trains associés, les navettes, ainsi que les deux types de passagers. Ensuite, nous procédons à la formulation mathématique du problème sous la forme d'un programme linéaire mixte en nombres entiers. Nous expliquons les notations utilisées, définissons les variables, présentons la fonction objectif et détaillons les contraintes.

3.1 Description du problème

Notre problème comporte trois acteurs principaux, chacun ayant ses propres caractéristiques.

- **Gares** : les gares ferroviaires sont les dépôts centraux, d'où et/ou vers lesquels les passagers souhaitent être transportés. Ce sont également les endroits où les véhicules doivent commencer et terminer leurs itinéraires. Chaque gare a son propre emplacement et un nombre de véhicules disponibles en début de journée, ce nombre pouvant être nul.

  De plus, chaque gare est associée à un seul train. Ce dernier est responsable, d'une part, de déposer des passagers à la gare (c'est-à-dire de générer les demandes de livraison associées à la gare) à son heure d'arrivée, supposée être 0 ; et d'autre part, de transporter des passagers (après avoir été pris en charge par les véhicules et amenés à la gare avant le départ du train) depuis la gare à son heure de départ.

  Lorsque plus de deux trains doivent être associés à une même gare, celle-ci peut être divisée en plusieurs gares — toutes situées au même endroit — et les trains de la gare d'origine sont répartis équitablement entre elles. Ainsi, chaque nouvelle gare est associée à un seul train, qui amène des passagers à la gare et en transporte également.
  Toutefois, comme le système de transport est destiné à des zones de faible densité, nous pouvons supposer que les trains sont assez espacés dans le temps. Cela signifie qu'il y a des intervalles assez longs entre les heures d'arrivée (resp. de départ) des trains, ce qui permet de considérer que chaque gare peut être associée à un seul train, assumant à la fois le rôle d'un train entrant (amenant des passagers) et d'un train sortant (emportant des passagers).
- **Navettes** : ce sont des véhicules utilisés pour transporter les passagers. Au départ, chaque véhicule est situé dans une gare et a une capacité limitée en charge de passagers

qu'il peut transporter en même temps.

- **Passagers** : il s'agit des passagers qui doivent être servis, chaque passager se trouvant soit dans une gare, soit à l'extérieur des gares. Le premier cas signifie que le passager doit être livré de la gare à un endroit situé autour de cette gare, tandis que le second signifie que le passager doit être récupéré à son emplacement et transporté vers une gare. Chaque passager est associé à une quantité de chargement et d'un temps de trajet maximum dans le véhicule qu'il doit ne pas dépasser. Certains passagers situés à l'extérieur des gares ne peuvent être servis avant une certaine heure que l'on appelle l'heure de disponibilité du passager.

  Noter que les passagers provenant des gares (c'est-à-dire les demandes de livraison) peuvent être amenés à attendre. Cette situation peut survenir lorsqu'un véhicule d'une gare se rend dans une autre pour satisfaire ses demandes de livraison. Par conséquent, ces passagers de type livraison devront attendre au moins le temps nécessaire pour que le véhicule se déplace de sa gare d'origine à la gare où les passagers attendent. Toutefois, aucun temps d'attente maximal n'est imposé à ces passagers.

Outre la nécessité de respecter les contraintes liées aux caractéristiques des éléments mentionnés (comme les heures de départ des trains, les capacités des véhicules, les durées maximales de trajet et les dates de début de service), certaines hypothèses, que nous énumérons ci-dessous, sont également imposées.

- Sur un même itinéraire, les requêtes de livraison doivent précéder celles de collecte (principe des *Backhauls*) ;
- les véhicules commencent et terminent leurs trajets dans une gare ;
- un véhicule peut desservir des passagers d'une autre gare que celle d'origine ;
- les requêtes de livraison (reps. collecte) d'un même itinéraire doivent provenir (resp. aller) à une seule et même gare, c-à-d. l'itinéraire ne peut pas contenir deux requêtes de livraison (resp. collecte) provenant de (resp. allant aux) deux gares différentes ;
- un véhicule peut effectuer une seule route au plus ;
- tous les itinéraires débutent à l'heure 0, et l'horizon est limité à [0, heure de départ du dernier train].

3.2 Réseau associé

Le problème étudié dans ce mémoire est modélisé par un graphe orienté $G = (\mathcal{V}, \mathcal{A})$, où $\mathcal{V}$ est l'ensemble des nœuds et $\mathcal{A}$ est l'ensemble des arcs.

On définit :

- $\mathcal{P} \subset \mathcal{V}$: l'ensemble des nœuds associés aux requêtes de collecte (*Pickup nodes*),
- $\mathcal{D} \subset \mathcal{V}$: l'ensemble des nœuds associés aux requêtes de livraison (*Delivery nodes*).

Ces ensembles sont subdivisés selon les gares :

$$\mathcal{P} = \bigcup_{i \in \mathcal{S}} \mathcal{P}_i \text{ et } \mathcal{D} = \bigcup_{i \in \mathcal{S}} \mathcal{D}_i,$$

où $\mathcal{S} = \{0, \ldots, r-1\}$ est l'ensemble des indices des gares, et chaque $\mathcal{P}_i$ (resp. $\mathcal{D}_i$) est l'ensemble des nœuds de collecte (resp. de livraison) associés à la gare i.

On introduit aussi :

- $\mathcal{O}^+ = \{o_0^+, \ldots, o_{r-1}^+\} \subset \mathcal{V}$: l'ensemble des nœuds de dépôts de départ,
- $\mathcal{O}^- = \{o_0^-, \ldots, o_{r-1}^-\} \subset \mathcal{V}$: l'ensemble nœuds de dépôts d'arrivée.

En pratique, chaque couple dépôt de départ / dépôt d'arrivée correspond généralement au même emplacement. Pour tout $i \in \mathcal{S}$, chaque dépôt de départ o_i^+ est associé à un dépôt d'arrivée o_i^-. Autrement dit, chaque gare est représentée par deux nœuds de dépôts, un nœud source (dépôt de départ) et un nœud puits (dépôt d'arrivée). Ainsi, l'ensemble des nœuds de G peut être écrit comme suit,

$$\mathcal{V} = \mathcal{O}^+ \cup \mathcal{D} \cup \mathcal{P} \cup \mathcal{O}^-.$$

Le graphe G n'est pas complet. Ainsi, une construction bien détaillée de $\mathcal{A}$ est donnée ci-dessous, ainsi qu'un exemple de G qui est donné dans la Figure 3.1 pour une instance de deux gares ayant chacune deux requêtes de collecte et deux requêtes de livraison. Chaque sous-ensemble défini lors de la construction de $\mathcal{A}$ a une couleur différente dans la Figure 3.1 ; celle-ci est mentionnée quand chaque sous-ensemble est défini.

- Les nœuds de $\mathcal{O}^+$ doivent tous être liés les uns aux autres pour former une clique, afin de permettre aux véhicules de servir les passagers (demandes de livraison) d'autres gares si nécessaire. Ainsi, cet ensemble d'arcs peut être exprimé par $\{(i,j) \mid i,j \in \mathcal{O}^+; i \neq j\} \subset \mathcal{A}$ et est coloré en vert.
- Chaque nœud de dépôt de départ o_i^+, $i \in \mathcal{S}$, doit être lié aux nœuds de livraison et de collecte de la gare correspondante i, d'où cet ensemble d'arcs est exprimé par $\{(o_i^+, j) \mid i \in \mathcal{S}, j \in \mathcal{P}_i \cup \mathcal{D}_i\} \subset \mathcal{A}$, et est coloré en rouge.
- Pour permettre à un véhicule d'être inutilisé si nécessaire, des arcs doivent être définis à partir de chaque nœud de dépôt de départ o_i^+ jusqu'à son nœud de dépôt d'arrivée correspondant o_i^- pour $i \in \mathcal{S}$. Donc, cet ensemble d'arcs peut être exprimé par $\{(o_i^+, o_i^-) \mid i \in \mathcal{S}\} \subset \mathcal{A}$, et est coloré en orange.

- Au sein de chaque ensemble des nœuds de collecte $\mathcal{P}_i$ (resp. nœuds de livraison $\mathcal{D}_i$) pour tout $i \in \mathcal{S}$, les véhicules doivent être autorisés à se déplacer entre n'importe quels des nœuds $\mathcal{P}_i$ (resp. $\mathcal{D}_i$). En d'autres termes, si un véhicule commence à servir une requête de collecte (ou de livraison) de la gare i, il peut répondre à n'importe quelle autre requête de collecte (ou de livraison) de la gare i, mais il ne peut pas se déplacer pour répondre à une requête de collecte (ou de livraison) d'une autre gare. Ce qui signifie que les nœuds $\mathcal{P}_i$ (resp. $\mathcal{D}_i$) doivent former une clique entre eux. Ainsi, on peut formuler cet ensemble d'arcs par $\bigcup_{t\in\mathcal{S}}\{(i,j) \mid i,j \in \mathcal{P}_t; i \neq j\} \subset \mathcal{A}$ (resp. $\bigcup_{t\in\mathcal{S}}\{(i,j) \mid i,j \in \mathcal{D}_t; i \neq j\} \subset \mathcal{A}$). Les arcs de cet ensemble sont colorés en bleu.
- Comme notre problème suit le principe des *Backhauls*, qui consiste à servir les livraisons toujours avant les collectes dans le même itinéraire, aucun arc ne doit être défini de $\mathcal{P}$ à $\mathcal{D}$. D'autre côté, un arc doit être défini de chaque nœud de livraison en $\mathcal{D}$ à chaque nœud dans $\mathcal{P}$, pour permettre à un véhicule de servir non seulement les requêtes de collecte de la dernière gare où il se trouvait, mais aussi toute requête de collecte de n'importe quelle autre gare. On a donc l'ensemble d'arcs $\{(i,j) \mid i \in \mathcal{D}, j \in \mathcal{P}\} \subset \mathcal{A}$ qui est coloré en mauve.
- Un véhicule dont l'itinéraire ne contient que des requêtes de livraison peut terminer son itinéraire dans la gare la plus proche de la dernière requête de livraison qu'il a servie dans son itinéraire. Cela peut se faire en reliant chaque nœud de livraison au nœud de dépôt d'arrivée le plus proche. Toutefois, même si le nœud de livraison est lié à tous les nœuds de dépôt d'arrivée, l'optimisation s'occupe de choisir le dépôt d'arrivée le plus proche. D'où l'ensemble d'arcs $\{(i, o_j^-) \mid i \in \mathcal{D}, j \in \mathcal{S}\} \subset \mathcal{A}$ coloré en noir.
- Lorsqu'un véhicule dessert des nœuds de collecte dans $\mathcal{P}_j$ d'une certaine gare $j \in \mathcal{S}$, il doit rester dans la même gare à la fin de son itinéraire. Donc chaque nœud de collecte dans $\mathcal{P}_j$ doit être relié au nœud de dépôt d'arrivée de la gare correspondante o_j^-. Alors, cet ensemble d'arcs est formulé par $\{(i, o_j^-) \mid i \in \mathcal{P}_j, j \in \mathcal{S}\} \subset \mathcal{A}$ et est coloré en jaune.

Par conséquent, $\mathcal{A}$ peut être exprimé comme suit,

$$\begin{aligned}\mathcal{A} = &\{(i,j) \mid i,j \in \mathcal{O}^+; i \neq j\} \cup \{(o_i^+, j) \mid i \in \mathcal{S}, j \in \mathcal{P}_i \cup \mathcal{D}_i\} \cup \{(o_i^+, o_i^-) \mid i \in \mathcal{S}\} \\ &\cup \left(\bigcup_{t\in\mathcal{S}}\{(i,j) \mid i,j \in \mathcal{P}_t; i \neq j\} \subset \mathcal{A}\right) \cup \left(\bigcup_{t\in\mathcal{S}}\{(i,j) \mid i,j \in \mathcal{D}_t; i \neq j\}\right) \\ &\cup \{(i,j) \mid i \in \mathcal{D}, j \in \mathcal{P}\} \cup \{(i, o_j^-) \mid i \in \mathcal{D}, j \in \mathcal{S}\} \cup \{(i, o_j^-) \mid i \in \mathcal{P}_j, j \in \mathcal{S}\}.\end{aligned}$$

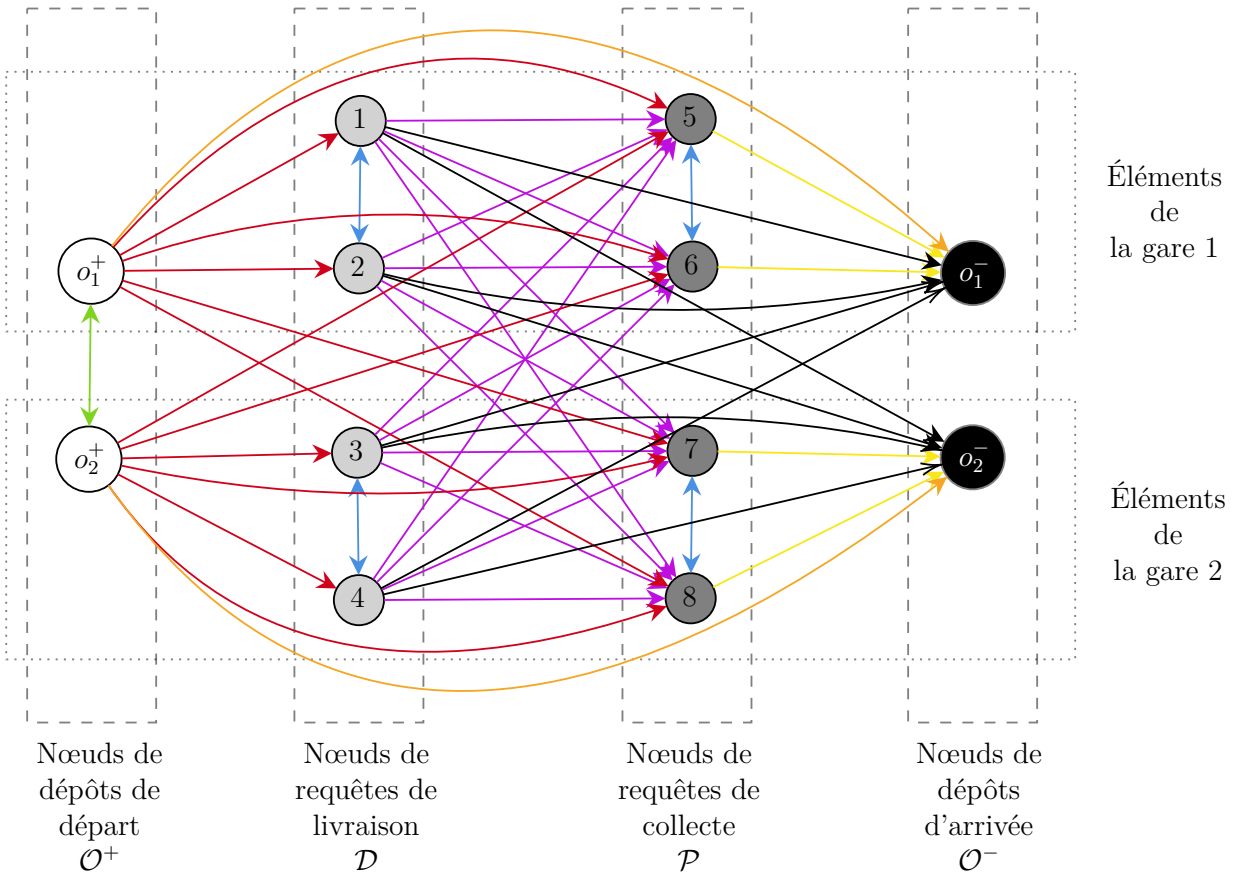


FIGURE 3.1 Un exemple du graphe associé G d'une instance de 2 gares, chacune avec deux requêtes de livraison et deux requêtes de collecte.

Soit $\mathcal{K}$ l'ensemble des véhicules disponibles. Chaque arc $(i, j) \in \mathcal{A}$ a un temps de déplacement $t_{ij}^k \geq 0$ et un coût de déplacement $c_{ij}^k \geq 0$ qui dépendent du véhicule $k \in \mathcal{K}$ utilisé. Tous les temps de déplacement et tous les coûts sont positifs.

Notons u_i la gare associée au passager $i \in \mathcal{P} \cup \mathcal{D}$, c-à-d. la gare d'où vient le passager i si $i \in \mathcal{D}$, et la gare où va le passager i, si $i \in \mathcal{P}$. Chaque passager $i \in \mathcal{P} \cup \mathcal{D}$ est caractérisé par une charge $q_i \geq 0$, une durée de service d_i et un temps de trajet maximal dans le véhicule L_i qui doit être respecté. Les temps de trajet maximum sont définis comme $L_i = SP(i, o_{u_i}^{\pm}) + \Delta_i$, où $SP(i, o_{u_i}^{\pm})$ est le temps de trajet du plus court chemin entre le nœud dépôt de la gare associée u_i et le lieu de service, c-à-d., le chemin le plus court entre le nœud de dépôt de départ associé $o_{u_i}^+ \in \mathcal{O}^+$ et le nœud de livraison $i \in \mathcal{D}$ si i s'agit d'une requête de livraison. Sinon, si i est une requête de collecte, le chemin sera donc le plus court chemin entre le nœud de collecte $i \in \mathcal{P}$ et le nœud de dépôt d'arrivée associé $o_{u_i}^- \in \mathcal{O}^-$. D'autre part, Δ_i est le temps de déviation autorisé pour le passager i, qui est un temps supplémentaire que le véhicule peut utiliser pour

dépasser le temps de trajet idéal du passager. Ce dernier est considéré comme le temps du chemin le plus court entre la gare associée et le lieu de service, c-à-d., $SP(i, o^{\pm}_{u_i})$. Cela vient du fait que tout passager préférerait sûrement atteindre sa destination le « plus tôt» possible. Le plus tôt possible signifie le temps minimum qu'un véhicule doit utiliser pour transporter un passager entre deux points, qui est en effet $SP(i, o^{\pm}_{u_i})$. Par exemple, dans chaque itinéraire, le premier passager qui fait partie des requêtes de livraison $\mathcal{D}$ et le dernier passager qui fait partie des requêtes de collecte $\mathcal{P}$ sont les passagers chanceux qui seront transportés dans leur temps de trajet idéal.

Il n'y a aucune obligation de servir tous les passagers, mais le coût de ne pas servir un passager $i \in \mathcal{P} \cup \mathcal{D}$ est $w_i \geq 0$. Le coût de ne pas servir le passager i est censé être suffisamment élevé pour décourager le non-service de ce passager, à moins qu'il ne soit vraiment optimal de le laisser sans service. Par exemple :

$$w_i > \text{le coût minimum pour servir le passager } i.$$

Pour les requêtes de collecte, une heure de service au plus tôt $e_i \geq 0$ est imposée pour chaque nœud associé $i \in \mathcal{P}$, pour lequel le véhicule ne peut pas commencer à desservir le nœud avant. Cela est dû à la nature du problème, qui consiste à optimiser un système de transport à la demande. Dans ce genre de système, les passagers choisissent la date au plus tôt à laquelle ils peuvent être servis.

Chaque véhicule $k \in \mathcal{K}$ a une capacité maximale Q_k. Aussi, soit s_k la gare où le véhicule k a commencé son itinéraire (la gare d'origine de k où il a été initialement localisé).

Une heure de départ T_i du train associé à la gare $i \in \mathcal{S}$ est imposée. Un véhicule ne peut pas entrer dans la gare après cette heure.

Le problème consiste à trouver les itinéraires optimaux qui commencent aux nœuds $\mathcal{O}^+$ et se terminent aux nœuds $\mathcal{O}^-$, qui visitent le nombre maximum de nœuds dans $\mathcal{P} \cup \mathcal{D}$ (servent le nombre maximum de requêtes de collecte et livraison) en utilisant les ressources limitées telles que le nombre de véhicules disponibles et les heures de départ des trains, tout en maintenant un bon compromis entre le bon service à la clientèle et/ou la minimisation des coûts.

3.3 Formulation mathématique

Dans cette section, nous formulons notre problème sous forme d'un programme linéaire mixte en nombres entiers. Nous commencerons par introduire les notations nécessaires, puis nous décrirons les variables de décision, la fonction objectif et les différentes contraintes.

Notations

Le Tableau 3.1 résume la notation utilisée pour le modèle mathématique.

n	Le nombre de passagers à collecter (type *Pickup*).
m	Le nombre de passagers à déposer (type *Delivery*)
r	Le nombre de gares.
$\mathcal{S}$	L'ensemble des gares.
$\mathcal{O}^+$	L'ensemble des nœuds de dépôts de départ associés aux gares.
$\mathcal{O}^-$	L'ensemble des nœuds de dépôts d'arrivée associés aux gares.
$\mathcal{P}$	L'ensemble des nœuds associés aux passagers à collecter.
$\mathcal{D}$	L'ensemble des nœuds associés aux passagers à déposer.
$\mathcal{K}$	L'ensemble des véhicules.
w_i	Le coût de ne pas servir le passager $i \in \mathcal{P} \cup \mathcal{D}$.
q_i	La charge sur le véhicule au nœud $i \in \mathcal{P} \cup \mathcal{D}$.
e_i	L'heure au plus tôt de début de service au nœud $i \in \mathcal{P}$.
d_i	La durée de service au nœud $i \in \mathcal{O}^+ \cup \mathcal{P} \cup \mathcal{D}$.
u_i	La gare associée au passager $i \in \mathcal{P} \cup \mathcal{D}$.
T_i	L'heure de départ du train associé à la gare $i \in \mathcal{S}$.
L_i	Le temps du voyage maximum du passager $i \in \mathcal{P} \cup \mathcal{D}$.
$SP(i,j)$	Le temps nécessaire pour parcourir le chemin le plus court entre les sommets $i, j \in \mathcal{V}$.
Δ_i	La déviation de temps autorisée pour le nœud $i \in \mathcal{P}$.
Q_k	La capacité du véhicule $k \in \mathcal{K}$.
s_k	La gare d'origine du véhicule $k \in \mathcal{K}$.
c_{ij}^k	Coût du déplacement du nœud $i \in \mathcal{V}$ au nœud $j \in \mathcal{V}$ en utilisant le véhicule $k \in \mathcal{K}$.
t_{ij}^k	Temps nécessaire pour passer du nœud $i \in \mathcal{V}$ au nœud $j \in \mathcal{V}$ en utilisant le véhicule $k \in \mathcal{K}$.

Tableau 3.1 Notations utilisées pour le modèle mathématique

Variables

Le modèle mathématique proposé est un programme linéaire mixte en nombres entiers, dont les variables sont les suivantes.

- $x_{ij}^k \in \{0,1\}$: une variable binaire qui vaut 1 si le véhicule $k \in \mathcal{K}$ utilise l'arc $(i,j) \in \mathcal{A}$, et 0 sinon ;
- $B_i^k \geq 0$: une variable continue positive qui correspond au début du temps de service du véhicule $k \in \mathcal{K}$ au nœud $i \in \mathcal{V}$;
- $Q_i^k \geq 0$: une variable continue positive qui correspond à la charge du véhicule $k \in \mathcal{K}$ après avoir visité le nœud $i \in \mathcal{V}$;
- $\zeta_i \in \{0,1\}$: une variable binaire qui vaut 1 si le passager $i \in \mathcal{P} \cup \mathcal{D}$ n'est pas servi, et 0 sinon.

Fonction objectif

Dans un tel problème de transport où les véhicules transportent des passagers et non des marchandises, le service à la clientèle est très important. Par conséquent, outre la fonction objectif classique qui minimise les coûts totaux, nous présentons ici différentes formulations de la fonction objectif.

- Minimisation des coûts de trajet totaux, à savoir

$$\min \alpha(x) = \min \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^k x_{ij}^k. \tag{3.1}$$

- Minimisation des temps d'attente des passagers à collecter avant le début de service, formulée par

$$\min \beta(B) = \min \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{P}} \left(B_i^k - e_i \right). \tag{3.2}$$

  L'objectif $\min \beta(B)$ consiste à minimiser le temps entre le moment où le passager est disponible pour être servi et le moment où le véhicule commence à le servir. Cette fonction se concentre uniquement sur les passagers de type *Pickup* qui sont les passagers qui se trouvent à l'extérieur des gares et qui attendent d'être collectés. Chacun de ces passagers est caractérisé par une heure de disponibilité e_i. Cette fonction ne prend pas en compte les passagers de type *Delivery* ou les passagers qui se trouvent dans les gares et qui attendent d'être livrés. Le temps de disponibilité de ces derniers dépend de l'heure d'arrivée des trains, qui ne peut pas être modifiée par le passager.
- Minimisation de la moyenne des dates d'arrivée des véhicules en gare en fin de trajet, à savoir

$$\min \gamma(B) = \min \frac{1}{|K|} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{O}^-} B_i^k. \tag{3.3}$$

  Cet objectif vise à minimiser la longueur des itinéraire, ou en d'autres termes, à réduire le nombre d'heures d'utilisation de chaque véhicule.
- Minimisation des coûts des passagers non servis

$$\min \delta(\zeta) = \min \sum_{i \in \mathcal{P} \cup \mathcal{D}} w_i \zeta_i. \tag{3.4}$$

La fonction objectif du modèle peut donc être l'une de ces formulations ou une combinaison de certaines ou de toutes. Dans le reste de ce chapitre, nous considérons l'objectif de la minimisation d'une agrégation des coûts totaux de déplacements plus les coûts totaux des

passagers non servis, à savoir

$$\min \alpha(x) + \delta(\zeta) = \min \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}} c_{ij}^k x_{ij}^k + \sum_{i \in \mathcal{P} \cup \mathcal{D}} w_i \zeta_i, \tag{3.5}$$

car ce sont les deux principaux objectifs que la société SNCF souhaite atteindre.

De toute façon, chacune des fonctions α, β, γ et δ peuvent être utilisées comme des facteurs ou des indices pour évaluer la qualité des solutions et les comparer.

Contraintes

- Chaque passager est servi au plus une fois :

$$\sum_{k \in \mathcal{K}} \sum_{j:(i,j) \in \mathcal{A}} x_{ij}^k + \zeta_i = 1, \qquad \forall i \in \mathcal{P} \cup \mathcal{D}. \tag{3.6}$$

- Tous les véhicules sont utilisés :

$$\sum_{j:(o_{s_k}^+, j) \in \mathcal{A}} x_{o_{s_k}^+ j}^k = 1, \qquad \forall k \in \mathcal{K}. \tag{3.7}$$

- Le flot est conservé :

$$\sum_{i:(i,j) \in \mathcal{A}} x_{ij}^k - \sum_{i:(j,i) \in \mathcal{A}} x_{ji}^k = 0 \qquad \forall j \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{O}^+ \setminus \{o_{s_k}^+\}, k \in \mathcal{K}. \tag{3.8}$$

  Comme indiqué, le flot n'est pas conservé au nœud de départ $o_{s_k}^+$ du véhicule k, qui correspond au nœud de dépôt de départ de la gare d'origine du véhicule k. En effet, s'il était conservé à ce nœud, le véhicule devrait rentrer à ce nœud à partir d'autres nœuds, ce qui est impossible.

- La charge des véhicules à un nœud dépend de la charge cumulée de l'itinéraire et respecte la capacité du véhicule, ce qui est garanti par les trois ensembles de contraintes suivants :

$$Q_i^k + q_j - Q_k(1 - x_{ij}^k) \leq Q_j^k, \qquad \forall (i,j) \in \mathcal{A} \setminus (\mathcal{D} \times \mathcal{P}), k \in \mathcal{K}; \tag{3.9}$$

$$q_j + Q_k(1 - x_{ij}^k) \geq Q_j^k, \qquad \forall (i,j) \in \mathcal{D} \times \mathcal{P}, k \in \mathcal{K}; \tag{3.10}$$

$$q_i \leq Q_i^k \leq Q_k, \qquad \forall i \in \mathcal{V}, k \in \mathcal{K}. \tag{3.11}$$

  Contrairement à d'autres formulations des problèmes de collecte et livraison où les nœuds de collecte ont généralement un signe de charge différent de celui des nœuds de

livraison, nous considérons ici des charges positives sur tous les nœuds, qu'il s'agisse des nœuds de collecte ou des nœuds de livraison. Cela est dû au fait que les véhicules ne mélangent pas les collectes et les livraisons lors de l'exécution de leurs routes, car les routes sont formées selon le principe de *Backhauls*. Par conséquent, il est nécessaire de trouver un moyen de garantir que la charge dans les véhicules soit réinitialisée au minimum chaque fois que le véhicule commence à desservir les nœuds de collecte, qui sont toujours desservis après les nœuds de livraison dans ce problème.

Ainsi, les contraintes (3.9) appliquent l'accumulation des charges sur tous les nœuds de ramassage et de livraison en utilisant tous les arcs du réseau sauf l'ensemble d'arcs $\mathcal{D} \times \mathcal{P}$ qui contient les arcs de l'ensemble des nœuds $\mathcal{D}$ vers l'ensemble des nœuds $\mathcal{P}$. Ces arcs sont considérés comme un pont permettant de passer des nœuds de livraison aux nœuds de collecte. Lorsqu'un véhicule k se déplace des nœuds de livraison $\mathcal{D}$ aux nœuds de collecte $\mathcal{P}$, c-à-d. lorsqu'il utilise un arc de $\mathcal{D} \times \mathcal{P}$, la charge dans le véhicule est réinitialisée au minimum grâce aux contraintes (3.10) et à la partie gauche des contraintes (3.11). En d'autres termes, lorsque $x_{ij}^k = 1$ où $(i, j) \in \mathcal{D} \times \mathcal{P}$, alors d'après la contrainte (3.10), la charge cumulée du véhicule au nœud j est bornée supérieurement par q_j,

$$q_j + Q_k(1 - x_{ij}^k) \geq Q_j^k \Longrightarrow q_j \geq Q_j^k.$$

D'autre part, la partie gauche de la contrainte (3.11) limite la charge cumulée du véhicule au nœud j inférieurement par q_j. D'où la charge cumulée du véhicule au nœud j vérifie $Q_j^k = q_j$. Par conséquent, la charge dans le véhicule est réinitialisée.

- L'heure d'arrivée à un nœud correspond à la durée de son arc entrant et à la durée de service du nœud précédent :

$$B_i^k + d_i + t_{ij}^k - \max_{h \in \mathcal{O}^-}\{T_h\}(1 - x_{ij}^k) \leq B_j^k, \qquad \forall (i, j) \in \mathcal{A}, k \in \mathcal{K}. \tag{3.12}$$

- Les temps de disponibilité de début du service au niveau des nœuds de collecte sont respectés :

$$B_i^k \geq e_i, \qquad \forall i \in \mathcal{P}, k \in \mathcal{K}. \tag{3.13}$$

Rappelons que les passagers de livraison (nœuds de livraison $\mathcal{D}$) ont tous la même heure de disponibilité, qui est l'heure 0.

- Les heures de départ des trains sont respectées :

$$B_i^k \leq T_i, \qquad \forall i \in \mathcal{O}^-, k \in \mathcal{K}. \tag{3.14}$$

- Le temps passé dans les véhicules par les passagers, respecte le temps maximal de trajet des passagers :

$$B_i^k - (B_{o_{u_i}^+}^k + d_{o_{u_i}^+}) \leq L_i = SP(i, o_{u_i}^+) + \Delta_i, \qquad \forall i \in \mathcal{D}, k \in \mathcal{K}; \tag{3.15}$$

$$B_{o_{u_i}^-}^k - (B_i^k + d_i) \leq L_i = SP(i, o_{u_i}^-) + \Delta_i, \qquad \forall i \in \mathcal{P}, k \in \mathcal{K}. \tag{3.16}$$

- Chaque variable est définie dans son ensemble respectif :

$$x_{ij}^k \in \{0, 1\}; \qquad \forall (i, j) \in \mathcal{A}, k \in \mathcal{K}, \tag{3.17}$$

$$\zeta_i \in \{0, 1\}; \qquad \forall i \in \mathcal{P} \cup \mathcal{D}, \tag{3.18}$$

$$B_i^k \geq 0; \qquad \forall i \in \mathcal{V}, k \in \mathcal{K}, \tag{3.19}$$

$$Q_i^k \geq 0; \qquad \forall i \in \mathcal{V}, k \in \mathcal{K}. \tag{3.20}$$

Ci-dessous se trouve la forme compacte du modèle, où l'objectif est la minimisation d'une agrégation des coûts totaux de déplacements et les coûts totaux des passagers non servis (3.5).

$$
(MIP):\begin{cases}
\min \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k + \sum_{i\in\mathcal{P}\cup\mathcal{D}} w_i\zeta_i & \\
\text{s.à}, \sum_{k\in\mathcal{K}} \sum_{j:(i,j)\in\mathcal{A}} x_{ij}^k + \zeta_i = 1 & \forall i \in \mathcal{P}\cup\mathcal{D} \\
\sum_{j:(o_{s_k}^+,j)\in\mathcal{A}} x_{o_{s_k}^+ j}^k = 1 & \forall k \in \mathcal{K} \\
\sum_{i:(i,j)\in\mathcal{A}} x_{ij}^k - \sum_{i:(j,i)\in\mathcal{A}} x_{ji}^k = 0 & \forall j \in \mathcal{P}\cup\mathcal{D}\cup\mathcal{O}^+ \setminus \{o_{s_k}^+\}, k \in \mathcal{K} \\
Q_i^k + q_j - Q_k(1 - x_{ij}^k) \leq Q_j^k & \forall (i,j) \in \mathcal{A}\setminus(\mathcal{D}\times\mathcal{P}), k \in \mathcal{K} \\
q_j + Q_k(1 - x_{ij}^k) \geq Q_j^k & \forall (i,j) \in \mathcal{D}\times\mathcal{P}, k \in \mathcal{K} \\
q_i \leq Q_i^k \leq Q_k & \forall i \in \mathcal{V}, k \in \mathcal{K} \\
B_i^k + d_i + t_{ij}^k - \max_{h\in\mathcal{O}^-}\{T_h\}(1 - x_{ij}^k) \leq B_j^k & \forall (i,j) \in \mathcal{A}, k \in \mathcal{K} \\
B_i^k \geq e_i & \forall i \in \mathcal{P}, k \in \mathcal{K} \\
B_i^k \leq T_i & \forall i \in \mathcal{O}^-, k \in \mathcal{K} \\
B_i^k - (B_{o_{u_i}^+}^k + d_{o_{u_i}^+}) \leq L_i = SP(i, o_{u_i}^+) + \Delta_i & \forall i \in \mathcal{D}, k \in \mathcal{K} \\
B_{o_{u_i}^-}^k - (B_i^k + d_i) \leq L_i = SP(i, o_{u_i}^-) + \Delta_i & \forall i \in \mathcal{P}, k \in \mathcal{K} \\
x_{ij}^k \in \{0,1\} & \forall (i,j) \in \mathcal{A}, k \in \mathcal{K} \\
\zeta_i \in \{0,1\} & \forall i \in \mathcal{P}\cup\mathcal{D} \\
B_i^k \geq 0 & \forall i \in \mathcal{V}, k \in \mathcal{K} \\
Q_i^k \geq 0 & \forall i \in \mathcal{V}, k \in \mathcal{K}
\end{cases}
$$

CHAPITRE 4 Méthodologie de résolution

Dans ce chapitre, nous présentons notre méthode de résolution basée sur la génération de colonnes. Plus précisément, nous détaillons la décomposition du problème en un problème maître et en sous-problèmes, ainsi que la manière dont l'algorithme de résolution permet de résoudre le problème maître. Par ailleurs, nous décrivons la modélisation des sous-problèmes sous la forme de problèmes de plus court chemin avec contraintes de ressources, et expliquons comment les résoudre à l'aide de la programmation dynamique.

4.1 Résolution par génération de colonnes et l'arrondi des valeurs fractionnaires dans un contexte de recherche en profondeur

Il est possible de résoudre notre problème directement en utilisant la formulation explicite présentée dans le chapitre précédent. Cette formulation est dite compacte ou non décomposée, contrairement à la reformulation de Dantzig-Wolfe qui est basée sur la décomposition du problème en problème maître et sous-problème(s).

Le modèle MIP peut être résolu à l'aide d'un solveur commercial pour les programmes mathématiques linéaires en nombres entiers comme CPLEX de l'entreprise IBM ou Gurobi Optimizer de l'entreprise Gurobi Optimization. Cependant, en raison de la faiblesse de la relaxation linéaire du MIP, le temps d'exécution devient rapidement prohibitif pour des instances de grande taille. Nous choisissons donc d'exploiter la structure bloc-angulaire de notre MIP et de le décomposer en un problème maître et des sous-problèmes adéquats. Nous envisageons également d'appliquer des stratégies d'arrondi (*rounding strategies*) aux variables fractionnaires de la solution optimale du problème maître, afin de faciliter la génération de solutions entières, chaque fois que ce dernier est résolu à l'optimalité et que la solution n'est pas encore entière. Cela donne lieu à l'algorithme itératif de génération de colonnes avec heuristique de plongée (*Diving heuristic*) décrit dans les paragraphes suivants.

4.1.1 Problème maître

La génération de colonnes ne s'applique généralement pas directement sur la formulation originale du problème, mais sur une reformulation du problème comportant un très grand nombre de variables, mais relativement peu de contraintes.

Dans le MIP, seules les contraintes d'affectation (3.6) lient tous les véhicules ensemble,

tandis que les contraintes restantes traitent chaque véhicule séparément. Par conséquent, nous pouvons décomposer le problème original en : un sous-problème ayant une structure d'un problème du plus court chemin avec contraintes de ressources, pour chaque véhicule ; et un problème maître. Autrement dit, le problème maître est défini par les contraintes (3.5) et (3.6), y compris la fonction objectif et l'affectation de chaque passager à un seul véhicule au plus. Le reste des contraintes appartient au sous-problème qui a une fonction objectif modifiée et qui se décompose en $|\mathcal{K}|$ sous-problèmes indépendants, un pour chaque véhicule [61].

Les chemins pouvant être réalisés par le véhicule k peuvent être représentés par l'ensemble Ω_k. Chaque chemin p dans Ω_k peut être représenté par les valeurs binaires x^k_{ijp}, où x^k_{ijp} est égal à 1 si le véhicule k se rend directement du nœud i au nœud j sur le chemin p, et 0 sinon. On peut exprimer toute solution x^k_{ij} du problème maître comme une combinaison convexe d'un nombre fini de chemins élémentaires. Par conséquent, nous avons :

$$
\begin{aligned}
x^k_{ij} &= \sum_{p\in\Omega_k} x^k_{ijp}\lambda^k_p, && \forall (i,j)\in\mathcal{A}, k\in\mathcal{A};\\
\sum_{p\in\Omega_k} \lambda^k_p &= 1, && \forall k\in\mathcal{K};\\
\lambda^p_k &\geq 0, && \forall k\in\mathcal{K}, p\in\Omega_k.
\end{aligned}
$$

On peut également définir c^k_p le coût d'un trajet p effectué par un véhicule k, et a^k_{ip} le nombre de fois qu'un passager i est visité par le véhicule k sur le trajet p. Par conséquent :

$$
\begin{aligned}
c^k_p &= \sum_{(i,j)\in\mathcal{A}} c^k_{ij}x^k_{ijp}, && \forall k\in\mathcal{K}, p\in\Omega_k;\\
a^k_{ip} &= \sum_{j:(i,j)\in\mathcal{A}} x^k_{ijp}, && \forall i\in\mathcal{V}, k\in\mathcal{K}, p\in\Omega_k.
\end{aligned}
$$

En substituant ces deux dernières valeurs dans (3.5) et (3.6), nous obtenons la formulation révisée appelée le MP :

$$
(\mathrm{MP}) : \begin{cases}
\min \sum_{k \in \mathcal{K}} \sum_{p \in \Omega_k} c_p^k \lambda_p^k + \sum_{i \in \mathcal{P} \cup \mathcal{D}} w_i \zeta_i & & (4.1) \\
\text{s.à}, \sum_{k \in \mathcal{K}} \sum_{p \in \Omega_k} a_{ip}^k \lambda_p^k + \zeta_i = 1 \quad \forall i \in \mathcal{P} \cup \mathcal{D} & (\pi_i) & (4.2) \\
\sum_{p \in \Omega_k} \lambda_p^k = 1 \quad \forall k \in \mathcal{K} & (\sigma_k) & (4.3) \\
\lambda_p^k \geq 0 \quad \forall k \in \mathcal{K}, p \in \Omega_k & & (4.4) \\
\zeta_i \geq 0 \quad \forall i \in \mathcal{P} \cup \mathcal{D} & & (4.5)
\end{cases}
$$

Dans la version en nombres entiers du problème maître MP, la variable λ_p^k peut être interprétée comme une variable binaire qui prend la valeur 1 si la route p est attribuée au véhicule k, et 0 sinon.

La méthodologie de génération de colonnes limite les colonnes du problème maître à celles qui ont déjà été générées, ce qui donne lieu au problème maître restreint (RMP). Si les routes générées pour le véhicule k sont indiquées par $\Omega'_k \subseteq \Omega_k$, alors le RMP est créé en remplaçant Ω_k par Ω'_k dans le problème maître.

4.1.2 Sous-problème

L'objectif de RMP est de déterminer un ensemble de routes à coût minimal à partir des routes existantes dans le RMP. Une fois que le RMP est résolu, la valeur de chaque variable duale liée à chaque contrainte de RMP peut être obtenue. Ces valeurs duales sont ensuite incorporées dans la fonction objectif du sous-problème. Chaque fois que le sous-problème est résolu, de nouvelles routes sont générées selon les besoins et insérées dans l'ensemble $\Omega' = \bigcup_{k \in \mathcal{K}} \Omega'_k$.

Si nous définissons les variables duales π_i et σ_k pour les contraintes (4.2) et (4.3), respectivement, alors le coût réduit de la variable λ_p^k peut être exprimé comme :

$$
\overline{c}_p^k = c_p^k - \left(\sum_{i \in \mathcal{P} \cup \mathcal{D}} \pi_i a_{ip}^k \right) - \sigma_k.
$$

Le sous-problème associé au véhicule k peut donc être formulé sous forme d'un programme linéaire mixte en nombres entiers $SPMIP^k$ (*Subproblem of vehicle k as a Mixed Integer Program*) comme ci-dessous.

$$
(SPMIP^k) : \begin{cases}
\min \sum_{(i,j)\in\mathcal{A}} (c_{ij}^k - \pi_i) x_{ij} - \sigma_k & \\
\text{s.à,} \sum_{j:(s_k,j)\in\mathcal{A}} x_{s_k j}^k = 1 & \\
\sum_{i:(i,j)\in\mathcal{A}} x_{ij}^k - \sum_{i:(j,i)\in\mathcal{A}} x_{ji}^k = 0 & \forall j \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{O}^+ \setminus \{s_k\} \\
Q_i^k + q_j - Q_k(1 - x_{ij}^k) \leq Q_j^k & \forall (i,j) \in \mathcal{A} \setminus (\mathcal{D} \times \mathcal{P}) \\
q_j + Q_k(1 - x_{ij}^k) \geq Q_j^k & \forall (i,j) \in \mathcal{D} \times \mathcal{P} \\
q_i \leq Q_i^k \leq Q_k & \forall i \in \mathcal{V} \\
B_i^k + d_i + t_{ij}^k - \max_{h\in\mathcal{O}^-}\{T_h\}(1 - x_{ij}^k) \leq B_j^k & \forall (i,j) \in \mathcal{A} \\
B_i^k \geq e_i & \forall i \in \mathcal{P} \\
B_i^k \leq T_i & \forall i \in \mathcal{O}^- \\
B_i^k - (B_{u_i}^k + d_{u_i}) \leq L_i = SP(i, o_{u_i}^+) + \Delta_i & \forall i \in \mathcal{D} \\
B_{u_i}^k - (B_i^k + d_i) \leq L_i = SP(i, o_{u_i}^-) + \Delta_i & \forall i \in \mathcal{P} \\
x_{ij}^k \in \{0,1\} & \forall (i,j) \in \mathcal{A} \\
\zeta_i \in \{0,1\} & \forall i \in \mathcal{P} \cup \mathcal{D} \\
B_i^k \geq 0 & \forall i \in \mathcal{V} \\
Q_i^k \geq 0 & \forall i \in \mathcal{V}
\end{cases}
$$

4.1.3 Description de l'algorithme général

Sachant que le nombre de variables de décision dans le MP est exponentiel, il est donc difficile de le résoudre directement. De plus, il est inutile d'énumérer toutes les routes de $\Omega = \bigcup_{k\in\mathcal{K}} \Omega_k$, puisque seules quelques unes de ces routes sont utilisées dans la solution optimale de MP. On utilise donc à la place le RMP_l, le problème maître restreint à une itération l du processus de la génération de colonnes. RMP_l ne considère que les routes de l'ensemble $\Omega_l \subseteq \Omega$, qui est un sous-ensemble des routes réalisables générées avant l'itération l.

Pour déterminer Ω_{l+1}, nous procédons comme suit. À chaque itération l de l'algorithme de génération de colonnes, nous résolvons le RMP_l à l'aide d'un solveur linéaire tel que Gurobi. Nous utilisons ensuite les valeurs des variables duales fournies par la solution optimale de ce problème pour modifier le poids des arcs dans $\mathcal{G}_k$ (voir la section 4.2.2), le graphe associé au véhicule k, afin de refléter leur coût réduit. Nous pouvons ainsi résoudre SP_k, c-à-d., le sous-problème du véhicule k. La résolution des sous-problèmes nous permet d'identifier les

routes dont le coût réduit est strictement négatif à l'itération l, c-à-d., celles qui améliorent la solution actuelle de RMP_l.

Après avoir résolu le sous-problème, nous ajoutons au RMP_l un certain nombre de variables de décision binaires correspondant aux routes à coût réduit négatif identifiées par l'algorithme de résolution du problème du plus court chemin. Nous ajoutons également une colonne au RMP_l correspondant à chacun de ces chemins et incrémentons le compteur d'itération de l à $l+1$. Nous répétons cette procédure itérative jusqu'à ce que la valeur du plus court chemin fournie par la résolution du sous-problème soit supérieure ou égale à zéro.

Si h est le nœud actuel dans l'arbre de recherche (lors de la première exécution de l'algorithme de génération de colonnes, on a $h = 0$), alors il est possible que la solution optimale de RMP_h^l fournie à la fin de l'algorithme de génération de colonnes au niveau du nœud h soit fractionnaire. Étant donné que le problème maître est en nombres entiers, il est nécessaire de s'assurer qu'au moins une solution entière sera fournie à la fin de l'algorithme. Pour ce faire, nous fixons certaines variables en appliquant un arrondi supérieur à un ensemble de variables ayant une valeur fractionnaire dans la solution optimale actuelle du problème maître, tout en fixant d'autres variables qui ont déjà atteint leurs valeurs maximales. On crée un nouveau nœud de branchement $h+1$ ayant h comme parent et on ajoute la fixation des variables décrites au RMP_{h+1}^0 avant de redémarrer l'algorithme de génération de colonnes. Cette dernière procédure qui aide à obtenir une solution entière est appelée l'heuristique de plongée (*Diving heuristic*) [62]. Elle commence au nœud initial de l'arbre de recherche et fixe de manière répétitive une ou plusieurs routes fractionnaires (c-à-d., elle fixe les variables λ_p^k fractionnaires dans la solution finale du processus de génération de colonnes) dans le cadre d'une recherche en profondeur d'abord sans retour en arrière (*depth-first search without backtracking*).

Une heuristique de plongée peut être considérée comme une heuristique de recherche dans un arbre de *Branch-and-Bound* basé sur la programmation linéaire. La recherche plonge en profondeur dans l'arbre d'énumération en sélectionnant une branche de manière heuristique à chaque nœud, comme illustré dans la Figure 4.1. La règle de branchement utilisée dans un tel contexte est généralement assez différente de celle utilisée dans une approche exacte. Dans une heuristique de plongée, on ne se préoccupe pas d'équilibrer l'arbre, mais on vise à trouver rapidement de bonnes solutions entières.

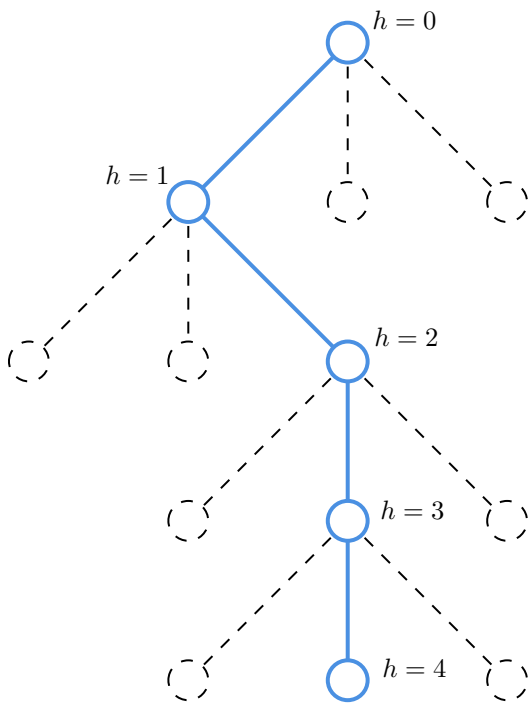


FIGURE 4.1 Illustration d'une heuristique de plongée dans un arbre d'énumération

Notre algorithme, qui peut être appelé une heuristique de plongée basée sur la génération de colonnes $CGDH$ (*Column Generation-based Diving Heuristic*), se termine lorsque la solution de la relaxation linéaire de RMP_h^l est entière ou si le temps maximum autorisé est dépassé. Noter que la solution triviale où aucun passager n'est servi, c-à-d., la solution dans laquelle $\zeta_i = 1$ pour tout $i \in \mathcal{P} \cup \mathcal{D}$, est toujours disponible, donc le problème est toujours réalisable et borné.

Le pseudo-code de l'algorithme général est décrit ci-dessous dans Algorithme 1. Plus de détails sont donnés dans les sections suivantes. Soit :

— $\Omega_{h,k}^l \subseteq \Omega_k$: l'ensemble restreint des routes réalisables du véhicule k considérées à l'itération l dans le nœud de branchement h ;

— $\Omega_h^l = \bigcup_{k \in \mathcal{K}} \Omega_{h,k}^l$: l'ensemble restreint des routes réalisables considérées à l'itération l dans le nœud de branchement h ;

— $D_{RP,h}^l$: le domaine réalisable du problème maître restreint au nœud h de branchement à l'itération l de l'algorithme de génération de colonnes ;

— $(\pi, \sigma)_h^l$: la solution duale calculée à partir de la solution optimale de RMP_h^l ;

— $SP_{h,k}^l$: le sous-problème du véhicule k au nœud h de branchement à l'itération l de l'algorithme de génération de colonnes ;

— $z(SP^l_{h,k})$: la valeur minimale de $SP^l_{h,k}$;
— $MaxCol$: le nombre maximal de routes à coût négatif qui peuvent être pris en compte à partir de la résolution d'un sous-problème ;
— $Col(SP^l_{h,k})$: un ensemble de routes contenant au plus $MaxCol$ routes ayant les coûts les plus négatifs trouvées lors de la résolution de $SP^l_{h,k}$;
— $LimTem$: la limite autorisée du temps d'exécution pour l'algorithme général ;
— $LimIte$: le nombre maximum d'itérations autorisées pour l'algorithme général.

Algorithme 1 : Heuristique de plongée basée sur la génération de colonnes

1 $h = 0$, $l = 0$, continuer=Vrai ;
2 construire Ω_0^0 en générant un ensemble initial de routes ;
3 **tant que** *continuer=Vrai* **faire**
4 résoudre RMP_h^l et fournir les valeurs duales $(\pi, \sigma)_h^l$;
5 **pour** $k \in \mathcal{K}$ **faire**
6 mettre à jour $SP_{h,k}^l$ en utilisant les valeurs de $(\pi, \sigma)_h^l$;
7 résoudre $SP_{h,k}^l$ et fournir $z(SP_{h,k}^l)$ et $Col(SP_{h,k}^l)$;
8 ajouter les routes de $Col(SP_{h,k}^l)$ à Ω_h^l ;
9 **si** $\min_{k \in \mathcal{K}}\{z(SP_{h,k}^l)\} \geq 0$ **alors**
10 **si** *la solution* $(\lambda, \zeta)_h^l$ *est fractionnaire* **alors**
11 $\Omega_{h+1}^0 = \Omega_h^l$
12 **pour** $k \in \mathcal{K}$ *et* $p \in \Omega_{h+1,k}^0$ **faire**
13 **si** $\lambda_p^{k*} = 1$ **alors**
14 ajouter la contrainte $\lambda_p^k \geq 1$ au $D_{RP,h+1}^0$ pour fixer la varibale λ_p^k ;
15 fixer la varibale $\lambda_{p'}^{k'}$ en ajoutant la contrainte $\lambda_{p'}^{k'} \geq 1$ au $D_{RP,h+1}^0$ dont $(k', p') = \arg\max \left\{\lambda_p^{k*} \mid \lambda_p^{k*} < 1\right\}$;
16 $h \longleftarrow h + 1$;
17 $l \longleftarrow 0$;
18 $temps \longleftarrow 0$;
19 **sinon**
20 $continuer = Faux$;
21 **sinon si** $temps > LimTem$ *ou* $l > LimIte$ **alors**
22 $continuer = Faux$;
23 **sinon**
24 $l \longleftarrow l + 1$;
25 Résoudre RMP_h^l en utilisant l'ensemble final de routes Ω_h^l et sans tenir compte la fixation des variables qui a été appliquée aux relaxations et fournir la dernière meilleure solution.

L'Algorithme 1 commence par l'initialisation de paramètres à la ligne 1. Ensuite, il génère l'ensemble des routes initiales à la ligne 2. L'ensemble initial des routes peut être généré à l'aide de n'importe quelle heuristique. Cependant, l'ensemble des routes triviales où les

véhicules ne desservent aucun passager et dont les routes commencent et se terminent à la gare d'origine, est toujours suffisant pour démarrer l'algorithme. Les lignes 4 à 24 contiennent le cœur de l'algorithme général itératif ; elles se répètent jusqu'à ce qu'un critère d'arrêt soit effectué dans l'une des lignes 20 ou 22. Les lignes 4 à 8 contiennent le processus de génération de colonnes et sont colorées en bleu. À la ligne 4, le RMP correspondant est résolu et les valeurs duales correspondantes sont obtenues. Ensuite, le sous-problème de chaque véhicule est créé à la ligne 6 et résolu à la ligne 7. Les routes obtenues à partir de chaque sous-problème sont ajoutées à l'ensemble actuel de routes à la ligne 8. La ligne 9 vérifie si le processus de génération de colonnes doit être arrêté ou non en vérifiant les valeurs optimales des sous-problèmes. Si elles ont toutes des valeurs positives, alors la génération de colonnes est arrêtée et un processus de plongée peut être appliqué. Sinon, s'il y a plus de temps et plus d'itérations disponibles, l'Algorithme 1 passe à la ligne 24 où il met à jour le nombre d'itérations et recommence le processus de génération de colonnes. Mais si plus de temps ou d'itérations ne sont disponibles, alors le processus itératif est arrêté et l'algorithme passe à la ligne 25. Sinon, il démarre un processus de plongée à la ligne 10. Après avoir mis à jour le nœud de recherche actuel à la ligne 11, le processus de plongée est implémenté dans les lignes 12 à 15 colorées en orange. Les variables qui sont égaux à 1 dans la solution fractionnaire actuelle sont fixés à la ligne 14, la même chose se fait à la ligne 15 pour la variable qui a la valeur la plus proche de 1 mais pas égale à 1. Après le processus de plongée, les paramètres sont mis à jour dans les lignes 16 à 18. La ligne 20 n'est atteinte que si la solution courante est entière ; le processus itératif est alors arrêté et l'algorithme passe à la ligne 25.

La ligne 25 de l'Algorithme 1 consiste à résoudre le RMP en nombres entiers en considérant le dernier ensemble restreint de routes trouvé Ω_h^l ; la solution trouvée à cette étape est considérée comme la meilleure solution trouvée par l'Algorithme 1. À cette étape, la résolution de RMP en nombres entiers peut prendre trop de temps à être exécutée en raison du grand nombre de colonnes qui ont été générées. Cependant, grâce à l'heuristique de plongée, une solution entière est toujours disponible, qui est la solution de la dernière RMP_h^l trouvée en utilisant les contraintes initiales de MP original plus toutes les fixations des varibales générées par l'heuristique de plongée.

Un organigramme simplifié de l'Algorithme 1 est donné dans la Figure 4.2, dont le processus de génération de colonnes (resp. l'heuristique de plongée) étant représenté par la partie bleue (resp. orange) qui corresponde aux lignes bleues (resp. oranges) dans l'Algorithme 1.

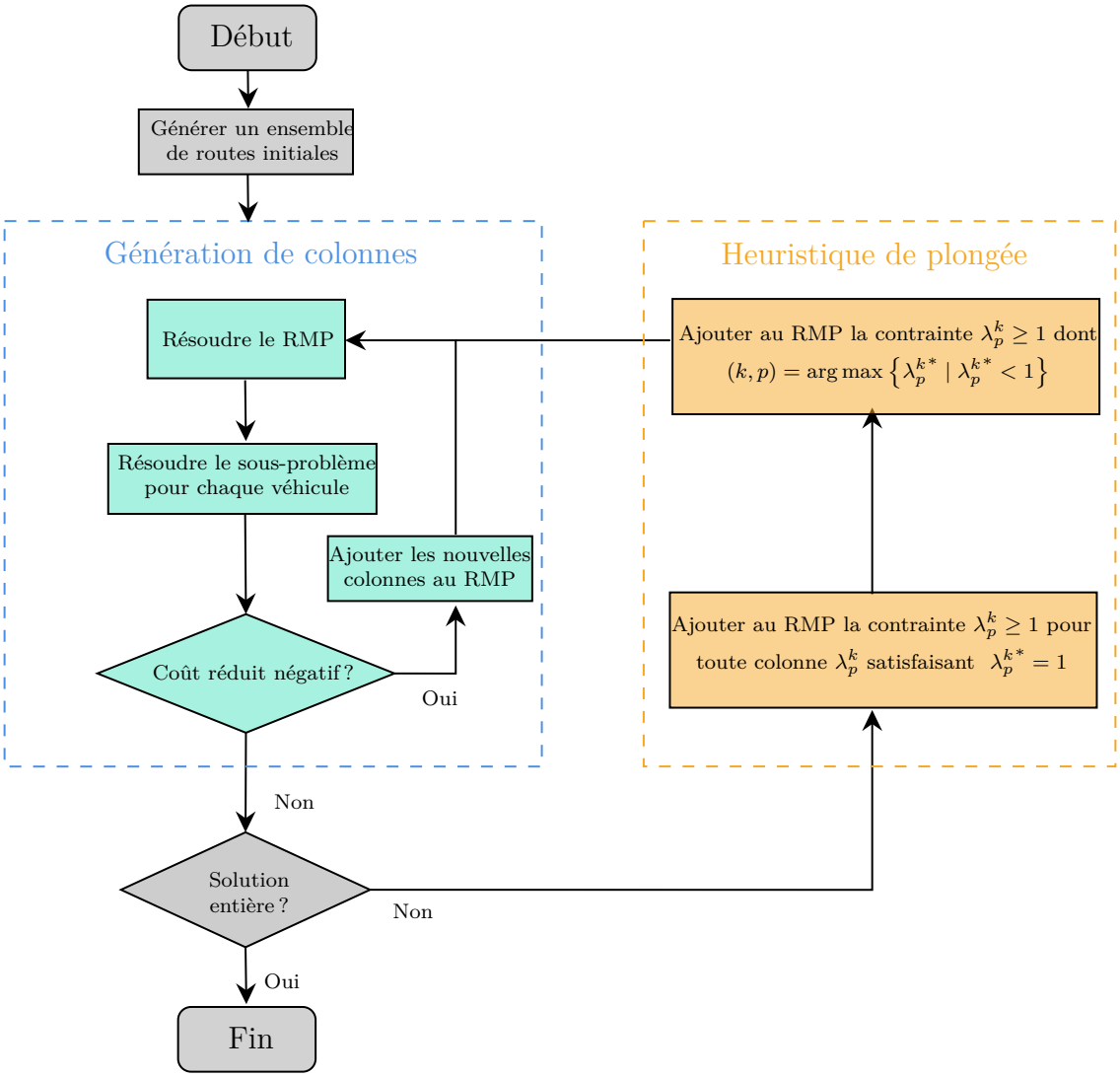


FIGURE 4.2 Organigramme de la procédure de résolution heuristique de plongée basée sur la génération de colonnes (CGDH)

4.2 Résolution du sous-problème

Même si le programme en nombres entiers du sous-problème $SPMIP$ peut être résolu à l'aide de techniques d'optimisation traditionnelles, nous pouvons appliquer une approche de programmation dynamique pour résoudre efficacement le sous-problème en le modélisant comme un problème de plus court chemin avec contraintes de ressources sur un réseau orienté.

4.2.1 Formulation générique du problème de plus court chemin avec contraintes de ressources

Lorsqu'aucune contrainte n'est imposée sur la présence de cycles, la résolution du sous-problème du véhicule k est équivalente à la résolution d'un problème de plus court chemin avec contraintes de ressources («*Shortest Path Problem with Resource Constraints*» - SPPRC) sur le graphe $\mathcal{G}_k$ associé [61]. Ce problème peut être énoncé comme suit : Étant donné un réseau avec une source et un puits composé d'un ensemble de nœuds connectés entre eux par des arcs auxquels sont associés des coûts ainsi qu'un ensemble de ressources dont la consommation est définie sur les arcs et dont la consommation totale est limitée sur chacun des nœuds, quel est le chemin ayant le coût minimum commençant à la source et se terminant au puits ? Ce type de problème permet d'identifier les routes à coût réduit strictement négatif qui peuvent améliorer la solution courante du problème maître restreint [61].

Une ressource est une quantité qui s'accumule le long des arcs parcourus par un chemin dans le réseau considéré. Si on prend le VRPTW comme exemple, les ressources dans ce cas sont la quantité totale livrée (ou récupérée) ainsi que le temps accumulé le long d'un chemin.

Notons $\mathcal{R}$ l'ensemble fini des R ressources considérées, où $R = |\mathcal{R}|$ et la consommation de la ressource $s \in \mathcal{R}$ sur un arc (i, j) est donnée par τ_{ij}^s. Chaque chemin commençant par le nœud source et terminant au nœud i, a un vecteur de consommation $\mathbf{r}_i \in \mathbb{R}^R$ indiquant la consommation totale de ressources au nœud i. La consommation totale de la ressource s est donnée sous forme générique par la fonction d'extension $f_{ij}^s(\mathbf{r}_i) : \mathbb{R}^R \longrightarrow \mathbb{R}$. Cette fonction d'extension procure une borne inférieure sur la valeur que peut prendre la consommation totale de chacune des ressources au nœud j après avoir visité l'arc (i, j). Il faut noter que le coût total est également considéré comme une ressource, car il s'agit d'une quantité qui s'accumule sur chaque arc. Cependant, il est traité séparément, car il est généralement non borné.

La consommation totale de chaque ressource est également bornée sur chaque nœud. Ces contraintes prennent la forme de fenêtres de consommation de ressources définies par une borne inférieure et une autre supérieure de la forme $[a_i^s, b_i^s]$ où $a_i^s, b_i^s \in \mathbb{N}$ pour chaque ressource s et chaque nœud i. Si nous avons atteint le nœud u et que la consommation est toujours inférieure à a_u^s, alors nous la fixons à a_u^s. Ceci permet de modéliser plusieurs concepts, comme l'attente si la ressource considérée est le temps cumulé et que les livraisons doivent avoir lieu dans un intervalle de temps spécifique. Ainsi, r_j^s la consommation totale de la ressource s au niveau du nœud j est bornée comme suit : $\tilde{a}_j^s \leq r_j^s \leq b_j^s$ où $\tilde{a}_j^s = \max_{i \in N^-(j)}\{f_{ij}^s(r_i^s), a_j^s\}$, et $N^-(j)$ est l'ensemble de nœuds dont j est l'un de leurs successeurs. Les notations de cette section sont inspirées de [63].

4.2.2 Formulation spécifique du problème de plus court chemin avec contraintes de ressources pour notre problème

Dans notre problème particulier, nous résolvons un problème SPPRC pour chaque véhicule sur un graphe $\mathcal{G}_k = (\mathcal{V}_k, \mathcal{A}_k)$ associé au véhicule $k \in \mathcal{K}$ initialement localisé à la gare s_k. Ce graphe est une version modifiée du graphe $G = (\mathcal{V}, \mathcal{A})$ présenté dans la section 3.2. Le graphe $\mathcal{G}_k$ contient deux nœuds supplémentaires, un nœud source global noté $Source$ et un nœud puits global noté $Puits$, donc $\mathcal{V}_k = \mathcal{V} \cup \{Source, Puits\}$. Il contient aussi un seul arc sortant de $Source$ vers $o^+_{s_k}$ le nœud dépôt de départ de la gare d'origine du véhicule k, ainsi qu'un arc de chaque nœud dépôt d'arrivée d'une gare vers $Puits$. Contrairement à G, dans $\mathcal{G}_k$, l'ensemble de nœuds $\mathcal{O}^+$ (nœuds de dépôts de gares) ne forme pas une clique, mais un graphe étoile dont le nœud $o^+_{s_k}$ est le nœud central, et est relié à chacun des autres nœuds de dépôts de départ d'autres gares par un arc sortant. Ainsi, l'ensemble d'arcs de $\mathcal{G}_k$ peut être exprimé formellement comme suit,

$$\begin{aligned}
\mathcal{A}_k = &\{(Source, o^+_{s_k})\} \cup \{(o^+_{s_k}, o^+_j) \mid j \neq s_k, j \in \mathcal{S}\} \cup \{(o^+_i, j) \mid i \in \mathcal{S}, j \in \mathcal{P}_i \cup \mathcal{D}_i\} \\
&\cup \{(o^+_i, o^-_i) \mid i \in \mathcal{S}\} \cup \left(\bigcup_{t \in \mathcal{S}} \{(i,j) \mid i \neq j \in \mathcal{P}_t\} \subset \mathcal{A}\right) \cup \left(\bigcup_{t \in \mathcal{S}} \{(i,j) \mid i \neq j \in \mathcal{D}_t\}\right) \\
&\cup \{(i,j) \mid i \in \mathcal{D}, j \in \mathcal{P}\} \cup \{(i, o^-_j) \mid i \in \mathcal{D}, j \in \mathcal{S}\} \cup \{(i, o^-_j) \mid i \in \mathcal{P}_j, j \in \mathcal{S}\} \\
&\cup \{(o^-_j, Puits), j \in \mathcal{S}\}.
\end{aligned}$$

La Figure 4.3 montre le réseau d'un sous-problème particulier de l'instance montrée dans la Figure 3.1, elle représente le graphe associé de SPPRC correspondant au sous-problème d'un véhicule k initialement situé à la gare 1.

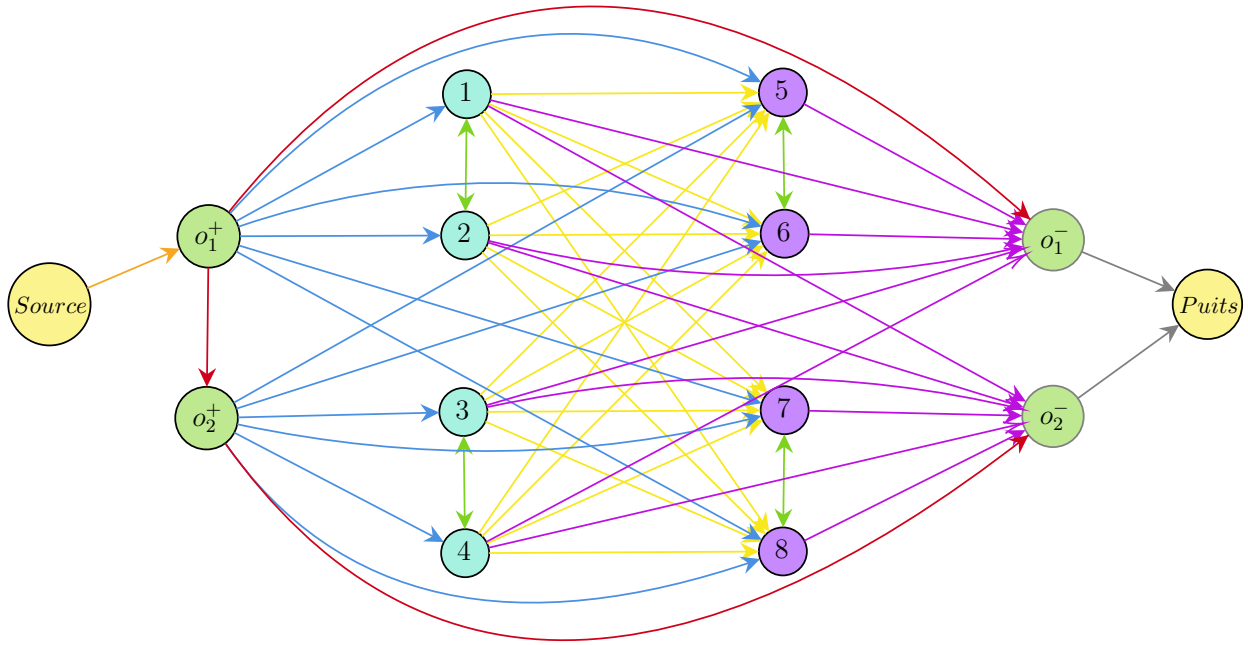


FIGURE 4.3 Illustration d'un réseau du SPPRC associé à un sous-problème d'un véhicule initialement situé à la gare 1. Consulter les Tableaux 4.1 et 4.2 pour les détails des couleurs utilisées.

Les coûts réduit $\overline{c}_{ij}$ sur chaque arc $(i,j) \in \mathcal{A}_k$ sont calculés à partir des valeurs des variables duales associées aux contraintes du problème maître restreint à l'itération précédente de l'algorithme de génération de colonnes, et sont définis comme ci-dessous.

- Le coût de l'arc unique sortant de *Source* vers $o^+_{s_k}$ qui est le nœud de dépôt de départ de la gare d'origine du véhicule k, à savoir l'arc $(Source, o^+_{s_k})$, est $\overline{c}_{Source,o^+_{s_k}} = -\sigma_k$.
- Les coûts des arcs sortant des nœuds de dépôts de départ $\mathcal{O}^+$, donnés par $(i,j) \in \mathcal{A}_k$ tels que $i \in \mathcal{O}^+$ et $j \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{O}^-$, sont $\overline{c}_{ij} = c_{ij}$.
- Les coûts d'arcs ayant un nœud de collecte ou de livraison comme une extrémité initiale, à savoir les arcs $(i,j) \in \mathcal{A}_k$, avec $i \in \mathcal{P} \cup \mathcal{D}$ et $j \in \mathcal{P} \cup \mathcal{D} \cup \mathcal{O}^-$, sont $\overline{c}_{ij} = c_{ij} - \pi_i$.
- Les coûts d'arcs sortant de nœuds de dépôts d'arrivée, donnés par $(o_i^-, Puits) \in \mathcal{A}_k$, $i \in \mathcal{S}$, sont $\overline{c}_{o_i^-,Puits} = 0$.

Dans notre cas, seules deux ressources sont nécessaires pour modéliser le sous-problème comme un problème SPPRC : la charge cumulée et le temps cumulé (passé). La ressource de la charge cumulée notée *cha* est utilisée pour suivre les charges sur les véhicules afin de ne pas dépasser les capacités disponibles. Tandis que la ressource de temps cumulé notée *tem* est

utilisée pour suivre le temps passé par les passagers dans les véhicules afin de ne pas dépasser le temps de trajet maximal pour chaque passager, ainsi que pour suivre l'heure d'arrivée des véhicules aux gares à la fin du trajet afin de ne pas dépasser les heures de départ. On considère donc $\mathcal{R} = \{cha, tem\}$, un ensemble de $|\mathcal{R}| = R = 2$ ressources.

Concernant la ressource de la charge cumulée *cha*, chaque passager $i \in \mathcal{P} \cup \mathcal{D}$ possède une demande/charge entière et positive q_i. La ressource de la charge cumulée est donc bornée par la fenêtre de consommation $[a_i^{cha}, b_i^{cha}]$ où $a_i^{cha} = q_i$ si $i \in \mathcal{P} \cup \mathcal{D}$, et $a_i^{cha} = 0$ sinon ; et où $b_i^{cha} = Q_k$ pour tout $i \in \mathcal{V}_k$. D'autre part, comme la charge de tous les passagers est positive, qu'il s'agisse d'un passager de collecte ou de livraison, alors un mécanisme de réinitialisation de la ressource de la charge cumulée *cha* doit être défini à chaque fois qu'un véhicule commence à servir des passagers de collecte. Pour cela, la définition de la consommation de la ressource *cha* sur l'arc $(i, j) \in \mathcal{A}_k$ suit la logique suivante :

- Soit l'arc $(i, j) \in \mathcal{A}_k$, si l'extrémité finale n'est pas un nœud de passager, c-à-d., $j \notin \mathcal{P} \cup \mathcal{D}$, alors la consommation de la ressource *cha* sur cet arc est $\tau_{ij}^{cha} = 0$.
- Sinon, si l'extrémité finale est un nœud de passager et l'extrémité initiale est soit un nœud de passager de même type soit pas un nœud de passager, c-à-d., soit $j \in \mathcal{D}$ et $i \notin \mathcal{P}$, soit $j \in \mathcal{P}$ et $i \notin \mathcal{D}$, alors la consommation de la ressource *cha* sur cet arc est $\tau_{ij}^{cha} = q_j$.
- Sinon, si l'extrémité finale est un nœud de passager et l'extrémité initiale est un nœud de passager de l'autre type, c-à-d., $j \in \mathcal{P}$ et $i \in \mathcal{D}$ (le cas $j \in \mathcal{D}$ et $i \in \mathcal{P}$ n'est pas considéré car aucun arc est défini de l'ensemble $\mathcal{P}$ vers l'ensemble $\mathcal{D}$), alors la consommation de la ressource *cha* sur cet arc est $\tau_{ij}^{cha} = -Q_k$. La définition de la consommation de la ressource *cha* sur les arcs de ce cas, fonctionne comme un mécanisme qui permet de réinitialiser la consommation de cette ressource sur le véhicule, à chaque fois que ce dernier commence à servir des passagers de collecte $\mathcal{P}$. En effet, si la consommation de *cha* sur un arc (i, j) est $-Q_k$, alors lorsque le véhicule atteint l'extrémité finale de cet arc qui est le premier passager de collecte dans la route du véhicule k, la consommation sera réinitialisée à q_j grâce à la contrainte de la fenêtre de consommation $[a_j^{cha}, b_j^{cha}]$.

Par conséquent,

$$\tau_{ij}^{cha} = \begin{cases} 0, & \text{si } j \notin \mathcal{P} \cup \mathcal{D}\,; \\ q_j, & \text{si } j \in \mathcal{D} \text{ et } i \notin \mathcal{P} \text{ ou } j \in \mathcal{P} \text{ et } i \notin \mathcal{D}\,; \\ -Q_k, & \text{si } j \in \mathcal{P} \text{ et } i \in \mathcal{D}. \end{cases}$$

Pour la ressource de temps cumulé *tem*, on sépare l'ensemble des nœuds en 4 catégories afin

de définir les fenêtres de consommation pour cette ressource.

- Pour $i \in \{Source, Puits\}$, il n'y a aucune restriction sur la consommation de tem, alors dans ce cas, on peut considérer $[a_i^{tem}, b_i^{tem}] = [0, \max_{h \in \mathcal{O}^-}\{T_h\}]$.
- Pour $i \in \mathcal{D}$, la seule restriction de temps associée aux passagers de livraison $\mathcal{D}$ est le temps de trajet maximum passé par le passager dans le véhicule. Ainsi, la borne supérieure de la fenêtre de consommation de la ressource tem doit être égale au temps de trajet maximum du passager concerné, c-à-d, $b_i^{tem} = L_i$. Cela implique que le temps cumulé entre le départ du véhicule de la gare et l'arrivée du véhicule au point de service du passager i sera toujours inférieur au temps de trajet maximum L_i. Alors, dans ce cas $[a_i^{tem}, b_i^{tem}] = [0, L_i]$.
- Pour les passagers de collecte $i \in \mathcal{P}$, deux restrictions de temps sont imposées, le temps de trajet maximum L_i et l'heure de disponibilité e_i. Rappelons que T_{u_i} est l'heure de départ du train associé à la gare correspondante au passager i (la gare où i souhaite atteindre), alors b_i^{tem} la borne supérieure de la fenêtre de consommation de tem au niveau de $i \in \mathcal{P}$ peut être définie comme $b_i^{tem} = T_{u_i} - t_{i,o_{u_i}^-}$, où $t_{i,o_{u_i}^-}$ est le temps nécessaire pour se déplacer directement de i vers $o_{u_i}^-$. D'autre part, pour garantir que le service du passager i ne commence qu'à/après sa date de disponibilité e_i, alors il faut s'assurer que $a_i^{tem} \geq e_i$. Par ailleurs, une manière pour assurer que le temps passé dans le véhicule par le passager i ne dépasse pas L_i, est de supposer que $a_i^{tem} \geq T_{u_i} - L_i$. En combinant les deux dernières hypothèses sur a_i^{tem}, on trouve $a_i^{tem} = \max\{e_i, T_{u_i} - L_i\}$. Par conséquent, pour $i \in \mathcal{P}$, on a $[a_i^{tem}, b_i^{tem}] = [\max\{e_i, T_{u_i} - L_i\}, T_{u_i} - t_{i,o_{u_i}^-}]$.
- Il reste l'ensemble des nœuds de dépôts de départ et d'arrivée $\mathcal{O}^+ \cup \mathcal{O}^-$. Ces nœuds correspondent aux gares, donc la seule restriction de temps au niveau de ces nœuds est les heures de départ des trains. Alors, pour $o_i^{\pm} \in \mathcal{O}^+ \cup \mathcal{O}^-$, on suppose que $b_i^{tem} = T_i$, ce qui garantit que les véhicules n'entrent jamais dans la gare i après l'heure de départ de son train associé. Ainsi, pour tout nœud $o_i^{\pm}$ de dépôt de départ/arrivée de la gare $i \in \mathcal{S}$, on a $[a_i^{tem}, b_i^{tem}] = [0, T_i]$.

La consommation de la ressource tem sur tous les arcs est égale au temps nécessaire pour parcourir l'arc plus la durée de service au niveau du nœud final de l'arc (l'addition de la durée de service est appliquée seulement si le nœud final est un nœud de passager de l'ensemble $\mathcal{P} \cup \mathcal{D}$), sauf pour l'arc sortant du nœud $Source$ et pour les arcs entrant dans le nœud $Puits$;

dans ce cas, la consommation de la ressource *temp* est nulle. Par conséquent,

$$\tau_{ij}^{tem} = \begin{cases} 0, & \text{si } i = Source \text{ ou } j = Puits\,; \\ t_{i,j} + d_j, & \text{sinon, si } j \in \mathcal{P} \cup \mathcal{D}\,; \\ t_{i,j}, & \text{sinon.} \end{cases}$$

Le Tableau 4.1 indique les informations sur chaque arc $(i,j) \in \mathcal{A}_k$ sous forme d'un vecteur ligne $(\overline{c}_{i,j}, \tau_{ij}^{cha}, \tau_{ij}^{tem})$ contenant respectivement, le coût réduit, la consommation de la ressource *cha*, et la consommation de la ressource *tem* associés à l'arc (i,j). Les couleurs des cases du Tableau 4.1 correspondent aux arcs associés du réseau représenté dans la Figure 4.3.

$i \in$ \ $j \in$	$\{Source\}$	$\{o_{s_k}^+\}$	$\mathcal{O}^+ \setminus \{o_{s_k}^+\}$	$\mathcal{D}$	$\mathcal{P}$	$\mathcal{O}^-$	$\{Puits\}$
$\{Source\}$	-	$(-\sigma_k, 0, 0)$	-	-	-	-	-
$\{o_{s_k}^+\}$	-	-	$(c_{ij}, 0, t_{ij})$	$(c_{ij}, q_j, t_{ij} + d_j)$	$(c_{ij}, q_j, t_{ij} + d_j)$	$(c_{ij}, 0, t_{ij})$	-
$\mathcal{O}^+ \setminus \{o_{s_k}^+\}$	-	-	-	$(c_{ij}, q_j, t_{ij} + d_j)$	$(c_{ij}, q_j, t_{ij} + d_j)$	$(c_{ij}, 0, t_{ij})$	-
$\mathcal{D}$	-	-	-	$(c_{ij} - \pi_i, q_j, t_{ij} + d_j)$	$(c_{ij} - \pi_i, -Q_k, t_{ij} + d_j)$	$(c_{ij} - \pi_i, 0, t_{ij})$	-
$\mathcal{P}$	-	-	-	-	$(c_{ij} - \pi_i, q_j, t_{ij} + d_j)$	$(c_{ij} - \pi_i, 0, t_{ij})$	-
$\mathcal{O}^-$	-	-	-	-	-	-	$(0, 0, 0)$
$\{Puits\}$	-	-	-	-	-	-	-

TABLEAU 4.1 Informations sur les arcs

Le Tableau 4.2 résume les informations sur les fenêtres de consommation de ressources sur chaque type de nœuds. Les couleurs des cases de chaque type de nœuds dans le Tableau 4.2 correspondent aux nœuds associés du réseau représenté dans la Figure 4.3.

Ressource \ $i \in$	$\{Source\}$	$\mathcal{O}^+$	$\mathcal{D}$	$\mathcal{P}$	$\mathcal{O}^-$	$\{Puits\}$
cha	$[0, Q_k]$	$[0, Q_k]$	$[q_i, Q_k]$	$[q_i, Q_k]$	$[0, Q_k]$	$[0, Q_k]$
tem	$[0, \max_{h \in \mathcal{O}^-}\{T_h\}]$	$[0, T_i]$	$[0, L_i]$	$[\max\{e_i, T_{u_i} - L_i\}, T_{u_i} - t_{i,o_{u_i}^-}]$	$[0, T_i]$	$[0, \max_{h \in \mathcal{O}^-}\{T_h\}]$

TABLEAU 4.2 Informations sur les fenêtres de consommation de ressources sur les nœuds

Dans notre problème particulier, on suppose que les fonctions d'extension $f_{ij}^s(\mathbf{r}_i)$ pour le coût réduit, la charge cumulée, et le temps cumulé sont décomposables par ressource. Elles peuvent être exprimées sous la forme $f_{ij}^s(\mathbf{r}_i) = r_i^s + \tau_{ij}^s$. Le coût réduit total (resp. le temps cumulé) d'une route est donc donné par la somme des coûts réduits (resp. des temps de déplacement) des arcs empruntés (resp. plus les temps de service des passagers visités), et la

charge cumulée est donnée par la somme des charges des passagers visités par cette route en tenant en compte le remise à jour avant les noeuds de collecte.

Afin de bien comprendre la complexité de SPPRC et l'importance d'ajouter des contraintes de ressources, nous renvoyons le lecteur intéressé à [64, Section 4.2.3] qui contient une comparaison bien détaillée entre le SPPRC et l'une de ses variantes simples bien connue : le problème du plus court chemin («*Shortest Path Problem*» - SPP).

4.2.3 Algorithmes d'étiquetage basés sur la programmation dynamique

Pour la résolution exacte des SPPRCs, les algorithmes d'étiquetage (*Labeling algorithm*) basés sur la programmation dynamique sont prédominants [61]. Dans les algorithmes d'étiquetage, une étiquette fait référence à un chemin partiel depuis le nœud source donné, qui est le nœud *Source* dans notre cas, jusqu'à un nœud $i \in \mathcal{V}_k$. Cette étiquette contient les informations sur la consommation de ressources le long de ce chemin et fait référence à son prédécesseur, l'étiquette du chemin partiel à partir duquel l'étiquette actuelle a été étendue. En commençant par le chemin partiel initial $p = (Source)$, les algorithmes d'étiquetage peuvent être décrits comme suit.

Algorithme 2 : Schéma général d'algorithmes d'étiquetage

1 sélectionner un chemin partiel non traité $p = (Source, \ldots, i)$;
2 **pour** $j \in N^+(i)$ **faire**
3 étendre le chemin partiel p en utilisant l'arc (i, j);
4 **si** *le nouveau chemin partiel* $(p, j) = (Source, \ldots, i, j)$ *est réalisable* **alors**
5 stocker le nouveau chemin partiel (p, j) afin de pouvoir l'étendre ultérieurement;

Les algorithmes d'étiquetage permettent d'effectuer les étapes de l'Algorithme 2 de manière implicite à l'aide d'attributs spécifiques, qui sont les valeurs de ressources de l'étiquette. En particulier, l'étape d'extension (3) propage directement les étiquettes à l'aide de fonctions d'extension de ressources (*«Resource Extension Functions»* - REFs) ; pour plus d'informations sur ces fonctions, voir l'article de synthèse de Desaulniers et al. [65]. La répétition des étapes de l'Algorithme 2 crée tous les chemins possibles commençant au noeud *Source*. Des procédures de dominance sont aussi invoquées pour identifier et éliminer les chemins partiels et leurs étiquettes qui ne peuvent conduire à une solution optimale de SPPRC.

Le schéma d'étiquetage de l'Algorithme 2 est connu comme un algorithme d'étiquetage monodirectionnel avant (*monodirectional forward labeling algorithm*), car il étend les chemins partiels uniquement dans le sens avant. Pour de nombreux types de SPPRC, la direction de

propagation peut être inversée, ce qui donne suite à un autre schéma d'étiquetage souvent appelé un algorithme d'étiquetage monodirectionnel arrière monodirectionnel (*monodirectional backward labeling algorithm*). Cela signifie que le premier chemin partiel est $p = (Puits)$, où $Puits$ est le nœud du puits dans notre cas, et les chemins partiels $P = (j, \ldots, Puits)$ sont ensuite étendus par rapport à l'orientation d'un arc (i, j) créant de nouveaux chemins partiels $p' = (i, j, ..., Puits)$. L'étiquetage monodirectionnel arrière ne peut être calculé que si un inverse de chaque REF est défini. Pour plus de détails sur les propriétés des REFs, nous renvoyons le lecteur à [66].

Il faut également mentionner qu'il existe un autre type d'étiquetage appelé l'étiquetage bidirectionnel, introduit pour la première fois par Righini et Salani en 2006 [67], où ils ont présenté un étiquetage bidirectionnel afin d'éviter l'explosion d'étiquettes qui peut être rencontrée lors de la mise en œuvre d'un schéma d'étiquetage monodirectionnel. Dans l'étiquetage bidirectionnel, des chemins partiels sens avant et sens arrière sont créés, mais traités uniquement jusqu'à un point dit à mi-chemin (*half-way point*). Le point à mi-chemin est défini à l'aide d'une ressource monotone (par exemple, dans notre cas, la ressource de temps cumulé *tem* est une ressource monotone, tandis que la ressource de charge cumulée *cha* ne l'est pas), souvent comme le point médian de son domaine réalisable. Lorsque l'étiquetage se termine, des étiquettes sens avant et arrière appropriées doivent être fusionnées pour obtenir des chemins $Source - Puits$ réalisables complets. Consulter [68] pour les applications des algorithmes d'étiquetage bidirectionnel pour la résolution de plusieurs problèmes, tels que le problème des tournées des véhicules électriques avec des fenêtres de temps ainsi que les problèmes des tournées des véhicules avec l'ordonnancement des chauffeurs de camions.

Pour notre problème, on a implémenté un algorithme d'étiquetage monodirectionnel avant pour générer les routes. Plus de détails sur cette implémentation sont donnés ci-dessous.

4.2.4 Adaptation d'un algorithme d'étiquetage monodirectionnel avant pour résoudre les sous-problèmes

Comme pour les algorithmes d'étiquetage standard, chaque chemin partiel $p = (Source, \ldots, i)$ est associé à une étiquette unique $\mathscr{L}_i = (\mathscr{L}^{\text{coût}}, \mathscr{L}^{cha}, \mathscr{L}^{tem})$ correspond à un chemin commençant au nœud $Source$ et se terminant au nœud i, et qui possède les attributs (ressources) suivants :

— $\mathscr{L}^{\text{coût}}$ le coût réduit cumulé jusqu'au nœud i ;

— $\mathscr{L}^{cha}$ la charge cumulée jusqu'au nœud i ;

— $\mathscr{L}^{tem}$ le temps cumulé jusqu'au nœud i.

Notons $\rho(\mathscr{L}_i)$ l'étiquette parent de $\mathscr{L}_i$, ce qui est nécessaire pour reconstruire le chemin entre *Source* et i. Aussi, notons $|p|$ la longueur de $p = (p_0, p_1, \ldots, p_{|p|-1})$, c-à-d., le nombre de nœuds visités par p. Alors, dans notre problème particulier, on sait bien que $p_0 = Source$ et que si $|p| > 1$, alors $p_1 = o^+_{s_k}$ forcément, car le seul arc sortant de *Source* est celui qui relie *Source* avec le nœud dépôt de départ de la gare d'origine du véhicule k, notée s_k.

On décrit ensuite l'extension d'une étiquette $\mathscr{L}_i$ appartenant à un chemin partiel réalisable $p = (Source, \ldots, i)$ le long d'un arc $(i, j) \in \mathcal{A}_k$ jusqu'au nœud j. Les REFs créent une nouvelle étiquette $\mathscr{L}_j = (\overline{c}(\mathscr{L}_j), \mathscr{L}_j^{cha}, \mathscr{L}_j^{tem})$ ayant comme étiquette parent $\rho(\mathscr{L}_j) = \mathscr{L}_i$ et comme attributs les valeurs suivantes :

$$\overline{c}(\mathscr{L}_j) = \overline{c}(\mathscr{L}_i) + \overline{c}_{ij}; \tag{4.6}$$

$$\mathscr{L}_j^{cha} = \max\{a_j^{cha}, \mathscr{L}_i^{cha} + \tau_{ij}^{cha}\}; \tag{4.7}$$

$$\mathscr{L}_j^{tem} = \max\{a_j^{tem}, \mathscr{L}_i^{tem} + \tau_{ij}^{tem}\}. \tag{4.8}$$

Noter que le mécanisme de la réinitialisation de la ressource *cha* lorsqu'un véhicule commence à desservir les nœuds de $\mathcal{P}$, peut être défini à l'aide des REFs. Cela se fait comme suit : au lieu de mettre les consommations négatives $-Q_k$ sur certains arcs, on suppose que $\tau_{ij}^{cha} = q_j$ si $j \in \mathcal{P} \cup \mathcal{D}$ et $\tau_{ij}^{cha} = 0$ sinon, et on modifie la REF (4.7) par

$$\mathscr{L}_j^{cha} = \begin{cases} \mathscr{L}_i^{cha} + \tau_{ij}^{cha}, & \text{si } j \notin \mathcal{P}\,; \\ \tau_{ij}^{cha}, & \text{sinon.} \end{cases} \tag{4.9}$$

La nouvelle étiquette $\mathscr{L}_j$, pour $j \in \mathcal{V}_k \setminus \{Source, Puits\}$ et le chemin partiel associé $(p, j) = (0, \ldots, i, j)$ sont réalisables si et seulement si les conditions suivantes sont vérifiées :

$$\mathscr{L}_j^{cha} \leq b_j^{cha}, \tag{4.10}$$

$$\mathscr{L}_j^{tem} \leq \begin{cases} b_j^{tem} + \tau_{s_k,p_2}^{tem}, & \text{si } |p| \geq 3 \text{ et } p_2 \in \mathcal{O}^+ \text{ et } j \in \mathcal{D}; \\ b_j^{tem}, & \text{sinon.} \end{cases} \tag{4.11}$$

La Condition (4.10) est claire, tandis que la Condition (4.11) peut nécessiter plus d'explications. Rappelons que dans notre problème particulier, le véhicule k est non seulement censé pouvoir servir les passagers allant de/vers sa gare d'origine notée s_k, mais également les pas-

sagers d'autres gares, en particulier, les requêtes/passagers de type livraison d'autres gares. Cette hypothèse peut affecter la vérification du temps cumulé tem lorsque le véhicule k dessert les passagers de livraison d'une autre gare, car le temps passé entre la gare d'origine du véhicule k et la gare où les passagers de livraison attendent, ne doit pas être pris en compte lors de la vérification du temps de trajet maximal de ces passagers de livraison.

En d'autres termes, si le véhicule k est censé servir le passager $j \in \mathcal{D}$ situé à une gare s autre que s_s, alors lorsque le véhicule k atteint le nœud de service du passager j, le temps passé lorsque le véhicule k s'est déplacé de s_k à s, doit être soustrait du temps cumulé tem lors de la comparaison de ce dernier avec b_j^{tem} la borne supérieure de la fenêtre de consommation de ressources tem au nœud de service du passager j, qui est également égale à son temps de trajet maximal. Parce que le passager j est monté dans k au moment où k a visité o_s^+, qui est le moment à partir duquel on devrait commencer à compter le temps passé par le passager j dans le véhicule k.

Cette situation se produit uniquement lorsque p contient au moins 3 nœuds, tel que le troisième nœud visité par p est un nœud dépôt de départ d'une gare ($p_2 \in \mathcal{O}^+$), et le dernier nœud visité par j est un nœud de livraison ($j \in \mathcal{D}$). Dans ce cas, p est réalisable si et seulement si la Condition (4.10) est vérifiée et $\mathscr{L}_j^{tem} - \tau_{s_k,p_2}^{tem} \leq b_j^{tem} = \mathscr{L}_j$.

La Figure 4.4 présente un exemple d'un chemin partiel $p = (Source, o_{s_k}^+, o_s^+, j)$ coloré en rouge, associé à un véhicule k initialement localisé à la gare s_k. Selon ce chemin partiel, le véhicule k s'est déplacé de sa gare d'origine s_k à la gare s en passant par l'arc ondulé $(o_{s_k}^+, o_s^+)$, et cela pour récupérer le passager de type livraison j de la gare s et le transporter vers son nœud de service j.

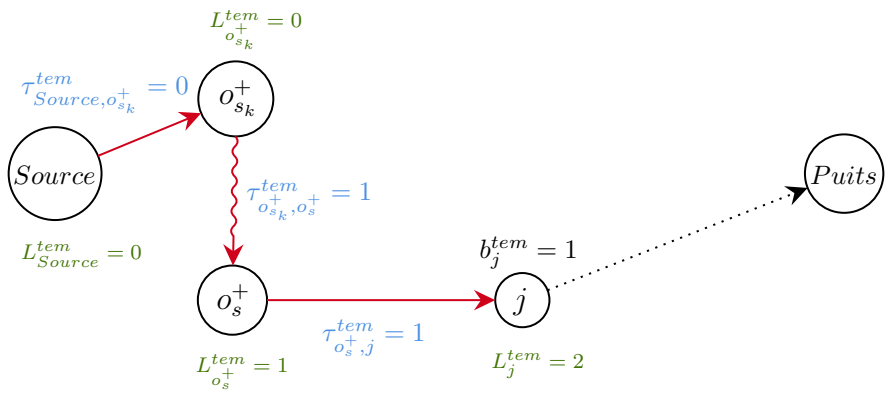


FIGURE 4.4 Exemple d'un chemin partiel du véhicule k desservant un passager de type livraison d'une autre gare

Comme on peut le voir, $\mathscr{L}_j^{tem} = 2 > b_j^{tem} = 1$, ce qui signifie que le chemin partiel p peut

être considéré comme un chemin partiel irréalisable. Cependant, ce n'est pas vrai, car le passager j n'était pas dans le véhicule k lorsque ce dernier a passé l'arc $(o^+_{s_k}, o^+_s)$. D'où, au lieu de comparer $\mathscr{L}^{tem}_j$ avec b^{tem}_j, on devrait comparer $\mathscr{L}^{tem}_j - \tau_{o^+_{s_k}, o^+_s}$ avec b^{tem}_j. Dans ce cas, on peut vérifier que p est réalisable (pour le temps de trajet maximal de j) car $\mathscr{L}^{tem}_j - \tau_{o^+_{s_k}, o^+_s} = 1 \leq b^{tem}_j$.

Les règles de dominance permettent d'éliminer les chemins non-optimaux ou redondants lors de la recherche. Ces règles contribuent à réduire la complexité des calculs en éliminant les chemins ne conduisant pas à une solution optimale. Dans notre cas, on utilise la règle de dominance basique, énoncée comme suit. Soient $\mathscr{L}$ et $\mathscr{L}'$ deux étiquettes de chemins partiels différents p et p', respectivement, qui se terminent au même nœud i. On dit que $\mathscr{L}$ domine $\mathscr{L}'$ si chaque ressource utilisée dans $\mathscr{L}$ est inférieure ou égale à $\mathscr{L}'$, c-à-d., si toutes les conditions suivantes sont vérifiées :

$$\mathscr{L}^{\text{coût}} \leq \overline{c}(\mathscr{L}') \tag{4.12}$$

$$\mathscr{L}^{cha} \leq \mathscr{L}'^{cha} \tag{4.13}$$

$$\mathscr{L}^{tem} \leq \mathscr{L}'^{tem} \tag{4.14}$$

Les étiquettes dominées ne feront pas partie de la solution finale et peuvent donc être éliminées de l'analyse ultérieure. Si il n'y a aucune dominance entre $\mathscr{L}$ et $\mathscr{L}'$, alors les deux étiquettes sont incomparables, ce qui signifie qu'on ne peut pas dire que l'une est meilleure que l'autre.

L'étiquette du chemin initial pour $p = (Source)$ est définie comme $\mathscr{L}_{Source} = (0, 0, 0)$, où $\rho(\mathscr{L}_{Source})$ est non défini, car l'étiquette initiale $\mathscr{L}_{Source}$ n'a aucune étiquette parent associée. Le SPPRC associé au sous-problème du véhicule k peut être résolu en utilisant l'algorithme d'étiquetage monodirectionnel avant basé sur le pseudo-code présenté dans l'Algorithme 3.

Algorithme 3 : Pseudo-code de l'algorithme d'étiquetage monodirectionnel avant utilisé pour générer les routes réalisable à coût réduit négatif

Données : Graphe $\mathcal{G}_k$

1 $\mathcal{U} = \{\mathscr{L}_{Source}\}$;
2 $\mathcal{L}_i = \emptyset$ pour tout nœud i ;
3 **tant que** $\mathcal{U} \neq \emptyset$ **faire**
4 sélectionner $\mathscr{L}_i$ une étiquette dans $\mathcal{U}$;
5 $\mathcal{U} = \mathcal{U} \setminus \{\mathscr{L}_i\}$;
6 **si** *aucune étiquette dans* $\mathcal{L}_i$ *domine* $\mathscr{L}_i$ **alors**
7 **si** $i \neq Puits$ **alors**
8 $\mathcal{L}_i = \mathcal{L}_i \cup \{\mathscr{L}_i\}$;
9 **sinon**
10 **si** $\overline{c}(\mathscr{L}_{Puits}) < 0$ **alors**
11 $\mathcal{L}_{Puits} = \mathcal{L}_{Puits} \cup \{\mathscr{L}_{Puits}\}$;
12 Étendre $\mathcal{L}_i$ le long de tous les arcs sortant de i;
13 Ajouter toutes les extensions réalisables à $\mathcal{U}$;
14 trier les éléments de $\mathcal{L}_{Puits}$ de manière croissante selon leurs coûts réduits associés;
15 **retourner** *les chemins (routes) correspondants aux premières* $MaxCol$ *étiquettes dans* $\mathcal{L}_{Puits}$.

Dans la ligne 1 de l'Algorithme 3, l'étiquette initiale $\mathscr{L}_{Source}$ associée au nœud source $Source$ est créée et insérée dans $\mathcal{U}$. Ici, $\mathcal{U}$ désigne l'ensemble des étiquettes non encore traitées et $\mathcal{L}_i$ est l'ensemble des étiquettes déjà traitées au niveau du nœud i (chemins « partiels » se terminant au nœud i). Les lignes 4 à 13 sont répétées tant qu'il y a des étiquettes non encore traitées. À la ligne 4, une nouvelle étiquette non traitée est sélectionnée en choisissant la première étiquette dans $\mathcal{U}$. Cette étiquette choisie est ensuite supprimée de $\mathcal{U}$ à la ligne 5. La ligne 5 vérifie si l'étiquette $\mathscr{L}_i$ peut être supprimée à l'aide des règles de dominance (4.12)-(4.14). Si l'étiquette $\mathscr{L}_i$ ne peut pas être supprimée (elle est donc non dominée), elle est stockée dans $\mathcal{L}_i$ l'ensemble des étiquettes traitées au niveau du nœud i à la ligne 7 seulement si ce dernier n'est pas le nœud $Puits$. Sinon, si i est le nœud $Puits$, alors seulement les étiquettes $\mathscr{L}_{Puits}$ ayant un coût réduit négatif sont stockées dans $\mathcal{L}_{Puits}$. À la ligne 11, de nouvelles étiquettes sont créées en étendant l'étiquette $\mathscr{L}_i$ le long de tous les arcs $(i, j) \in \mathcal{A}_k$ en utilisant les REFs (4.6)-(4.8). Les étiquettes résultantes sont réalisables seulement si elles satisfont les conditions (4.10)-(4.11). Toutes les étiquettes correspondant aux extensions réalisables de l'étiquette $\mathscr{L}_i$ sont insérées dans $\mathcal{U}$ dans la ligne 13. Les étiquettes de $\mathcal{L}_{Puits}$ sont ensuite triées par ordre croissant en fonction de leurs coûts réduits dans la ligne 14. Enfin, dans la dernière ligne 15, les

$MaxCol$ premières étiquettes de l'ensemble ordonné $\mathcal{L}_{Puits}$ sont renvoyées. Si $\mathcal{L}_{Puits}$ contient moins de NCI étiquettes, alors toutes ses étiquettes seront renvoyées. Ici $MaxCol$, tel qu'il est défini à la Section 4.1.3, est un paramètre fixé par l'utilisateur représentant le nombre maximum de colonnes qui peuvent être ajoutées au MP chaque fois qu'un sous-problème est résolu.

L'Algorithme 3 est implémenté dans la ligne 4 de l'Algorithme 1 pour résoudre les sous-problèmes.

Les arcs dans notre $SPPRC$ sont pondérés à l'aide de variables duales, ce qui peut entraîner des poids négatifs dans le réseau du $SPPRC$. Ainsi, comme notre objectif est de résoudre les sous-problèmes de manière optimale, nous considérons uniquement des solutions élémentaires, où chaque nœud est visité au plus une fois.

Pour garantir une élimination complète des sous-tours, nous avons utilisé la méthode proposée par Beasley et Christofides [69], dans laquelle une ressource binaire supplémentaire est introduite pour chaque nœud $i \in \mathcal{V}_k$. Cette nouvelle ressource servira à vérifier si un nœud a été visité ou non.

CHAPITRE 5 Résultats numériques

Dans ce chapitre, nous commençons par présenter les données, les paramètres généraux et la procédure de génération d'instances utilisées pour tester notre modèle mathématique et notre méthode de résolution. Ensuite, nous exposons les résultats obtenus afin d'évaluer la performance de cette dernière, ainsi que la qualité des solutions fournies, à l'aide d'un ensemble d'indices de performance.

5.1 Paramètres Généraux

Tous les tests ont été effectués sur un ordinateur avec un processeur AMD Ryzen 5 3500U avec 2,1 GHz et 12 GB de RAM. Le modèle mathématique MIP et l'algorithme $CGDH$ ont été implémentés en Python, où :

- les modèles mathématiques linéaires MIP et le problème maître MP sont résolus à l'aide de la bibliothèque Python-MIP [1] et du solveur Gurobi [2] ;
- les algorithmes utilisés pour résoudre le SPPRC ont été implémentés à l'aide de la bibliothèque CSPY [70] ;
- une visualisation de solutions est fournie à l'aide d'un code basé sur la bibliothèque Matplotlib [3] ;
- aucune limite n'est imposée sur le nombre total de colonnes générées au cours de l'algorithme $CGDH$ (le paramètre $MaxCol$ est considéré comme suffisamment grand) ;
- aucune limite n'est imposée sur le nombre total d'itérations effectuées au cours de l'algorithme $CGDH$ (le paramètre $LimIte$ est considéré comme suffisamment grand).

Comme notre problème est nouveau et n'a jamais été abordé dans la littérature, il n'existe pas d'instances de la littérature à utiliser pour la comparaison. Ainsi, nous utilisons un ensemble d'instances générées pour tester le modèle MIP et la méthode de résolution $CGDH$. Notre expérimentation a été réalisée sur 40 instances réparties en 4 catégories selon le nombre de gares dans chaque instance, variant entre 2 et 5 gares, et générées en utilisant la procédure de génération décrite dans la section suivante. Le temps d'exécution est limité à 20 minutes pour toutes les instances testées.

La Figure 5.1 montre une solution optimale trouvée après l'implémentation du MIP. Cette figure est une capture de la visualisation fournie par le code pour illustrer les solutions, où les

1. `https://www.python-mip.com/`
2. `https://www.gurobi.com/`
3. `https://matplotlib.org/`

cercles, les carrés et les triangles sont respectivement des gares, des demandes de livraisons et des demandes de collecte. Une gare et ses requêtes associées ont toutes la même couleur. Un véhicule utilisé est représenté par une route ayant la même couleur que la gare d'origine du véhicule. Les informations sur les nœuds circulaires sont : T, l'heure de départ du train de la gare associée (donnée) ; tandis que les informations sur les nœuds carrés et triangulaires sont : e, l'heure de disponibilité du passager (donnée) ; B, l'heure d'arrivée du véhicule au nœud (variable) ; L, le temps maximal du trajet du passager (donnée) ; et R, le temps passé dans le véhicule par passager (variable). Le rectangle rouge contient des informations sur les véhicules utilisés, où chaque ligne contient respectivement le style et la couleur de la route attribuée, les parenthèses contenant la gare d'origine et l'ID (l'identité) du véhicule, la capacité du véhicule (données), le nombre de passagers servis (variable) et l'heure d'arrivée du véhicule à la gare finale dans sa route (variable). Cette solution est la solution optimale de l'instance Ins12(G3D6P6K4), l'une des instances testées dans le Tableau 5.1.

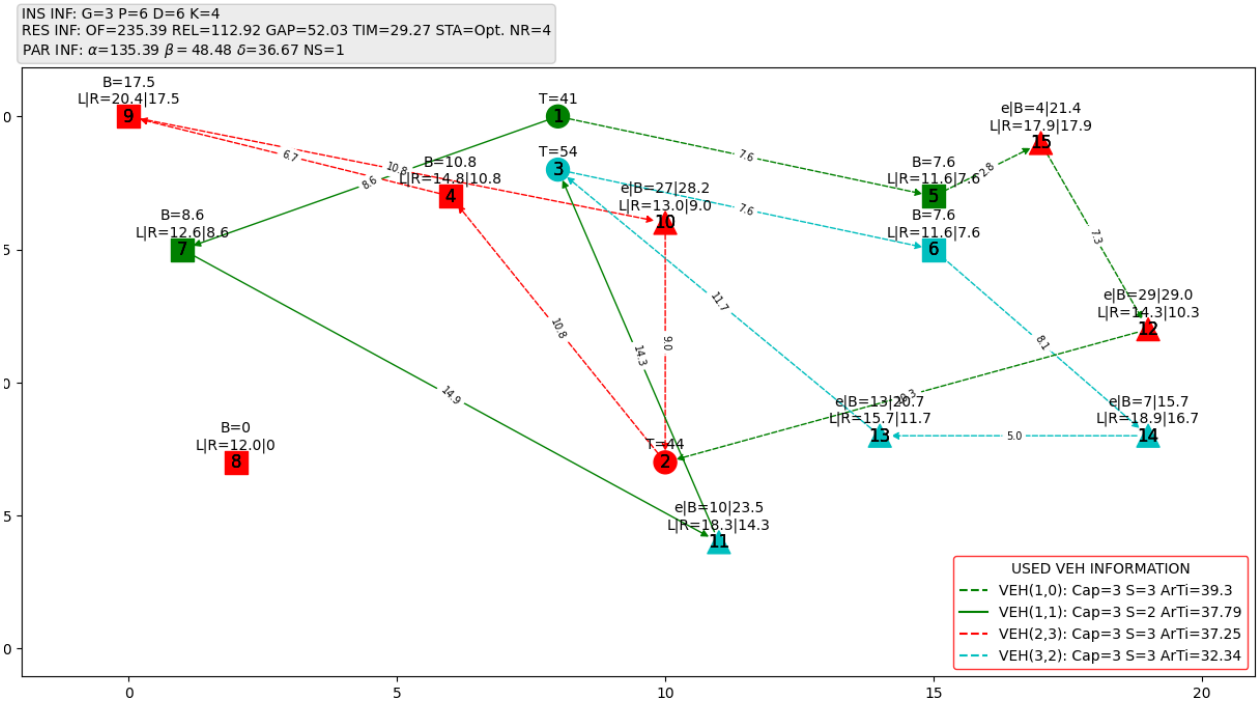


FIGURE 5.1 Visualisation d'une solution trouvée après l'exécution du code

5.2 Génération d'instances

Afin de tester notre méthode de résolution $CGDH$ et le modèle mathématique MIP et d'évaluer leur efficacité, nous avons créé plusieurs instances en utilisant un générateur personnalisé pour notre problème. Les réseaux sont générés en plaçant des sommets aléatoirement sur une

grille carrée de côté $l = 20$. Tous les paramètres des instances générées sont considérés comme des entiers. Le choix de paramètres aléatoires suit une distribution uniforme notée $U[a, b]$, où a et b sont respectivement la plus petite et la plus grande valeur que peut prendre le paramètre aléatoire. Le générateur est configuré à l'aide des hypothèses et paramètres suivants.

- La seule donnée nécessaire pour démarrer le générateur est le nombre de gares $|S|$;
- le nombre de passagers de collecte $|\mathcal{P}|$ (resp. livraison $|\mathcal{D}|$) est choisi aléatoirement dans $U[|\mathcal{S}|, 5 \times |\mathcal{S}|]$;
- le nombre de véhicules $|\mathcal{K}|$ est choisi aléatoirement dans $U[\frac{|\mathcal{P}|+|\mathcal{D}|}{3}, 2\frac{|\mathcal{P}|+|\mathcal{D}|}{3}]$;
- la gare associée à un passager et la gare d'origine d'un véhicule sont choisies aléatoirement sur l'ensemble $\mathcal{S}$;
- les coordonnées (x, y) de chaque gare, passager de collecte, et passager de livraison sont choisies aléatoirement dans $U[0, l] \times U[0, l]$;
- la distance entre deux nœuds i et j est calculée à l'aide de la distance euclidienne $\|\mathbf{i} - \mathbf{j}\|$;
- la vitesse est la même pour tous les véhicules ;
- le temps t_{ij}^k entre deux nœuds i et j est considéré comme égal à la distance entre eux ;
- les heures de départ des trains sont choisies aléatoirement dans $U[diag, 2 \times diag]$, où $diag = \sqrt{l^2 + l^2} = \sqrt{2}l$ est la longueur de la diagonale de la grille ;
- les charges des passagers sont unitaires, $q_i = 1$ pour tous $i \in \mathcal{P} \cup \mathcal{D}$;
- les capacités des véhicules sont choisies aléatoirement dans $U[\frac{|\mathcal{P}|+|\mathcal{D}}{|\mathcal{K}|}, 2\frac{|\mathcal{P}|+|\mathcal{D}}{|\mathcal{K}|}]$;
- la déviation autorisée est $\Delta = l/5$;
- les heures de disponibilités des passagers de collecte e_i sont choisies aléatoirement dans $U[0, T_{u_i} - t_{i,u_i}]$, où u_i est la gare de destination de i ;
- le coût de ne pas servir le passager $i \in \mathcal{P} \cup \mathcal{D}$ est $w_i = 100$;
- les coûts de déplacement sont égaux aux temps de déplacement, $c_{ij}^k = t_{ij}^k$ pour tous $i, j \in \mathcal{S} \cup \mathcal{P} \cup \mathcal{D}$, $k \in \mathcal{K}$.

5.3 Résultats

Comme aucune méthode de résolution n'a été développée auparavant pour notre problème, nous analysons ainsi la performance de l'algorithme de résolution $CGDH$ et la qualité du modèle mathématique MIP en utilisant les bornes inférieures et supérieures.

Les résultats numériques sont regroupés selon le nombre de gares $|\mathcal{S}|$, et pour chaque ensemble d'instances ayant le même nombre de gares, les instances sont ordonnées selon le nombre de

passagers à servir $|\mathcal{P} \cup \mathcal{D}|$. Les détails des performances de MIP et $CGDH$ pour chaque instance testée sont présentés dans les Tableaux 5.1, dont TL signifie que la limite de temps d'exécution de 20 minutes a été atteinte, et NT signifie qu'aucune solution réalisable n'a été trouvée pendant le temps d'exécution limité. Les colonnes de ce tableau sont données par :

- L'ensemble des quatre colonnes sous «Informations sur l'instance» contient les informations sur l'instance testée, où :
  - Nom : le nom de l'instance dont le chiffre après Ins, G, D, P, et K, est l'identifiant de l'instance, le nombre de gares $|\mathcal{S}|$, le nombre de passagers de livraison $|\mathcal{D}|$, le nombre de passagers de collecte $|\mathcal{P}|$, et le nombre de véhicules disponibles $|\mathcal{K}|$, respectivement.
  - N : le nombre total de passagers dans l'instance $|\mathcal{P} \cup \mathcal{D}|$.
  - TC : la capacité totale disponible $\sum_{k \in \mathcal{K}} Q_k$.
  - F : la différence entre le nombre total de passagers et la capacité totale $N - TC$.
- L'ensemble des quatre colonnes sous «MIP» et «$CGDH$» contient respectivement les informations de l'exécution du modèle MIP et l'algorithme $CGDH$, où :
  - UB : la valeur de la fonction objectif de la meilleure solution entière trouvée, qui est également considérée comme une borne supérieure sur la valeur optimale. Les valeurs indiquées en **gras** les meilleures UB.. Le symbole * dans les UB de $CGDH$ indique que l'optimalité de la borne est prouvée à travers la comparaison avec la borne inférieure correspondante.
  - LB : la valeur optimale de la relaxation linéaire du MIP, et aussi la valeur optimale du MP au nœud racine de $CGDH$. C'est également considérée comme une borne inférieure sur la valeur optimale du problème.
  - $GAP\%$: le pourcentage de l'écart d'intégrité (*Integrality gap*) calculé par $GAP = \frac{UB-LB}{UB}$.
  - $Tem.(s)$: le temps d'exécution en **secondes** effectué pour trouver la solution correspondante.
  - NS : le nombre de passagers non servis dans la solution trouvée. Les valeurs indiquées en rouge indiquent les NS minimaux.

Noter que dans certaines instances, il se peut que la capacité totale soit strictement inférieure au nombre total de passagers, c-à-d., $TC < N$. Mais cela ne signifie pas que nous n'avons pas une capacité totale suffisante pour servir tous les passagers. En raison du principe des *Backhauls*, les seuls cas où nous pouvons être sûrs que certains passagers ne seront jamais servis sont les cas $TC < |\mathcal{D}|$ ou $TC < |\mathcal{P}|$.

Par contre, même si $TC < N$, il se peut que tous les passagers puissent être servis sans dépasser la capacité totale, ou que le nombre total de passagers servis puisse dépasser la capacité totale sans violer la contrainte de capacités des véhicules. On peut voir cela dans les solutions fournies par $CGDH$ des instances Ins7, Ins18, Ins20, Ins23, Ins27, et Ins28. Par exemple, dans la solution fournie par $CGDH$ (et le MIP) de Ins7, on a $NS = 0$, malgré que $D = N - TC = 2 > 0$. Ainsi, comme on ne peut pas fournir une borne inférieure valide pour NS, alors F est utilisé comme référence pour évaluer la valeur de NS pour les solutions trouvées. Lorsque NS est proche ou inférieur à F (ou encore lorsque la quantité $NS - D$ est proche de zéro ou négative), le nombre de passagers non servis (ou de passagers servis) est relativement bon.

	Informations sur l'instance				*MIP*					*CGDH*				
	Nom	N	TC	F	UB	LB	$Gap\%$	$Tem.(s)$	NS	UB	LB	$Gap\%$	$Tem.(s)$	NS
2 Gares	Ins0(G2D2P3K1)	5	5	0	**144.27**	31.72	78.01	0.08	1	**144.27***	144.27	0	0.02	1
	Ins1(G2D3P2K1)	5	5	0	**139.54**	32.96	76.38	0.05	1	**139.54***	139.54	0	0.02	1
	Ins2(G2D3P4K2)	7	6	1	**163.19**	75.58	53.69	0.1	1	**163.19***	163.19	0	0.03	1
	Ins3(G2D3P7K3)	10	9	1	**459.83**	81.05	82.37	6.61	4	**459.83***	459.83	0	0.11	4
	Ins4(G2D5P7K4)	12	12	0	**149.86**	76.36	49.05	15.14	0	**149.86***	149.86	0	0.15	0
	Ins5(G2D6P7K4)	13	12	1	**214.88**	100.24	53.35	84.43	1	**214.88***	214.88	0	0.13	1
	Ins6(G2D5P9K4)	14	12	2	**335.4**	108.05	67.78	TL	2	**335.4***	335.4	0	1	2
	Ins7(G2D5P9K4)	14	12	2	**99.81**	79.06	20.79	4.92	0	**99.81***	99.81	0	0.3	0
	Ins8(G2D10P7K5)	17	15	2	**558.25**	122.63	78.03	TL	4	**558.25***	558.25	0	0.37	4
	Ins9(G2D9P9K6)	18	18	0	**147.32**	105.04	28.7	141.68	0	**147.32***	147.32	0	0.55	0
3 Gares	Ins10(G3D7P5K4)	12	12	0	**222.59**	116.33	47.74	42.95	1	**222.59**	220.31	1.03	0.88	1
	Ins11(G3D3P9K4)	12	12	0	**287.68**	116.22	59.6	20.94	2	**287.68***	287.68	0	0.13	2
	Ins12(G3D6P6K4)	12	12	0	**235.39**	112.92	52.03	34.24	1	**235.39***	235.39	0	0.12	1
	Ins13(G3D7P8K5)	15	15	0	253.58	119.53	52.86	TL	1	**250.76**	226.08	9.84	0.32	1
	Ins14(G3D11P5K5)	16	15	1	422.42	142.13	66.35	TL	3	**418.82***	418.82	0	1.46	3
	Ins15(G3D5P12K5)	17	15	2	**493.27**	164.1	66.73	TL	3	**493.27***	493.27	0	0.36	3
	Ins16(G3D9P8K5)	17	15	2	447.46	152.06	66.02	TL	3	**444.25***	444.25	0	0.46	3
	Ins17(G3D12P5K5)	17	15	2	**442.53**	144.28	67.4	TL	3	**442.53**	441.08	0.33	0.55	3
	Ins18(G3D12P7K6)	19	18	1	**198.95**	154.74	22.22	93.79	0	**198.95**	198.69	0.13	0.82	0
	Ins19(G3D13P8K7)	21	21	0	323.33	124.82	61.4	TL	1	**296.06**	288.48	2.56	3.29	1
4 Gares	Ins20(G4D7P9K5)	16	15	1	**149.58**	101.27	32.3	329.17	0	**149.58***	149.58	0	0.27	0
	Ins21(G4D13P6K6)	19	18	1	1112.71	172.22	84.52	TL	9	**567.06***	563.56	0.62	0.71	4
	Ins22(G4D7P18K8)	25	24	1	542.65	186.17	65.69	TL	3	**522.61***	522.61	0	5.21	3
	Ins23(G4D9P17K8)	26	24	2	323.23	155.84	51.79	TL	1	**292.35***	292.35	0	2.05	1
	Ins24(G4D15P13K9)	28	27	1	412.31	220.07	46.63	TL	1	**394.51***	394.51	0	1.8	1
	Ins25(G4D10P20K10)	30	30	0	NT	NT	NT	TL	NT	**587.36**	573.94	2.29	3.77	3
	Ins26(G4D15P18K11)	33	33	0	321.59	212.24	34	TL	0	**284.81***	284.81	0	11.2	0
	Ins27(G4D20P15K11)	35	33	2	1464.12	204.75	86.02	TL	11	**437.33**	402.98	7.85	9.44	1
	Ins28(G4D17P18K11)	35	33	2	1518.87	211.98	86.04	TL	11	**405.16**	362.14	10.62	6.96	1
	Ins29(G4D17P19K12)	36	36	0	NT	NT	NT	TL	NT	**380.4**	379	0.37	8.6	0
5 Gares	Ins30(G5D10P7K5)	17	15	2	381.73	133.86	64.93	TL	2	**378.39***	378.39	0	0.38	2
	Ins31(G5D12P9K7)	21	21	0	344.12	182.86	46.86	TL	1	**337.05***	337.05	0	0.69	1
	Ins32(G5D13P9K7)	22	21	1	905.81	193.29	78.66	TL	6	**551.07***	551.07	0	1.01	3
	Ins33(G5D18P6K8)	24	24	0	1324.2	170.72	87.11	TL	11	**645.17**	602	6.69	1.41	4
	Ins34(G5D19P6K8)	25	24	1	1145.91	192.64	83.19	TL	9	**661.8***	661.8	0	1.36	4
	Ins35(G5D16P14K10)	30	30	0	561.4	236.45	57.88	TL	2	**415.07**	346.07	16.62	6.04	1
	Ins36(G5D6P24K10)	30	30	0	1988.12	213.47	89.26	TL	17	**787.4***	787.4	0	2.52	5
	Ins37(G5D23P10K11)	33	33	0	1771.48	283.56	83.99	TL	14	**710.8**	664.48	6.52	3.95	4
	Ins38(G5D18P19K12)	37	36	1	NT	NT	NT	TL	NT	**447.4**	445.75	0.37	10.7	1
	Ins39(G5D21P25K15)	46	45	1	NT	NT	NT	TL	NT	**563.63***	563.63	0	52.41	1

TABLEAU 5.1 Résultats des tests du MIP et $CGDH$

Le Tableau 5.2 résume les résultats du Tableau 5.1. Les colonnes de ce tableau sont données par :

- $\overline{Tem.}(s)$: le temps d'exécution moyen en secondes ;
- $Trou.\%$: le pourcentage d'instances pour lesquelles une solution réalisable a été trouvée ;
- $Opt.\%$: le pourcentage d'instances pour lesquelles une solution optimale a été trouvée, l'optimalité pour les solutions de l'algorithme $CGDH$ est atteinte si $UB_{CGDH} = LB_{CGDH}$;

- $Opt. + \%$: le pourcentage d'instances pour lesquels une solution optimale a été trouvée par l'algorithme $CGDH$, qui vérifie soit $UB_{CGDH} = LB_{CGDH}$ ou bien UB_{CGDH} avec UB_{MIP} si UB_{MIP} est optimale.
- $\overline{GAP}\%$: l'écart d'intégrité moyen ;
- $\overline{M}$: la moyenne des quantités $M = NS - D$.

	MIP					$CGDH$					
	$\overline{Tem.}(s)$	$Trou.\%$	$Opt.\%$	$\overline{GAP}\%$	$\overline{M}$	$\overline{Tem.}(s)$	$Trou.\%$	$Opt.\%$	$Opt. + \%$	$\overline{GAP}\%$	$\overline{M}$
2 Gares	265.3	100	100	58.82	0.5	0.27	100	100	100	0	0.5
3 Gares	739.19	100	60	56.23	1	0.84	100	50	80	1.39	1
4 Gares	1391.15	80	12.5	60.87	3.75	5	100	50	50	2.17	0.8
5 Gares	1500	80	0	73.98	7	8.05	100	60	60	3.02	2

Tableau 5.2 Comparaison des résultats

Comme on peut le constater au Tableau 5.2, l'algorithme $CGDH$ a toujours été capable de trouver une solution entière réalisable, et cela dans un temps d'exécution moyen très court de $3.54s$. En outre, il a fourni une solution qui peut être prouvée comme étant optimale en utilisant son GAP correspondant pour 65% des cas. Ce pourcentage peut même être amélioré à 72.5% lorsqu'on utilise les UB_{MIP} pour prouver l'optimalité des solutions de $CGDH$ où le GAP n'est pas nul. La moyenne des $\overline{M}$ est 1.075 qui est très proche de 0, et signifie que les solutions $CGDH$ servent toujours presque tous les passagers en utilisant les ressources disponibles.

En comparant les solutions du MIP et de l'algorithme $CGDH$, on peut constater que $CGDH$ a fourni les meilleure bornes supérieures UB dans tous les cas, et a également amélioré considérablement les bornes inférieures LB de tous les cas, fournissant des GAP proches de 0 pour tous les cas, et même des GAP égaux à 0 pour 26 instances sur les 40 instances testées au total.

5.4 Analyse de la qualité des solutions

Dans cette section, nous analysons la qualité des solutions en utilisant les indices α, β, et γ décrits dans la section 3.3 et NS le nombre de passagers non servis. Ces indices peuvent également être des objectifs inclus dans la fonction objectif, c'est pourquoi nous avons testé le MIP en utilisant différentes fonctions objectifs pour voir l'impact de ces modifications sur les solutions. Ici, nous supposons que la minimisation des coûts des passagers non servis $\min \delta$ est toujours un objectif, et nous ajoutons un ou plusieurs autres objectifs liés aux indices décrits comme suit.

— $MIP_{\alpha+\delta}$: minimisation des coûts de déplacement et de passagers non servis.
— $MIP_{\beta+\delta}$: minimisation des temps d'attente des passagers avant le début du service et des coûts des passagers non servis.
— $MIP_{\gamma+\delta}$: minimisation des temps d'arrivée des véhicules aux gares et des coûts des passagers non servis.
— $MIP_{\alpha+\beta+\gamma+\delta}$: minimisation de la somme de tout ce qui précède et des coûts des passagers non servis.

Le Tableau 5.3 résume la fonction objective de chaque modèle, ainsi que l'objectif de l'algorithme $CGDH$ qui est toujours $\min \alpha + \delta$.

	$MIP_{\alpha+\delta}$	$MIP_{\beta+\delta}$	$MIP_{\gamma+\delta}$	$MIP_{\alpha+\beta+\gamma+\delta}$	$CGDH$
Fonction objectif	$\min \alpha + \delta$	$\min \beta + \delta$	$\min \gamma + \delta$	$\min \alpha + \beta + \gamma + \delta$	$\min \alpha + \delta$

Tableau 5.3 La fonction objectif de chaque modèle

Pour réaliser cette analyse, nous avons testé les 20 instances Ins0,...,Ins19 en utilisant les quatre versions mentionnées du MIP et de l'algorithme $CGDH$ avec une limite de temps de 10 minutes. Noter que comme le coût de ne pas servir un passager est le même pour tous les passagers dans ces instances testées, $w_i = 100$ pour tout $i \in \mathcal{P} \cup \mathcal{D}$, alors $NS = \delta/100$.

Le Tableau 5.4 montre la valeur de chaque indice dans la solution correspondante. Presque tous les tests des quatre modèles dans ce tableau ont dépassé la limite de temps, ce qui explique l'absence de mention des valeurs optimales dans ce tableau.

	$MIP_{\alpha+\delta}$				$MIP_{\beta+\delta}$				$MIP_{\gamma+\delta}$				$MIP_{\alpha+\beta+\gamma+\delta}$				$CGDH$			
Instance	NS	α	β	γ	NS	α	β	γ	NS	α	β	γ	NS	α	β	γ	NS	α	β	γ
Ins0(G2D2P3K1)	**1**	**44.27**	**31.83**	**44.27**	**1**	**44.27**	**31.83**	**44.27**	**1**	**44.27**	**31.83**	**44.27**	**1**	**44.27**	**31.83**	**44.27**	**1**	**44.27**	**31.83**	**44.27**
Ins1(G2D3P2K1)	**1**	**39.54**	**7.12**	**39.54**	**1**	**39.54**	**7.12**	**39.54**	**1**	**39.54**	**7.12**	**39.54**	**1**	**39.54**	**7.12**	**39.54**	**1**	**39.54**	**7.12**	**39.54**
Ins2(G2D3P4K2)	**1**	**63.19**	**56.16**	**31.59**	**1**	**63.19**	**56.16**	**31.59**	**1**	**63.19**	**56.16**	**31.59**	**1**	**63.19**	**56.16**	**31.59**	**1**	**63.19**	**56.16**	**31.59**
Ins3(G2D3P7K3)	**4**	**59.83**	35.1	24.59	**4**	78.44	**15.9**	26.26	**4**	**59.83**	21.14	**19.94**	**4**	**59.83**	21.14	**19.94**	**4**	**59.83**	63.31	34
Ins4(G2D5P7K4)	**0**	**149.86**	124.17	48.41	**0**	160.32	**87.75**	**43.15**	**0**	162.8	**87.75**	**43.15**	**0**	159.23	**87.75**	**43.15**	**0**	**149.86**	140.23	50.16
Ins5(G2D6P7K4)	**1**	**114.88**	51.87	32.62	**1**	115.06	**49.93**	32.62	**1**	115.06	**49.93**	32.62	**1**	115.06	**49.93**	32.62	**1**	**114.88**	51.87	**32.57**
Ins6(G2D5P9K4)	**2**	136.36	80.52	38.79	4	170.43	99.44	43.2	4	**126.92**	76.24	**35.23**	**2**	135.4	**74.86**	39.05	**2**	135.4	128.9	44.46
Ins7(G2D5P9K4)	**0**	**99.81**	68.37	33.19	**0**	114.08	**50.23**	**32.43**	**0**	115.34	60.99	32.87	**0**	114.08	**50.23**	**32.43**	**0**	**99.81**	112.89	37.11
Ins8(G2D10P7K5)	**4**	158.27	69.93	34.01	5	164.02	50.03	33.91	7	**140.53**	**41.72**	**25.77**	4	187.08	76.2	37.89	**4**	158.25	100.4	39.72
Ins9(G2D9P9K6)	**0**	147.32	77.73	33.54	3	207.13	48.45	37.02	9	**145.68**	**13.45**	**24.28**	**0**	187.33	70.96	38.53	**0**	147.32	113.91	36.78
Ins10(G3D7P5K4)	**1**	**122.59**	49.81	34.71	**1**	125.71	**45.16**	34.55	**1**	124.85	**45.16**	**33.55**	**1**	124.3	**45.16**	**33.55**	**1**	**122.59**	65.77	38.7
Ins11(G3D3P9K4)	**2**	**87.68**	65.96	28.54	**2**	93.61	**57.28**	26.37	**2**	100.67	70	**26.25**	**2**	**87.68**	59.25	26.87	**2**	**87.68**	122.97	35.67
Ins12(G3D6P6K4)	**1**	**135.39**	**48.48**	**36.67**	**1**	**135.39**	**48.48**	**36.67**	**1**	**135.39**	**48.48**	**36.67**	**1**	**135.39**	**48.48**	**36.67**	**1**	**135.39**	111.45	46.37
Ins13(G3D7P8K5)	**1**	156.59	28.64	36.42	2	181.17	24.1	36.68	3	**141.34**	21.73	**34.53**	2	162.37	**19.79**	36.84	**1**	150.76	44.5	37.97
Ins14(G3D11P5K5)	**3**	138.52	32.77	28.5	5	**98.81**	**24.47**	**20.84**	**3**	137.39	50.4	27.48	4	149.89	44.73	30.09	**3**	118.82	78.11	30.61
Ins15(G3D5P12K5)	**3**	193.27	83.2	40.93	5	195.22	56.21	41.32	5	182.66	**33.72**	39.75	4	**146.3**	31.91	**33.54**	**3**	193.27	113.64	44.94
Ins16(G3D9P8K5)	**3**	147.46	106.08	30.88	**3**	149.19	92.88	**30.27**	**3**	155.48	100.31	33.19	**3**	149.19	97.36	31.16	**3**	**144.25**	**146.74**	38.13
Ins17(G3D12P5K5)	**3**	154.53	**41.84**	31.27	**3**	174.92	54.62	35.41	4	148.04	52.33	**29.75**	**3**	161.34	58.88	32.27	**3**	**142.53**	63.92	33.03
Ins18(G3D12P7K6)	**0**	**198.95**	99.61	**33.16**	2	257.38	86.87	42.97	1	226.83	**81.47**	37.8	2	259.84	102.13	43.31	**0**	**198.95**	131.29	36.35
Ins19(G3D13P8K7)	**1**	223.33	75.6	35.03	4	258.97	84.01	37.1	7	209.02	41.7	31.78	11	**184.26**	**30.93**	**26.87**	**1**	196.06	84.84	31.62

Tableau 5.4 Comparaison des qualités des solutions

Comme on peut le constater, l'algorithme $CGDH$ a fourni le nombre minimum de passagers non servis NS pour toutes les instances. Il a également fourni les coûts de déplacement minimum pour la majorité des instances, à l'exception de certaines instances, comme Ins6 par exemple, où la solution du modèle $MIP_{\gamma+\delta}$ a fourni une meilleure valeur de α. Mais cela est dû au fait que la solution du modèle $MIP_{\gamma+\delta}$ a servi moins de passagers que celle de $CGDH$. D'autre part, les solutions $CGDH$ n'ont pas fourni les valeurs minimales pour les indices β et γ, cela est dû au fait que la solution de $CGDH$ sert toujours le nombre maximum de passagers, en d'autres termes, les valeurs de NS fournies par $CGDH$ sont toujours minimales.

Pour rendre la comparaison plus fiable, nous considérons les moyennes de performance suivantes :

— $\overline{\alpha+\delta}$: la moyenne des coûts de déplacement plus les coûts des passagers non servis.
— $\overline{\beta+\delta}$: la moyenne des temps d'attente plus les coûts des passagers non servis
— $\overline{\gamma+\delta}$: la moyenne des temps d'arrivée à la gare plus les coûts des passagers non servis.
— $\overline{\delta}$: la moyenne des coûts des passagers non servis.

Les coûts des passagers non servis sont considérés dans chacune des trois premières moyennes, afin de rendre la comparaison des solutions équilibrée et pour éviter des cas comme celui de l'instance Ins6 où la solution du $MIP_{\gamma+\delta}$ semble fournir une meilleure valeur de l'indice α que celle de $CGDH$, mais cela est dû au fait que la solution de $MIP_{\gamma+\delta}$ a servi moins de passagers que celle de $CGDH$.

Le Tableau 5.5 montre les moyennes de performances de chaque modèle, où les valeurs en **gras** sont les valeurs minimales correspondantes.

	$MIP_{\alpha+\delta}$	$MIP_{\beta+\delta}$	$MIP_{\gamma+\delta}$	$MIP_{\alpha+\beta+\gamma+\delta}$	$CGDH$
$\overline{\alpha+\delta}$	288.58	381.34	418.74	368.28	**285.13**
$\overline{\beta+\delta}$	**221.74**	293.55	339.58	288.24	248.49
$\overline{\gamma+\delta}$	**194.83**	275.31	323.0	269.51	198.18
$\overline{\delta}$	**160**	240	290	234	**160**

Tableau 5.5 Comparaison des moyennes de performance

Comme on peut le voir dans le Tableau 5.5, l'algorithme $CGDH$ a fourni les valeurs minimales des moyennes de performance $\overline{\alpha+\delta}$ et $\overline{\delta}$, tandis qu'il a dépassé les valeurs minimales des moyennes de performance $\overline{\beta+\delta}$ et $\overline{\gamma+\delta}$ égales à 221.74 et 194.83 respectivement, fournies par le modèle $MIP_{\alpha+\delta}$, seulement par une augmentation de 12.06% et 1.71% respectivement. Il était prévu que les modèles $MIP_{\beta+\delta}$ et $MIP_{\gamma+\delta}$ fournissent les meilleures valeurs pour $\overline{\beta+\delta}$

et $\overline{\gamma + \delta}$, respectivement. Cependant, en raison de la limite de temps, la plupart des tests ont atteint cette limite sans converger vers des solutions optimales. Cela explique pourquoi les résultats attendus n'ont pas été observés.

Nous pouvons conclure que les solutions fournies par le $MIP_{\alpha+\delta}$ ont en général la meilleure qualité en termes des moyennes de performances définies. Cependant, les solutions du $CGDH$ ont presque le même niveau de qualité et peuvent être obtenues dans un temps d'exécution bien inférieur à celles du $MIP_{\alpha+\delta}$.

CHAPITRE 6 Conclusion

Le problème de tournées de véhicules défini et traité dans ce travail n'a jamais été étudié auparavant dans la littérature. Ce nouveau problème consiste à combiner les contraintes de temps de trajet maximum des passagers avec le principe du « *Backhauls* » dans les problèmes de collecte et livraison. Le nouveau problème traité dans ce travail n'a pas seulement un aspect théorique et enrichit la littérature, mais il s'agit d'un problème réel qui servirait à optimiser les systèmes de transport liés au réseau ferroviaire.

6.1 Synthèse des travaux

Pour commencer, nous avons exploré différents travaux sur des problèmes ayant des aspects communs avec le nôtre tels que le principe du « *Backhauls* » et le temps de trajet maximum.

Nous avons ensuite formulé notre problème sur un réseau routier où les sommets représentent les gares et les lieux de desserte des passagers, et les arcs représentent les mouvements valides entre les sommets, où l'objectif est de servir le maximum de passagers et de minimiser les coûts de déplacement.

Par la suite, nous avons fourni une décomposition de notre problème en un problème maître et un sous-problème, où le sous-problème peut être modélisé comme un problème de plus court chemin avec des contraintes de ressources. Afin de résoudre le problème maître, nous avons décrit une méthode de résolution qui consiste en une heuristique de plongée basée sur la génération de colonnes. Aussi, pour résoudre les sous-problèmes, nous avons fourni une adaptation de l'algorithme d'étiquetage monodirectionnel pour résoudre efficacement les problèmes de plus court chemin avec contraintes de ressources en utilisant seulement deux ressources, soit le temps et la charge cumulés.

Finalement, nous avons utilisé un total de 40 instances générées de manière synthétique afin d'évaluer l'efficacité de notre méthode. Grâce à nos résultats compétitifs, nous avons démontré que l'algorithme de résolution proposé pourrait être efficace pour résoudre notre problème. En effet, il est parvenu à résoudre les 40 instances testées avec un écart moyen d'intégrité (GAP) de seulement $1,64\%$ et ce, dans un temps moyen de $3,54s$. Nous avons également évalué la qualité des solutions fournies par notre algorithme de résolution en utilisant un ensemble d'indices, tels que le nombre de clients non servis, les coûts de déplacement, les temps d'attente avant le début du service et les heures d'arrivée aux gares. Cette analyse a montré que la qualité des solutions de l'algorithme est très élevée et fournit un bon compromis

des indices décrits.

6.2 Limitations de la solution proposée

Bien que nos résultats empiriques démontrent la compétitivité de notre algorithme en termes de rapidité et de fourniture de solutions à faibles coûts de déplacement, il ne fournit pas de solutions avec faibles temps d'attente de passagers et temps d'arrivée aux gares. Cela peut être dû aux définitions des fenêtres temporelles que nous avons imposées pour modéliser le sous-problème comme un SPPRC avec seulement deux ressources.

6.3 Perspectives de travaux futurs

Plusieurs possibilités sont envisagées pour le développement de travaux connexes. Une extension directe de notre méthode serait de l'intégrer dans un cadre *Branch-and-Price* afin qu'elle soit une méthode exacte plutôt qu'une heuristique. De plus, d'autres méthodes de résolution peuvent être développées pour résoudre le sous-problème comme il consiste en un élément très important de notre problématique ; un bon choix serait une bonne adaptation d'un schéma d'étiquetage bidirectionnel qui est très utilisé dans la littérature pour résoudre efficacement les SPPRCs.

Une autre perspective serait l'adaptation de notre méthode de résolution telle qu'elle est actuellement afin qu'elle puisse être utilisée pour résoudre une version dynamique et/ou stochastique de notre problème.

Une généralisation du problème peut également être envisagée, dans laquelle celui-ci serait défini sur une journée entière. Cette approche permettrait de prendre en compte plusieurs trains par gare, ainsi que plusieurs séquences de livraison et de collecte par véhicule.

RÉFÉRENCES

[1] G. B. Dantzig et J. H. Ramser, “The truck dispatching problem,” *Management Science*, vol. 6, n°. 1, p. 80–91, 1959. [En ligne]. Disponible : https://doi.org/10.1287/mnsc.6.1.80

[2] SNCF, “SNCF group companies,” 2024, https://www.groupe-sncf.com/en/group/about-us/companies.

[3] I. Deif et L. Bodin, “Extension of the Clarke and Wright algorithm for solving the vehicle routing problem with backhauling,” dans *Proceedings of the Babson conference on software uses in transportation and logistics management.* Babson Park, MA, 1984, p. 75–96.

[4] SNCF, “Ma course SNCF,” 2024, https://www.groupe-sncf.com/fr/innovation/mobilite-territoires/ma-course.

[5] M. W. Savelsbergh et M. Sol, “The general pickup and delivery problem,” *Transportation Science*, vol. 29, n°. 1, p. 17–29, 1995.

[6] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia et G. Laporte, “Static pickup and delivery problems : a classification scheme and survey,” *Top*, vol. 15, p. 1–31, 2007.

[7] M. Chassaing, “Problèmes de transport à la demande avec prise en compte de la qualité de service,” Thèse de doctorat, Université Blaise Pascal-Clermont-Ferrand II, 2015.

[8] M. Goetschalckx et C. Jacobs-Blecha, “The vehicle routing problem with backhauls,” *European Journal of Operational Research*, vol. 42, n°. 1, p. 39–51, 1989.

[9] Y. Dumas, J. Desrosiers et F. Soumis, “The pickup and delivery problem with time windows,” *European Journal of Operational Research*, vol. 54, n°. 1, p. 7–22, 1991.

[10] B. Golden, A. Assad, L. Levy et F. Gheysens, “The fleet size and mix vehicle routing problem,” *Computers & Operations Research*, vol. 11, n°. 1, p. 49–66, 1984.

[11] Á. Felipe, M. T. Ortuño, G. Righini et G. Tirado, “A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges,” *Transportation Research Part E : Logistics and Transportation Review*, vol. 71, p. 111–128, 2014.

[12] J.-F. Cordeau et G. Laporte, “The dial-a-ride problem (DARP) : Variants, modeling issues and algorithms,” *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 1, p. 89–101, 2003.

[13] ——, “A tabu search heuristic for the static multi-vehicle dial-a-ride problem,” *Transportation Research Part B : Methodological*, vol. 37, n°. 6, p. 579–594, 2003.

[14] D. M. Stein, “Scheduling dial-a-ride transportation systems,” *Transportation Science*, vol. 12, n°. 3, p. 232–249, 1978.

[15] J.-F. Cordeau et G. Laporte, “The dial-a-ride problem : models and algorithms,” *Annals of Operations Research*, vol. 153, p. 29–46, 2007.

[16] Y. Molenbruch, K. Braekers et A. Caris, “Typology and literature review for dial-a-ride problems,” *Annals of Operations Research*, vol. 259, p. 295–325, 2017.

[17] S. C. Ho, W. Y. Szeto, Y.-H. Kuo, J. M. Leung, M. Petering et T. W. Tou, “A survey of dial-a-ride problems : Literature review and recent developments,” *Transportation Research Part B : Methodological*, vol. 111, p. 395–421, 2018.

[18] J.-F. Cordeau, “A branch-and-cut algorithm for the dial-a-ride problem,” *Operations Research*, vol. 54, n°. 3, p. 573–586, 2006.

[19] S. Ropke, J.-F. Cordeau et G. Laporte, “Models and branch-and-cut algorithms for pickup and delivery problems with time windows,” *Networks*, vol. 49, n°. 4, p. 258–272, 2007.

[20] M. Riedler et G. Raidl, “Solving a selective dial-a-ride problem with logic-based benders decomposition,” *Computers & Operations Research*, vol. 96, p. 30–54, 2018.

[21] D. Gaul, K. Klamroth et M. Stiglmayr, “Event-based MILP models for ridepooling applications,” *European Journal of Operational Research*, vol. 301, n°. 3, p. 1048–1063, 2022.

[22] M. J. Santos, P. Amorim, A. Marques, A. Carvalho et A. Póvoa, “The vehicle routing problem with backhauls towards a sustainability perspective : A review,” *Top*, vol. 28, n°. 2, p. 358–401, 2020.

[23] B. Golden, E. Baker, J. Alfaro et J. Schaffer, “The vehicle routing problem with backhauling : two approaches,” dans *Proceedings of the twenty-first annual meeting of the SE TIMS, Myrtle Beach, SC, USA*, 1985.

[24] D. Dumez, C. Tilk, I. Stefan, F. Lehuédé, K. Olkis et O. Péton, “The two-echelon vehicle routing problem with backhauls and capacitated satellites,” dans *ROADEF 2023*, 2023.

[25] Sukhpal et K. Kumar, “Multi-trip multi-compartment vehicle routing problem with backhauls,” *International Journal of System Assurance Engineering and Management*, vol. 15, p. 1–18, 2023.

[26] C. Wang, Z. Cao, Y. Wu, L. Teng et G. Wu, “Deep reinforcement learning for solving vehicle routing problems with backhauls,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, n°. 3, p. 4779–4793, 2025.

[27] M. Karimi, F. Camiat, G. Desaulniers et M. G. and, “An exact branch-and-price-and-cut algorithm for a practical and large-scale dial-a-ride problem,” *Journal of the Operational Research Society*, p. 1–15, 2024. [En ligne]. Disponible : https://doi.org/10.1080/01605682.2024.2412214

[28] S. N. Parragh, “Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem,” *Transportation Research Part C : Emerging Technologies*, vol. 19, n°. 5, p. 912–930, 2011. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0968090X10001075

[29] K. Braekers, A. Caris et G. K. Janssens, “Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots,” *Transportation Research Part B : Methodological*, vol. 67, p. 166–186, 2014. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0191261514000800

[30] A. Schulz et C. Pfeiffer, “A branch-and-cut algorithm for the dial-a-ride problem with incompatible customer types,” *Transportation Research Part E : Logistics and Transportation Review*, vol. 181, p. 103394, 2024. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S1366554523003824

[31] T. Garaix, C. Artigues, D. Feillet et D. Josselin, “Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation,” *Computers & Operations Research*, vol. 38, n°. 10, p. 1435–1442, 2011. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0305054810003102

[32] T. Gschwind et S. Irnich, “Effective handling of dynamic time windows and its application to solving the dial-a-ride problem,” *Transportation Science*, vol. 49, n°. 2, p. 335–354, 2015. [En ligne]. Disponible : https://doi.org/10.1287/trsc.2014.0531

[33] E. Queiroga, Y. Frota, R. Sadykov, A. Subramanian, E. Uchoa et T. Vidal, “On the exact solution of vehicle routing problems with backhauls,” *European Journal of Operational Research*, vol. 287, n°. 1, p. 76–89, 2020. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0377221720304008

[34] M. Gendreau et J.-Y. Potvin, “Metaheuristics in combinatorial optimization,” *Annals of Operations Research*, vol. 140, p. 189–213, 2005.

[35] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller et E. Teller, “Equation of State Calculations by Fast Computing Machines,” *The Journal of Chemical Physics*, vol. 21, n°. 6, p. 1087–1092, 06 1953. [En ligne]. Disponible : https://doi.org/10.1063/1.1699114

[36] S. Kirkpatrick, C. D. Gelatt et M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, n°. 4598, p. 671–680, 1983. [En ligne]. Disponible : https://www.science.org/doi/abs/10.1126/science.220.4598.671

[37] J. W. Baugh Jr, G. K. R. Kakivaya et J. R. Stone, “Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing,” *Engineering Optimization*, vol. 30, n°. 2, p. 91–123, 1998. [En ligne]. Disponible : https://doi.org/10.1080/03052159808941240

[38] I. Zidi, K. Mesghouni, K. Zidi et K. Ghedira, “A multi-objective simulated annealing for the multi-criteria dial a ride problem,” *Engineering Applications of Artificial Intelligence*, vol. 25, n°. 6, p. 1121–1131, 2012. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0952197612000802

[39] İlker Küçükoğlu et N. Öztürk, “An advanced hybrid meta-heuristic algorithm for the vehicle routing problem with backhauls and time windows,” *Computers & Industrial Engineering*, vol. 86, p. 60–68, 2015. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0360835214003453

[40] M. O. M. Javad et B. Karimi, “A simulated annealing algorithm for solving multi-depot location routing problem with backhaul,” *International Journal of Industrial and Systems Engineering*, vol. 25, n°. 4, p. 460–477, 2017. [En ligne]. Disponible : https://www.inderscienceonline.com/doi/abs/10.1504/IJISE.2017.083043

[41] F. Glover et C. McMillan, “The general employee scheduling problem. an integration of ms and ai,” *Computers & Operations Research*, vol. 13, n°. 5, p. 563–573, 1986. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/030505488690050X

[42] C. Duhamel, J.-Y. Potvin et J.-M. Rousseau, “A tabu search heuristic for the vehicle routing problem with backhauls and time windows,” *Transportation Science*, vol. 31, n°. 1, p. 49–59, 1997. [En ligne]. Disponible : https://doi.org/10.1287/trsc.31.1.49

[43] P. Shaw, “Using constraint programming and local search methods to solve vehicle routing problems,” dans *Principles and Practice of Constraint Programming — CP98*, M. Maher et J.-F. Puget, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 1998, p. 417–431.

[44] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt et G. Dueck, “Record breaking optimization results using the ruin and recreate principle,” *Journal of Computational Physics*, vol. 159, n°. 2, p. 139–171, 2000. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0021999199964136

[45] S. Ropke et D. Pisinger, “An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows,” *Transportation Science*, vol. 40, n°. 4, p. 455–472, 2006. [En ligne]. Disponible : https://doi.org/10.1287/trsc.1050.0135

[46] S. Jain et P. Van Hentenryck, “Large neighborhood search for dial-a-ride problems,” dans *Principles and Practice of Constraint Programming – CP 2011*, J. Lee, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2011, p. 400–413.

[47] S. N. Parragh et V. Schmid, “Hybrid column generation and large neighborhood search for the dial-a-ride problem,” *Computers & Operations Research*, vol. 40, n°. 1, p. 490–497, 2013. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0305054812001694

[48] S. Ropke et D. Pisinger, “A unified heuristic for a large class of vehicle routing problems with backhauls,” *European Journal of Operational Research*, vol. 171, nº. 3, p. 750–775, 2006. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0377221704005831

[49] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI : University of Michigan Press, 1975.

[50] B. Rekiek, A. Delchambre et H. A. Saleh, “Handicapped person transportation : An application of the grouping genetic algorithm,” *Engineering Applications of Artificial Intelligence*, vol. 19, nº. 5, p. 511–520, 2006. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0952197606000339

[51] A. García-Nájera, J. A. Bullinaria et M. A. Gutiérrez-Andrade, “An evolutionary approach for multi-objective vehicle routing problems with backhauls,” *Computers & Industrial Engineering*, vol. 81, p. 90–108, 2015. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0360835214004586

[52] C. Cubillos, N. Rodriguez et B. Crawford, “A study on genetic algorithms for the DARP problem,” dans *Bio-inspired Modeling of Cognitive Tasks*, J. Mira et J. R. Álvarez, édit. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007, p. 498–507.

[53] K. Ganesh et T. Narendran, “Cloves : A cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up,” *European Journal of Operational Research*, vol. 178, nº. 3, p. 699–717, 2007. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0377221706001214

[54] G. Desaulniers, J. Desrosiers et M. M. Solomon, *Column generation.* Springer Science & Business Media, 2006, vol. 5.

[55] Y. Dumas, J. Desrosiers et F. Soumis, “Large scale multi-vehicle dial-a-ride problems,” Groupe d’études et de recherche en analyse des décisions, GERAD, Montréal QC H3T 2A7, Canada, Les Cahiers du GERAD G-89-30, sept. 1989. [En ligne]. Disponible : https://www.gerad.ca/fr/papers/G-89-30

[56] J. Desrosiers, Y. Dumas, F. Soumis, S. Taillefer et D. Villeneuve, “An algorithm for mini-clustering in handicapped transport,” Groupe d’études et de recherche en analyse des décisions, GERAD, Montréal QC H3T 2A7, Canada, Les Cahiers du GERAD G-91-02, janv. 1991. [En ligne]. Disponible : https://www.gerad.ca/fr/papers/G-91-02

[57] I. Ioachim, J. Desrosiers, Y. Dumas, M. M. Solomon et D. Villeneuve, “A request clustering algorithm for door-to-door handicapped transportation,” *Transportation Science*, vol. 29, nº. 1, p. 63–78, 1995. [En ligne]. Disponible : https://doi.org/10.1287/trsc.29.1.63

[58] S. N. Parragh, J.-F. Cordeau, K. F. Doerner et R. F. Hartl, “Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints.” *OR Spectrum*, vol. 34, n°. 3, p. 593–633, 2012. [En ligne]. Disponible : http://dblp.uni-trier.de/db/journals/ors/ors34.html#ParraghCDH12

[59] N. Rahmani, B. Detienne, R. Sadykov et F. Vanderbeck, “A Column Generation Based Heuristic for the Dial-A-Ride Problem,” dans *International Conference on Information Systems, Logistics and Supply Chain (ILS)*, Bordeaux, France, juin 2016. [En ligne]. Disponible : https://inria.hal.science/hal-01425755

[60] S. Gélinas, M. Desrochers, J. Desrosiers et M. M. Solomon, “A new branching strategy for time constrained routing problems with application to backhauling,” *Annals of Operations Research*, vol. 61, p. 91–109, 1995.

[61] S. Irnich et G. Desaulniers, “Shortest path problems with resource constraints,” *Les Cahiers du GERAD ISSN*, vol. 711, p. 2440, 2004.

[62] M. Van Den Eeckhout, M. Vanhoucke et B. Maenhout, “A column generation-based diving heuristic to solve the multi-project personnel staffing problem with calendar constraints and resource sharing,” *Computers & Operations Research*, vol. 128, p. 105163, 2021. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S030505482030280X

[63] C. Gauvin, G. Desaulniers et M. Gendreau, “A branch-cut-and-price algorithm for the vehicle routing problem with stochastic demands,” *Computers & Operations Research*, vol. 50, p. 141–153, 2014. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0305054814000835

[64] C. Gauvin, “Un algorithme de génération de colonnes pour le problème de tournées de véhicule avec demandes stochastiques,” Mémoire de maîtrise, École Polytechnique de Montréal, 2012.

[65] G. Desaulniers, J. Desrosiers, I. loachim, M. M. Solomon, F. Soumis et D. Villeneuve, *A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems*, T. G. Crainic et G. Laporte, édit. Boston, MA : Springer US, 1998. [En ligne]. Disponible : https://doi.org/10.1007/978-1-4615-5755-5_3

[66] S. Irnich, “Resource extension functions : Properties, inversion, and generalization to segments,” *OR Spectrum*, vol. 30, n°. 1, p. 113–148, 2008.

[67] G. Righini et M. Salani, “Symmetry helps : Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints,” *Discrete Optimization*, vol. 3, n°. 3, p. 255–273, 2006. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S1572528606000417

[68] C. Tilk, A.-K. Rothenbächer, T. Gschwind et S. Irnich, “Asymmetry matters : Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster,” *European Journal of Operational Research*, vol. 261, n°. 2, p. 530–539, 2017. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0377221717302035

[69] J. E. Beasley et N. Christofides, “An algorithm for the resource constrained shortest path problem,” *Networks*, vol. 19, n°. 4, p. 379–394, 1989. [En ligne]. Disponible : https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230190402

[70] D. T. Sanchez, “cspy : A python package with a collection of algorithms for the (resource) constrained shortest path problem,” *Journal of Open Source Software*, vol. 5, n°. 49, p. 1655, 2020. [En ligne]. Disponible : https://cspy.readthedocs.io/en/latest/index.html

[71] T. R. Sexton et Y.-M. Choi, “Pickup and delivery of partial loads with “soft” time windows,” *American Journal of Mathematical and Management Sciences*, vol. 6, n°. 3-4, p. 369–398, 1986.

[72] I. H. Osman et S. Salhi, “Local search strategies for the vehicle fleet mix problem,” *Modern Heuristic Search Methods*, p. 131–153, 1996.

[73] X. Li, P. Tian et Y. Aneja, “An adaptive memory programming metaheuristic for the heterogeneous fixed fleet vehicle routing problem,” *Transportation Research Part E : Logistics and Transportation Review*, vol. 46, n°. 6, p. 1111–1127, 2010.

[74] H. Hernández-Pérez et J.-J. Salazar-González, “A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery,” *Discrete Applied Mathematics*, vol. 145, n°. 1, p. 126–139, 2004.

[75] P. Chalasani et R. Motwani, “Approximating capacitated routing and delivery problems,” *SIAM Journal on Computing*, vol. 28, n°. 6, p. 2133–2149, 1999.

[76] H. Hernández-Pérez et J.-J. Salazar-González, “Heuristics for the one-commodity pickup-and-delivery traveling salesman problem,” *Transportation Science*, vol. 38, n°. 2, p. 245–255, 2004.

[77] F. Wang, A. Lim et Z. Xu, “The one-commodity pickup and delivery travelling salesman problem on a path or a tree,” *Networks*, vol. 48, n°. 1, p. 24–35, 2006.

[78] S. Anily et R. Hassin, “The swapping problem,” *Networks*, vol. 22, n°. 4, p. 419–433, 1992.

[79] G. N. Frederickson et D. Guan, “Preemptive ensemble motion planning on a tree,” *SIAM Journal on Computing*, vol. 21, n°. 6, p. 1130–1152, 1992.

[80] S. Ropke et J.-F. Cordeau, “Branch and cut and price for the pickup and delivery problem with time windows,” *Transportation Science*, vol. 43, n°. 3, p. 267–286, 2009.

[81] T. R. Sexton et L. D. Bodin, “Optimizing single vehicle many-to-many operations with desired delivery times : I. scheduling,” *Transportation Science*, vol. 19, n°. 4, p. 378–410, 1985.

[82] C. E. Cortés, M. Matamala et C. Contardo, “The pickup and delivery problem with transfers : Formulation and a branch-and-cut solution method,” *European Journal of Operational Research*, vol. 200, n°. 3, p. 711–724, 2010.

[83] M. E. Lübbecke, “Combinatorially simple pickup and delivery paths,” *Central European Journal of Operations 'Research'*, vol. 12, p. 405–417, 2004.

[84] G. Nagy et S. Salhi, “Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries,” *European Journal of Operational Research*, vol. 162, n°. 1, p. 126–141, 2005.

[85] J. Brandao, “A new tabu search algorithm for the vehicle routing problem with backhauls,” *European Journal of Operational Research*, vol. 173, n°. 2, p. 540–555, 2006.

[86] A. Wade et S. Salhi, “An investigation into a new class of vehicle routing problem with backhauls,” *Omega*, vol. 30, n°. 6, p. 479–487, 2002.

[87] N. Christofides, “Worst-case analysis of a new heuristic for the travelling salesman problem,” dans *Operations Research Forum*, vol. 3, n°. 1. Springer, 2022, p. 20.

[88] M. Gendreau, G. Laporte et D. Vigo, “Heuristics for the traveling salesman problem with pickup and delivery,” *Computers & Operations Research*, vol. 26, n°. 7, p. 699–714, 1999.

[89] T.-Y. Hu et C.-P. Chang, “A revised branch-and-price algorithm for dial-a-ride problems with the consideration of time-dependent travel cost,” *Journal of Advanced Transportation*, vol. 49, n°. 6, p. 700–723, 2015. [En ligne]. Disponible : https://onlinelibrary.wiley.com/doi/abs/10.1002/atr.1296

[90] S. N. Parragh, J. Pinho de Sousa et B. Almada-Lobo, “The dial-a-ride problem with split requests and profits,” *Transportation Science*, vol. 49, n°. 2, p. 311–334, 2015. [En ligne]. Disponible : https://doi.org/10.1287/trsc.2014.0520

[91] S. Mitrović-Minić et G. Laporte, “The pickup and delivery problem with time windows and transshipment,” *INFOR : Information Systems and Operational Research*, vol. 44, n°. 3, p. 217–227, 2006. [En ligne]. Disponible : https://doi.org/10.1080/03155986.2006.11732749

[92] G. B. Dantzig et P. Wolfe, “Decomposition principle for linear programs,” *Operations Research*, vol. 8, n°. 1, p. 101–111, 1960. [En ligne]. Disponible : https://doi.org/10.1287/opre.8.1.101