

Titre: Nouvelle approche d'optimisation pour le problème d'affectation
Title: des unités ferroviaires

Auteur: Pierre Mordant
Author:

Date: 2023

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Mordant, P. (2023). Nouvelle approche d'optimisation pour le problème
Citation: d'affectation des unités ferroviaires [Mémoire de maîtrise, Polytechnique
Montréal]. PolyPublie. <https://publications.polymtl.ca/57076/>

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/57076/>
PolyPublie URL:

**Directeurs de
recherche:** Louis-Martin Rousseau, & Guy Desaulniers
Advisors:

Programme: Maîtrise recherche en mathématiques appliquées
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

**Nouvelle approche d'optimisation pour le problème d'affectation des unités
ferroviaires**

PIERRE MORDANT

Département de mathématiques et de génie industriel

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

Décembre 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

**Nouvelle approche d'optimisation pour le problème d'affectation des unités
ferroviaires**

présenté par **Pierre MORDANT**

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

Louis-Martin ROUSSEAU, membre et directeur de recherche

Guy DESAULNIERS, membre et codirecteur de recherche

François SOUMIS, membre

REMERCIEMENTS

Je tiens à remercier tout d'abord mon directeur de recherche Louis-Martin Rousseau et mon codirecteur de recherche Guy Desaulniers, qui ont été d'une grande aide tout au long de la maîtrise et qui se sont rendus particulièrement disponibles quand il le fallait. Je leur en suis très reconnaissant.

Je tiens également à remercier l'entreprise GIRO, pour avoir proposé ce sujet et pour nous avoir fourni les données dont ils disposaient. Les conseils et indications de toute l'équipe ont été précieux pour le cadrage et le bon déroulement de cette maîtrise.

Je remercie le GERAD pour leur soutien financier et matériel au cours de cette maîtrise.

Je souhaite aussi remercier Michel Gendreau et François Soumis pour avoir accepté de faire partie de ce jury.

Enfin je souhaite remercier tous les collègues et amis qui m'ont accompagné au cours de ces deux années à Montréal. Étudier et vivre à vos côtés a été une très belle expérience dont je garderai un excellent souvenir.

RÉSUMÉ

Nous nous intéressons au problème d'affectation des unités ferroviaires. Il s'agit d'un problème de couverture, où des voyages prévus au préalable doivent être associés à des unités ferroviaires. Nous avons travaillé sur ce problème sur un horizon d'une journée, en cherchant à obtenir une solution périodique qui puisse être appliquée sur un intervalle de temps plus long. L'assignation doit respecter un certain nombre de contraintes, notamment la disponibilité des différents types d'unités, la périodicité quotidienne des stationnements dans les dépôts, et des contraintes de couplage et de découplage des unités selon des règles prédéfinies. Ce problème peut être résolu efficacement en découpant en deux phases distinctes : calcul des compositions de trains optimales en ignorant certaines contraintes puis affectation des trains à ces compositions. Cette approche a notamment été utilisée par Giro avec lesquels nous avons travaillé au cours de cette maîtrise. Cette approche heuristique ne permet cependant pas de connaître l'erreur par rapport à la valeur optimale du problème. Nous avons donc cherché à modéliser ce problème en une seule phase, par un problème de plus court chemin dans un graphe avec contraintes supplémentaires, où chaque chemin décrit un enchaînement potentiel de voyages couverts par une unique unité.

Nous résolvons ce modèle linéaire par génération de colonnes, avant d'utiliser un solveur en nombres entiers sur les colonnes retenues. Nous avons travaillé sur des instances construites aléatoirement et sur des instances fournies par la SNCF. La plus grosse de nos instances inclut 367 voyages à couvrir, répartis entre 26 gares. Les temps de résolution sur cette instance et celles de taille comparable nous ont poussé à utiliser des techniques heuristiques pour accélérer la résolution du problème en nombres entiers. Nous présentons nos résultats, comparés avec ceux de Giro, et avec les meilleures bornes que nous connaissons pour ce problème. Nous présentons enfin les limites de notre modèle, et les pistes d'amélioration de notre méthode.

ABSTRACT

We interested ourselves in the problem of allocating rail units. This is a hedging problem, where pre-scheduled trips have to be associated with railway units. We have worked on this problem over a one-day horizon, seeking to obtain a periodic solution that can be applied over a longer time interval. The assignment must respect a number of constraints, including the availability of different types of units, the daily periodicity of parking in depots, and constraints on coupling and uncoupling units according to predefined rules.

This problem can be solved efficiently by dividing it into two distinct phases: calculating optimal train compositions while ignoring certain constraints, and then assigning trains to these compositions. This approach was used in particular by Giro, with whom we worked during the course of this master's degree. However, this heuristic approach does not allow us to know the error with respect to the optimal value of the problem. We therefore sought to model this problem in a single phase, by means of a shortest path problem in a graph with additional constraints, where each path describes a potential sequence of journeys covered by a single unit.

We solve this linear model by column generation, before using an integer solver on the selected columns. We worked on randomly constructed instances and on instances provided by SNCF. The largest of our instances included 367 trips to cover, spread over 26 stations. Solving times on this instance and others of comparable size prompted us to use heuristic techniques to speed up solving the integer problem. We present our results, compared with those of Giro, and with the best bounds we know for this problem. Finally, we present the limitations of our model, and suggest ways of improving our method.

TABLE DES MATIÈRES

REMERCIEMENTS	iii
RÉSUMÉ	iv
ABSTRACT	v
TABLE DES MATIÈRES	vi
LISTE DES TABLEAUX	viii
LISTE DES FIGURES	ix
LISTE DES SIGLES ET ABRÉVIATIONS	x
CHAPITRE 1 INTRODUCTION	1
1.1 Définitions et concepts de base	1
1.2 Éléments de la problématique	3
1.3 Objectifs de recherche	4
1.4 Plan du mémoire	5
CHAPITRE 2 REVUE DE LITTÉRATURE	6
2.1 Génération de colonnes	6
2.2 Affectation des unités ferroviaires	7
2.2.1 Premiers travaux	8
2.2.2 Modèles par flots	9
2.2.3 Modèles par chemins	10
2.2.4 Explorations de nouvelles problématiques	11
CHAPITRE 3 MODÉLISATION MATHÉMATIQUE DU PROBLÈME	13
3.1 Idée générale du modèle	13
3.2 Construction du sous-problème	13
3.3 Construction du problème maître	16
3.3.1 Variables du problème maître	16
3.3.2 Modèle	17
3.3.3 Contraintes de couplage	20

CHAPITRE 4	MÉTHODES DE RÉOLUTION DU PROBLÈME	22
4.1	Relaxation linéaire du problème	22
4.2	Résolution en nombres entiers	24
4.2.1	Limitation de la taille des graphes	25
4.2.2	Réduction du nombre de variables	26
4.2.3	Utilisation d'heuristiques	27
CHAPITRE 5	RÉSULTATS NUMÉRIQUES	31
5.1	Instances et jeux de données	31
5.1.1	Jeux de données construits aléatoirement	31
5.1.2	Jeux de données réels	34
5.2	Résultats obtenus sur le jeu de données aléatoires	35
5.3	Résultats et performances pour les instances réelles	37
5.3.1	Résultats pour la relaxation linéaire	37
5.3.2	Résolution du problème en nombres entiers	38
CHAPITRE 6	CONCLUSION	44
6.1	Synthèse des travaux	44
6.2	Limitations de la solution proposée	45
6.3	Améliorations futures	45
RÉFÉRENCES		47

LISTE DES TABLEAUX

Tableau 5.1	Caractéristiques des patrons de rame	32
Tableau 5.2	Tailles des instances utilisées	33
Tableau 5.3	Caractéristiques des instances SNCF	34
Tableau 5.4	Caractéristiques des sous-problèmes liées aux instances réelles	35
Tableau 5.5	Résultats obtenus en énumérant les chemins	36
Tableau 5.6	Résultats obtenus avec une méthode de génération de colonnes	36
Tableau 5.7	Performances obtenues avec la méthode de génération de colonnes . .	37
Tableau 5.8	Résultats obtenus par génération de colonnes sur les trois instances .	37
Tableau 5.9	Résultats d'un solveur en nombres entiers avec limite de temps d'attente maximal sur l'instance de Bretagne	39
Tableau 5.10	Résultats de l'heuristique des quantiles pour un temps d'attente maximal de 1h. Meilleure borne connue : 952517	40
Tableau 5.11	Résultats de l'heuristique des quantiles pour un temps d'attente maximal de 3h. Meilleure borne connue : 507613	40
Tableau 5.12	Résultats de l'heuristique des quantiles sans temps d'attente maximal. Meilleure borne connue : 426221	40
Tableau 5.13	Paramètres choisis selon l'avancement de l'algorithme de résolution heuristique	41
Tableau 5.14	Meilleurs résultats entiers identifiés	42
Tableau 5.15	Comparaison de nos meilleurs résultats avec ceux de Giro et avec les meilleures bornes connues	43

LISTE DES FIGURES

Figure 1.1	Exemple schématique de réseau ferroviaire	2
Figure 3.1	Représentation schématique du sous-problème	15
Figure 5.1	Évolution de l'objectif, du nombre d'unités et du nombre de colonnes générées selon le nombre d'itérations	38
Figure 5.2	Comparaison de la valeur de la solution entière trouvée et de la valeur après post-traitement	39
Figure 5.3	Évolution de la valeur de la solution pour le jeu de données des Hauts de France	42

LISTE DES SIGLES ET ABRÉVIATIONS

GC	Génération de colonnes
MIP	Mixed-integer programming
PP	Post-traitement
RMH	Restricted Master Heuristic
SP	Sous-problème

CHAPITRE 1 INTRODUCTION

Un des enjeux majeurs de l'industrie ferroviaire est de déterminer comment faire rouler ses trains de manière à satisfaire ses voyageurs tout en augmentant sa rentabilité. Dans ce but, de nombreux acteurs publics et privés sont impliqués dans la construction de l'offre ferroviaire. Celle-ci se déroule en trois étapes principales : l'aménagement des lignes de chemin de fer, le choix des horaires des voyages et l'organisation des trains pour couvrir les voyages prévus, qui est le problème sur lequel nous avons travaillé. Le surcoût potentiel de devoir acheter et entretenir un train non nécessaire est tel qu'il est très important pour les compagnies ferroviaires de résoudre ce problème de la meilleure manière possible.

Il s'agit d'un sujet de recherche opérationnelle, qui a été beaucoup étudié (voir [1] qui présente une comparaison de différents algorithmes utilisés) mais dont la complexité algorithmique justifie l'exploration régulière de nouvelles méthodes de résolution. Résoudre ce problème de manière exacte sur des réseaux étendus est encore très difficile. La recherche de nouvelles heuristiques pour obtenir des solutions rapidement est ainsi très intéressante, d'autant plus que celles-ci peuvent être adaptées aux problématiques de modification du planning en cas d'imprévu sur un réseau.

1.1 Définitions et concepts de base

Le problème d'affectation des unités ferroviaires est dans la continuité de deux autres problèmes liés. Tout d'abord, il faut aménager un réseau ferroviaire capable d'accueillir des trains. Ce réseau est composé d'un certain nombre d'éléments :

- Des gares, qui vont accueillir les voyageurs cherchant à se déplacer. Une de leurs caractéristiques intéressantes est la taille de leurs quais, qui permet ou interdit à certains trains de s'y arrêter. Il est également possible d'y faire stationner des trains pendant une certaine période de temps.
- Des voies ferrées, routes fixes définissant les déplacements possibles des trains.
- Des dépôts, qui indiquent où les trains peuvent stationner quand ils ne sont pas utilisés. Généralement, les dépôts sont associés à une gare dont ils sont très proches. Chaque dépôt peut être caractérisé par une capacité maximale d'accueil, ainsi que par sa distance à cette gare.

Ces éléments permettent de définir un réseau ferroviaire comme celui présenté sur la figure 1.1

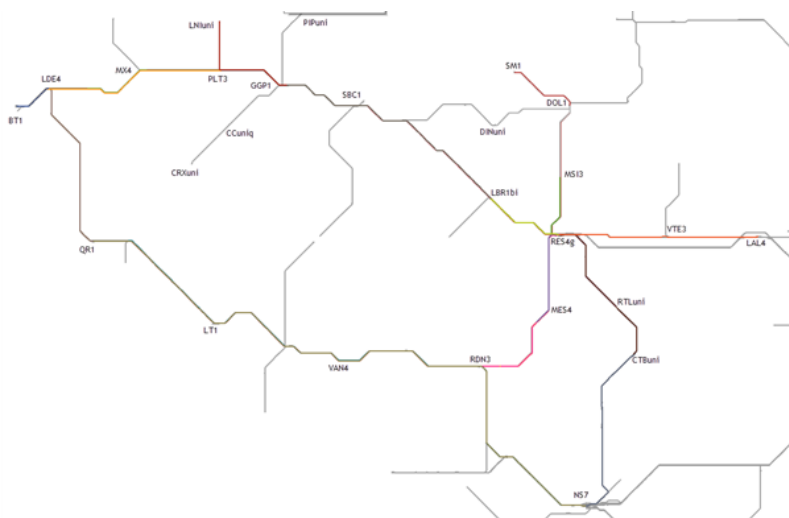


FIGURE 1.1 Exemple schématique de réseau ferroviaire

Une fois ce réseau ferroviaire défini, l'opérateur ferroviaire peut définir des voyages pouvant accueillir des passagers. Pour cela, il doit prendre en compte la demande des voyageurs qui souhaitent se déplacer à une certaine heure vers une certaine ville. Il s'agit donc de proposer des trajets à des horaires précis, parcourant un certain nombre de gares avant d'atteindre leur terminus.

Le problème d'affectation des unités ferroviaires se pose une fois cette grille horaire de voyages définie. L'opérateur dispose de plusieurs données :

- Les types d'unités possibles. Chaque unité est définie par sa taille, qui peut lui interdire l'accès à certaines gares, et par sa capacité indiquant combien de personnes peuvent voyager à son bord. Chaque unité est autopropulsée : nous n'avons ainsi pas à considérer des unités locomotives et des unités wagons, chaque unité remplissant la même fonction. Chaque type d'unités est associé à un coût d'achat spécifique. Les unités peuvent se coupler, c'est-à-dire s'associer l'une à l'autre de manière à former un train plus long pouvant accueillir plus de passagers. De la même manière, elles peuvent se découpler pour couvrir plus de voyages différents. On dit qu'un couplage possible de deux types d'unités est un patron de rame, et on peut énumérer l'ensemble des patrons de rames possibles, puisque le nombre de types d'unités différents est limité.
- Les voyages possibles. Les voyages à couvrir sont définis au préalable par la grille horaire fournie, avec une demande en passagers attendue. On retrouve également les règles définissant les possibilités de voyages à vide, ou deadhead. Ces voyages consistent à déplacer une unité à moindre coût entre deux gares. Ils permettent de rajouter de la flexibilité au réseau, pour qu'une unité puisse intervenir dans n'importe quelle gare

si le modèle y trouve un intérêt.

- Les règles métier, qui correspondent à des contraintes de notre problème. Parmi celles-ci, on retrouve la présence d'un temps de battement minimal après l'arrivée d'un voyage pour permettre à l'équipage de prendre une pause, ainsi que celle d'un temps nécessaire pour effectuer les opérations de couplage et de découplage. On retrouve aussi un certain nombre de règles autorisant ou interdisant certaines opérations de couplage et de découplage dans certaines gares.

Ce problème peut être étudié sur plusieurs échelles temporelles. La première possibilité est de l'étudier à l'échelle d'une journée, en définissant un régime périodique. On chercherait ainsi à ce qu'il y ait autant d'unités de chaque type partant de chaque dépôt, pour que la solution quotidienne puisse être répétée chaque jour. La seconde possibilité est de l'étudier sur une échelle de temps plus longue. Cette possibilité plus réaliste mais plus complexe permet par exemple d'intégrer la nécessité de faire des maintenances régulières pour entretenir la flotte d'unités. Nous avons choisi de nous concentrer sur une approche quotidienne, pour pouvoir développer un modèle plus performant qui pourrait ensuite être adapté à une échelle de temps plus longue.

1.2 Éléments de la problématique

Historiquement, dans de nombreuses grandes compagnies ferroviaires, ce problème était traité à la main par des agents organisant les unités de la manière la plus efficace qu'ils trouvaient. Néanmoins, les recherches sur ce sujet et l'augmentation des puissances de calcul ont justifié que les opérateurs ferroviaires et des entreprises spécialisées développent leur modèle pour résoudre le problème d'affectation des unités ferroviaires.

Dans ce contexte, l'entreprise Giro a développé un modèle et une méthode de résolution du problème d'affectation des unités ferroviaires qu'elle vend à un certain nombre d'acteurs ferroviaires dont la Société Nationale des Chemins de fer Français (SNCF), le transporteur ferroviaire historique français. L'approche de Giro est organisée en deux étapes :

- Dans un premier temps, un modèle détermine des patrons de rames optimaux pour chaque voyage à couvrir. Cette étape est modélisée comme un problème de flot dans un graphe avec des contraintes supplémentaires, où les noeuds correspondent à des voyages couverts par un certain patron de rame, et les arcs à des transitions possibles entre les voyages avec les patrons de rames compatibles. Un sous-réseau espace-temps est également inclus pour modéliser les entrées et sorties de dépôt en milieu de journée.

- Après la résolution de ce modèle, on dispose du patron de rame qui doit couvrir chaque voyage, ainsi que d'enchaînements de voyages couverts par le même patron de rame.
- Dans un second temps, l'approche va déterminer l'horaire de chaque unité, c'est-à-dire la liste des voyages qu'elle va couvrir au cours d'une journée, auxquelles on rajoute les opérations techniques de couplage ou découplage, ainsi que les mouvements de sortie et de rentrée à un dépôt. La complexité de cette étape est réduite par l'utilisation des patrons de rames et des enchaînements de voyages déterminés dans la première étape. Elle est résolue par génération de colonnes, méthode que nous développerons dans la section 2.1.

Cette approche en deux étapes permet d'obtenir des résultats pertinents avec des temps de calcul relativement faibles. Néanmoins, le défaut de cette méthode réside dans sa structure en deux étapes. Comme deux modèles différents avec des objectifs et des paramètres différents ont été optimisés de manière indépendante, Giro ne peut pas évaluer l'écart entre sa solution et la solution optimale théorique. La question se pose alors d'explorer une approche plus directe. En effet, si le problème était résolu en faisant appel à un unique modèle d'optimisation, il serait alors possible de trouver des bornes minorant la valeur de la solution optimale. On pourrait ainsi justifier l'efficacité des solutions entières trouvées.

Cependant, si Giro a adopté cette approche, c'est à cause de la grande complexité du problème. En effet, le nombre de journées possibles pour une unité, c'est à dire le nombre de listes possibles de voyages successifs couverts par cette unité au cours de la journée, augmente exponentiellement avec le nombre de voyages à couvrir et le nombre de patrons de rame possibles. Chercher naïvement à énumérer tous ces enchaînements possibles et à résoudre un problème en nombres entiers avec ces variables devient alors beaucoup trop long. L'enjeu est ainsi de trouver des méthodes permettant d'accélérer la résolution du problème sans sacrifier la qualité de la solution.

1.3 Objectifs de recherche

L'objectif de cette maîtrise est d'explorer une approche plus directe que celle de Giro, en une seule phase. Nous pourrions ainsi dans un premier temps déterminer des bornes permettant de quantifier l'efficacité des solutions de Giro. Dans un second temps, notre objectif est d'obtenir des solutions entières pertinentes pour nos instances. Cependant, en utilisant une approche en une seule phase, la taille des problèmes que nous devons résoudre sera nettement plus grande, et la résolution de ces problèmes se fait en temps exponentiel. Nous devons donc explorer des algorithmes de résolution et des heuristiques nous permettant d'améliorer les

résultats de Giro en maintenant des temps de calcul raisonnables.

1.4 Plan du mémoire

La suite de ce mémoire est constituée de quatre chapitres. Dans le chapitre 2, nous rappelons tout d'abord le principe de l'algorithme de génération de colonnes puis explorons les travaux déjà effectués sur le sujet d'affectation des unités ferroviaires. Dans le chapitre 3, nous rentrons dans le détail de la modélisation formelle du problème que nous proposons. Nous expliquerons dans le chapitre 4 les différents algorithmes et heuristiques que nous avons appliqués pour résoudre notre modèle. Nous présenterons enfin dans le chapitre 5 les instances sur lesquelles nous avons travaillé et les résultats que nous avons obtenus sur celles-ci, avant de récapituler brièvement les travaux effectués et les pistes d'amélioration.

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans cette section, nous survolons l'état de l'art sur ce sujet. Dans un premier temps, nous rappelons le principe de l'algorithme de génération de colonnes, qui est essentiel dans l'application de ce genre de méthodes. Nous présentons ensuite les travaux déjà effectués sur l'affectation d'unités ferroviaires et leurs spécificités.

2.1 Génération de colonnes

Notre travail de maîtrise repose en grande partie sur l'algorithme de génération de colonnes, introduit en 1960 par Dantzig et Wolfe [2]. Barnhart et al. [3] ainsi que Desrosiers et Lübbecke [4] décrivent l'intérêt théorique de cette méthode ainsi que quelques implémentations dans des problèmes classiques. Costa et al. [5] présentent les dernières contributions aux algorithmes de branch-price-and-cut, appliquées au problème de tournées de véhicules.

Nous rappelons donc ici le principe théorique de cet algorithme. Nous rentrerons dans le détail de l'application de cet algorithme à notre problème dans le chapitre 4.1.

L'algorithme de génération de colonnes est conçu pour résoudre des problèmes linéaires comportant un trop grand nombre de variables pour que leur énumération soit possible ou pertinente. Le problème doit pouvoir être séparé en deux problèmes qui seront résolus successivement de façon itérative :

- Un problème maître restreint : il prend en entrée un sous-ensemble des variables et retourne une solution du problème global.
- Un sous-problème : il permet d'identifier quelles sont les variables pas encore prises en compte par le problème maître restreint dont l'ajout permettrait potentiellement d'améliorer sa solution.

L'intérêt de cette méthode est de n'avoir à sélectionner qu'une fraction de toutes les variables possibles pour arriver à la solution optimale. Au lieu de considérer l'ensemble complet de toutes les variables possibles, nous initialisons un ensemble restreint de variables qui garantissent la faisabilité du problème. La génération de colonnes repose sur le principe que, pour un problème de minimisation, rajouter en base une variable de coût réduit non négatif ne peut pas améliorer la solution du problème maître restreint. Le sous-problème est donc utilisé pour chercher les variables de coût réduit minimal. Le but étant de ne pas avoir à énumérer ces variables, il faut que le sous-problème dispose d'une structure lui permettant d'être résolu de manière très efficace. Par exemple, les problèmes de flot à coût minimal ou

de plus court chemin dans un graphe sont particulièrement adaptés.

Ainsi, le problème maître restreint est résolu, et ses variables duales sont calculées et utilisées pour actualiser les paramètres du sous-problème. Il est à noter qu'il est possible d'avoir plusieurs sous-problèmes distincts, auquel cas il faut actualiser les paramètres de chacun de ces sous-problèmes. Ces paramètres doivent permettre de calculer le coût réduit de chacune des variables potentielles du problème maître. Il est ainsi possible de déterminer les variables de coût réduit minimal pour chacun de ces sous-problèmes, et de les ajouter à l'ensemble des variables considérées. Si aucune de ces variables n'a un coût réduit négatif, la solution du problème maître restreint ne peut plus être améliorée et correspond à la solution du problème maître non restreint. Le processus est décrit par l'Algorithme 1.

Algorithme 1 : Génération de colonnes

- 1 Création d'un ensemble de variables tel que la restriction du problème maître à cet ensemble de variables soit réalisable.
 - 2 Résolution du problème maître restreint avec cet ensemble de variables.
 - 3 Actualisation des coûts réduits des variables avec les variables duales associées aux contraintes pertinentes.
 - 4 Résolution des sous-problèmes pour identifier les variables de coûts réduits minimaux.
 - 5 **Si** les solutions des sous-problèmes sont toutes de coût non-négatif :
 - 6 **Alors** la solution du problème maître ne peut plus être améliorée et est optimale.
 - 7 **Sinon** :
 - 8 On rajoute les variables de coût réduit négatif à l'ensemble des variables considérées et on reprend depuis l'étape 2
-

2.2 Affectation des unités ferroviaires

Dans cette section, nous ferons tout d'abord un point sur les premiers travaux effectués sur l'affectation des unités ferroviaires dans la section 2.2.1. Nous explorerons ensuite les modélisations utilisant des problèmes de flot à coût minimal dans un graphe (section 2.2.2) et les modélisations utilisant des problèmes de plus court chemin (section 2.2.3). Enfin, dans la section 2.2.4), nous étudierons les méthodes les plus récentes et les nouveaux aspects de la planification ferroviaire qu'elles intègrent directement au problème d'affectation des unités ferroviaires.

2.2.1 Premiers travaux

Nous nous intéressons au problème d'affectation des unités ferroviaires dans le cadre des voyages de passagers. Néanmoins, un certain nombre de travaux portent sur les transports de marchandises. Ziarati et al. [6] notamment cherchent à déterminer le nombre de locomotives nécessaires pour transporter les différents wagons de marchandises requis. Les contraintes sont assez différentes puisqu'il leur faut respecter une puissance minimale pour déplacer les trains. Les horaires des voyages sont plus souples que ceux des trains de passagers, et il n'existe pas de contrainte de nombre de sièges suffisants pour accueillir les voyageurs. Nous nous concentrons donc sur les articles traitant de voyages passagers.

Nous faisons ensuite la distinction entre les types d'unités considérées. En effet, certains articles traitent de ce problème en considérant une distinction entre les wagons et les locomotives. Les wagons peuvent être couplés de manière à former un train plus long. Chaque train doit néanmoins contenir une locomotive. C'est le cas dans les travaux de Cordeau et al [7]. Il y est développé un modèle cherchant à assigner les locomotives et les wagons de manière optimale et simultanée à un ensemble prédéfini de voyages à couvrir. Les voyages sont ainsi couverts avec des chemins de locomotives et de wagons. Le modèle est associé à une structure de flot dans un graphe dont le coût est minimisé. Il est résolu par une approche de branch-and-bound où les relaxations linéaires sont résolues par génération de colonnes.

Les travaux de Lingaya et al [8] s'inscrivent dans la continuité de cette première étude. Au lieu de considérer locomotives et wagons, cette étude ne s'intéresse qu'à l'affectation des wagons à la grille horaire. Elle prend notamment en compte la position des wagons dans un patron de rame, ce qui est au coeur des recherches futures sur le sujet. Les modèles des articles [7] et [8] sont testés tous les deux sur le réseau de chemins de fer canadien.

D'autres travaux comme ceux de Abbink et al. [9] publiés en 2004 sont aussi liés à ce sujet. L'objectif est toutefois légèrement différent : il s'agit pour eux non plus de minimiser les coûts de transport, mais de maximiser l'organisation des types d'unités de manière à ce qu'un maximum de gens puisse être transporté. La fonction objectif correspond ainsi à une minimisation de la pénurie de sièges dans un train lors des heures de pointe du matin. Leurs variables de décision ne correspondent alors pas au comportement individuel d'une unité, mais plutôt au nombre d'unités de chaque type qui doivent être utilisées. Leur modèle a été testé sur le réseau néerlandais.

Par la suite, les modèles proposés dans la littérature se sont principalement concentrés sur l'optimisation de l'assignation d'unités auto-propulsées. Ils se sont décomposés en deux prin-

cipales catégories, que nous présentons dans les sections 2.2.2 et 2.2.3.

2.2.2 Modèles par flots

La première catégorie de modèles que nous pouvons étudier adopte une approche dite par flots. De la même manière que dans [9], les modèles utilisés ne cherchent pas à modéliser le comportement d’une unique unité mais à comprendre comment organiser la flotte dans son ensemble. Les travaux de Alfieri et al. [10] proposent un des premiers modèles du genre. Il s’agit ici de déterminer le nombre optimal d’unités de chaque type pour chaque voyage à couvrir. Deux modèles sont introduits :

- Le premier a une structure de problème de flot à coût minimal dans un graphe espace-temps. Chaque noeud correspond à un événement, un moment où l’état du réseau change (départ ou arrivée de voyage notamment) et chaque arc à une transition possible entre deux événements.
- Le second rajoute l’ordre des patrons de rames choisis : en effet, il n’est pas possible de le considérer avec le premier modèle. Les auteurs introduisent alors un graphe de transition entre les différents patrons de rame. Il s’agit alors de déterminer un enchaînement de patrons de rame, c’est-à-dire un chemin dans le graphe de transition pour chaque train, respectant les contraintes additionnelles et minimisant le coût du flot du premier modèle.

Le premier modèle peut être résolu par un solveur en nombres entiers. Le second est résolu par des réductions successives du graphe jusqu’à identifier une solution réalisable. Ce modèle a été appliqué au réseau néerlandais.

Peeters et al. [11] développent l’approche proposée précédemment, en introduisant des techniques de résolution par génération de colonnes. Les graphes de transitions sont réutilisés pour déterminer les patrons de rame, mais la résolution du modèle s’effectue en s’intéressant au comportement individuel des trains (et pas des unités). La génération de colonnes intervient donc pour déterminer des chemins optimaux dans ce graphe de transitions, tout en respectant les contraintes additionnelles et minimisant le problème de multi-flot, ce qui permet d’améliorer les temps de résolution.

Ces approches permettent d’obtenir une compréhension globale du réseau et de l’organisation optimale des unités. Il est possible de reconstruire le comportement individuel de chaque unité, puisqu’un flot en nombres entiers peut être décomposé en plusieurs chemins unitaires (voir [12]). Cependant, il est impossible de rajouter des contraintes sur le comportement individuel des unités. Par exemple, les opérations de maintenance ne peuvent être prises en

compte.

2.2.3 Modèles par chemins

Les approches dites par chemins ont pour but au contraire de permettre la prise en compte de ces contraintes. Au lieu de résoudre un modèle par flots donnant l'organisation générale des types d'unités dans le réseau, on cherche à comprendre le chemin suivi par chaque unité individuellement. Cacchiani et al [13] présentent ainsi un modèle en nombres entiers suivant cette approche. Leur sous-problème se présente sous la forme d'un graphe où chaque noeud correspond à un voyage à couvrir, et les arcs sont les enchaînements possibles de ces voyages. Les variables de décision sont ainsi directement liées à un chemin possible pour une unité, c'est-à-dire l'enchaînement de voyages qu'elle va devoir couvrir. Les auteurs notent la complexité de ce modèle et introduisent une heuristique reformulant les contraintes de capacité des unités pour accélérer la résolution.

Les travaux de Haahr et al. [14] développent cette approche dans le cas de réaffectation des unités après un incident. La modélisation est similaire, à ceci près qu'elle est moins contrainte car elle autorise de ne pas couvrir certains voyages, ou de les couvrir avec des patrons de rame inadaptés (tout en pénalisant fortement ces écarts). Le graphe du sous-problème est différent et correspond plutôt à un graphe espace-temps : un noeud correspond maintenant à un événement (arrivée ou départ d'un voyage, opération de couplage, ...) ayant lieu dans une gare, et les arcs décrivent les transitions possibles entre ces noeuds. Les voyages à couvrir sont notamment représentés par des arcs. L'objectif de ces travaux est avant tout de trouver une solution réalisable dans la limite de temps considérée. Ce modèle est résolu par une approche de branch-and-price.

Il est notable que ces deux approches ne permettent pas de prendre en compte l'ordre des unités dans un patron de rame. En effet, on se contente de choisir des chemins assurant que tous les noeuds voyages sont couverts, et on ne s'intéresse pas aux relations entre les chemins choisis. Kroon et al. [1] présentent un modèle par chemins capable d'intégrer les contraintes de couplage et de découplage décrivant l'ordre des compositions au sein d'un patron de rame. Il est inspiré de celui décrit dans [14] mais prend en compte les différences d'ordre dans les patrons de rames. Ce modèle est lui aussi résolu par un algorithme de branch-and-price sur le réseau de chemin de fer néerlandais et sur le réseau de métro de Copenhague.

Nous avons choisi d'explorer une modélisation similaire à celle de [13]. Dans le cadre de notre problématique, choisir une approche par chemins fait sens pour s'opposer au modèle de Giro qui utilise une approche par flots pour déterminer des patrons de rames, et qui reconstruit

ensuite les chemins individuels des unités. De plus, utiliser un réseau espace-temps comme présenté dans [14] et [1] ne nous paraissait pas la meilleure approche sur nos instances. En effet, bien que [1] propose des résolutions d’instances comportant jusqu’à 850 voyages, la structure du réseau étudiée semble très différente avec beaucoup moins de voyages deadhead possibles que dans nos instances, ce qui aurait multiplié le nombre de noeuds.

2.2.4 Explorations de nouvelles problématiques

D’autres modélisations cherchent également à modifier l’objectif du problème et à intégrer d’autres problématiques de conception du réseau ferroviaire. La question des maintenances par exemple est une question récurrente lors de la planification ferroviaire. Maroti et Kroon [15] proposent déjà en 2005 une modélisation fondée sur une minimisation d’un flot multi-types pour intégrer des maintenances urgentes à une affectation d’unités préexistantes. Giacco et al. [16] proposent en 2014 une modélisation permettant de déterminer en deux étapes l’assignation des équipages aux unités et le planning de maintenance des unités. Ces approches sont toutefois distinctes de l’optimisation de l’assignation des unités ferroviaires, et les travaux récents chercheront à intégrer la gestion des maintenances dans une même étape.

Des travaux plus récents cherchent ainsi à intégrer cette question directement dans le modèle d’assignation des unités ferroviaires. Mira et al. [17] proposent un modèle linéaire en nombres entiers pour aborder ce problème. Le sous-problème est ainsi décrit comme un ensemble de tâches, pouvant être soit des voyages à couvrir soit des opérations de maintenance. Un certain nombre d’unités doivent ainsi couvrir une tâche de maintenance au cours de la journée. Le modèle est résolu de manière heuristique de façon à obtenir une solution robuste. Les auteurs effectuent en effet des simulations pour tester l’efficacité du modèle si certaines maintenances prennent plus de temps que d’autres. Amorosi et al. étendent cette modélisation dans [18]. Ils proposent de considérer des niveaux de maintenances, définis par le kilométrage parcouru par chaque unité. Le nombre de maintenances à effectuer n’est ainsi plus fixe mais dépend du planning des unités qui est déterminé en même temps. Ils prennent également en compte les contraintes de capacité dans les entrepôts de maintenance. Néanmoins, le réseau est simplifié pour ignorer les possibilités de voyages deadhead.

Au-delà de la question des maintenances, d’autres aspects de la planification ferroviaire sont pris en compte dans la modélisation de l’affectation des unités ferroviaires. On peut citer par exemple les travaux de Canca et al. [19] qui intègrent la question du nombre et de la localisation des dépôts dans le modèle d’assignation des unités ferroviaires. Ils déterminent dans

une première phase le nombre d'unités nécessaires, puis dans un second temps les chemins parcourus par chaque unité chaque jour. Dans un troisième temps, ils utilisent un algorithme génétique pour déterminer les chemins parcourus par les unités de manière hebdomadaire et les localisations des dépôts de manière à minimiser les temps de voyages à vide. Leurs modèles sont résolus pour une instance liée au réseau ferroviaire de Séville, avec des résultats prometteurs.

Nous avons de notre côté cherché à étendre la modélisation proposée par [13] (chapitre 3) et à la résoudre par génération de colonnes (section 2.1). Nous avons ensuite exploré un certain nombre d'heuristiques pour résoudre le modèle en nombres entiers, que nous présentons dans la section 4.2.

CHAPITRE 3 MODÉLISATION MATHÉMATIQUE DU PROBLÈME

Nous présenterons dans un premier temps le principe du modèle que nous implémentons dans la section 3.1. Nous présenterons ensuite les deux composantes principales de notre modèle, à savoir le sous-problème dans la section 3.2, et le problème maître dans la section 3.3.

3.1 Idée générale du modèle

A la manière de ce qui a été fait dans [1], il nous a paru pertinent d'explorer une modélisation du problème où les principales variables de décision correspondent au comportement individuel d'une unité au cours d'une journée, c'est-à-dire à l'ensemble des voyages qu'elle va effectuer et les opérations de couplage et de découplage qu'elle va subir. De cette façon, nous pouvons séparer nos contraintes en deux catégories : la faisabilité d'un enchaînement de voyages à couvrir et de dépôts par une seule unité, et la faisabilité d'une certaine combinaison de chemins suivis par des unités différentes.

Dans le but de résoudre nos algorithmes par génération de colonnes comme développé dans la section 2.1, nous avons cherché à construire un modèle en deux parties :

- Un sous-problème modélisant le comportement d'une unité au cours d'une journée et permettant de déterminer les chemins réalisables, présenté dans la section 3.2.
- Un problème maître partant de l'ensemble des chemins possibles pour une unité, et qui cherche à déterminer l'ensemble des chemins les plus intéressants pour avoir une solution du problème complet, présenté dans la section 3.3.

Au lieu de considérer tous les chemins possibles pour une unité, nous utiliserons le sous-problème pour déterminer les chemins les plus intéressants pour améliorer les résultats du problème maître, d'où l'intérêt de cette modélisation. Dans ce but, nous utiliserons les variables duales du problème maître pour modifier le coût des arcs du sous-problème. Nous présenterons ces modifications dans la section 4.1.

3.2 Construction du sous-problème

Le sous-problème a pour but d'identifier des chemins prometteurs. Il se présente sous la forme d'un problème de chemins dans un graphe orienté, que nous représentons sur la figure 3.1. Nous présentons tout d'abord quelques ensembles principaux :

- $\mathcal{G}$: Ensemble des gares du réseau

- $\mathcal{T}$: Ensemble des voyages à couvrir, c'est à dire requis par l'emploi du temps préalablement fixé. A chaque voyage $t \in \mathcal{T}$, on peut associer les gares $s_t, e_t \in \mathcal{G}$ correspondant aux gares de départ et d'arrivée du voyage t respectivement.
- $\mathcal{D}$: Ensemble des dépôts du réseau. Chaque dépôt peut être associé à une gare à laquelle il est rattaché : on définit par conséquent la notation $g_d \in \mathcal{G}$ indiquant la gare correspondant au dépôt.

Les noeuds de ce graphe sont de trois types :

- Des noeuds "dépôts d'entrée". Il y en a un par dépôt $d \in \mathcal{D}$. Ils correspondent au lieu d'où l'unité commencera sa journée.
- Des noeuds "dépôts de sortie". Il y en a un par dépôt $d \in \mathcal{D}$. Ils correspondent au lieu où l'unité terminera sa journée.
- Des noeuds "voyages". Il y en a un par voyage $t \in \mathcal{T}$.

Les arcs de ce graphe indiquent les connections possibles entre ces différents noeuds :

- Pour un dépôt $d \in \mathcal{D}$ et un voyage $t \in \mathcal{T}$, si on a $g_d = s_t$, c'est-à-dire si la gare associée au dépôt est celle de départ du voyage, on crée l'arc (d, t) . Si on a $g_d = e_t$, c'est-à-dire si la gare associée au dépôt est celle d'arrivée du voyage, on crée l'arc (t, d)
- Pour un dépôt $d \in \mathcal{D}$ et un voyage $t \in \mathcal{T}$ tels que $g_d \neq s_t$, s'il est possible d'effectuer un voyage à vide entre g_d et s_t avant l'heure de départ du voyage t , on rajoute l'arc (d, t) . De la même manière, on peut rajouter l'arc (t, d) si $g_d \neq e_t$ mais qu'il est possible d'effectuer un voyage à vide entre e_t et g_d après l'heure d'arrivée du voyage t .
- Pour deux voyages $t_1, t_2 \in \mathcal{T}$ tels que $e_{t_1} = s_{t_2}$ et que le temps entre l'heure d'arrivée de t_1 et l'heure de départ de t_2 est supérieur à un temps de pause minimal, on crée l'arc (t_1, t_2) .
- Pour deux voyages $t_1, t_2 \in \mathcal{T}$ tels que $e_{t_1} \neq s_{t_2}$ et tels qu'il est possible d'effectuer un voyage à vide entre les gares e_{t_1} et s_{t_2} entre l'heure d'arrivée de t_1 et l'heure de départ de t_2 , tout en respectant les temps de pause minimaux dans chaque gare, on crée l'arc (t_1, t_2) .

A priori les voyages deadhead pourraient relier n'importe quelles gares entre elles sous l'hypothèse réaliste que le réseau ferroviaire est connexe. Cependant, la liste des voyages possibles est fournie dans le jeu de données réelles décrit dans la section 5.1.2 ce qui permet de limiter le nombre d'arcs dans ces cas. On peut alors ignorer complètement les contraintes de compatibilité des voyages deadhead avec les autres voyages effectués sur le réseau ferroviaire.

Le coût d'un arc correspond à la somme des coûts des activités cachées dans cet unique arc :

- Les manoeuvres de Pullout ou de Pullin correspondant respectivement à sortir ou

- rentrer une unité dans un dépôt
- L'attente dans une gare jusqu'au prochain départ
- Les voyages deadhead nécessaires

Chacune de ces activités a un coût proportionnel au temps passé pour cette activité, l'attente coûtant moins cher par minute que le deadhead ou les manoeuvres de dépôts. De plus, sur chaque arc partant d'un noeud dépôt d'entrée, on rajoute un coût fixe qui correspond au coût de mobilisation d'une unité. Ce coût est significativement plus élevé que les autres coûts calculés puisque la première priorité est de diminuer le nombre d'unités nécessaires. Il dépend du type de l'unité considérée. Nous avons ainsi un sous-problème différent pour chaque type d'unité.

On trie au préalable les noeuds voyages par ordre d'heure de départ croissante. On peut ainsi présenter un sous-problème constitué de d dépôts et n voyages à couvrir comme dans la figure 3.1. Dans cette figure, les noeuds trips correspondent aux noeuds voyages.

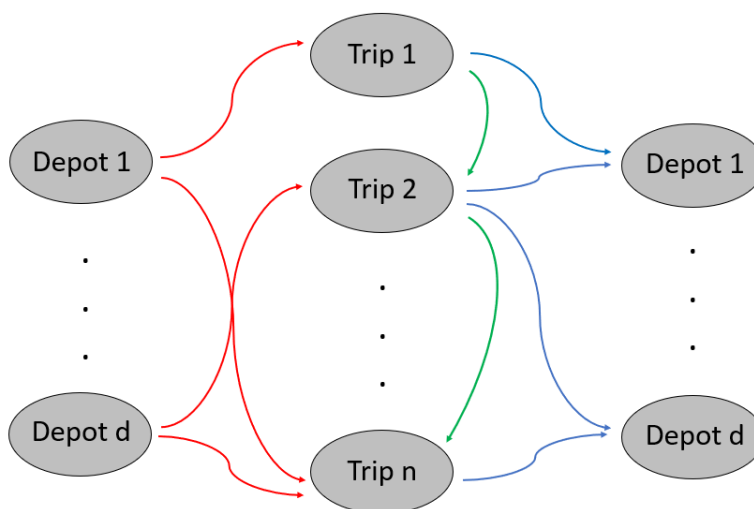


FIGURE 3.1 Représentation schématique du sous-problème

Cette formulation nous permet de déterminer rapidement la totalité des chemins possibles sur ce réseau ferroviaire, qu'on peut regrouper dans un ensemble $\mathcal{P}$. On peut également associer un coût total à chaque chemin $p \in \mathcal{P}$, qui correspond à la somme des coûts des arcs qui le composent. Ce coût dépend du type de l'unité considérée, puisque le coût fixe de mobilisation d'une unité est différent selon son type.

Cependant, cette formulation pourra facilement être adaptée pour sélectionner les chemins les

plus intéressants pour la résolution du modèle complet. En effet, nous chercherons à générer les colonnes qui pourront améliorer la solution courante du problème maître restreint. Le sous-problème correspondra donc à un problème de plus court chemin, où les coûts des arcs auront été modifiés pour prendre en compte les résultats du problème maître restreint. En particulier, les variables duales associées aux contraintes dépendant du choix des chemins seront prises en compte dans le calcul des coûts du sous-problème, ce que nous présenterons dans la section 4.1.

3.3 Construction du problème maître

Nous disposons de l'ensemble $\mathcal{P}$ des chemins réalisables pour une unité. Soit $\mathcal{U}$ l'ensemble des types d'unités disponibles. Notre modèle cherchera donc à utiliser des variables de décision entières, de la forme x_p^u , indiquant combien d'unités de type $u \in \mathcal{U}$ suivent le chemin $p \in \mathcal{P}$ au cours d'une journée. Cependant, un certain nombre d'autres contraintes doivent être respectées, que nous décrirons dans la section 3.3.1 avant d'introduire la formalisation du modèle dans la section 3.3.2.

3.3.1 Variables du problème maître

En premier lieu, nos unités ne voyagent pas forcément seules puisqu'elles peuvent être couplées entre elles. En effet, certains voyages requièrent des compositions particulières pour des raisons d'affluence attendue ou de longueur des quais. On définit donc $\mathcal{C}$ l'ensemble des compositions possibles d'unités ferroviaires, aussi appelées patrons de rames. L'ordre est important : en effet, selon la disposition du quai, une unité devra être libérée avant l'autre et partir, si nécessaire, avant l'autre. Par exemple, si on dispose de deux types d'unités A et B ($\mathcal{U} = \{A, B\}$), l'ensemble des compositions possibles pourrait être $\mathcal{C} = \{A, B, A - A, B - B, A - B, B - A\}$, comme il pourrait être $\mathcal{C} = \{A, B, A - A, A - B, B - A, A - A - A\}$ selon les spécifications techniques de ces types d'unités. En pratique, nous avons travaillé avec des instances comportant jusqu'à 7 types d'unités donnant 18 compositions différentes. Dans tous nos jeux de données, les patrons de rames "mixtes", c'est-à-dire composés de deux types différents d'unités, sont limités aux deux premiers types $\{A\}$ et $\{B\}$. Enfin, aucun patron de rame n'est composé de plus de trois unités différentes.

Chaque voyage peut être couvert par un certain nombre de compositions mais dans la majorité des cas pas par toutes. Il est donc nécessaire d'introduire des variables binaires y_c^t indiquant si un voyage $t \in \mathcal{T}$ est couvert par une composition $c \in \mathcal{C}$, qu'on devra comparer

aux données binaires $\alpha_{t,c}$ indiquant si la composition c est autorisée pour le voyage t . La valeur des variables y_c^t devra également être cohérente avec les valeurs des x_p^u pertinents.

Aussi, nous devons prendre en compte quelles opérations de couplage et de découplage sont possibles dans quelles gares. Il n'est a priori pas possible de réaliser tous les types d'opérations partout. Il est également important de garder la trace des différents retournements du train et donc de la composition au cours d'un arc emprunté par plusieurs unités. Pour ces raisons, nous devons introduire l'ensemble $\mathcal{R}$ correspondant à l'ensemble des arcs du sous-problème. En effet nous devons nous déterminer quels arcs sont parcourus par quelle composition pour déterminer quand les changements de composition ont lieu. On introduit donc les variables binaires z^r indiquant si l'arc r est parcouru au moins une fois par un des chemins retenus. On introduit ensuite les variables binaires $z_{c,c'}^r$, qu'on souhaite égales à 1 si et seulement si l'arc r est parcouru, le noeud de départ de cet arc, qu'on note s_r , est parcouru avec la composition $c \in \mathcal{C}$ et le noeud d'arrivée de cet arc, noté e_r , est parcouru avec la composition $c' \in \mathcal{C}$. Dans le cas où s_r est un noeud dépôt, cette variable vaudra 1 si la composition quittant ce dépôt est c et que les autres conditions sont remplies. De la même manière, si e_r est un noeud dépôt, cette variable vaudra 1 si la composition arrivant à ce dépôt est c' et que les autres conditions sont remplies. Dans le cas où s_r et e_r sont des noeuds voyages et pas des noeuds dépôts, cela revient à la relation suivante.

$$z_{c,c'}^r = 1 \iff y_c^{s_r} = 1 \text{ et } y_{c'}^{e_r} = 1 \text{ et } z^r = 1 \quad (3.1)$$

En utilisant ces variables, on pourra interdire les changements de composition impossibles entre deux voyages spécifiques. On pourra également calculer les coûts de changement de composition.

3.3.2 Modèle

On peut introduire le modèle mathématique utilisé pour ce problème, qui fait appel à la notation suivante.

Ensembles

- $\mathcal{P}$: Ensemble des chemins réalisables pour une unité
- $\mathcal{U}$: Ensemble des types d'unités disponibles
- $\mathcal{C}$: Ensemble des compositions possibles
- $\mathcal{T}$: Ensemble des voyages à couvrir

- $\mathcal{D}$: Ensemble des dépôts du réseau
- $\mathcal{R}$: Ensemble des arcs du sous-problème

Données

- $C_p^u \in \mathbb{R}$: coût du chemin $p \in \mathcal{P}$ pour une unité de type $u \in \mathcal{U}$
- $C_{c,c'} \in \mathbb{R}$: coût de changement de la composition $c \in \mathcal{C}$ à la composition $c' \in \mathcal{C}$
- $v_{t,p} \in \{0, 1\}$: binaire indiquant si le chemin $p \in \mathcal{P}$ passe par le noeud correspondant au voyage $t \in \mathcal{T}$
- $v_{r,p} \in \{0, 1\}$: binaire indiquant si le chemin $p \in \mathcal{P}$ passe par l'arc $r \in \mathcal{R}$
- $s_r \in \mathcal{T} \cup \mathcal{D}$: dépôt ou voyage correspondant au noeud de départ de l'arc $r \in \mathcal{R}$
- $e_r \in \mathcal{T} \cup \mathcal{D}$: dépôt ou voyage correspondant au noeud d'arrivée de l'arc $r \in \mathcal{R}$
- $\delta_{d,p}^S \in \{0, 1\}$: binaire indiquant si le dépôt $d \in \mathcal{D}$ est le dépôt de départ du chemin $p \in \mathcal{P}$
- $\delta_{d,p}^E \in \{0, 1\}$: binaire indiquant si le dépôt $d \in \mathcal{D}$ est le dépôt d'arrivée du chemin $p \in \mathcal{P}$
- $\alpha_{t,c} \in \{0, 1\}$: binaire indiquant si le voyage t peut être couvert par la composition $c \in \mathcal{C}$
- $n_{u,c} \in \mathbb{N}$: entier indiquant le nombre d'unités de type $u \in \mathcal{U}$ dans la composition $c \in \mathcal{C}$
- $N \in \mathbb{N}$: le nombre maximal d'unités par composition

Avec ces notations, on peut formuler le problème comme le programme en nombres entiers suivant :

$$\text{Min}_{x_p^u} \sum_{u \in \mathcal{U}} \sum_{p \in \mathcal{P}} C_p^u x_p^u + \sum_{r \in \mathcal{R}} \sum_{c \in \mathcal{C}} \sum_{c' \in \mathcal{C}} z_{c,c'}^r C_{c,c'} \quad (3.2)$$

$$\text{sujet à : } \sum_{p \in \mathcal{P}} v_{t,p} x_p^u = \sum_{c \in \mathcal{C}} n_{u,c} y_c^t \quad \forall t \in \mathcal{T}, \forall u \in \mathcal{U} \quad (3.3)$$

$$\sum_{c \in \mathcal{C}} y_c^t = 1 \quad \forall t \in \mathcal{T}, \quad (3.4)$$

$$y_c^t \leq \alpha_{t,c} \quad \forall t \in \mathcal{T}, c \in \mathcal{C}, \quad (3.5)$$

$$\sum_{p \in \mathcal{P}} (\delta_{d,p}^S - \delta_{d,p}^E) x_p^u = 0 \quad \forall u \in \mathcal{U}, d \in \mathcal{D} \quad (3.6)$$

$$\sum_{p \in \mathcal{P}} v_{r,p} \left(\sum_{u \in \mathcal{U}} x_p^u \right) \geq z^r \quad \forall r \in \mathcal{R}, \quad (3.7)$$

$$\sum_{p \in \mathcal{P}} v_{r,p} \left(\sum_{u \in \mathcal{U}} x_p^u \right) \leq N z^r \quad \forall r \in \mathcal{R} \quad (3.8)$$

$$\sum_{c,c' \in \mathcal{C}} z_{c,c'}^r = z^r \quad \forall r \in \mathcal{R} \quad (3.9)$$

$$\sum_{\substack{r \in \mathcal{R} \\ s_r=t}} \sum_{c' \in \mathcal{C}} z_{c,c'}^r \geq y_c^t \text{ et } \sum_{\substack{r \in \mathcal{R} \\ s_r=t}} \sum_{c' \in \mathcal{C}} z_{c,c'}^r \leq N y_c^t \quad \forall t \in \mathcal{T}, c \in \mathcal{C} \quad (3.10)$$

$$\sum_{\substack{r \in \mathcal{R} \\ e_r=t}} \sum_{c \in \mathcal{C}} z_{c,c'}^r \geq y_{c'}^t \text{ et } \sum_{\substack{r \in \mathcal{R} \\ e_r=t}} \sum_{c \in \mathcal{C}} z_{c,c'}^r \leq N y_{c'}^t \quad \forall t \in \mathcal{T}, c' \in \mathcal{C}, \quad (3.11)$$

$$\text{Contraintes techniques de couplage} \quad (3.12)$$

$$x_p^u \in \mathbb{N} \quad \forall u \in \mathcal{U}, p \in \mathcal{P} \quad (3.13)$$

$$y_c^t \in \{0, 1\} \quad \forall t \in \mathcal{T}, c \in \mathcal{C} \quad (3.14)$$

$$z^r \in \{0, 1\} \quad \forall r \in \mathcal{R} \quad (3.15)$$

$$z_{c,c'}^r \in \{0, 1\} \quad \forall r \in \mathcal{R}, c \in \mathcal{C}, c' \in \mathcal{C} \quad (3.16)$$

Il s'agit d'un problème de couverture : on souhaite que tous les trajets soient couverts par une composition exactement, comme le décrivent les contraintes (3.4). Cette composition doit faire partie des compositions possibles pour ce trajet comme imposé par (3.5). Dans l'implémentation, nous supprimons ces contraintes en ne définissant que les variables y_c^t acceptables. La bonne définition des variables y_c^t par rapport aux variables de choix x_c^u est imposée par les contraintes (3.3). Se rajoutent ensuite des contraintes additionnelles : on souhaite que la solution trouvée soit périodique pour pouvoir répéter la journée. Pour chaque type d'unité et pour chaque dépôt, on doit donc avoir autant d'unités qui partent et arrivent (3.6). Les contraintes (3.7) à (3.11) permettent de bien définir les variables z^r et $z_{c,c'}^r$, ainsi que de

retrouver l'équivalence (3.1). On calcule ainsi correctement la fonction de coût (3.2), qui dépend du coût des chemins choisis, et du coût des opérations de couplage et de découplage. On peut noter que seules les contraintes (3.3), (3.6), (3.7), (3.8) dépendent directement des variables x_p^u , ce qui sera utile pour la résolution du modèle.

3.3.3 Contraintes de couplage

Aux contraintes (3.3) - (3.11), il faut ajouter les contraintes techniques (3.12) régissant quels enchaînements de composition sont autorisés. Celles-ci sont directement liées aux données considérées. En effet, bien que deux chemins soient possibles indépendamment l'un de l'autre, il est possible que le patron de rame, utilisé lors de leur association lors d'un voyage $t \in \mathcal{T}$ soit contraint. En particulier, l'ordre des unités est concerné. Selon la gare d'arrivée du voyage t , soit l'unité située à l'avant soit l'unité à l'arrière devra partir en second. L'unité partant en second est donc inutilisable en attendant le départ de la première, ce qui interdit un certain nombre de chemins qui étaient a priori possibles dans le graphe du sous-problème.

Nous définissons un sous-ensemble de compositions $\tilde{\mathcal{C}} := \{\{u_1 - u_2\} \in \mathcal{C} \mid u_1 \neq u_2, u_1, u_2 \in \mathcal{U}\}$. Les contraintes d'ordre de couplage et de découplage ne s'appliquent en effet qu'aux patrons de rames constitués de deux unités de type différents. Les contraintes que nous présentons ne concernent donc que les patrons de rame de cet ensemble $\tilde{\mathcal{C}}$.

Nous pouvons définir les trois types de contraintes suivantes :

Couplage : Les jeux de données fournissent toutes les règles métier sur l'ordre dans lequel les unités doivent se coupler selon leur origine. Nous en avons une pour chaque couple de trajets (voyages obligatoires, deadhead et sorties de dépôts) qui finissent dans la même gare. Pour chacun de ces couples, il est indiqué si l'unité arrivant en premier devra être à l'avant, ou à l'arrière du patron de rame créé. Nous pouvons ainsi définir des données booléennes $\chi_{r,r'}$ qui indiquent pour chaque couple d'arcs si l'unité venant de l'arc r va devoir être couplée à l'avant (valeur 0) ou à l'arrière (valeur 1). Nous rajoutons ainsi des contraintes au modèle pour interdire les mauvaises transitions de patrons de rame :

$$z_{\{u_1\},\{u_1-u_2\}}^r \leq \sum_{\substack{r' \in \mathcal{R} \\ e_{r'} = e_r, r' \neq r}} z_{r'} (1 - \chi_{r,r'}), \quad \forall \{u_1 - u_2\} \in \tilde{\mathcal{C}}, r \in \mathcal{R} \quad (3.17)$$

$$z_{\{u_2\},\{u_1-u_2\}}^r \leq \sum_{\substack{r' \in \mathcal{R} \\ e_{r'} = e_r, r' \neq r}} z_{r'} \chi_{r,r'}, \quad \forall \{u_1 - u_2\} \in \tilde{\mathcal{C}}, r \in \mathcal{R} \quad (3.18)$$

Découplage : De la même manière, les jeux de données fournissent les règles indiquant dans quel ordre doivent se découpler les unités. Elles se présentent sous la forme d'un couple de trajets, le premier couvert par le patron mixte, et le suivant couvert par une unité uniquement, et un booléen indiquant si le découplage se fait par l'avant ou non. Le second trajet pourra être couvert par l'unité à l'avant si et seulement si la règle métier indique qu'il s'agit d'un découplage par l'arrière (l'unité arrière est retirée et reste en gare alors que l'unité avant part) ou qu'il s'agit d'un découplage par l'avant mais que l'unité arrière est déjà partie. Pour modéliser cela, nous pouvons utiliser les données fournies pour créer des booléens ϕ_r de valeur 1 si le découplage sur l'arc r se fait par l'avant (et donc que l'unité à l'arrière part en premier) et de valeur 0 sinon. Nous pouvons aussi définir des données $\omega_{r,r'}$ qui indiquent pour deux arcs r et r' provenant du même noeud de départ si l'unité de l'arc r devra se déplacer en première (valeur 0) ou si c'est l'unité de l'arc r' (valeur 1). Ces données se déterminent assez efficacement en vérifiant l'heure de départ des voyages e_r et $e_{r'}$, ou des voyages deadhead préalables si besoin. Nous rajoutons donc des contraintes de la forme :

$$z_{\{u_1-u_2\},u_1}^r \leq \sum_{\substack{r' \in \mathcal{R} \\ s_{r'}=s_r, r' \neq r}} z_{r'} \omega_{r,r'} + (1 - \phi_r), \quad \forall \{u_1 - u_2\} \in \tilde{\mathcal{C}}, r \in \mathcal{R} \quad (3.19)$$

$$z_{\{u_1-u_2\},u_2}^r \leq \sum_{\substack{r' \in \mathcal{R} \\ s_{r'}=s_r, r' \neq r}} z_{r'} \omega_{r,r'} + \phi_r, \quad \forall \{u_1 - u_2\} \in \tilde{\mathcal{C}}, r \in \mathcal{R} \quad (3.20)$$

Retournement : Enfin, les jeux de données fournissent les règles de retournement si deux trajets sont enchaînés. En effet, un patron de rame peut arriver en bout de voie, et repartir dans l'autre sens, ce qui le retourne. Pour cela, des règles métiers pour chaque enchaînement de trajets nous sont fournies avec un booléen pour chacun indiquant s'il y a retournement ou non. Nous définissons ainsi les données ψ_r pour chaque arc de notre sous-problème, de valeur 1 si le patron de rame se retourne au cours de la transition et de valeur 0 sinon. Nous rajoutons ainsi les contraintes :

$$z_{\{u_1-u_2\},\{u_1-u_2\}}^r \leq 1 - \psi_r, \quad \forall \{u_1 - u_2\} \in \tilde{\mathcal{C}}, r \in \mathcal{R} \quad (3.21)$$

$$z_{\{u_1-u_2\},\{u_2-u_1\}}^r \leq \psi_r, \quad \forall \{u_1 - u_2\} \in \tilde{\mathcal{C}}, r \in \mathcal{R} \quad (3.22)$$

Avec toutes ces règles métiers supplémentaires, nous pouvons chercher à résoudre nos modèles.

CHAPITRE 4 MÉTHODES DE RÉOLUTION DU PROBLÈME

Pour résoudre le modèle (3.2) - (3.22), nous procédons en plusieurs étapes. Tout d'abord, dans le but d'obtenir une borne minimale du coût de la solution que nous cherchons, nous résolvons la relaxation linéaire de ce modèle par génération de colonnes comme expliqué dans la section 4.1. Nous cherchons ensuite à obtenir une solution entière pour ce modèle comme développé dans la section 4.2.

4.1 Relaxation linéaire du problème

La valeur optimale de la relaxation linéaire du modèle (3.2) - (3.22) permet d'obtenir une borne inférieure sur la valeur optimale du modèle à nombres entiers. Nous pouvons ainsi non seulement comparer les résultats de nos différentes méthodes de résolution en nombre entiers mais aussi estimer leur qualité par rapport à une valeur minimale qui n'est pas atteignable mais dont on peut supposer qu'elle se rapproche de la solution minimale.

La première étape est d'initialiser un ensemble de chemins (aussi appelés colonnes) qui garantissent la faisabilité du problème maître restreint. Une manière naïve de s'en occuper est de considérer l'ensemble des chemins de la forme "Dépôt de départ - Voyage - Dépôt de fin". Le nombre de ces chemins est contrôlé puisqu'il y a en au plus $|\mathcal{T}| * |\mathcal{D}|^2$. Il n'est théoriquement pas garanti que cet ensemble de chemins fournisse une solution réalisable au problème maître restreint. Cependant, si tous les chemins de cette forme sont possibles, le problème est réalisable. Par conséquent, si on en a besoin, il suffit d'ajouter des chemins de cette même forme qui n'étaient a priori pas possibles, quitte à leur donner un coût très élevé de manière à ce que des chemins possibles prennent leur place dans les premières itérations de la génération de colonnes. En pratique, ce problème n'a pas été rencontré.

A l'issue de la résolution du problème maître restreint, on peut obtenir les valeurs des variables duales, notamment les variables $(\lambda_{u,t})_{u \in \mathcal{U}, t \in \mathcal{T}}$ correspondant aux contraintes (3.3), $(\pi_{u,d})_{u \in \mathcal{U}, d \in \mathcal{D}}$ correspondant aux contraintes (3.6), $(\mu_r^-)_{r \in \mathcal{R}}$ correspondant aux contraintes (3.7) et $(\mu_r^+)_{r \in \mathcal{R}}$ aux contraintes (3.8). Le coût réduit $\bar{c}_p^u$ d'un chemin $p \in \mathcal{P}$ dans le sous-problème de type $u \in \mathcal{U}$ vaut alors :

$$\bar{c}_p^u = c_p^u - \sum_{t \in \mathcal{T}} \lambda_{u,t} v_{t,p} - \sum_{d \in \mathcal{D}} (\delta_{d,p}^S - \delta_{d,p}^E) \pi_{u,d} - \sum_{r \in \mathcal{R}} (\mu_r^+ - \mu_r^-) v_{r,p} \quad (4.1)$$

On note $C_{r,u}$ et $\overline{C}_{r,u}$ le coût original et le coût ajusté d'un arc $r \in \mathcal{R}$ du sous-problème de type $u \in \mathcal{U}$. On initialise $\overline{C}_{r,u}$ comme $C_{r,u}$ et on modifie sa valeur en suivant la procédure suivante :

- Si $s_r = t \in \mathcal{T}$ alors $\overline{C}_{r,u} = \overline{C}_{r,u} - \lambda_{u,t}$.
- Si $s_r = d \in \mathcal{D}$ alors $\overline{C}_{r,u} = \overline{C}_{r,u} - \pi_{u,d}$
- Si $e_r = d \in \mathcal{D}$ alors $\overline{C}_{r,u} = \overline{C}_{r,u} + \pi_{u,d}$
- $\overline{C}_{r,u} = \overline{C}_{r,u} - (\mu_r^+ - \mu_r^-)$

De cette manière, le coût d'un chemin $p \in \mathcal{P}$ dans ce sous-problème actualisé correspondra bien au coût réduit du chemin p dans le problème maître.

On peut alors chercher le plus court chemin dans ce graphe. On note que la structure de ce graphe se prête très bien à un tri topologique. La méthode est notamment expliquée dans [20]. En effet, on peut assez naturellement définir un ordre sur les sommets de telle sorte que pour tous $i, j \in \llbracket 1, |\mathcal{T}| + 2 * |\mathcal{D}| \rrbracket$ tels que $j > i$, il n'existe aucun moyen d'accéder au sommet i depuis le sommet j . On pourrait ainsi ordonner la liste des sommets de la sorte en mettant les dépôts de départ avant les voyages rangés par heure de départ de croissante avant les dépôts d'arrivée. En effet, il n'est pas possible d'accéder à un dépôt de départ depuis n'importe quel noeud et il n'est pas possible d'accéder à n'importe quel noeud depuis n'importe quel noeud dépôt d'arrivée. Comme on a trié les noeuds par heure de départ croissante, si une unité couvre le voyage j , elle ne peut pas couvrir ensuite le voyage i qui est parti avant le départ du voyage j . Ainsi, avec cet ordre de sommets, on peut aussi trier la liste des arcs du graphe de telle sorte que, pour $r, r' \in \mathcal{R}$ tels que $r > r'$, on ait que le noeud s_r soit "plus grand" que $s_{r'}$ ou identique, auquel cas on aurait que e_r est "plus grand" que $e_{r'}$. Trier la liste des arcs dans cet ordre est particulièrement intéressant pour déterminer le plus court chemin dans un graphe. En effet, le plus court chemin pour accéder au voyage i peut être intégralement déterminé par les arcs provenant de sommets plus petits que celui-ci. On peut donc faire l'algorithme de plus court chemin présent dans l'Algorithme 2, qui cherche à déterminer le plus court chemin entre chaque noeud dépôt départ et chaque noeud dépôt d'arrivée :

A tout instant de l'algorithme, la valeur $D[t][d]$ correspond au coût du plus court chemin connu entre le noeud "dépôt de départ d " et le noeud t . En particulier, une fois qu'on a atteint un arc $r \in \mathcal{R}$, on peut être certain que le plus court chemin pour accéder à s_r et tous les noeuds "plus petits" ont été trouvés.

Cet algorithme présente donc plusieurs avantages. La structure de la liste d'arcs permet de n'avoir à parcourir la liste qu'une unique fois pour trouver les plus courts chemins, contrairement à l'algorithme plus général de Bellman-Ford qui requiert de la parcourir $n - 1$ fois où n est le nombre de sommets du graphe dans le pire des cas ([21]). Par ailleurs, on peut égale-

Algorithme 2 : Plus courts chemins depuis chaque noeud dépôt de départ

```

1 Données : la liste des arcs du graphe triée topologiquement
2 Initialiser : D : Dictionnaire qui à chaque sommet associe  $[\infty \text{ pour } d \in \mathcal{D}]$ 
3 Initialiser : P : Dictionnaire qui à chaque sommet associe [None pour  $d \in \mathcal{D}$ ]
4 Initialiser :  $l_P$  : Liste vide
5  $D[d_{\text{départ}}][d] = 0$  pour  $d \in \mathcal{D}$ 
6 Pour chaque arc  $r \in \mathcal{R}$  :
7     Pour chaque dépôt  $d \in \mathcal{D}$  :
8         Si  $D[e_r][d] > D[s_r][d] + \overline{C}_r$  Alors :
9              $D[e_r][d] = D[s_r][d] + \overline{C}_r$ 
10             $P[e_r][d] = r$ 
11 Pour chaque dépôt  $d \in \mathcal{D}$  et  $d_2 \in \mathcal{D}$  :
12     Si  $D[d_{2\text{arrivée}}][d] < 0$ 
13         Reconstruction du plus court chemin entre le dépôt de départ  $d$  et le dépôt
            d'arrivée  $d_2$  en utilisant le dictionnaire P, qu'on ajoute à  $l_P$ 
14 Renvoyer liste-chemins

```

ment très facilement déterminer de cette manière les plus courts chemins entre chaque dépôt de départ et chaque dépôt d'arrivée, et pas seulement le chemin de coût réduit minimal. Cela permet d'identifier beaucoup plus rapidement un grand nombre de chemins intéressants pour chaque itération de génération de colonnes. Le test d'arrêt de l'algorithme de génération de colonnes (voir ligne 5 de l'Algorithme 1) est toujours faisable : si l'Algorithme 2 ne renvoie qu'une liste vide, cela signifie qu'aucun chemin n'a de coût réduit strictement négatif, et que l'on a trouvé la solution optimale du problème maître. Sinon, on rajoute tous les chemins de coût réduit strictement négatifs trouvés et on résout cette nouvelle version du problème maître restreint.

A l'issue de l'algorithme de génération de colonnes, on dispose de la solution minimale de la relaxation linéaire de ce problème, ainsi que de l'ensemble des colonnes générées. Néanmoins, il ne s'agit que d'un indicateur puisque le saut d'intégrité entre le coût de cette solution linéaire optimale et celui de la solution entière optimale est inconnu et peut être significatif. On continue néanmoins vers la résolution du problème en nombre entiers.

4.2 Résolution en nombres entiers

Les principales difficultés de ce projet ont surtout été rencontrées lors de la résolution en nombre entiers de ce problème. En effet, si l'algorithme de génération de colonnes est connu pour être très performant et résoudre efficacement des problèmes linéaires, il est beaucoup

plus compliqué d'obtenir des solutions entières de bonne qualité. La méthode naïve que nous avons adopté en premier lieu a été de résoudre le modèle en nombres entiers en ne prenant en compte que les chemins issus de la génération de colonnes linéaire. Cette approche est présentée sous le nom anglais de *restricted master heuristic* dans [22]. Cependant, nous avons vite rencontré les limites de cette méthode, que ce soit du point de vue de la qualité de la solution trouvée ou du point de vue des temps de calculs exponentiels.

Nous avons donc exploré un certain nombre de méthodes de résolution, que ce soit en limitant le nombre d'arcs des graphes (section 4.2.1), en essayant de réduire le nombre de variables considérées (section 4.2.2), ou en explorant des métaheuristiques pour résoudre le problème (section 4.2.3). Les résultats de ces différentes méthodes seront présentés dans la section 5.3

4.2.1 Limitation de la taille des graphes

Une des idées que nous avons eu pour faciliter la résolution d'un problème en nombre entiers était de limiter la taille du graphe. En effet, une intuition que l'on pourrait avoir sur ce problème est qu'il ne serait pas efficace de faire attendre une unité dans une gare pendant trop longtemps. Ce serait en effet une perte d'argent et de productivité alors qu'aucun voyage obligatoire supplémentaire n'est couvert. En utilisant cette idée, nous avons donc essayé de reconstruire le graphe de notre sous-problème pour limiter fortement le nombre d'arcs, et donc le nombre de chemins possibles. Nous décrivons cette méthode dans l'Algorithme 3.

Algorithme 3 : Limitation de la taille des graphes

- 1 **Données** : $h_{\max}$ une heure maximale d'attente
 - 2 **Initialiser** : Graphe du sous-problème et sa liste d'arcs
 - 3 **Pour** chaque arc du graphe entre deux voyages à couvrir :
 - 4 Calcul du temps entre l'arrivée du noeud précédent et le départ du prochain noeud
 - 5 **Si** ce temps est supérieur à $h_{\max}$
 - 6 Supprimer cet arc du graphe
-

Cet algorithme assez intuitif permet ainsi de diminuer le nombre d'arcs et d'obtenir des solutions entières de manière beaucoup plus rapide. Cependant, plus $h_{\max}$ est faible, plus la solution trouvée sera de faible qualité. On peut alors essayer de réaméliorer la qualité de la solution en post-traitement. Nous avons adopté une approche naïve, en cherchant à "raccrocher les morceaux" entre eux. Si on observe qu'une unité finit son chemin dans un dépôt, mais qu'au moins $h_{\max}$ heures plus tard, une autre unité du même type y commence son chemin à elle, on peut fusionner ces deux chemins en rajoutant un arc d'attente entre

le dernier voyage du 1er chemin et le 1er voyage du second. Nous définissons alors $h_p^{\text{début}}$ et h_p^{fin} pour $p \in \mathcal{P}$ correspondant aux heures de départ et de fin du chemin p . Nous définissons également l'opération de fusion $\oplus$ entre deux chemins $p_1, p_2 \in \mathcal{P}$. Nous rappelons également les données binaires $\delta_{d,p}^S$ et $\delta_{d,p}^E$ indiquant respectivement si un chemin $p \in \mathcal{P}$ commence et finit au dépôt $d \in \mathcal{D}$. Le processus est résumé dans l'Algorithme 4.

Algorithme 4 : Post-traitement

```

1 Données :  $P$  : Pile des chemins choisis pour un type d'unité spécifique
2 Initialiser :  $L$  : Liste des chemins finaux
3 Tant que  $P$  n'est pas vide :
4   Pour chaque chemin  $c_1$  de la pile  $P$  :
5     Initialiser :  $\text{Fus} := \text{Faux}$ 
6      $P = P \setminus \{c_1\}$ 
7     Pour chaque chemin  $c_2$  de la pile  $P$  :
8       Si  $\exists d \in \mathcal{D}, \delta_{d,c_1}^E = \delta_{d,c_2}^S = 1$  alors :
9         Si  $h_{c_1}^{\text{fin}} + h_{\text{max}} \leq h_{c_2}^{\text{début}}$  alors :
10            $\text{Fus} = \text{Vrai}$ 
11            $c_3 = c_1 \oplus c_2$ 
12            $P = P \setminus \{c_2\}$ 
13            $P = P \cup \{c_3\}$ 
14     Si  $\text{Fus} = \text{Faux}$  alors :
15        $L = L \cup \{c_1\}$ 
16 Retourner  $L$ 

```

Ce post-traitement nous permet de retrouver facilement une solution entière de meilleure qualité que la solution entière précédente. Nous l'utiliserons dans quasiment tous nos algorithmes.

4.2.2 Réduction du nombre de variables

Dans un second temps, nous avons cherché à réduire le nombre de colonnes que nous considérons dans le modèle en nombres entiers. Cette fois-ci, il ne s'agit pas de réduire le nombre d'arcs du graphe, mais bien de chercher à considérer moins de chemins réels en espérant trouver parmi ceux-ci les chemins optimaux. Pour cela, nous appliquons une génération de colonnes sur le graphe complet pour déterminer un ensemble de colonnes intéressantes. La méthode naïve utilisée jusque là consiste à intégrer cet ensemble de colonnes dans un solveur en nombres entiers, et à trouver la meilleure solution entière. Ici, nous cherchons à réduire encore plus cet ensemble de colonnes. Nous récupérons le coût réduit de chacune de ces colonnes, déjà calculé par le solveur linéaire pendant la résolution du problème maître restreint, et ne gardons qu'une certaine proportion dont les coûts réduits sont les plus faibles grâce à

un quantile fixé au préalable. La méthode est détaillée dans l’Algorithme 5.

Algorithme 5 : Sélection de colonnes par leur coût réduit

- 1 **Données :** q compris entre 0 et 1, N nombre d’itérations maximal de la génération de colonnes, P_{ini} ensemble de colonnes initiales
 - 2 **Initialiser :** $\hat{\mathcal{P}} = P_{\text{ini}}$
 - 3 **Tant que** la GC n’a pas atteint l’optimalité, ou que N n’est pas atteint :
 - 4 Nouvelle itération de la GC : ajout de colonnes dans $\hat{\mathcal{P}}$
 - 5 **Données :** $\bar{C}_p$ coût réduit pour tous les $p \in \hat{\mathcal{P}}$
 - 6 Déterminer le quantile $\hat{C}_q$ d’ordre q de l’ensemble $\{\bar{C}_p | p \in \hat{\mathcal{P}}\}$
 - 7 **Initialiser :** $\mathcal{P}_{\text{reduit}} = \emptyset$:
 - 8 **Pour** chaque colonne $p \in \hat{\mathcal{P}}$:
 - 9 **Si** $\bar{C}_p \leq \hat{C}_q$ **alors :**
 - 10 $\mathcal{P}_{\text{reduit}} = \mathcal{P}_{\text{reduit}} \cup \{p\}$
 - 11 **Retourner :** $\mathcal{P}_{\text{reduit}}$
-

Ainsi, cet algorithme permet de retenir un certain pourcentage des colonnes qui sont censées être les colonnes les plus intéressantes. Il faut toutefois y rajouter un ensemble de colonnes garantissant que le problème admette une solution, nommé P_{ini} dans l’algorithme. Cet ensemble peut être créé comme l’ensemble de tous les chemins unitaires, c’est-à-dire l’ensemble $\{(d_{\text{depart}}, t), (t, d_{\text{arrivee}})\} | t \in \mathcal{T}, d_{\text{depart}} \in \mathcal{D}, d_{\text{arrivee}} \in \mathcal{D}\}$.

La quantité de variables à considérer dans notre modèle en nombre entiers est bien plus faible que la taille de l’ensemble de colonnes initial. Cependant, rien ne garantit que les colonnes permettant de construire la solution entière optimale aient un coût réduit faible pour la relaxation linéaire du problème. Par conséquent, si on choisit un quantile q trop faible, on risque de perdre beaucoup de colonnes nécessaires pour obtenir des solutions entières intéressantes. L’écart de la meilleure solution que nous pourrions obtenir avec la solution entière optimale devrait donc augmenter quand q diminue, mais le temps nécessaire pour obtenir cette solution devrait diminuer. Nous présentons des résultats obtenus pour les différentes valeurs de ces quantiles dans la section 5.3.

4.2.3 Utilisation d’heuristiques

Enfin, nous avons eu besoin de rechercher d’autres types de méthodes permettant d’obtenir des solutions entières pertinentes. En effet, les approches utilisant un post-traitement sont trop aléatoires pour garantir une optimalité du modèle en nombres entiers, et la méthode en utilisant des quantiles sacrifie trop vite la qualité de la solution trouvée. Nous avons donc eu recours à des heuristiques liées à la structure du problème.

La solution entière optimale est constituée d'un certain nombre de colonnes qui couvrent tous les voyages. Cependant, comme chaque voyage ne peut être couvert que par deux unités au maximum (ou trois dans de rares cas), il paraît très probable qu'on puisse identifier un ensemble de colonnes au sein de cette solution qui couvre un ensemble de voyages et de telle sorte que toutes les autres colonnes ne couvrent aucun de ces voyages. La solution entière pourrait ainsi être décomposée comme la réunion de deux solutions entières optimales sur deux instances complètement disjointes.

Cela nous a donc indiqué qu'il est possible d'approcher la solution entière optimale en décomposant l'instance en deux sous-instances disjointes, résolues séparément. Le choix de cette décomposition pose question ; il est très probable qu'il existe une décomposition idéale permettant d'aboutir à la solution optimale, mais nous ignorons comment la trouver. Nous avons donc décidé d'utiliser une approche itérative.

En partant d'une solution initiale, nous choisissons de manière aléatoire une décomposition de cette solution en deux solutions entières de deux instances disjointes, si possible de tailles équivalentes. Cette décomposition est développée dans l'Algorithme 6. Pour cela, nous sélectionnons un voyage, puis retrouvons les chemins couvrant ce voyage. Nous devons ensuite prendre en compte tous les voyages couverts par ces chemins, puis retrouver pour chacun d'entre eux les chemins les couvrant. Nous procédons ainsi jusqu'à avoir épuisé la liste des voyages à traiter. Cela revient à identifier une composante connexe du graphe constitué des noeuds du sous-problème et uniquement des arcs empruntés par au moins une unité de la solution. Ce processus est décrit dans les lignes 4 à 13 de l'Algorithme 6.

La condition de la ligne 3 impose de relancer la recherche d'une nouvelle composante connexe jusqu'à avoir obtenu un ensemble de composantes connexes dont le nombre de voyages atteint la proportion demandée. Il serait idéal que $\mathcal{T}_1$ et $\mathcal{T} \setminus \mathcal{T}_1$ soient de même taille. Cependant, avec cet algorithme, nous ne maîtrisons pas la taille de $\mathcal{T}_1$: si la dernière composante connexe trouvée contient trop de voyages, nos deux ensembles peuvent être déséquilibrés. La valeur de α doit donc être inférieure à 0.5, de manière à éviter ce scénario. En pratique, nous avons constaté qu'en choisissant $\alpha = 0.4$, il est rare d'observer que $\mathcal{T}_1$ contienne plus de 60% des voyages, en plus de garantir qu'il en contienne plus de 40%. Les résultats présentés dans la section 5.3 ont donc été obtenus avec cette valeur.

Une fois les deux instances définies, nous cherchons de manière indépendante les meilleures solutions entières possibles sur ces deux instances, et nous les fusionnons pour former une solution entière du problème complet. Cette solution est nécessairement de coût inférieur ou égal à celui de la solution précédente, puisque le coût de chaque morceau de la nouvelle solution est inférieur ou égal à celui du morceau correspondant de l'ancienne solution. Cette

Algorithme 6 : Décomposition du graphe

```

1 Données :  $\mathcal{P}_{sol}$  ensemble de chemins de la solution,  $\alpha$  paramètre entre 0 et 1
   indiquant la proportion à partir de laquelle on considère la décomposition
   acceptable.
2 Initialiser :  $\mathcal{T}_1 = \emptyset$  l'ensemble des voyages de la première instance de cette
   décomposition et  $\mathcal{P}_1 = \emptyset$  l'ensemble des colonnes de la solution couvrant  $\mathcal{T}_1$ 
3 Tant que  $\frac{|\mathcal{T}_1|}{|\mathcal{T}|} < \alpha$  :
4   Initialiser :  $t_0$  sélectionné aléatoirement dans  $\mathcal{T} \setminus \mathcal{T}_1$ 
5   Initialiser :  $T := \{t_0\}$  pile de voyages et  $\mathcal{P}_{t_0} = \emptyset$  un ensemble de chemins.
6   Tant que  $T \neq \emptyset$  :
7     Initialiser  $t \in T$ 
8      $T = T \setminus \{t\}$ 
9      $\mathcal{T}_1 = \mathcal{T}_1 \cup \{t\}$ 
10    Pour tous les chemins  $p \in \mathcal{P}_{sol}$  couvrant le trajet  $t$  :
11       $\mathcal{P}_{t_0} = \mathcal{P}_{t_0} \cup \{p\}$ 
12      Pour tous les voyages  $t_p$  couverts par le chemin  $p$  tels que  $t_p \notin \mathcal{T}_1$  :
13         $T = T \cup \{t_p\}$ 
14     $\mathcal{P}_1 = \mathcal{P}_1 \cup \mathcal{P}_{t_0}$ 
15 Renvoyer :  $\mathcal{T}_1$  et  $\mathcal{P}_1$ 

```

solution n'est en revanche pas nécessairement optimale : il s'agit de la meilleure solution de la restriction du problème qui impose cette décomposition précise. Nous procédons donc itérativement en redéfinissant une nouvelle décomposition, et en cherchant à améliorer à nouveau la solution.

Les seules contraintes globales brisées lors de la décomposition en deux sous-instances sont les contraintes de périodicité de dépôt. En effet, il est possible d'obtenir une décomposition où toutes les colonnes partant d'un dépôt A et arrivant à un dépôt B sont dans la première sous-instance, et où toutes celles commençant en B et finissant en A sont dans la seconde. Nous devons donc modifier ces contraintes. Au lieu de forcer une balance départ/arrivée nulle dans chaque dépôt pour chaque type, nous contraignons le problème pour que la balance départ/arrivée de chaque dépôt et type d'unité soit identique à la balance départ/arrivée de la sous-solution initiale pour ce dépôt et ce type d'unité. Ainsi, la réunion des deux sous-solutions obtenues équilibre cette balance départ/arrivée pour chaque dépôt et type d'unité, ce qui nous permet de retrouver la contrainte de périodicité.

Cette heuristique dépend de nombreux paramètres. En effet, nous devons décider d'un nombre d'itérations indiquant combien de fois nous allons répéter ce processus, ainsi que de tous les

paramètres de résolution (nombre d'itérations maximal de la GC, marge du solveur MIP, temps laissé au solveur MIP notamment) pour la résolution des deux sous-instances pour chaque itération. Trouver les paramètres idéaux est donc une tâche complexe, dépendant fortement de l'instance étudiée. Nous présenterons les paramètres choisis et nos résultats en fin de section 5.3.

CHAPITRE 5 RÉSULTATS NUMÉRIQUES

Nous avons appliqué ces algorithmes à des jeux de données et nous allons présenter nos résultats dans ce chapitre. Dans la section 5.1, nous introduirons les jeux de données sur lesquels nous avons travaillé. Nous présenterons ensuite dans les sections 5.2 et 5.3 respectivement, les résultats et performances pour nos algorithmes sur les instances artificielles, et sur les instances réelles. Tous les résultats et temps de calculs ont été obtenus avec le solveur CPLEX, en exécutant les codes en local et en parallélisant quand cela était possible.

5.1 Instances et jeux de données

Au cours de cette maîtrise, nous avons eu l'opportunité de travailler sur plusieurs jeux de données différents. Dans un premier temps, nous avons travaillé nos algorithmes avec des jeux de données construits aléatoirement, de tailles variables. Ces jeux de données nous ont permis de développer progressivement nos modèles et de rajouter progressivement nos contraintes et les évolutions techniques. Dans un second temps, nous avons eu l'opportunité de travailler sur des données provenant de la SNCF, le transporteur ferroviaire français. Ces jeux de données nous ont été fournis avec leur aval par l'entreprise GIRO, qui nous ont accompagné et conseillé tout au long de ce projet.

5.1.1 Jeux de données construits aléatoirement

Dans un premier temps, nous avons développé des jeux de données de manière aléatoire pour obtenir nos premiers résultats.

Nous créons tout d'abord un ensemble de gares et de dépôts selon le nombre souhaité donné en entrée. Nous considérons que toutes les gares sont reliées entre elles, et que toutes les gares sont reliées à tous les dépôts. Les temps de trajet entre chaque connexion sont déterminés de manière aléatoire, entre 1 et 60 minutes pour deux gares et entre 1 et 30 minutes entre une gare et un dépôt. Par ailleurs, nous considérons que les trajets à couvrir entre deux gares ne durent pas forcément le même temps que le temps théorique, et que le temps de voyage va être un peu plus long. Nous créons donc une liste de voyages dont la taille est donnée en entrée. Chaque voyage relie deux gares aléatoires, avec une heure de départ qui est déterminée aléatoirement entre 8h00 et 20h00 et avec un temps de trajet qui est un multiple du temps de trajet théorique entre les deux gares. Le facteur de multiplication est choisi aléatoirement entre 1 et 2. Nous attribuons aussi une demande aléatoire à chaque voyage, entre 1 et 300

passagers pour déterminer quel patron de rame peut couvrir quel voyage.

A partir de ces paramètres, nous pouvons construire les sous-problèmes pour notre modèle. Pour cela, nous avons aussi défini deux types d'unités différents, A et B, pouvant accueillir respectivement 120 et 180 passagers et coûtant respectivement 5000 et 5500. Ces 2 unités peuvent être couplées en 5 patrons de rame différents : $\{A\}$, $\{B\}$, $\{A-A\}$, $\{B-B\}$ et $\{A-B\}$, dont les caractéristiques et coûts sont résumés dans le tableau 5.1.

L'unité B est ainsi beaucoup plus intéressante a priori, et est nécessaire pour couvrir environ 20% des trajets. Néanmoins, 40 % des trajets peuvent être couverts par une unité A seule. Il est donc probablement intéressant pour le modèle d'investir également dans des unités de ce type. Il est à noter qu'ici, la composition $\{A-B\}$ pourrait indifféremment être appelée $\{B-A\}$; on ne prend pas en compte les contraintes d'ordre. Implémenter un ensemble de règles métier de couplage de manière aléatoire aurait été fastidieux et très peu réaliste. Nous nous contentons donc de ces 5 patrons de rame, et implémentons les règles de couplage dans les jeux de données réels.

Les trois seuls paramètres variables lors de la création de ce jeu de données sont les nombres de dépôts, de gares et de voyages à couvrir. Ces paramètres permettent de déterminer la taille du sous-problème que nous allons étudier et sont donc les paramètres les plus importants et les seuls que nous souhaitons modifier. Cependant, nous nous sommes contentés de modifier le nombre de voyages qui permettaient de complexifier naturellement le réseau et le sous-problème. Nous avons donc maintenu fixés à 2 le nombre de dépôts et à 10 le nombre de gares total du réseau. Nous avons fait varier ce paramètre dans un grand échantillon de valeurs, que nous présentons dans le tableau 5.2.

Comme vu dans la modélisation du sous-problème, le nombre de noeuds correspond directement au nombre de voyages à couvrir auxquels on ajoute deux fois le nombre de dépôts, soit ici 4. Les arcs du sous-problème correspondent au nombre de transitions entre deux voyages à couvrir, ou entre un voyage à couvrir et un noeud dépôt. La conception du réseau, où toutes les gares sont reliées entre elles et aux dépôts, fait qu'un arc relie deux noeuds s'il y

TABLEAU 5.1 Caractéristiques des patrons de rame

Patrons	A	B	A-A	B-B	A-B
Capacité	120	180	240	360	300
Prix	5000	5500	10000	11000	10500
Trajets pouvant être couverts (%)	40	60	80	100	100

TABLEAU 5.2 Tailles des instances utilisées

Nombre de voyages	Noeuds du SP	Arcs du SP	Nombre de chemins possibles
5	9	21	40
10	14	48	380
20	24	144	3036
30	34	348	14192
40	44	669	42148
50	54	992	78316
60	64	1382	176724
100	104	3559	2108996

a suffisamment de temps entre les deux voyages pour intégrer un éventuel voyage deadhead et les temps d'attente minimaux. Le nombre d'arcs du sous-problème va donc augmenter quadratiquement avec le nombre de noeuds du sous-problème (donc le nombre de voyages à couvrir). Enfin, le nombre de chemins possibles dépend exponentiellement du nombre de voyages. En effet, dans le pire des cas, rajouter un seul noeud voyage relié à tous les autres noeuds voyages et à tous les noeuds dépôts multiplie le nombre de chemins possibles par deux (pour chaque chemin du sous-problème sans ce voyage on peut choisir d'inclure ce nouveau voyage ou pas). En réalité, les noeuds ne sont pas tous reliés entre eux, mais le nombre de chemins augmente tout de même de façon exponentielle, ce qu'on observe sur les instances.

Ces différentes instances présentent un certain nombre d'avantages :

- Pour les plus petites instances, on peut faire tourner nos modèles très rapidement. Cela permet de les tester, puisque l'on peut énumérer tous les chemins possibles et utiliser un modèle en nombres entiers sur ces chemins en un temps de calcul raisonnable.
- La taille des plus grandes instances se rapproche de la taille des instances réelles, ce qui nous a permis de commencer à tester les performances des différents algorithmes. Nous avons aussi pu vérifier qu'énumérer tous les chemins possibles dans un modèle en nombres entiers devient irréaliste pour ces instances.

Ce jeu de données a toutefois des limites évidentes : la structure "en étoile" où toutes les gares sont reliées entre elles saisit très mal la complexité des réseaux ferroviaires réels. La structure du sous-problème et la densité des arcs sera donc différente. Par ailleurs, les temps de trajet ne sont probablement pas assez divers, ce qui modifiera également la forme du sous-problème. Enfin, il est évidemment difficile d'implémenter les différentes contraintes techniques liées au réseau de voies ferrées, à l'ordre des unités ou des différents voyages deadhead possibles et à quelles heures. Pour toutes ces raisons, il était nécessaire et plus intéressant de travailler sur des instances réelles.

5.1.2 Jeux de données réels

Nous avons pu travailler sur trois jeux de données issues du réseau ferroviaire français. Ces données correspondent aux Trains Express Régionaux (TER) de trois régions : la Bretagne, la Normandie et les Hauts de France. Pour chacun de ces réseaux, nous disposons des voyages à couvrir, avec leurs horaires et leurs patrons de rame autorisés qui sont décrits dans le tableau 5.3.

Chacune de ces instances a ses spécificités. Nous avons commencé à travailler avec le jeu de données de Bretagne, dont le nombre de voyages est le plus élevé. Cependant, il n'admet que deux types d'unités qui peuvent être couplés entre eux deux à deux pour former donc six patrons de rames. Nous avons obtenu ensuite le jeu de données de Normandie, qui contient significativement moins de voyages à couvrir, et qui rajoute deux types d'unités, qui ne peuvent se coupler qu'entre eux deux à deux ce qui rajoute quatre patrons de rames. Enfin, nous avons travaillé sur le jeu de données des Hauts de France, qui contient un nombre de voyages à couvrir similaire à celui de Bretagne, et un nombre de types d'unités encore un peu plus élevés. Un des patrons de rame possible contient trois fois un certain type d'unité. Quel que soit le nombre de types de l'instance, il n'y a toujours que deux types différents qui peuvent se coupler entre eux. Les contraintes pour distinguer l'ordre des unités ne s'appliquent donc qu'à ces deux types-là quoi qu'il arrive, ce qui limite la complexité supplémentaire du modèle.

En plus de la liste des voyages à couvrir, viennent d'autres trajets :

- La liste des sorties de dépôt possibles
- La liste des rentrées en dépôt possibles
- La liste des voyages deadhead possibles

Chacune de ces listes a été créée suite à un prénettoyage des données fait par Giro, qui déterminait toutes les heures de départ possibles et toutes les heures d'arrivée possibles selon quels voyages et/ou quel dépôt ce trajet doit relier. Nous avons toutefois rajoutés un certain nombre de trajets entre les dépôts et les voyages, sans quoi notre problème n'était pas réalisable. Cela est dû à des règles métier que nous n'avons pas implémentées, qui autorisaient

TABLEAU 5.3 Caractéristiques des instances SNCF

Réseau	Voyages	Gares	Types d'unités différents	Patrons possibles
Normandie	114	20	4	10
Hauts de France	356	22	7	17
Bretagne	367	26	2	6

une unité à stationner sur certaines voies pendant la nuit au lieu de rentrer dans un dépôt. Pour le jeu de données de Normandie, nous avons ainsi une liste de 406 voyages deadhead possibles, 652 possibilités pour sortir une unité d'un dépôt et 658 possibilités pour faire rentrer une unité dans un dépôt.

Avec toutes ces données, nous avons pu construire le graphe de notre sous-problème en respectant les différentes règles métiers. Les tailles des sous-problèmes sont décrites dans le tableau 5.4.

5.2 Résultats obtenus sur le jeu de données aléatoires

Dans un premier temps, nous allons présenter nos résultats sur les instances que nous avons générées aléatoirement. La première méthode pour obtenir des résultats sur ces instances, et la raison pour laquelle nous avons choisi de commencer avec de petites instances est de chercher à énumérer tous les chemins possibles et à optimiser un modèle en nombres entiers dessus. Cette méthode naïve donne des résultats efficaces sur les instances les plus modestes. On présente dans le tableau 5.5 les instances et le nombre de chemins considérés, ainsi que les coûts optimaux trouvés, le nombre final d'unités à utiliser, et le temps nécessaire pour résoudre ce modèle.

On y observe bien qu'à partir de 60 voyages à couvrir, le modèle devient incapable de fournir des solutions entières dans le temps imparti de 300 secondes. Le nombre de variables à considérer devient trop important pour un solveur en nombres entiers.

Nous avons donc continué avec notre idée initiale de résoudre jusqu'à optimalité la relaxation linéaire du modèle avec une génération de colonnes, et de résoudre ensuite le modèle en nombres entiers avec les colonnes ainsi obtenues. Dans le tableau 5.6, nous présentons pour chaque instance, le coût de la solution linéaire optimale z_{PL} qui est déterminée pendant la génération de colonnes, le coût de la meilleure solution entière connue grâce aux méthodes d'énumération z_{PNE} , et le coût de la solution entière identifiée après la génération de co-

TABLEAU 5.4 Caractéristiques des sous-problèmes liées aux instances réelles

Réseau	Noeuds du SP	Arcs du SP
Normandie	154	1609
Hauts de France	400	10290
Bretagne	419	10040

TABLEAU 5.5 Résultats obtenus en énumérant les chemins

Nombre de voyages	Chemins possibles	Coût optimal	Unités	Temps de calcul (s)
5	40	28490	5	0.001
10	380	38220	5	0.03
20	3036	67285	10	0.43
30	14192	101410	15	1.1
40	42148	116790	18	3.2
50	78316	150860	25	36.4
60	176724	?	?	>300
100	2108996	?	?	>300

lonnes par la restricted master heuristic z_{RMH} . Nous y déterminons également, quand cela est possible, l'écart relatif entre les coûts z_{PNE} et z_{RMH} . Nous présentons ensuite dans le tableau 5.7 les performances temporelles de cette méthode, en indiquant le nombre de chemins sélectionnés par la génération de colonnes, leur proportion par rapport au nombre de chemins totaux et le temps de calcul nécessaire pour aboutir à une solution entière.

Ces résultats nous indiquent plusieurs choses. Tout d'abord, on remarque que le coût des solutions entières identifiées après la génération de colonnes est très proche de celui des solutions entières optimales, ici 1.86% dans le pire des cas mais plus souvent de l'ordre du dixième de %. Par ailleurs, pour les instances de plus de 50 voyages, notre méthode permet d'obtenir une solution dans le délai des 300 secondes, ce qui n'est plus le cas quand on cherche à énumérer les chemins possibles. Nous constatons aussi que la proportion de variables retenues après la génération de colonnes décroît plus la taille de l'instance augmente, ce qui est encourageant pour l'application à nos instances réelles. Nous constatons aussi cependant que l'écart entre le coût de la meilleure solution linéaire et le coût de la meilleure solution entière semble particulièrement significatif avec cette modélisation. Tant que nous pouvons

TABLEAU 5.6 Résultats obtenus avec une méthode de génération de colonnes

Voyages	z_{PL}	z_{PNE}	z_{RMH}	Écart (%)
5	22231	28490	28490	0
10	29445	38220	38350	0.34
20	51861	67285	67305	0.03
30	73830	101410	101895	0.48
40	93303	116790	118965	1.86
50	115575	150860	151615	0.50
60	158564	?	205205	?
100	202701	?	255135	?

TABLEAU 5.7 Performances obtenues avec la méthode de génération de colonnes

Voyages	Chemins	Chemins générés	Proportion de chemins générés (%)	Temps (s)
5	40	3	7.5	3.0
10	380	70	18.4	3.0
20	3036	235	7.7	7.9
30	14192	466	3.3	7.7
40	42148	699	1.7	13.2
50	78316	1052	1.3	38.5
60	176724	1310	0.7	35.4
100	2108996	3164	0.2	250.5

calculer les solutions entières optimales, nous savons que nous en sommes proches, mais pour les plus grandes instances nous devons nous contenter de cet écart entre les coûts de la meilleure solution linéaire et de la meilleure solution entière trouvées, qui n'est pas forcément révélateur de la qualité de la solution. C'est un problème que nous avons retrouvé dans la suite. Nous avons ensuite appliqué ces méthodes aux instances du réseau ferroviaire français.

5.3 Résultats et performances pour les instances réelles

Après avoir obtenu ces résultats sur les instances générées aléatoirement, nous avons appliqué ces algorithmes sur nos jeux de données issus de la SNCF.

5.3.1 Résultats pour la relaxation linéaire

Dans un premier temps, nous avons appliqué l'algorithme de génération de colonnes aux trois instances, en résolvant jusqu'à optimalité. Les résultats sont présentés dans le tableau 5.8. Nous y avons indiqué le nombre de noeuds du sous-problème, le coût de la solution de la relaxation linéaire, le nombre d'unités qui sont nécessaires pour cette solution et le temps de calcul nécessaire à l'obtention de cette solution.

Tout d'abord, le jeu de données de Normandie est de taille suffisamment réduite pour que

TABLEAU 5.8 Résultats obtenus par génération de colonnes sur les trois instances

Réseau	Noeuds du SP	Coût optimal	Unités	Temps de calcul
Normandie	154	157125	25	2'18"
Hauts de France	400	412195	77	1h32'17"
Bretagne	419	426221	76	52'17"

la génération de colonnes se résolve jusqu'à optimalité en un temps réduit. Au contraire, les jeux de données de Bretagne et des Hauts de France nécessitent plus de temps de calcul pour obtenir la meilleure solution. En particulier, la génération de colonnes pour le jeu de données des Hauts de France est extrêmement longue. Cela est dû au plus grand nombre de patrons de rames possibles pour cette instance, qui multiplie la complexité du problème. Cependant, au bout de 25 minutes on obtient un résultat à 1% de la valeur finale. Nous pouvons ainsi arrêter la génération de colonnes bien avant l'optimalité et être confiant d'avoir une solution proche du coût de la meilleure solution. Pour justifier cette affirmation, nous présentons l'évolution de la meilleure solution linéaire connue, du nombre d'unités nécessaires ainsi que du nombre de colonnes générées lors de l'algorithme de génération de colonnes appliqué au jeu de données de Bretagne sur la figure 5.1.

Nous pouvons maintenant chercher à résoudre ces instances pour avoir une solution entière.

5.3.2 Résolution du problème en nombres entiers

Nous pouvons appliquer nos différents algorithmes et heuristiques à ces jeux de données. Les résultats que nous présentons dans les prochaines sous-sections sont des résultats obtenus sur le jeu de données de Bretagne.

Réduction du nombre d'arcs : Dans un premier temps, nous avons réduit la taille du graphe en rajoutant une contrainte de temps d'attente maximal. Nous avons effectué la génération de colonnes pendant 20 itérations, et intégré les colonnes résultantes dans un solveur en nombres entiers. Nous avons enfin appliqué notre méthode de post-traitement aux solutions entières trouvées. Nous compilons dans le tableau 5.9 les meilleurs résultats trouvés pour différentes heures d'attente maximales. La colonne de temps indique le temps qu'il a fallu pour identifier cette solution. Enfin nous indiquons l'écart avec le coût de la meilleure

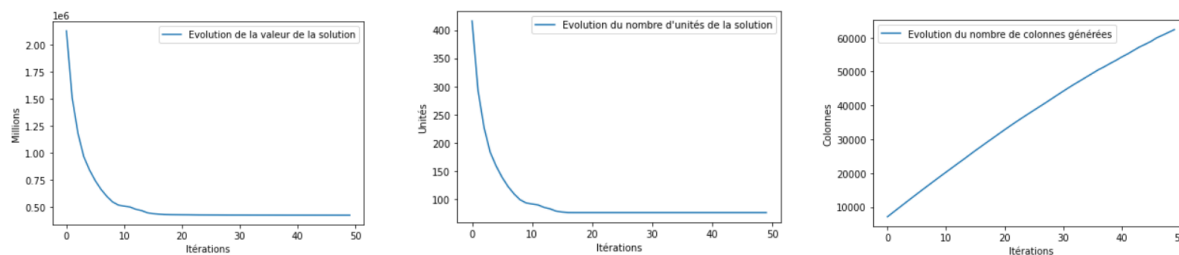


FIGURE 5.1 Évolution de l'objectif, du nombre d'unités et du nombre de colonnes générées selon le nombre d'itérations

solution linéaire identifiée.

TABLEAU 5.9 Résultats d'un solveur en nombres entiers avec limite de temps d'attente maximal sur l'instance de Bretagne

Attente max (h)	Arcs du SP	Valeur	Unités	Temps	Écart (%)
1	3414	546227	89	11'19"	28.2
2	4004	553169	85	29'29"	29.8
3	4613	?	?	>2h	?

Ces résultats restent assez éloignés de la meilleure borne linéaire connue. Cependant, il ne s'agit pas de la valeur associée à la solution entière optimale du problème. Il est en effet très probable qu'il existe un saut d'intégrité entre le coût de la solution linéaire optimale et le coût de la solution entière optimale qui ne soit pas négligeable. Cependant, les valeurs que nous avons pour l'instant ne peuvent pas nous satisfaire. Par ailleurs, nous présentons sur la figure 5.2, les valeurs de quelques solutions entières identifiées au cours de la résolution, ainsi que leur valeur après post-traitement. Nous pouvons ainsi remarquer le caractère aléatoire de ce post-traitement. Nous n'avons aucun contrôle sur la qualité de la solution finale, seulement sur la qualité de la solution de la restriction à 1h d'attente maximale. C'est évidemment un problème que nos résultats soient hasardeux et non uniquement dépendant de la qualité des résultats de l'optimisation.

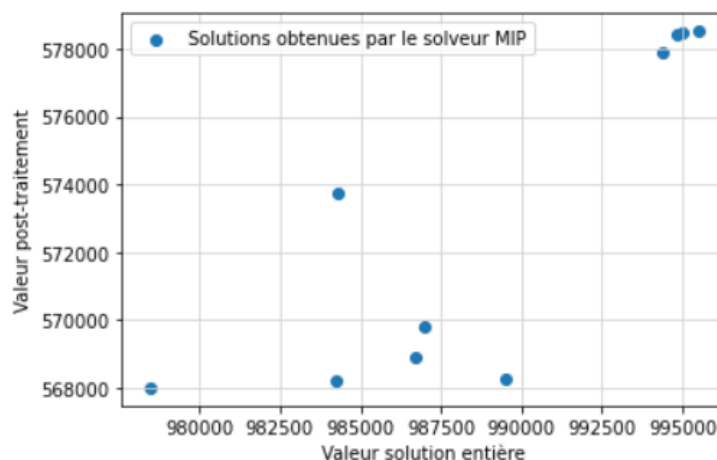


FIGURE 5.2 Comparaison de la valeur de la solution entière trouvée et de la valeur après post-traitement

Utilisation de quantiles : Nous avons donc cherché à utiliser la méthode des quantiles présentée dans la section 4.2.2. Nous donnons les résultats obtenus pour des valeurs de

contraintes d'attente maximale de 1h (tableau 5.10), de 3h (tableau 5.11) et sans contrainte (tableau 5.12). Pour chacune de ces valeurs, nous présentons pour quelques valeurs de quantile différentes le nombre de colonnes retenues, la meilleure valeur de la fonction objectif trouvée dans le temps de résolution accordé, et le temps nécessaire pour trouver cette solution. Nous avons imposé au solveur en nombres entiers un temps de résolution maximal de 2 heures, et lui avons accordé une marge de tolérance de 5%. A noter que nous n'utilisons pas de post-traitement ici, pour comprendre à quel point on s'approche du coût de la solution linéaire de la restriction correspondante. Nous indiquons en légende de chacun de ces tableaux le coût de la solution de la relaxation linéaire associée, et donc une borne sur le coût des solutions entières.

TABLEAU 5.10 Résultats de l'heuristique des quantiles pour un temps d'attente maximal de 1h. Meilleure borne connue : 952517

Quantile	Nombre de colonnes	Valeur	Temps
10%	7156	961068	5'08"
20%	11178	973900	7'10"
30%	14572	978189	8'51"
50%	20672	977985	15'33"
100%	25280	961753	24'46"

TABLEAU 5.11 Résultats de l'heuristique des quantiles pour un temps d'attente maximal de 3h. Meilleure borne connue : 507613

Quantile	Nombre de colonnes	Valeur	Temps
0.1%	3501	623375	41'32"
2.5%	5123	622345	1h15'
5%	7183	699641	1h27'

TABLEAU 5.12 Résultats de l'heuristique des quantiles sans temps d'attente maximal. Meilleure borne connue : 426221

Quantile	Nombre de colonnes	Valeur	Temps
0.1%	3494	818140	45'
1.0%	3745	791930	1h08'
2.0%	4972	778041	1h09'

Ces tableaux nous permettent d'obtenir des informations intéressantes. Dans un premier temps, nous pouvons observer, que pour un temps d'attente maximal de 1h, nous obtenons des solutions dont les coûts sont proches de la meilleure borne linéaire connue à 5% près. Notre méthode peut donc sembler pertinente. De plus, quand nous l'appliquons au problème

avec un temps d'attente maximal de 3h, nous obtenons des solutions entières en moins de 2h. Nous observons néanmoins que dès que nous gardons plus de 5% des colonnes, la qualité de la solution se dégrade. En effet, on ne trouve qu'une solution de bien moins bonne qualité au bout d'un temps plus long. Le temps nécessaire au solveur pour retrouver une solutions de même coût que les précédentes semble ainsi plus long. Cet effet est d'autant plus visible sur les résultats sans aucune contrainte d'attente maximale, où une solution médiocre est trouvée. Cela montre ainsi les limites de cette approche.

Heuristique dichotomique Enfin, nous avons appliqué l'algorithme présenté dans la section 4.2.3. Nous avons tout d'abord décidé du nombre d'itérations que nous effectuons au cours de cet algorithme. Pour nos instances, nous avons observé que 10 itérations étaient un bon compromis entre temps de calcul et qualité de la solution finale. Pour chaque itération, nous avons dû ensuite définir avec quelles valeurs de paramètres les deux sous-instances sont résolues. Les paramètres que nous considérons sont le nombre d'itérations de la génération de colonnes, le temps laissé au solveur MIP pour identifier la meilleure solution, et la tolérance sur le saut d'optimalité du solveur MIP. Nous présentons les paramètres choisis dans le tableau 5.13.

TABLEAU 5.13 Paramètres choisis selon l'avancement de l'algorithme de résolution heuristique

Itérations restantes	Itérations de la GC	Temps du MIP (s)	Tolérance du MIP (%)
Entre 6 et 10	20	600	5
Entre 3 et 5	30	1200	3
Entre 1 et 2	40	1800	1

Ces paramètres sont définis de manière à avoir une résolution de plus en plus précise de l'algorithme au fil de l'amélioration de la solution. Les premières itérations permettent d'obtenir rapidement une solution grossière pour chaque sous-instance, et les dernières itérations nous permettent d'obtenir des solutions plus précises et de trouver des améliorations plus fines. Nous avons testé plusieurs ensembles de paramètres, et celui-ci a été le plus efficace. Nous présentons sur la figure 5.3 l'évolution de la valeur de la solution pour le jeu de données des Hauts de France au fil des itérations de notre heuristique de dichotomie.

Nous observons une amélioration régulière de la solution. L'inflexion lors de la dernière itération semble indiquer qu'avec encore plus de temps de calcul et des paramètres différents, il pourrait être possible de continuer à améliorer la meilleure solution connue. Cependant, avec les paramètres fournis actuellement, l'algorithme tourne déjà pendant environ 5h. Nous pouvons par ailleurs utiliser notre méthode de post-traitement pour chercher à améliorer

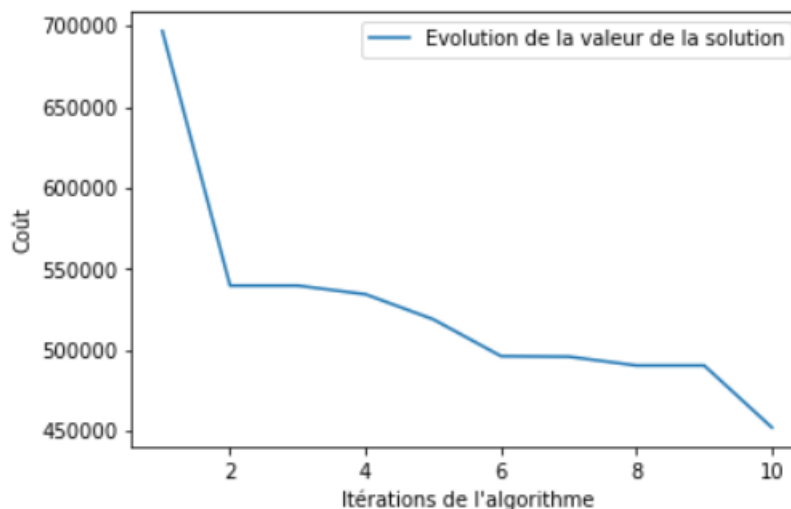


FIGURE 5.3 Évolution de la valeur de la solution pour le jeu de données des Hauts de France

encore un peu la qualité des solutions obtenues.

Résultats finaux Nous présentons finalement, pour chaque jeu de données, les meilleures solutions entières que nous avons obtenues. Pour le jeu de données de Normandie, il nous a suffi d'intégrer les colonnes générées dans un solveur en nombres entiers pour obtenir une excellente solution. Pour les deux autres, nos meilleurs résultats ont été obtenus en appliquant notre méthode de post-traitement aux résultats obtenus à l'issue de l'heuristique de dichotomie. Les résultats et temps de calculs sont présentés dans le tableau 5.14. Nous présentons de plus dans le tableau 5.15 les résultats obtenus par Giro pour les comparer aux nôtres, en rappelant la meilleure borne linéaire que nous avons identifiée.

Nous pouvons observer que nous obtenons des résultats comparables à ceux de Giro. Le jeu de données de Normandie prouve que notre méthode est capable de résoudre les petites instances en quelques minutes. La méthode de Giro prouve toutefois son efficacité sur les plus grandes instances ; leurs résultats sont meilleurs et leur temps de calculs plus faibles selon

TABLEAU 5.14 Meilleurs résultats entiers identifiés

Réseau	Coût optimal	Unités	Temps de calcul	Méthode utilisée
Normandie	157523	25	3'42"	RMH
Hauts de France	442659	81	5h	Dichotomie + PP
Bretagne	477893	82	5h	Dichotomie + PP

TABLEAU 5.15 Comparaison de nos meilleurs résultats avec ceux de Giro et avec les meilleures bornes connues

	Coûts	Coûts Giro	Meilleure borne linéaire
Normandie	157523	157224	157125
Hauts de France	442659	432791	412195
Bretagne	497893	479815	426221

leurs estimations. La différence entre les coûts obtenus correspond à un écart de deux unités ferroviaires pour le jeu de données des Hauts de France et de six unités pour le jeu de données de Bretagne. Ce sont des coûts supplémentaires pour une entreprise ferroviaire qui ne sont vraiment pas négligeables.

Nous pouvons aussi observer un écart significatif entre les bornes linéaires des données des Hauts de France et de Bretagne avec les solutions entières correspondantes. Cela peut indiquer une possible marge d'amélioration dans la recherche de solutions entières intéressantes, et donc un gain significatif pour les opérateurs ferroviaires.

CHAPITRE 6 CONCLUSION

6.1 Synthèse des travaux

Au cours de cette maîtrise, nous avons eu l'opportunité de travailler sur le problème d'affectation des unités ferroviaires à un emploi du temps donné. Nous avons tout d'abord modélisé le problème comme un problème de recherche d'un plus court chemin dans un graphe avec des contraintes supplémentaires. Nous avons pu résoudre la relaxation linéaire de notre modèle par un algorithme de génération de colonnes, avant de nous attaquer à la résolution du modèle en nombres entiers. Nous avons exploré plusieurs approches :

- Réduire la taille du graphe du sous-problème de manière à accélérer la résolution, et utiliser un post-traitement naïf pour retrouver une solution entière au problème complet. Cette approche a permis d'améliorer significativement les temps de calculs et de commencer à identifier des solutions entières. Cependant, le caractère aléatoire de notre post-traitement ne permet pas de l'utiliser de manière fiable dans un algorithme d'optimisation.
- Réduire le nombre de variables considérées dans notre modèle, en ne gardant que les variables dont le coût réduit dans la relaxation linéaire est faible. Nous avons ainsi pu obtenir des solutions pour les problèmes non contraints, et pour les problèmes contraints avec un temps d'attente maximal autorisé de 3h et plus. Nous avons observé que plus le nombre de colonnes considérées était faible, plus le temps de calcul nécessaire était faible. Cependant, les solutions obtenues sont également de moins bonne qualité. Le compromis entre ces deux facteurs semble ainsi impossible pour cette méthode.
- Approcher le problème par dichotomie, en résolvant itérativement des problèmes plus petits qui fournissent ensuite une solution globale du problème. Cette heuristique a fourni des résultats prometteurs sur les deux instances concernées. Les temps de calculs sont longs de manière à permettre au solveur d'identifier des solutions entières de bonne qualité pour chaque sous-instance à chaque itération. Nous pouvons aussi appliquer notre post-traitement à la solution finalement obtenue pour améliorer encore la qualité de la solution. Nous avons obtenu nos meilleures solutions entières de cette manière.

Nous avons ainsi obtenu d'une part les solutions de la relaxation linéaire des instances, et d'autre part des solutions entières de bonne qualité. En les comparant avec les solutions de Giro, nous avons observé que nos résultats étaient similaires pour la plus petite instance, mais que leur qualité se dégradait sur les plus grandes instances, tout en ne restant qu'à

quelques % d'écart.

6.2 Limitations de la solution proposée

Notre solution présente certaines limitations. Dans un premier temps, notre heuristique ne garantit aucunement de converger vers la solution optimale. En effet, rien ne garantit que la solution optimale puisse être divisée de cette manière en 2 sous-solutions. De plus, le nombre de subdivisions possibles est tel que trouver par hasard une subdivision respectée par la solution optimale est quasiment impossible. Les temps de calculs longs pour déterminer les solutions entières nous empêchent également d'en explorer un trop grand nombre.

De plus, l'algorithme dépend de nombreux paramètres, puisqu'on peut associer un jeu de paramètres à chaque résolution de chaque sous-instance. Les temps de calculs interdisent d'explorer un ensemble de valeurs de paramètres trop grands. Par conséquent, il est très complexe de choisir des paramètres pertinents, d'autant plus que chaque instance aurait besoin d'un jeu de paramètres adapté spécifiquement. Comme la qualité des solutions obtenues dépend fortement de ces paramètres, il s'agit assurément d'une limite de cet algorithme.

Enfin, nous n'avons pu tester notre algorithme que sur trois jeux de données, dont un qui a été résolu très rapidement. Pour juger l'efficacité de cette méthode, il serait nécessaire de l'appliquer à un plus grand nombre d'instances, dont, idéalement, on connaîtrait déjà la solution entière optimale.

Ces différents aspects limitent la qualité de la solution résultante et expliquent les très longs temps de calculs nécessaires pour obtenir des résultats similaires à ceux de Giro.

6.3 Améliorations futures

Plusieurs pistes d'amélioration de notre algorithme sont possibles :

- En ce qui concerne les performances temporelles du modèle, il serait pertinent d'intégrer cette méthode dans des codes optimisés comme ceux du logiciel Gencol. Au cours de cette maîtrise, tous nos codes ont été développés à la main, et exécutés en local. Les améliorer et les intégrer dans un solveur plus optimisé pourrait réduire les temps de calculs. Il serait également pertinent de les faire tourner avec plus de puissance de calcul. La résolution d'un problème en nombres entiers est très parallélisable et disposer d'un grand nombre de coeurs permettrait directement de diviser les tâches et le temps de résolution total.

- Notre algorithme pourrait être amélioré. En effet, actuellement, nous créons une subdivision de manière aléatoire. C’est la méthode la plus simple pour en créer une, et nous avons observé une amélioration régulière de la solution au fil des itérations. Cependant, il pourrait être intéressant d’explorer d’autres manières de créer cette subdivision. Par exemple, on pourrait garder en mémoire la subdivision précédente et construire une nouvelle subdivision en cherchant à s’en éloigner le plus possible. Créer une subdivision spécifique à la structure du problème pourrait également être une idée, par exemple en séparant les voyages par zone géographique ou par heure de départ.
- Enfin, le périmètre de l’étude pourrait être augmenté. Comme mentionné en introduction, nous avons choisi de résoudre un problème quotidien avec des contraintes de périodicité. Nous pourrions chercher à étendre l’échelle de temps considérée de manière à inclure des maintenances régulières dans le problème. Nous pourrions aussi rester à l’échelle de temps quotidienne, mais garantir qu’une certaine proportion des unités ait le temps d’effectuer une maintenance chaque jour.

Le problème de réoptimisation de l’affectation des unités ferroviaires après un imprévu (accidents sur une voie, panne d’une unité, grève, ...) est aussi un enjeu majeur pour les opérateurs ferroviaires. Si nos algorithmes ne semblent pas adaptés pour ce problème à cause de leurs temps de calculs trop longs pour être utilisés de manière opérationnelle, il pourrait tout de même être intéressant de repenser la modélisation du problème avec cet objectif.

RÉFÉRENCES

- [1] J. T. Haahr, J. C. Wagenaar, L. P. Veelenturf et L. G. Kroon, “A comparison of two exact methods for passenger railway rolling stock (re)scheduling,” *Transportation Research Part E : Logistics and Transportation Review*, vol. 91, p. 15–32, 2016.
- [2] G. B. Dantzig et P. Wolfe, “Decomposition principle for linear programs,” *Operations Research*, vol. 8, n^o. 1, p. 101–111, 1960.
- [3] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh et P. H. Vance, “Branch-and-Price : Column generation for solving huge integer programs,” *Operations Research*, vol. 46, n^o. 3, p. 316–329, May–June 1998.
- [4] J. Desrosiers et M. E. Lübbecke, “A primer in column generation,” dans *Column Generation*, G. Desaulniers, J. Desrosiers et M. M. Solomon, édit. Boston : Springer, April 2005, ch. 1, p. 1–32.
- [5] L. Costa, C. Contardo et G. Desaulniers, “Exact branch-price-and-cut algorithms for vehicle routing,” *Transportation Science*, vol. 53, n^o. 4, p. 946–985, July–August 2019.
- [6] K. Ziarati, F. Soumis, J. Desrosiers, S. Gélinas et A. Saintonge, “Locomotive assignment with heterogeneous consists at CN North America,” *European Journal of Operational Research*, vol. 97, n^o. 2, p. 281–292, March 1997.
- [7] J.-F. Cordeau, G. Desaulniers, N. Lingaya, F. Soumis et J. Desrosiers, “Simultaneous locomotive and car assignment at VIA Rail Canada,” *Transportation Research Part B : Methodological*, vol. 35, n^o. 8, p. 767–787, 2001.
- [8] N. Lingaya, J.-F. Cordeau, G. Desaulniers, J. Desrosiers et F. Soumis, “Operational car assignment at VIA Rail Canada,” *Transportation Research Part B : Methodological*, vol. 36, n^o. 9, p. 755–778, 2002.
- [9] E. Abbink, B. van den Berg, L. Kroon et M. Salomon, “Allocation of railway rolling stock for passenger trains,” *Transportation Science*, vol. 38, n^o. 1, p. 33–41, 2004.
- [10] A. Alfieri, R. Groot, L. Kroon et A. Schrijver, “Efficient circulation of railway rolling stock,” *Transportation Science*, vol. 40, n^o. 3, p. 378–391, 2006.
- [11] M. Peeters et L. Kroon, “Circulation of railway rolling stock : a branch-and-price approach,” *Computers & Operations Research*, vol. 35, n^o. 2, p. 538–556, 2008.
- [12] R. K. Ahuja, T. L. Magnanti et J. B. Orlin, “Network flows,” p. 41–45, 1988.
- [13] V. Cacchiani, A. Caprara et P. Toth, “Solving a real-world train unit assignment problem,” *Mathematical Programming*, vol. 124, p. 207–231, 07 2010.

- [14] J. Haahr, R. Lusby, J. Larsen et D. Pisinger, *A Branch-and-Price Framework for Railway Rolling Stock Rescheduling During Disruptions*. DTU Management Engineering, 2014.
- [15] G. Maróti et L. Kroon, “Maintenance routing for train units : the transition model,” *Transportation Science*, vol. 39, n^o. 4, p. 518–525, 2005.
- [16] G. L. Giacco, D. Carillo, A. D’Ariano, D. Pacciarelli et Á. G. Marín, “Short-term rail rolling stock rostering and maintenance scheduling,” *Transportation Research Procedia*, vol. 3, p. 651–659, 2014.
- [17] L. Mira, A. R. Andrade et M. C. Gomes, “Maintenance scheduling within rolling stock planning in railway operations under uncertain maintenance durations,” *Journal of Rail Transport Planning & Management*, vol. 14, p. 100177, 2020.
- [18] P. D. L. Amorosi et G. Giacco, “An integrated model for high-speed rolling-stock planning and maintenance scheduling,” *Engineering Optimization*, p. 1–22, 2023.
- [19] D. Canca et E. Barrena, “The integrated rolling stock circulation and depot location problem in railway rapid transit systems,” *Transportation Research Part E : Logistics and Transportation Review*, vol. 109, p. 115–138, 2018.
- [20] T. H. Cormen, C. E. Leiserson, R. L. Rivest et C. Stein, *Introduction to algorithms*. MIT Press and McGraw-Hill, 2001, section 22.4 Topological Sort.
- [21] R. Bellman, “On a routing problem,” *Quarterly of Applied Mathematics*, vol. 16, n^o. 1, p. 87–90, 1958.
- [22] C. Joncour, S. Michel, R. Sadykov, D. Sverdlov et F. Vanderbeck, “Column generation based primal heuristics,” *Electronic Notes in Discrete Mathematics*, vol. 36, p. 695–702, 2010.