UNIVERSITÉ DE NANTES

UFR DES SCIENCES ET TECHNIQUES

SCIENCES ET TECHNOLOGIES
DE L'INFORMATION ET DE MATHÉMATIQUES

Année 2012

N° attribué par la bibliothèque

Évaluation biobjectif de la capacité d'infrastructures ferroviaires par génération de colonnes hybride

---

THÈSE DE DOCTORAT
Discipline : INFORMATIQUE
Spécialité : RECHERCHE OPÉRATIONNELLE

Présentée
et soutenue publiquement par

Aurélien MEREL

le 31 octobre 2012 au LINA, devant le jury ci-dessous

Président	:	Pr. Frédéric SAUBION, Professeur	Université d'Angers
Rapporteurs	:	Pr. Anass NAGIH, Professeur	Université de Lorraine
		Pr. Joaquin RODRIGUEZ, Directeur de Recherche	IFSTTAR
		Pr. Paolo TOTH, Professeur	Université de Bologne
Examinateurs	:	Dr. David DE ALMEIDA, Docteur	SNCF Innovation et Recherche
		Dr. Xavier DELORME, Maître-Assistant	École des Mines de Saint-Étienne
		Pr. Frédéric SAUBION, Professeur	Université d'Angers

Directeur de thèse : Pr. Xavier GANDIBLEUX
Co-encadrante de thèse : Dr. Sophie DEMASSEY
Laboratoire : **L**ABORATOIRE D'**I**NFORMATIQUE DE **N**ANTES **A**TLANTIQUE.
2, rue de la Houssinière, BP 92 208 – 44 322 Nantes, CEDEX 3.

ÉVALUATION BIOBJECTIF DE LA CAPACITÉ D'INFRASTRUCTURES FERROVIAIRES PAR GÉNÉRATION DE COLONNES HYBRIDE

Biobjective railway infrastructure capacity assessment by hybrid column generation

Aurélien MEREL

⋈

favet neptunus eunti

Université de Nantes

Aurélien MEREL

Évaluation biobjectif de la capacité d'infrastructures ferroviaires par génération de colonnes hybride

x+145 p.

Ce document a été préparé avec LaTeX2e et la classe `these-LINA` version v. 1.30 de l'association de jeunes chercheurs en informatique LOGIN, Université de Nantes. La classe `these-LINA` est disponible à l'adresse :

`http://login.lina.sciences.univ-nantes.fr/`

Impression : these_finale_01.tex – 26/11/2012 – 12:28

Révision pour la classe : these-LINA.cls,v 1.30 2005/08/16 17:01:41 mancheron Exp

Résumé

Ce mémoire propose de s'intéresser aux problématiques liées à la capacité des infrastructures ferroviaires. Évaluer la capacité d'une infrastructure revient à estimer le trafic maximal qu'elle peut accueillir pendant une période donnée, en regard d'une demande de circulation donnée. L'estimation de capacité est partie intégrante de la gestion prévisionnelle du trafic ferroviaire, et se révèle une donnée essentielle pour les gestionnaires d'infrastructure et les exploitants ferroviaires, d'autant plus étant donné le contexte européen de libéralisation du marché. Calculer la capacité peut être réalisé par la résolution d'un problème sous-jacent appelé problème de saturation, consistant à insérer le maximum de circulation possible sur une infrastructure et soulevant d'importantes difficultés combinatoires : des temps de calcul rédhibitoires apparaissent pour certains jeux de données en utilisant les méthodes existantes. Une nouvelle méthode est donc proposée, basée sur une modélisation sous forme de programme linéaire en nombres entiers. Une contribution algorithmique est présentée, se basant sur plusieurs techniques : génération de colonnes, agrégation dynamique de contraintes et hybridation d'algorithmes exacts et approchés. Les tests comparatifs montrent une réduction significative du temps de résolution grâce à notre contribution, que nous implémentons dans un logiciel d'analyse de capacité, RECIFE PC. Une analyse des solutions nous conduit finalement à affiner le modèle mathématique en proposant deux formulations biobjectifs et une suggestion algorithmique visant à tirer profit de la génération de colonnes tout en réalisant une résolution biobjectif.

Mots-clés : recherche opérationnelle ; capacité d'infrastructure ferroviaire ; optimisation combinatoire ; génération de colonnes ; optimisation multiobjectif.

Abstract

The present thesis is focused on problematics related to railway infrastructure capacity. Evaluating an infrastructure's capacity can be seen as assessing the maximum amount of traffic that can pass through it within a given time period, depending on some circulation demand. Capacity assessment is an essential part of previsional planning of the railway traffic and is consequently an essential piece of data for infrastructure managers as well as for train operators. This is even more important in the current context of the European railway market being open to free competition. Capacity assessment can be achieved by solving an underlying problem, the saturation problem, which entails inserting as much traffic as possible inside an infrastructure and yields significant combinatorial difficulties. Prohibitive computation times are yielded by some data sets when using existing computation algorithms. A new solution method is consequently proposed, based on an integer linear programming modelling of the problem. An original algorithmic contribution is presented in order to solve this formulation by using several techniques, including column generation, dynamic constraint aggregation and hybridizations of exact and approximate solution algorithms. Comparative tests show a substantial reduction of the solving time thanks to our contribution, which we consequently implement into a capacity analysis software, RECIFE PC. An analysis of the solutions finally leads us to refine the mathematical model by proposing two biobjective formulations and an algorithmic suggestion aiming at benefiting from the column generation procedure while doing a biobjective resolution.

Keywords: operations research; railway infrastructure capacity; combinatorial optimization; column generation; multiobjective optimization.

Remerciements

La réalisation de ce travail de doctorat n'aurait pas été possible sans l'aide, les avis et les discussions techniques que j'ai eues avec Xavier Gandibleux et Sophie Demassey, respectivement Professeur à l'Université de Nantes et Maître-assistante à l'École des Mines de Nantes. Leur clairvoyance sur la pertinence de certaines pistes, leurs conseils, doutes ou encore encouragements ont été autant d'éléments absolument indispensables à la progression de ce travail. Je leur adresse de profonds remerciements et une gratitude toute particulière pour avoir partagé cette aventure à travers l'encadrement qu'ils m'ont fourni au long de ces quatre riches années.

Je remercie également les membres de mon comité de suivi de thèse, Xavier Delorme et Frédéric Saubion, respectivement Maître-assistant à l'École des Mines de Saint-Étienne et Professeur à l'Université d'Angers, pour avoir adressé à l'égard de mon travail un regard critique qui a contribué à m'aiguiller.

Certaines contributions de ce travail n'auraient sans doute pas existé sans l'accueil dont j'ai bénéficié en début de doctorat à l'Université d'Auckland, en Nouvelle-Zélande. Je tiens donc à remercier chaleureusement le Professeur Matthias Ehrgott de m'avoir accueilli amicalement et offert un espace de travail lors de mon séjour. Je remercie également Richard Lusby, avec qui j'ai eu des échanges fructueux et qui a également contribué à rendre mon séjour à Auckland inoubliable.

J'adresse également de sincères remerciements à Joaquin Rodriguez, Directeur de recherche dans l'unité ESTAS à l'Institut français des sciences et technologies des transports, de l'aménagement et des réseaux (IFSTTAR), pour les nombreuses informations et éclaircissements dont il m'a fait profiter, pour sa venue à Nantes dans le cadre de discussions autour de RECIFE ainsi que pour l'accueil dont j'ai bénéficié au site de Villeneuve d'Ascq de l'IFSTTAR. Je remercie également Grégory Marlière, Ingénieur de recherche à l'IFSTTAR, pour sa contribution et ses échanges autour de RECIFE et pour les démonstrations logicielles et l'accueil à l'IFSTTAR de Villeneuve d'Ascq. Je remercie à cette occasion une nouvelle fois Xavier Delorme qui a également consacré du temps à m'apporter des éclaircissements en matière ferroviaire. Enfin, je remercie Sonia Sobieraj et Rémy Chevrier, également de l'unité ESTAS de l'IFSTTAR, pour les moments passés en leur compagnie avec Joaquin Rodriguez et Grégory Marlière.

Je suis aussi reconnaissant envers le personnel administratif, en particulier Catherine Fourny, assistante de l'équipe TASC [1] basée à l'École des Mines de Nantes et Annie Lardenois, assistante de l'équipe OPTI localisée à l'Université de Nantes, pour leur constante bonne humeur, leur remarquable patience et l'aide fournie de nombreuses fois lors de l'organisation de déplacements.

Le temps passé à l'École des Mines aurait été bien différent sans la compagnie d'un grand nombre de personnes, et je tiens à exprimer ma reconnaissance à l'ensemble du personnel du Département informatique. Je souhaite également à ce titre remercier très chaleureusement tous ceux dont la présence à l'École des Mines a constitué un remarquable coup de pouce du fait de moments inoubliables passés ensemble, en particulier David, Jeff, Justine, Alexis et de très nombreux autres.

Les deux années exceptionnelles passées à l'Université de Nantes le sont notamment grâce à la bonne humeur régnant au sein des doctorants. Je remercie particulièrement à ce titre Thomas, Marie et Fabien pour avoir imprimé une bonne humeur constante que ce soit dans le bureau ou dans le groupe des doctorants. Merci plus généralement à tout le groupe pour les discussions et les nombreux et agréables moments passés ensemble.

1. Théorie, algorithmes et systèmes en contraintes.

Merci également, dans le désordre, à ceux qui m'ont accompagné ou soutenu dans cette aventure sans nécessairement le savoir : les anciens de l'École sur IRC, les anciens de Charles Péguy, Pierre-Yves et Guiv-Roger, le groupe de l'ERNM, Nagham, Khaled, Rima et bien d'autres.

Enfin, je remercie mes parents, mes proches et Stéphanie pour leur soutien ainsi que pour les moments de détente et d'apaisement.

Sommaire

CHAPITRE 1

Introduction

Ce mémoire présente les travaux de recherche et donne un aperçu des développements applicatifs que j'ai effectués au sein du LINA [1] depuis octobre 2008 dans le cadre de mon doctorat. Ces travaux ont été réalisés au sein des équipes de recherche TASC, établie à l'École de Mines de Nantes, et OPTI, basée à l'Université de Nantes. Ils relèvent en majeure partie du domaine de l'optimisation, et plus précisément de la programmation linéaire, celle-ci regroupant un ensemble de techniques visant à modéliser mathématiquement et à résoudre une très large variété de problèmes d'optimisation. Parmi ces problèmes, nous avons choisi de nous intéresser à la thématique de l'évaluation de la capacité des infrastructures ferroviaires, qui rentre dans l'activité plus large de la gestion prévisionnelle du trafic ferroviaire.

La question de la détermination de la capacité d'infrastructures ferroviaires est une problématique essentielle, puisqu'elle vise à estimer les limites qu'une infrastructure impose sur la circulation. La connaissance de telles limites est cruciale tant pour les gestionnaires d'infrastructure (entités gérant le réseau ferré) que pour les exploitants ferroviaires (entités faisant circuler du matériel roulant sur l'infrastructure) pour répondre au mieux à la demande de transport avec l'infrastructure existante ou encore pour concevoir de nouvelles infrastructures pertinemment. Cette question intervient de façon d'autant plus importante en Europe, où les directives de la Commission européenne imposent une libéralisation du marché ferroviaire, les infrastructures étant progressivement ouvertes à la libre concurrence pour les exploitants privés. Il s'agit donc d'une problématique actuelle dans le contexte européen, prenant une ampleur croissante depuis le début des années 2000 avec la mise en application progressive des directions de la Commission.

C'est dans ce contexte qu'a été lancé en 2000 le projet RECIFE [2], financé notamment par la région Nord-Pas-de-Calais et objet d'une collaboration entre l'Université de Valenciennes, l'IFSTTAR [3] et la SNCF [4]. Il a mené au développement de diverses contributions logicielles visant à valider les recherches algorithmiques permettant l'estimation de capacité d'infrastructures à un niveau local, c'est-à-dire se focalisant de façon détaillée sur certaines gares ou jonctions. Le présent travail, de part le problème auquel il s'intéresse et les contributions qu'il apporte, s'inscrit informellement comme une poursuite du travail effectué dans le cadre de RECIFE, en relevant de nouvelles problématiques et en contribuant au logiciel existant pour effectuer la validation des algorithmes proposés.

Ainsi, l'objectif premier de ce travail est de répondre à la nécessité d'effectuer des estimations de capacité d'infrastructure dans un temps acceptable pour l'utilisateur d'un logiciel dédié à ce type de calcul. Ce problème apparaît pour certains jeux de données fournis dans le cadre de RECIFE, et peut être un facteur handicapant non négligeable dans la conduite d'études de capacité. En effet, bien que les études de capacité sont réalisées de l'ordre de plusieurs mois à l'avance, la considération possible

1. Laboratoire d'informatique de Nantes-Atlantique.
2. Recherche sur la capacité des infrastructures ferroviaires.
3. Institut français des sciences et technologies des transports, de l'aménagement et des réseaux.
4. Société nationale des chemins de fer français.

de plusieurs scénarios de circulation implique plusieurs calculs distincts de capacité, dont l'addition des temps de résolution peut s'avérer extrêmement handicapant.

Les principales contributions du travail de recherche présenté dans ce mémoire sont d'ordres mathématique et algorithmique. La réponse apportée passe par deux axes majeurs. D'une part, l'étude de diverses contributions répondant à des problématiques sensiblement différentes mais restant dans le cadre de l'estimation de la capacité d'infrastructures ferroviaires nous fournit des outils de modélisation sous la forme de problèmes de programmation linéaire. En particulier, des éléments de modélisation fournis dans le cadre d'un projet réalisé pour les chemins de fer néerlandais constituent une base essentielle, réutilisée notamment au sein des développements réalisés dans le cadre de RECIFE. Des travaux plus récents réalisés dans le cadre d'une collaboration entre l'exploitant allemand DB [5] et l'Université d'Auckland (Nouvelle-Zélande) viennent également apporter des contributions particulièrement pertinentes d'un point de vue de l'efficacité de la résolution du problème.

D'autre part, des contributions algorithmiques originales sont apportées en se basant sur l'étude de travaux rentrant dans le cadre plus générique de la programmation linéaire. Ce domaine fournit en effet un grand nombre de contributions pertinentes pouvant être utilisée sur des problèmes spécifiques moyennant un travail de modélisation et d'adaptation. Parmi les techniques algorithmiques exploitées dans l'approche proposée ici, figurent principalement la génération de colonnes, l'agrégation dynamique de contraintes et l'hybridation avec une métaheuristique.

Les contributions développées dans le présent travail sont implémentées, testées et validées dans la plateforme logicielle issue du projet RECIFE, dont l'architecture est améliorée par la même occasion. Des ouvertures en termes de modélisation et d'algorithmique relevant du domaine de l'optimisation multiobjectif sont également proposées suite à une observation critique des résultats obtenus.

Nous commençons ainsi par présenter précisément la problématique de la capacité d'infrastructures ferroviaires au chapitre 2, en introduisant tout d'abord le contexte de la libéralisation du marché ferroviaire, les directives de la Commission associées et la façon dont celles-ci sont déjà implémentées en Europe. Les différents paramètres rendant cette problématique cruciale sont mis en avant, à savoir l'engorgement croissant des infrastructures, l'augmentation du nombre d'acteurs et l'hétérogénéité du matériel circulant sur les voies. Plusieurs problématiques sous-jacentes à l'évaluation de capacité d'infrastructures sont détaillées, et en particulier le problème de saturation dont la résolution permet l'obtention d'une estimation de la capacité. C'est ce dernier problème que nous nous employons à résoudre. Enfin, un aperçu de plusieurs plateformes logicielles existantes liées au calcul de capacité d'infrastructures ferroviaires est donné, nous permettant de quantifier l'existant et d'identifier les verrous restant dans notre cas.

Le chapitre 3 formalise les données à prendre en compte dans le cadre du problème de saturation (trains, infrastructure, etc.) puis présente la modélisation mathématique choisie pour le problème de saturation, tirant profit des contributions présentes dans la littérature et relatives au problème de saturation ou à des problèmes similaires liés à l'évaluation de capacité. Nous proposons une discussion sur les différentes propriétés de ces formulations d'un point de vue mathématique dans la perspective d'une résolution efficace. Ceci nous permet de privilégier un modèle, dit par disjonction de ressources. Pour compléter la formalisation du problème, les instances que nous traitons sont détaillées, permettant d'identifier précisément les caractéristiques entraînant des difficultés de résolution avec les approches existantes.

Ces éléments posés, le chapitre 4 est dédié à la méthode de résolution que nous proposons. Cela nécessite d'abord de s'intéresser aux algorithmes préalablement développés pour résoudre des formulations

5. Deutsche Bahn.

similaires, et ce afin d'identifier dans quelle mesure nous pouvons réutiliser ces travaux. Ces derniers incluent un algorithme de génération de colonnes, principe algorithmique générique que nous réutilisons et adaptons à notre cas. Ces adaptations sont donc présentées dans le détail. À notre algorithme est intégré une procédure dite d'agrégation dynamique de contraintes, dont l'utilisation vise à accélérer la procédure de génération de colonnes en réduisant la taille de la formulation considérée par celle-ci. Nous en décrivons le principe et les algorithmes développés pour l'implémenter. Enfin, l'ensemble de la procédure de génération de colonnes collabore avec la métaheuristique ACO [6] développée précédemment dans le cadre de RECIFE, et permettant finalement l'obtention de solutions au problème de saturation.

Le chapitre 5 propose des résultats comparatifs de résolution du problème de saturation entre d'une part la méthode décrite au chapitre 4 et d'autre part la méthode ayant fourni jusqu'ici les meilleurs résultats, à savoir la seule métaheuristique ACO. Les mesures montrent une significative réduction du temps de calcul sur une vaste majorité des instances ainsi qu'une consommation mémoire plus faible et une qualité de solutions similaire. Ceci valide ainsi notre contribution algorithmique. Nous étudions également l'impact individuel des différents composants de notre algorithme sur le comportement de celui-ci. Le chapitre propose ensuite un aperçu de la plateforme logicielle RECIFE PC, en détaillant le module dédié à la saturation au sein duquel notre nouvel algorithme a été mis en place. Ceci nous amène finalement à une observation plus détaillée, en termes ferroviaires, des solutions obtenues par résolution du problème de saturation. Nous identifions un manque de réalisme dans ces solutions, caractérisé par un déséquilibre entre les types de trains choisis pour être routés sur l'infrastructure.

Dans la perspective de pallier à ce déséquilibre, le chapitre 6 suggère finalement une approche de résolution biobjectif. Nous y introduisons le principe de l'optimisation multiobjectif puis présentons deux critères, dits critères d'équité, visant à minimiser les déséquilibres entre types de trains au sein d'une solution. Ces critères sont utilisés pour formaliser des problèmes d'optimisation biobjectifs, et sont comparés sur la base de leur pertinence en regard du but recherché et de leur adaptabilité à l'algorithme de résolution préalablement proposé. Une approche algorithmique combinant l'un de ces nouveaux critères à notre précédent algorithme est suggérée, constituant une ouverture à des estimations de capacité plus réalistes ainsi qu'à la possibilité d'envisager la réutilisation du principe de génération de colonnes dans le cas de problèmes biobjectifs.

6. *Ant Colony Optimization*, optimisation par colonies de fourmis.

CHAPITRE 2

Problématique de la capacité des infrastructures ferroviaires

Ce premier chapitre introduit le problème traité dans le cadre de cette thèse, à savoir le problème dit de *capacité d'infrastructures ferroviaires*, et le place dans le contexte actuel de libéralisation du marché ferroviaire. L'estimation de la capacité des infrastructures répond en effet à des enjeux importants au regard de la possibilité offerte à de nouveaux opérateurs privés de faire circuler du matériel roulant. Ces enjeux sont liés d'une part à la multiplication des opérateurs et d'autre part à une tendance à la hausse du trafic. La première partie de ce chapitre montre ainsi que la capacité d'une infrastructure est stratégique tant pour les gestionnaires d'infrastructure que pour les opérateurs.

Une fois les enjeux mis en lumière, la définition du problème de capacité nécessite en premier lieu de préciser les limites de l'infrastructure considérée ainsi que ses éléments caractéristiques. Selon l'objectif de l'étude, l'infrastructure considérée sera soit une vaste proportion d'un réseau, soit une zone restreinte telle qu'une gare ou une jonction. En second lieu, le problème de capacité lui-même doit être défini à travers les données que l'on cherche à quantifier ainsi que les contraintes à prendre en compte. Intuitivement, calculer la capacité d'une infrastructure consiste à estimer la quantité maximale de circulation qui peut y passer par unité de temps. Ce chapitre montre d'une part que les contraintes à prendre en compte afin d'avoir une approche réaliste sont multiples et propose d'autre part une discussion sur les deux principaux types de définition existant dans la littérature. Une fois la définition de référence choisie, le problème de capacité est décomposé en problèmes sous-jacents permettant de séparer les différentes contraintes pesant sur l'estimation de capacité.

Un aperçu des précédentes études de capacité est finalement présenté puis mis en regard avec la variante du problème auquel nous nous intéressons, à savoir l'étude de capacité à l'échelle d'une jonction ferroviaire, basé sur des données précises sur l'infrastructure et sur la demande de trafic. Il apparaît que les études réalisées précédemment ne répondent que de façon partielle au cas de figure qui nous intéresse.

2.1 Contexte du transport ferroviaire

2.1.1 Libéralisation du marché ferroviaire

Conformément aux directives européennes relatives à la libéralisation du marché ferroviaire [23, 24], les anciens opérateurs ferroviaires nationaux, tels que la SNCF [1] en France ou la SNCB [2] en Belgique, ne sont plus propriétaires des infrastructures. Ces opérateurs, ou *exploitants*, deviennent des entreprises de droit privé, et l'infrastructure est désormais gérée par des *gestionnaires d'infrastructure* pour le compte des États. En France, c'est par exemple RFF [3] qui a été créé pour gérer l'infrastructure.

Tout exploitant souhaitant faire circuler du matériel roulant sur une infrastructure doit en premier lieu adresser une demande de *sillon* au gestionnaire de cette infrastructure, un sillon étant la spécification temporelle exacte de l'utilisation d'une infrastructure permettant de relier deux points du réseau (le plus souvent, des gares). Le gestionnaire, souhaitant rentabiliser au mieux l'infrastructure et pouvant faire face à des demandes de sillons conflictuelles, est en charge d'attribuer ou de refuser les sillons demandés par les exploitants. Le gestionnaire peut également suggérer de décaler dans le temps certains sillons demandés, l'exploitant pouvant à son tour accepter ou refuser. Il s'agit donc d'un processus de négociation qui permet *in fine* aux exploitants de s'accorder avec le gestionnaire sur la circulation de leur matériel et le cas échéant d'établir une grille horaire complète. Toutefois, le règlement des conflits entre exploitants peut en pratique être facilité par des négociations préalables entre ceux-ci.

Un nombre significatif d'acteurs est déjà susceptible d'effectuer des demandes de sillon. En France, outre la SNCF, l'opérateur allemand DB [4] effectue du transport de marchandises. Du point de vue du transport de voyageurs, la SNCB assure par exemple des connexions locales entre Lille et certaines villes belges comme Tournai, Liège ou Gand avec des trains type intercité. DB assure des dessertes entre Strasbourg et certaines villes allemandes ainsi qu'entre Paris et Munich, et l'opérateur italien Thello exploite l'axe Paris-Milan par des liaisons nocturnes. À ces opérateurs viennent s'ajouter des entités transverses telles que Thalys et Eurostar, exploitant avec du matériel type TGV [5] les axes Paris-Bruxelles-Amsterdam et Paris-Bruxelles-Cologne pour le premier et Paris-Londres et Bruxelles-Londres pour le second. Enfin, des collectivités locales telles que les Conseils régionaux opèrent sur le réseau français *via* la SNCF.

Il apparaît donc clairement que le nombre d'exploitants est important et sera probablement amené à augmenter avec le développement de la concurrence.

2.1.2 Hétérogénéité du matériel roulant

Conséquence de l'augmentation du nombre d'exploitants, le matériel roulant utilisé sur les infrastructures tend à être de plus en plus hétérogène. Ce phénomène est également causé par le renouvellement progressif des flottes, impliquant que du matériel récent côtoie du matériel plus ancien.

Sans être exhaustif, il est possible de prendre conscience de cette hétérogénéité. On dénombre par exemple au moins 9 types de TGV différents sur le réseau français, dont plusieurs caractéristiques telles que le poids, la longueur, la vitesse maximale et la capacité de freinage varient significativement. Ainsi, les TGV TMST [6] exploités par Eurostar mesurent plus de 390 mètres, les autres modèles de TGV ex-

1. Société nationale des chemins de fer français.
2. Société nationale des chemins de fer belges.
3. Réseau ferré de France.
4. Deutsche Bahn.
5. Train à grande vitesse.
6. TransMancheSuperTrain.

ployés par la SNCF mesurant quant à eux environ 200 mètres. En ce qui concerne les trains régionaux circulant à une vitesse inférieure à 200 km/h, le trafic est principalement partagé entre des rames dites AGC [7] récentes et des rames classiques plus anciennes, à voitures et motrices séparées, ces dernières existant en de nombreuses versions différentes.

À cette hétérogénéité du matériel s'ajoutent des fluctuations ponctuelles liées à la gestion du matériel roulant par les exploitants. En effet, pour répondre à une forte demande de transport, il est courant de coupler deux rames TGV ou deux voire trois rames AGC. Ces modifications ponctuelles créent de nouvelles hétérogénéités au niveau du matériel circulant sur l'infrastructure.

Ces différentes configurations de matériel roulant doivent souvent partager certains éléments d'infrastructure, tout en respectant des normes de sécurité précises et communes. Lors de la circulation, le respect des règles de sécurité est imposé par le système de signalisation. En limitant la vitesse de circulation, il a notamment pour but de faire respecter les distances minimales de sécurité devant séparer les trains. Ainsi, les capacités d'accélération et de freinage d'un train peuvent forcer un autre train à altérer sa course. Par propagation, l'hétérogénéité des caractéristiques des trains ont donc un impact sur l'ensemble du trafic.

2.1.3 Densification du trafic

Une autre caractéristique importante est le regain d'intérêt actuel pour le transport ferroviaire et une augmentation du trafic en conséquence. Ce regain d'intérêt est notamment causé par la pertinence du transport ferroviaire comme moyen de déplacement peu polluant. De plus, les trains à grande vitesse concurrencent pertinemment le transport aérien sur certaines liaisons.

Des signes du dynamisme du marché ferroviaire peuvent être observés à travers les investissements réalisés dans le domaine, et il est possible de citer comme exemples :

- la construction de lignes à grande vitesse telles que la ligne Paris-Marseille et plus récemment la ligne est-européenne ayant pour but la réduction du temps de parcours sur l'axe Paris-Strasbourg-Munich ;
- l'apparition du trafic à grande vitesse dans certains pays, comme la Pologne qui envisage la circulation d'AGV [8] sur des lignes dédiées d'ici à 2020 ;
- le renouvellement de certaines voies afin d'en augmenter la vitesse de circulation commerciale, tel que le projet de ligne à grande vitesse Bretagne-Pays de Loire.

La densification du trafic implique la nécessité de dimensionner convenablement les nouvelles infrastructures, en fonction d'une estimation de la future demande de circulation. En outre, cette densification tend à saturer les infrastructures existantes, rendant nécessaire l'utilisation d'outils appropriés pour la conception des grilles horaires.

2.1.4 Problématique et enjeux

Les trois ressources nécessaires au transport ferroviaires sont les suivantes :

- le matériel roulant (motrices, voitures, etc.), géré par les exploitants ;
- l'infrastructure, prise en charge par le gestionnaire ;
- le personnel, dont la présence est bien entendue nécessaire à la fois chez les exploitants et le gestionnaire et qui doit posséder un certain nombre de qualifications selon la mission attribuée.

7. Automotrice à grande capacité.
8. Automotrice à grande vitesse.

Chaque ressource pouvant être un facteur limitant pour la circulation, une gestion prévisionnelle doit être réalisée pour en assurer une bonne utilisation. Comme indiqué par Delorme [20], le transport ferroviaire nécessite deux niveaux de gestion prévisionnelle, chacun d'entre eux étant caractérisé par l'horizon temporel qu'il considère :

1. le niveau stratégique, qui induit une prévision sur plusieurs années, inclut notamment le dimensionnement et la construction de nouvelles infrastructures ainsi que l'achat de matériel roulant, ces deux ressources nécessitant d'importants investissements et ayant une durée de vie de plusieurs dizaines d'années ;
2. le niveau tactique, qui, une fois les ressources disponibles fixées, consiste principalement en la réalisation du *plan de transport* par la négociation entre exploitants et gestionnaires, et ce avec un horizon temporel de plusieurs mois.

Bien qu'elle ne soit pas considérée *stricto sensu* comme de la gestion prévisionnelle, une gestion au niveau opérationnel est également requise, où la mise en œuvre du plan de transport doit être réalisée au mieux compte tenu des indisponibilités ponctuelles de certaines ressources (*blanc-travaux* sur les voies, indisponibilités du matériel roulant ou du personnel, etc.). L'augmentation du nombre d'exploitants, de l'hétérogénéité du matériel et du trafic global rend donc d'autant plus importante une planification précise à tous ces niveaux.

Cette planification nécessite d'identifier les limitations imposées par la ressource infrastructure sur une demande de circulation donnée. Au niveau stratégique, cette donnée est requise par les gestionnaires pour réaliser des choix avisés lors de la conception d'une nouvelle infrastructure. Au niveau tactique, elle permet d'évaluer la quantité maximale de circulation qui peut être accueillie sur une infrastructure existante en fonction d'une demande estimée. C'est une donnée pertinente à la fois pour les gestionnaires et les exploitants : les premiers peuvent estimer le plus grand sous-ensemble de la demande qu'une infrastructure peut accueillir, les seconds peuvent répondre au mieux à la demande de transport en utilisant des sillons n'entrant pas en conflit avec un éventuel trafic préalablement existant. L'identification des limitations de l'infrastructure devient donc partie intégrante de la négociation entre exploitants et gestionnaires.

En outre, d'autres critères sont à prendre en compte lors de la planification de l'utilisation de l'infrastructure. C'est le cas par exemple de la qualité de service, notion générique qui implique entre autres d'éviter au mieux l'amplification en cascade des retards.

La gestion prévisionnelle de l'infrastructure est donc une problématique à part entière, en particulier aux niveaux stratégique et tactique. Elle peut être abordée sous l'angle du calcul de la capacité des infrastructures ferroviaires. Pour des infrastructures simples telles qu'une voie à sens unique soumise à un trafic homogène, la capacité peut être intuitivement conçue comme le rapport entre la durée minimale de sécurité devant séparer deux trains et une durée temporelle arbitrairement choisie : la valeur obtenue correspond au nombre maximal de trains pouvant circuler pendant cette durée choisie. Comme le décrit Delorme [20], il existe quelques autres cas simples pour lesquels un nombre de trains maximal par unité de temps peut être calculé en guise d'estimation de la capacité.

Le problème se complexifie nettement si l'infrastructure considérée est plus complexe et si les trains possèdent des caractéristiques variées. Il nous est donc nécessaire d'expliciter dans un premier temps les éléments caractéristiques significatifs d'une infrastructure puis de préciser la définition de la capacité.

2.2 L'infrastructure ferroviaire

2.2.1 Composition

L'infrastructure ferroviaire forme un réseau au sein duquel des *nœuds* sont reliés par des *voies* sur lesquelles les trains peuvent circuler. Sur une voie, un train ne peut pas changer de direction, et deux trains circulant l'un derrière l'autre sur la même voie ne peuvent pas changer d'ordre (pas de dépassement possible). Ces propriétés font des voies des zones qualifiées d'homogènes. Par opposition, une zone hétérogène permet les changements de direction et d'ordre des trains. Les zones hétérogènes correspondent aux nœuds ferroviaires et sont donc situées à l'intersection de deux ou plusieurs voies, impliquant la présence d'aiguillages. Concrètement, les nœuds sont matérialisés par les jonctions et les gares, dont la typologie et la taille sont variables. Enfin, un *tronçon* désigne l'ensemble des voies reliant deux nœuds adjacents, et une *ligne* est une séquence de tronçons reliant deux nœuds qui ne sont pas nécessairement adjacents.

Outre sa typologie, l'infrastructure est dotée d'un ensemble de dispositifs visant à faire respecter les règles de sécurité. Il se compose des systèmes de signalisation et des équipements de détection, ces derniers permettant de connaître les mouvements des trains. L'infrastructure est divisée en *zones de détection* (ou simplement zones) contiguës, qui sont comme indiqué par Rodriguez [63] les seuls équipements permettant de connaître la position des trains. Une zone de détection est considérée soit comme libre soit comme occupée par un train, auquel cas la signalisation interdit tout autre train d'y pénétrer. Les zones sont regroupées en *cantons*, dont l'entrée est protégée par un élément de signalisation. Lorsqu'un train pénètre sur un canton, toutes les zones du canton sont simultanément considérées comme *réservées* par ce train. Ainsi, un train ne peut rentrer sur un canton que si toutes ses zones sont libres. La sortie d'un train d'une zone est le *dégagement*.

Les zones et le cantonnement garantissent ainsi le respect des distances de sécurité. Selon le nombre d'*aspects* possibles des signaux, une consigne de vitesse variable peut être donnée en fonction du nombre de cantons libres au-devant du train.

L'infrastructure considérée par une étude de capacité étant nécessairement limitée géographiquement, celle-ci possède des *points d'entrée* et *points de sortie* correspondant aux éléments par lesquels les trains peuvent respectivement entrer et sortir de l'infrastructure considérée. Un point d'entrée ou de sortie coïncide typiquement avec un nœud ferroviaire ou, à échelle plus fine, une zone de détection particulière.

Une étude de capacité ne cherche pas nécessairement à modéliser l'infrastructure à la zone près, la granularité dépendant de la précision requise pour l'étude.

2.2.2 Échelles d'étude

Les deux échelles usuelles utilisées pour les études de capacité sont dites « macroscopique » et « microscopique ». De par la différence dans l'infrastructure qu'elles prennent en compte, elles répondent à des besoins différents. À l'échelle macroscopique, ou échelle du réseau, l'infrastructure considérée est un « grand » sous-ensemble du réseau tel qu'une ligne entre deux nœuds majeurs. Le large aperçu fourni par de telles études est une information essentielle pour les décideurs, puisqu'elle correspond à la demande de transport qui consiste à relier des points distants du réseau. À l'échelle du réseau, les éléments clés de l'infrastructure à prendre en compte incluent la disposition des lignes, gares et jonctions, les limites de vitesse, les distances de sécurité et les propriétés du système de signalisation.

Lorsqu'elle est réalisée à l'échelle microscopique, ou échelle d'un nœud, une étude est focalisée sur une infrastructure nettement plus restreinte telle qu'une gare ou une jonction. Comme cela est mis en avant par Lindner [43], les nœuds sont le point de croisement de plusieurs lignes et sont à ce titre

susceptibles de constituer un goulot d'étranglement pour la circulation globale. Ils représentent donc un facteur limitant pour la capacité vue à échelle macroscopique et peuvent avoir un rôle amplificateur sur les délais. Des analyses de capacité à échelle microscopique sont donc indispensables pour estimer correctement la capacité d'un réseau et pour éviter des délais inattendus. À cette échelle, il est nécessaire de prendre en compte la description précise de l'infrastructure, à savoir l'ensemble des zones, cantons et propriétés du système de signalisation, ainsi que les courbes de vitesse des trains sur chacun des chemins permettant de traverser le nœud. Étant donné le caractère amplificateur des nœuds, ces données et la méthode de calcul de capacité doivent être précises : le modèle choisi doit prendre en compte les paramètres du problème de façon précise, afin qu'une solution trouvée lors de sa résolution ait soit la plus proche possible d'être applicable à la réalité.

Les approches macroscopique et microscopique sont donc clairement différentes. Elles sont cependant complémentaires pour obtenir un aperçu à la fois global et précis des limitations d'une infrastructure. Des techniques sont en outre développées dans le but de combiner les études réalisées à des échelles différentes, comme proposé par Schlechte [69].

2.3 Les problèmes liés à l'évaluation de capacité

2.3.1 Définir la notion de capacité

La notion de capacité est intuitivement facile à appréhender pour des exemples simples : la capacité d'un récipient peut être synonyme de son volume, c'est-à-dire de la quantité de matière que celui-ci peut contenir ; celle d'un véhicule de transport de marchandise peut être associée au volume ou au poids maximal qu'il peut transporter, tous deux étant des facteurs limitant la quantité de marchandises pouvant être contenue ; dans le domaine de l'électricité, la capacité mesure une quantité d'énergie pouvant être stockée. Ces exemples illustrent l'association usuellement réalisée entre la capacité d'un objet et la quantité de produit (matière, marchandise, électricité, etc) que celui-ci peut gérer (stocker, transporter, etc). La capacité se réfère en outre à ce pourquoi l'objet est principalement conçu.

L'infrastructure ferroviaire étant conçue pour la circulation du matériel roulant, sa capacité peut faire référence à la quantité maximale de matériel qui peut y circuler pendant un intervalle temporel donné. Cette quantité peut cependant varier significativement en fonction de paramètres qui ne sont pas liés à l'infrastructure, tels que les caractéristiques du matériel roulant ou une fréquence de passage voulue pour certains types de trains. Ce simple constat fait de la définition de la capacité d'infrastructures un problème à part entière pour lequel plusieurs propositions sont présentes dans la littérature.

L'UIC [9] [75] établit ainsi que la capacité d'une infrastructure n'existe pas en tant que telle, puisqu'elle dépend de la façon dont elle est utilisée. L'approche préfère se focaliser sur le calcul de la « consommation de capacité » résultant des quatre facteurs suivants :

- le nombre de trains devant circuler pendant la période définie ;
- la vitesse moyenne de circulation ;
- l'hétérogénéité du trafic ;
- la stabilité voulue pour la grille horaire, c'est-à-dire sa capacité à résister aux retards.

Ces quatre facteurs sont interdépendants, c'est-à-dire que la capacité consommée par l'un des facteur n'est plus disponible pour les autres. La méthode UIC mesure la capacité non utilisée en unité de temps : il s'agit de la différence entre l'intervalle temporel d'étude et la durée nécessaire pour faire circuler la demande de circulation considérée, cette dernière correspondant à la consommation totale de capacité. La

9. Union internationale des chemins de fer.

capacité est donc associée à une unité temporelle. Une définition comparable est suggérée par Burkolter [8] qui, partant du même constat de l'influence des caractéristiques de la demande sur la mesure de capacité, la définit comme la durée minimale requise pour faire circuler une demande de circulation donnée. En outre, à l'image de Kuckelberg et al. [38], les recommandations de l'UIC [75] ont engendré le développement de nombreuses méthodes de calcul tentant de s'y conformer, suivant donc une définition temporelle de la capacité.

Le constat de l'impact des caractéristiques échappant à l'infrastructure sur l'estimation n'empêche toutefois pas l'utilisation d'une mesure de capacité basée sur la quantité de circulation. Ainsi, Hachemane [34] estime que la quantité à mesurer est le nombre de trains maximal pouvant circuler pendant un intervalle temporel donné. L'ensemble des paramètres de la demande sont alors considérés comme des contraintes à respecter dans la recherche de ce maximum. La définition qu'il propose est la suivante :

Définition 2.1. Capacité (Hachemane [34])
La capacité d'un élément de réseau ferroviaire est le nombre maximal de trains qui, selon une structure des lignes, une structure de l'horaire et une qualité de service données, peuvent y circuler dans un intervalle de temps donné dans des conditions pratiques d'exploitation.

Cette définition est également adoptée dans les travaux de Delorme [20]. En outre, de nombreux travaux indépendants utilisent des définitions similaires, tels qu'Abril et al. [1], Confessore et al. [15] et Kontaxi et Ricci [36].

La difficulté qu'il y a à définir la capacité est donc perceptible, et est due, comme le précisent Landex et al. [40], au fait qu'il n'existe pas un unique paramètre pertinent à mesurer : il est possible de privilégier la mesure d'un paramètre et de considérer les autres comme des facteurs limitants, et vice-versa. Nous choisirons par la suite de suivre la définition proposée par Hachemane [34], sa pertinence étant justifiée principalement par deux raisons :

- privilégier la mesure du nombre de trains pouvant circuler pendant une période donnée est cohérent avec de nombreuses références, récentes ou plus anciennes ;
- cette définition permet la prise en compte de l'ensemble des contraintes pesant sur l'estimation (structure des lignes, structure de l'horaire, qualité de service, conditions d'exploitation), ce qui est cohérent avec les facteurs de consommation de capacité présentés par l'UIC [75].

Delorme [20] précise la signification de chacune de ces contraintes définies par Hachemane [34] dans cette définition, qui peuvent être résumées comme suit.

2.3.1.1 Structure des lignes

La structure des lignes définit le réseau auquel appartient l'élément considéré pour la mesure de capacité, à savoir les nœuds et voies connectés à cet élément, ainsi que les relations origine-destination des trains circulant sur le réseau et passant par l'élément considéré.

2.3.1.2 Structure de l'horaire

La structure de l'horaire consiste en une spécification visant à garantir la circulation d'une certaine quantité de certains types de trafic pendant la fenêtre de temps considérée. C'est, en d'autres termes, ce qu'Hachemane [34] nomme la « répartition à l'horaire des différents types de trains ». La structure de l'horaire peut également impliquer des contraintes additionnelles telles que la nécessité d'assurer des correspondances, réaliser des couplages/découplages, etc.

2.3.1.3 Conditions pratiques d'exploitation

Les conditions pratiques d'exploitation regroupent les contraintes imposées à la circulation sur l'infrastructure considérée. Il s'agit de respecter les règles de sécurité imposées par le système de signalisation afin de ne considérer que des parcours effectivement réalisables lors du calcul de capacité. Des paramètres supplémentaires peuvent également subvenir : l'UIC [75] mentionne par exemple d'éventuelles priorités entre les trains et des considérations environnementales.

2.3.1.4 Qualité de service

La qualité de service est un point essentiel tout en étant difficile à définir avec exactitude. Un critère usuel de qualité de service est la capacité d'une grille horaire à absorber les retards, aussi appelée *robustesse*. Selon les points de vue, d'autres critères peuvent être pris en compte sous cette dénomination.

2.3.1.5 Intervalle de temps

Pour terminer, l'intervalle de temps (ou horizon temporel) peut être fixé arbitrairement, coïncider avec une éventuelle cadence ou encore correspondre à l'écart entre la première entrée et la dernière sortie prévues par la structure de l'horaire. C'est cet intervalle temporel qui est prise en compte pour l'étude de capacité.

2.3.2 Problèmes sous-jacents

Les multiples facteurs interdépendants et quantités à mesurer pour estimer la capacité font de son calcul un problème complexe. Delorme [20] identifie plusieurs sous-problèmes, plus simples, inhérents à l'estimation de capacité, que nous résumons ici.

Celui sur lequel nous nous focaliserons est le problème de saturation, que Delorme [20] définit comme suit :

Définition 2.2. Saturation (Delorme [20])
Le problème de saturation consiste à introduire le maximum de circulations supplémentaires dans une grille horaire établie (éventuellement vide). Les trains ainsi ajoutés représentent la marge de capacité disponible (c'est-à-dire la capacité résiduelle, ou la capacité absolue lorsque la grille horaire établie est vide) de l'infrastructure par rapport à une offre. Les nouveaux trains sont habituellement choisis parmi un ensemble de trains saturants.

Le problème de saturation est donc un problème d'optimisation, dont une solution est une estimation de la capacité. Ce problème soulève à lui seul d'importantes difficultés combinatoires en raison de la taille volontairement grande de l'ensemble de trains saturants et du degré de liberté qu'il est possible de laisser à chaque train, notamment en terme de retards autorisés. Le résoudre efficacement comporte donc un défi algorithmique important.

Il n'inclut toutefois pas l'ensemble des contraintes et facteurs également consommateurs de capacité. Ainsi, dans l'optique de respecter une certaine structure d'horaire, le décideur peut établir des préférences pour certains types de trafic ou l'utilisation de certains parcours à travers l'infrastructure. Delorme [20] définit donc le problème d'optimisation des préférences, qui vise cette fois à maximiser la somme des valeurs de préférence associées à des choix de parcours et/ou de délai pour un ensemble de trains devant

circuler sur l'infrastructure. Également dans l'optique de la qualité de service, le problème de fluidification est défini par Delorme [20] dans le but de faire circuler une demande de circulation donnée tout en minimisant les retards imposés aux trains de la demande par rapport à une heure de passage nominale.

Un problème qui se pose couramment consiste enfin à déterminer si une infrastructure est capable de faire face à une offre de trafic fixe, sans autoriser les trains à être retardés. Delorme [20] définit pour ce faire le problème de faisabilité :

Définition 2.3. Faisabilité (Delorme [20])
Le problème de faisabilité consiste à vérifier la faisabilité du passage d'une combinaison donnée de trains (point et date d'entrée-sortie fixés) dans une infrastructure considérée. La question est alors de savoir s'il existe un routage (affectation d'un itinéraire et éventuellement d'un quai) de ces trains permettant de les faire tous passer sans retard.

Il peut être aisément comparé au problème de saturation : là où ce dernier cherche à déterminer la quantité de circulation maximale (optimisation), le problème de faisabilité cherche à déterminer si une demande fixée peut circuler entièrement (décision). La différence entre ces deux problèmes réside donc principalement dans le plus grand degré de liberté laissé à la demande de circulation considérée. Plus précisément, pour un horizon temporel égal à celui d'une instance du problème de faisabilité, une instance du problème de saturation matérialise ce degré de liberté accru par deux caractéristiques :

- un nombre de trains significativement plus élevé, dont il est certain que l'ensemble ne pourra pas être routé intégralement à l'intérieur de l'horizon temporel considéré ;
- une flexibilité plus importante en terme de délai maximal autorisé, augmentant en conséquence le nombre de chemins possibles pour chaque train.

Ainsi, le problème de faisabilité peut être vu comme une variante sous forme de problème de décision du problème de saturation.

2.3.3 Capacité pratique et capacité théorique

Les études de capacité se basent sur une modélisation – et donc une approximation – de la réalité. À ce titre, la distinction est couramment faite entre la capacité dite « théorique » et la capacité dite « pratique ». Abril et al. [1] développent notamment les différences qui existent entre les deux : la première est un calcul basé sur des conditions idéales de circulation, en considérant un trafic homogène et sans aléa, et une distance de sécurité minimale entre les trains. La capacité théorique est donc un majorant qu'il n'est pas possible d'atteindre. À l'inverse, la capacité pratique prend en compte un trafic dit « représentatif » de la réalité. Plus clairement, cette représentativité sous-tend la prise en compte des contraintes énoncées par Hachemane [34], de la façon la plus proche possible de la réalité.

Qualifier une estimation de capacité comme étant « pratique » au sens strict impliquerait que celle-ci soit vérifiée sur le terrain par la mise en place d'une circulation réelle, ce qui n'est bien entendu pas envisageable pour toutes les études. En cela, la plupart des études restent donc théoriques. Il est toutefois évident qu'il existe une gradation entre une estimation « purement théorique » et une valeur pratique, dépendant de la finesse avec laquelle les contraintes pesant sur l'estimation de capacité sont prises en compte.

Notre approche ne prétend pas proposer une estimation de capacité pratique au sens strict du terme. Nous verrons cependant qu'à la fois les données et le modèle choisis permettent de modéliser certaines contraintes significatives.

2.4 Méthodes existantes d'évaluation de la capacité

Des contributions très diverses sur le calcul de capacité sont présentes dans la littérature. Un effort de synthèse est réalisé par Kontaxi et Ricci [36] qui présentent des méthodes développées depuis les années 1950, comparent leurs caractéristiques et confrontent leurs hypothèses à travers une plateforme nommée RailCAT. Celle-ci permet la comparaison des méthodes sur divers critères (données d'entrée et de sortie, réalisme, précision, échelle d'étude, etc.). Les méthodes recensées sont principalement dédiées à l'échelle macroscopique et pour la plupart liées au réseau ferroviaire italien.

Lusby et al. [45] présentent également une étude exhaustive des méthodes de calcul de capacité d'infrastructures, précisant pour chacune d'elle le niveau d'étude à laquelle elle est dédiée (stratégique, tactique ou opérationnel), la modélisation du problème ainsi que l'algorithme de résolution utilisé.

Les méthodes de calcul sont souvent classifiées en plusieurs catégories, principalement en fonction de la précision des données qu'elles prennent en compte et des techniques de résolution utilisées pour parvenir à une estimation de capacité.

Les catégories que nous considérons ont été introduites par Hachemane [34] et sont également utilisées par Delorme [20] et Abril et al. [1]. Cette classification est toujours applicable aux méthodes les plus récentes, c'est pourquoi il convient de la réutiliser. Nous mentionnons toutefois quelques variantes de classification observées dans la littérature et les mettrons en lien avec celle-ci.

- Les méthodes analytiques, qui évaluent la capacité d'une infrastructure directement grâce à une formule calculatoire dans laquelle sont quantifiés divers paramètres influents sur celle-ci tels que la vitesse moyenne de circulation ou l'écart minimal de sécurité requis entre les trains. Cependant, dans le cas de nœuds complexes soumis à un trafic mixte, les hypothèses imposées par ces méthodes peuvent s'avérer simplistes et peu précises. La méthode proposée par l'UIC [75] est une méthode analytique, bien que ses spécifications laissent suffisamment de liberté pour le développement de méthodes classifiées comme méthodes de construction d'horaires.
- Les méthodes probabilistes, qui sont fondées sur une estimation probabiliste de la répartition des trains en entrée de l'infrastructure ou sur d'autres considérations telles que la probabilité que deux trains se trouvent au même moment sur le même élément de l'infrastructure. En incluant les variations liées à la réalité au sein de formules calculatoires, elles visent à en améliorer la précision tout en conservant un temps de calcul raisonnable.
- Les méthodes par simulation, qui, à partir de données précises sur l'infrastructure et les caractéristiques dynamiques des trains, permettent d'estimer avec exactitude la façon dont réagit l'infrastructure face à un trafic donné en entrée. Elles sont conçues pour fournir une excellente précision, mais leur objectif est en général davantage la validation d'une grille horaire issue d'une estimation de capacité plutôt que le calcul de la capacité même. Un exemple de logiciel largement utilisé permettant d'effectuer des simulations très précises est OpenTrack, décrit par Nash et Huerlimann [56].
- Les méthodes de construction d'horaires, dont le principe est d'insérer progressivement de la circulation supplémentaire dans l'infrastructure en sus d'une circulation déjà présente. Leur but est d'élaborer la grille horaire contenant le nombre le plus grand possible de trains, ceux-ci étant choisis parmi une liste dont la composition doit être pertinente. Elles reposent sur des modèles d'optimisation permettant la prise en compte de données précises comme la description des zones de l'infrastructure ou différents profils de vitesse disponibles pour les trains. Cela permet le calcul de grilles horaires optimales ou quasi-optimales tenant compte des contraintes telles que définies par Hachemane [34]. L'augmentation de la précision est toutefois réalisée au prix du temps de calcul et du développement de solutions algorithmiques complexes.

Les méthodes par construction d'horaire ne sont pas mentionnées nécessairement en ces termes dans la littérature. Du fait que beaucoup de ces méthodes ont été récemment développées grâce à l'utilisation de techniques d'optimisation, elles sont parfois référencées sous le terme de « méthodes par optimisation », notamment par Abril et al. [1] et Kontaxi et Ricci [36]. En outre, il est important de noter que parmi les méthodes utilisant des algorithmes d'optimisation, le terme de « construction d'horaire » doit être nuancé. En effet, même si les méthodes d'optimisation sont celles qui proposent en général les capacités de modélisation les plus fines, le niveau de réalisme de l'horaire qui est produit reste variable. Il convient donc de garder en mémoire que l'applicabilité de ces horaires peut être plus ou moins proche de la réalité.

Des classifications sensiblement différentes sont proposées dans la littérature, à l'image de celle donnée par Kontaxi et Ricci [36], qui bien que conservant la même manière de grouper les méthodes existantes, utilisent une terminologie différente. Toutefois, ces classifications restent cohérentes avec celle que nous considérons ici.

À condition de reposer sur une modélisation pertinente, les méthodes par construction sont de fait les plus à mêmes de proposer une estimation de la capacité proche d'une valeur pratique. C'est en conséquences à ces méthodes auxquelles nous nous intéressons, en mettant l'accent sur celles qui ont abouti à la réalisation d'une plateforme logicielle fonctionnelle. Un certain nombre de contributions additionnelles utilisant des technique d'optimisation mais n'étant pas forcément liée à une plateforme logicielle complète, sont également présentées.

2.4.1 Plateformes logicielles

2.4.1.1 CAPRES

Le logiciel CAPRES [10] [44] a été développé grâce à la contribution initiale de Hachemane [34]. Son objectif est l'évaluation de la capacité des infrastructures à échelle macroscopique, c'est-à-dire qu'il considère un ou plusieurs tronçons reliant plusieurs nœuds (gares, jonctions). Pour l'évaluation de la capacité, il s'intéresse aux deux problèmes suivants :

1. le problème d'élaboration, qui consiste à trouver l'ensemble des ordonnancements possibles permettant de faire circuler un ensemble de train sur une infrastructure donnée, sans toutefois fixer de façon exacte les dates de passage ;
2. le problème de saturation, qui consiste à introduire dans le réseau un ensemble de trains supplémentaires, sélectionnés depuis un ensemble prédéterminé (l'ensemble de trains saturants), jusqu'à ce qu'il ne soit plus possible de le faire, chaque train de l'ensemble saturant pouvant être ajouté plusieurs fois à des dates de passage différentes.

La méthode de résolution utilise des techniques de séparation du problème pour réduire l'espace de recherche et une méthode basée sur la programmation par contraintes pour la résolution.

Le modèle est construit autour de la notion d'événement. Un événement peut être soit l'arrivée soit le départ d'un train dans un nœud. Deux types de contraintes sur les événements sont utilisées dans le modèle :

– les contraintes potentielles, contraignant deux événements E et F à se produire à l'intérieur d'un intervalle temporel donné, permettant de modéliser par exemple le temps de parcours d'un train sur un tronçon ou sa durée d'arrêt en gare :

$$h_E + t^- \leq h_F \leq h_E + t^+ \tag{2.1}$$

10. Système d'aide à l'analyse de la capacité des réseaux ferroviaires.

où le couple (t^-, t^+) avec $t^- < t^+$ permet de définir la taille de cet intervalle ;
– les contraintes disjonctives, forçant une durée minimale devant s'écouler entre deux événements, permettant par exemple la modélisation des distances de sécurité :

$$h_E + t_1 \leq h_F \text{ ou } h_F + t_2 \leq h_E \tag{2.2}$$

où t_1 et t_2 représentent des durées permettant de définir cet intervalle de disjonction entre les événements.

Afin de prendre en compte le caractère limitant des nœuds sur la capacité globale, ceux-ci peuvent être modélisés selon l'un des trois types suivants :
– le nœud dit normal, permet de modéliser un dédoublement temporaire des voies sans qu'il y ait de croisement (et donc sans *cisaillement* possible), avec l'éventuelle présence de quais (en cas de gares de passage) ;
– le nœud de type J permet de modéliser une jonction contenant des aiguillages et où des phénomènes de cisaillement sont possibles, à l'intérieur duquel l'arrêt des trains est interdit, où des itinéraires peuvent être définis comme incompatibles (à cause du cisaillement) et devant être empruntés avec un temps de séparation minimal (contrainte disjonctive) ;
– le nœud de type D (domino) permet de modéliser les gares complexes, contenant des itinéraires incompatibles, des temps permettant d'assurer les correspondances, etc.

Toutefois, aucune optimisation n'est réalisée à échelle locale : la méthode se contente de vérifier si une circulation donnée (avec points et dates d'entrée et sortie fixés) peut effectivement traverser le nœud.

2.4.1.2 STATIONS

STATIONS est un projet issu des travaux proposés par Zwaneveld et al. [78, 79] et Kroon et al. [37]. Il s'agit d'un sous-ensemble du projet néerlandais DONS [11] qui est centré sur la conception de grilles horaires pour le réseau néerlandais. Le but de STATIONS est la résolution du problème de faisabilité à l'échelle microscopique.

Bien que n'étant pas à proprement parler focalisé sur un calcul de capacité (les auteurs s'intéressent à la faisabilité plutôt qu'à la saturation), le projet se base sur une modélisation sous forme de PLNE [12], et plus précisément sous la forme de *Node Packing Problem* (NPP) qui est un problème classique d'optimisation. La résolution se fait par l'utilisation d'une combinaison de méthodes liées à la PLNE.

La formulation proposée permet de modéliser :
– un ensemble de trains devant circuler ;
– pour chaque train, un ensemble d'itinéraires possibles à travers l'infrastructure, une date d'entrée nominale et un éventuel ensemble de délais autorisés pour chacun des itinéraires de ce train ;
– des itinéraires dits d'entrée et de sortie dans le cas des gares ;
– les couples itinéraire-délai incompatibles, en se basant sur l'occupation des zones de détection impliquée par chacun de ces couples possibles pour chaque train ;
– une éventuelle pondération de certaines associations train-itinéraire-delai afin de privilégier (ou prévenir) leur sélection.

L'objectif de la formulation est de maximiser la pondération totale des triplets train-itinéraire-délai sélectionnés.

La méthode de résolution est décrite en détail par Zwaneveld et al. [78, 79], Kroon et al. [37] se focalisant davantage sur les aspects de complexité liés au problème considéré. La résolution consiste en :

11. *Design Of Network Schedules*, en anglais.
12. Programme linéaire en nombres entiers.

– des pré-traitements sur les données visant à ne considérer que certaines zones-clés de l'infrastructure (aiguillages, intersections, quais, points d'entrée/sortie sur l'infrastructure) et à supprimer les itinéraires ne devant être de façon évidente pas pris en compte (appelés itinéraires de détour) ;
– des pré-traitements sur la formulation NPP visant à réduire sa taille et à en améliorer ses propriétés ;
– un algorithme de résolution exacte, incluant des heuristiques visant à rendre l'algorithme performant en fournissant des solutions de bonne qualité le plus rapidement possible.

L'originalité de ces travaux réside dans le fait qu'ils figurent parmi les premiers à réaliser une optimisation à échelle microscopique malgré le fait qu'il s'agisse du problème de faisabilité, et ce en prenant en compte des données précises à la fois sur l'infrastructure et sur la demande de circulation. En outre, le modèle mathématique qu'ils proposent a servi de base à plusieurs travaux portant sur les problèmes de saturation et de faisabilité à l'échelle d'un nœud.

2.4.1.3 RECIFE PC

Le projet RECIFE [13] est une étude scientifique visant à estimer la capacité des infrastructures ferroviaires à l'échelle microscopique, en modélisant finement l'infrastructure et en prenant en compte plusieurs facteurs de consommation de capacité. Le projet a conduit à l'implémentation de deux logiciels :

– RECIFE, présenté par Rodriguez [63], dont le but est d'ordonnancer un ensemble de trains à travers l'infrastructure, de manière à minimiser le retard généré, grâce à une modélisation et une résolution par la programmation par contraintes du comportement des trains dans l'infrastructure ;
– RECIFE PC, présenté par Rodriguez et al. [64] et Gandibleux et al. [33], dont le but est d'effectuer une étude de capacité à échelle microscopique, par saturation de l'infrastructure et réalisation d'étude de stabilité.

Dans l'optique du problème de capacité, nous nous concentrons ici sur RECIFE PC, qui permet l'étude du problème de saturation. Le logiciel propose plusieurs outils :

– un module d'optimisation permettant de traiter les problèmes de faisabilité et de saturation à échelle microscopique, et ainsi obtenir une estimation de capacité ;
– un module d'étude de la stabilité permettant de comparer le comportement des grilles horaires générées par le module d'optimisation face à l'introduction de délais sur la circulation des trains ;
– divers modules de visualisation et de statistiques permettant d'étudier en détail l'utilisation de l'infrastructure par une grille horaire.

Le présent travail est en outre destiné à résoudre certaines problématiques posées par des jeux de données utilisées dans le cadre du développement de RECIFE PC. À ce titre, un de nos objectifs est de contribuer à l'amélioration de cette plateforme. Celle-ci fait donc l'objet d'une présentation plus détaillée au chapitre 5, parallèlement à une présentation des contributions que nous y apportons. Cette présente section propose un succinct historique des différentes méthodes de calcul de la saturation implémentées dans RECIFE PC.

Le modèle mathématique pour la faisabilité et la saturation s'inspire du travail de Zwaneveld et al. [78, 79] et consiste en une formulation en programme linéaire en variables binaires de *Set Packing Problem* (SPP). À l'instar de STATIONS, des informations détaillées sur l'infrastructure et les temps de passage sur les zones de détection sont disponibles et rendent pertinente l'utilisation d'un modèle prenant en compte des données précises.

Les importantes difficultés combinatoires posées par le problème de saturation ont impliqué le développement de métaheuristiques pour la résolution de sa formulation SPP. Delorme et al. [21] ont proposé l'al-

13. Recherches sur la capacité des infrastructures ferroviaires.

gorithme GRASP dans cette optique, puis Gandibleux et al. ont proposé la métaheuristique ACO [32] pour laquelle il a été constaté des résultats de meilleure qualité.

Le module d'étude de stabilité est basé sur la modélisation proposée par Delorme et al. [22]. Elle se fait en introduisant un retard (appelé retard primaire) sur un train et en mesurant le total des retards (appelés retards secondaires) engendrés sur les autres trains. Grâce à cette approche, l'utilisateur peut identifier les grilles horaires saturées les plus résistantes aux retards, pour des valeurs de retard primaire diverses. Le module ne réordonnance pas les trains pour minimiser la somme des retards secondaires mais se contente de propager les délais de manière à ce que les contraintes de sécurités restent satisfaites. La réoptimisation de grilles horaires est l'objet d'études dédiées, comme en témoignent celles présentées par Nakamura et al. [55] et D'Ariano et al. [19].

Si la prise en compte de données détaillées sur la demande et sur l'infrastructure sont un avantage de RECIFE PC, cela se fait au prix de temps de calcul parfois rédhibitoires pour le problème de saturation. La résolution du problème avec une demande de trafic de l'ordre de plusieurs centaines de trains tout en pouvant choisir précisément leur date d'entrée (à l'ordre de la seconde près) ont entraîné des temps de résolution atteignant plusieurs dizaines d'heures.

2.4.1.4 MOM

Abril et al. [1] et Barber et al. [4] proposent également un logiciel d'estimation de la capacité d'infrastructures ferroviaires dédié aux études macroscopiques. Les études de cas présentées par Abril et al. [1] concernent des tronçons du réseau ferré espagnol, d'une longueur de l'ordre de la centaine de kilomètres.

Le logiciel, MOM [14], possède les fonctionnalités suivantes :

- optimisation de grilles horaires existantes ;
- planification de trains supplémentaires à l'intérieur d'une grille horaire existante ;
- validation et réalisation d'études de capacité théoriques ou pratiques ;
- simulation de l'impact de retards ou d'incidents ;
- mesure de la stabilité (robustesse) de grilles horaires ;
- replanification de grilles horaire en fonction des retards ou incidents.

Abril et al. [1] insistent sur la complémentarité des différentes méthodes d'évaluation de la capacité (analytique, par optimisation et par simulation). De ce fait, une des spécificités de MOM est d'intégrer à la fois des méthodes analytiques et des méthodes d'optimisation pour l'évaluation de la capacité. Deux méthodes analytiques sont proposées dans le but d'estimer la capacité d'un tronçon : l'une est une modification d'une formule proposée par l'UIC [73] pour suivre des recommandations plus récentes de cette même UIC [74] et l'autre est une méthode *ad-hoc* dédiée au trafic périodique (voir Abril et al. [2] et Salido et al. [68]). Les méthodes d'optimisation ont quant à elles pour but la validation des résultats des méthodes analytiques en proposant les fonctionnalités suivantes :

- compacter une grille horaire afin d'en identifier la capacité résiduelle ;
- calculer des grilles horaires alternatives saturées cohérentes avec une grille horaire déjà existante ;
- saturer une grille horaire avec de nouveaux trains.

MOM n'effectue pas d'optimisation à l'échelle des nœuds. On remarque en outre que la méthode UIC [75] est classifiée comme méthode d'optimisation, choix discutable mais justifié par le fait qu'elle modifie la grille horaire considérée en rapprochant les sillons au maximum les uns des autres.

14. En espagnol, *Modulo Optimizador de Mallas*.

2.4.2 Lignes directrices de l'Union internationale des chemins de fer

L'UIC [75] suggère une méthode présentée comme un effort d'unification dans la manière d'appréhender et de calculer la capacité des infrastructures, et ce auprès d'une vaste partie des opérateurs et gestionnaires d'infrastructures du Monde. Il ne s'agit ni d'une plateforme logicielle en tant que telle ni du traitement d'un cas précis d'étude de capacité d'infrastructure, mais son impact auprès des différents acteurs ferroviaires nécessite de fait de s'y intéresser. Le principe de la méthode, dite méthode par compactage, consiste à resserrer au maximum, sans en changer l'ordre, les sillons des trains déjà présents sur l'infrastructure sur une fenêtre de temps donnée. La capacité dite résiduelle est alors mesurée par la durée disponible restante pour faire passer du nouveau trafic.

La méthode par compactage se fait sur une infrastructure donnée, pour une grille horaire et une fenêtre temporelle données. Elle est divisée en deux phases :

1. le compactage de la grille horaire de base, permettant d'obtenir la durée disponible pour ajouter du trafic supplémentaire ;
2. l'enrichissement, c'est-à-dire l'ajout de trafic supplémentaire dans la fenêtre temporelle éventuellement dégagée par le compactage.

Après compactage, la consommation de capacité est mesurée par la durée écoulée entre le début de l'occupation du premier train et la fin de l'occupation du dernier train, à laquelle peuvent être ajoutés des suppléments comme des marges tampon pour améliorer la robustesse ou des temps de maintenance. Le taux d'occupation K est le rapport entre la consommation et la longueur de la fenêtre temporelle choisie pour l'étude. Si K est supérieur à une limite choisie arbitrairement, l'infrastructure est considérée comme trop utilisée. S'il est inférieur, on considère qu'il existe une capacité inutilisée. Dans ce cas, l'horaire de base est enrichi avec un sillon supplémentaire, le nouvel horaire est compressé à nouveau, et ainsi de suite jusqu'à ce que la limite soit atteinte. L'UIC [75] donne des valeurs typiques pour les limites de taux d'occupation K en fonction du type de trafic, résumées dans le tableau 2.1. Les taux limites sont différenciés selon que l'étude porte sur une fenêtre temporelle considérée comme une heure de pointe ou comme une période de trafic normal. Les lignes directrices de l'UIC [75] semblent donc permettre, selon le taux d'occupation calculé, d'estimer facilement si une infrastructure doit être considérée comme saturée ou non.

Lindner [43] présente toutefois une série de problèmes posés par l'application de la méthode par compactage, notamment à l'échelle macroscopique. Lors de la phase de compactage, l'UIC [75] impose que le nombre de trains soit invariant sur l'infrastructure pendant la fenêtre temporelle considérée. Aucun train ne peut donc entrer ou sortir de l'infrastructure considérée par un embranchement intermédiaire : tous les trains doivent entrer par le nœud initial et sortir par le nœud situé à l'autre extrémité de l'infrastructure étudiée. Dans le cas contraire, la méthode par compactage devrait être appliquée séparément de part et d'autre de l'embranchement. Par la suite, l'enrichissement étant réalisé sur la même portion d'infrastructure que le compactage, l'ajout d'un sillon sur une portion peut impliquer un conflit non détecté dans une portion connexe où le compactage a été réalisé indépendamment. L'UIC [75] ne présente en outre aucun exemple en présence de trafic mixte, ignorant ainsi les problèmes engendrés par ce cas de figure. De plus, le processus ne contient ni méthode ni critère sur la façon d'insérer les sillons additionnels, ce qui peut pourtant influer significativement sur la mesure de capacité résiduelle.

La restriction du compactage à une infrastructure au sein de laquelle le nombre de trains est constant pendant la fenêtre temporelle considérée implique qu'il est nécessaire d'appliquer la méthode un grand nombre de fois dans les nœuds complexes. Lindner [43] donne en outre des exemples où l'application de la phase d'enrichissement au sein d'un nœud aboutit à une estimation de capacité fortement en deçà de sa valeur. Pour terminer, l'ajout arbitraire séquentiel de sillons tel que décrit par la méthode ne cherche

pas à réaliser un quelconque ordonnancement des trains pour optimiser l'utilisation de l'infrastructure, ordonnancement pourtant crucial dans les nœuds pour estimer la quantité de trafic pouvant circuler.

2.4.2.1 Interprétations et applications de la méthode

De par sa généricité, la méthode de l'UIC [75] est plus souvent vue comme un ensemble de spécifications menant au développement de méthodes plus spécifiques. Chacune de ces méthodes interprète les lignes directrices de l'UIC [75], et se définissent en général comme « compatibles » avec celles-ci. Leur cadre d'application est plus restreint que ce qu'entend traiter l'UIC [75] par une méthode unique et générique. Cette restriction est nécessaire en raison des hypothèses particulières faites selon l'infrastructure et le trafic considérés. Nous proposons ici un aperçu de deux méthodes récentes cherchant à respecter les lignes directrices de l'UIC [75], l'une à échelle macroscopique et l'autre à échelle microscopique.

Kuckelberg et al. [38] s'appuient sur la méthode par compactage pour répondre à un problème posé par la transposition dans la loi allemande de la directive européenne permettant aux gestionnaires d'infrastructure et aux opérateurs de conclure des accords (appelés « accords cadres ») à long-terme. La loi allemande spécifie que les sillons réservés par un accord-cadre ne peuvent représenter plus de 75% de la capacité totale. Il est donc nécessaire d'estimer cette capacité.

L'étude porte sur le cas de lignes à deux voies, où chacune est *a priori* réservée à un sens de circulation, mais où des trains peuvent réserver des zones ponctuelles en sens opposé. La phase de compactage est agrémentée de deux améliorations visant à éviter les surestimations de consommation de capacité. L'amélioration par rapport à la méthode UIC [75] porte essentiellement sur la gestion des trains roulant en sens opposé et ne se croisant que de façon ponctuelle : en décomposant leur itinéraire, les auteurs parviennent à améliorer le compactage de la grille horaire. Il est fait état de l'application de la méthode à une échelle significative, c'est-à-dire sur environ 4 000 éléments d'infrastructure, concernés par un total d'environ 28 600 trains.

Libardo et al. [41] développent une méthode d'estimation de capacité d'infrastructure à échelle microscopique dont le principe s'apparente à une forme de méthode de compactage suivie d'une saturation de l'infrastructure. Les techniques employées pour arriver à ces fins s'appuient sur des modèles PLNE. Ainsi, bien qu'étant présentée comme compatible avec la méthode de l'UIC [75], ce travail illustre à nouveau la nécessité d'adapter et de concrétiser ladite méthode en fonction du cas à traiter. L'exemple applicatif est la gare de passage italienne de Camposampiero.

La résolution se déroule en deux phases. La première consiste à identifier les tuples d'itinéraires pouvant être activés en même temps sans créer de conflits. Chaque tuple retourné est saturant, c'est-à-dire qu'il n'est pas possible d'activer un itinéraire supplémentaire sans créer de conflit.

C'est la seconde phase qui reprend le principe de la méthode UIC [75]. Les auteurs proposent un modèle d'optimisation sous forme de PLNE pour chacune des étapes de compactage et d'enrichissement, ce qui n'est pas sans rappeler les approches proposées par Zwaneveld et al. [78, 79], Hachemane [34] et Delorme [20]. L'objectif pour le compactage est toutefois la minimisation du temps total nécessaire pour faire circuler une demande donnée, ce qui se rapproche davantage de la méthode suggérée par l'UIC [75].

La solution optimale pour le compactage consiste en l'activation de certains tuples, minimisant le temps total utilisé pour faire passer l'ensemble du trafic demandé. L'absence de solution réalisable signifie que la demande requiert une capacité supérieure à celle de l'infrastructure.

Le problème d'optimisation pour la seconde étape est identique, à ceci près que l'objectif devient la maximisation de la quantité de circulation sous contrainte de garder les tuples précédemment sélectionnés pour conserver la faisabilité de l'horaire de base.

Si la méthode proposée est simple et relativement aisée à appréhender, cela se fait au prix de la non prise en compte de certaines données d'entrée. Il n'est notamment pas possible de spécifier des heures de passage autorisées pour les trains de la demande de circulation, ce qui limite fortement la prise en compte des caractéristiques de cette dernière.

Cette approche montre en outre que si la méthode UIC [75] n'est pas en elle-même une méthode par optimisation, le suivi de ses recommandations peut mener à l'implémentation de telles approches.

2.4.3 Autres méthodes par optimisation

La littérature intègre finalement un certain nombre de contributions additionnelles utilisant des méthodes d'optimisation et étant liées à des études de capacité concrètes et précises, n'ayant pour certaines pas nécessairement conduit à la présentation détaillée d'une plateforme logicielle orientée utilisateur. Ces contributions sont à ce titre des éléments pertinents connexes au présent travail et il convient donc d'en fournir une vue d'ensemble.

À l'échelle macroscopique, Liebchen [42] propose de générer des grilles horaires optimisées, au sein desquelles des événements consécutifs comme l'arrivée puis le départ d'un train dans une station doivent se dérouler à l'intérieur d'un intervalle temporel donné. L'auteur se base pour la modélisation sur le problème d'ordonnancement d'événements périodiques, initialement défini par Serafini et Ukovich [70]. Les travaux de Cacchiani et al. [9] et Caprara et al. [10, 11] se focalisent sur l'optimisation de grilles horaires sur certains tronçons, en utilisant un modèle d'optimisation basé sur la PLNE et en proposant des méthodes de résolution adaptées à la fois heuristiques et exactes issues du domaine de la PLNE. Une modélisation PLNE est également utilisée au sein du logiciel commercial DÉMIURGE, dont une présentation est donnée par Labouisse et Djellab [39], utilisé par la SNCF pour résoudre la faisabilité et la saturation à échelle macroscopique. Borndörfer et Schlechte [7] étudient la saturation sur le réseau allemand à travers la résolution du problème dit d'allocation optimale de voie. Les auteurs présentent également un modèle de PLNE et un ensemble de méthodes algorithmiques pour sa résolution. L'approche est par la suite généralisée par Schlechte [69], qui suggère un nouveau processus de planification pour le réseau ferroviaire européen dans le cadre de la libéralisation de celui-ci et propose une intégration des méthodes d'optimisation et de simulation. Une combinaison de l'optimisation et de la simulation est également proposée par Confessore et al. [15] dans le cadre d'une étude de capacité sur une ligne italienne. Les auteurs proposent un module d'optimisation présenté comme étant une implémentation des lignes directrices de l'UIC [75].

Un nombre plus réduit d'études à échelle microscopique basées sur l'optimisation peuvent être trouvées dans la littérature. En premier lieu, Lusby et al. [46, 47, 48] proposent une modélisation PLNE et un algorithme de résolution dédiés au problème de faisabilité. La méthode proposée est notamment capable de prendre en compte un grand nombre de profils de vitesses pour chaque train, et résout de ce fait un problème particulièrement complexe en un temps de calcul de l'ordre de quelques minutes. D'Ariano et al. [19] proposent des algorithmes de réordonnancement visant à restaurer la faisabilité d'une grille horaire suite à des perturbations sur le réseau (par exemple, un train retardé ou une voie bloquée). Ce travail fait l'objet d'une implémentation dans la plateforme logicielle ROMA, présentée par D'Ariano et al. [18] et conçue pour l'aide à la décision au niveau opérationnel. Pour terminer, Van Egmond [76] vise un objectif comparable, en proposant de minimiser le temps requis pour faire circuler une demande donnée tout en assurant une certaine robustesse par l'introduction de temps additionnels.

L'aperçu non exhaustif de ces méthodes montre qu'un nombre important d'études liées à la capacité des infrastructures existe, utilisant notamment des techniques d'optimisation issues du domaine de la PLNE. Notre connaissance de la littérature montre également qu'un nombre nettement plus important

d'études sont focalisées sur l'échelle macroscopique par opposition à l'échelle microscopique, bien que nous ayons montré le rôle crucial joué par les nœuds ferroviaires sur la capacité d'un réseau.

2.5 Synthèse

Le contexte économique, particulièrement en Europe, montre l'intérêt croissant porté sur le transport ferroviaire, et implique à la fois une augmentation de la demande, du nombre d'acteurs présents sur le marché et du matériel circulant sur l'infrastructure. L'estimation de la capacité est devenu en conséquence une donnée cruciale dans les négociations entre exploitants et gestionnaires.

La capacité est une notion complexe à définir, prenant en compte un grand nombre de contraintes opérationnelles et pouvant être mesurée par divers indicateurs. Nous avons choisi de privilégier la mesure de la capacité par un nombre de trains pouvant circuler pendant une période temporelle fixée, telle que défini par Hachemane [34].

De nombreuses méthodes de calcul ont été développées, parmi lesquelles les méthodes par construction d'horaire utilisant des algorithmes d'optimisation autorisent la prise en compte de données précises, et conséquemment une estimation de capacité plus précise que pour les autres types d'approche. Les méthodes sont dédiées soit à l'échelle microscopique soit à l'échelle macroscopique. Peu d'études visant à obtenir une estimation de capacité ont été réalisée à l'échelle microscopique : CAPRES et MOM se situent à échelle macroscopique, et STATIONS se focalise sur le problème de faisabilité. En outre, la méthode générique proposée par l'UIC [75] n'est pas adaptée non plus, et ne fournit que peu d'éléments pour l'implémentation de méthodes plus spécifiques.

Les travaux présentés par Libardo et al. [41] présentent une méthode d'estimation de capacité à échelle microscopique, mais ne prennent pas en compte certaines données cruciales de la demande. La plateforme RECIFE PC, développée à partir des travaux de Delorme [20, 21] et Gandibleux et al. [32, 33] permet quant à elle la prise en compte de détails supplémentaires au sein de la demande de trafic. La combinatoire est toutefois une barrière importante pour le traitement des instances de grande taille modélisant une situation réelle avec un bon niveau de précision, entraînant des temps de résolution extrêmement longs.

Il est en conséquence nécessaire de développer des méthodes appropriées pour répondre à ce problème. De telles méthodes peuvent être construites en se basant sur une modélisation sous forme de problème de PLNE, et plus précisément de *Set Packing Problem* (SPP), permettant ensuite l'application de techniques algorithmiques adaptées à de telles formulations. Le prochain chapitre introduit donc les notions mathématiques nécessaires autour de la résolution de problèmes de PLNE, formalise le SPP et détaille la modélisation que nous privilégions pour la résolution, à savoir une modélisation dite par disjonction de ressources.

Type de ligne	Heure de pointe	Trafic normal
Trafic dédié intra-agglomération	85%	70%
LGV dédiée	75%	60%
Ligne à trafic mixte	75%	60%

Table 2.1 – Limites de taux d'occupation d'infrastructure selon le Code UIC 406.

CHAPITRE 3

Formalisations autour de la disjonction de ressources

Le chapitre précédent a permis de mettre en valeur de nombreuses approches existant dans la littérature pour réaliser des études de capacité d'infrastructures ferroviaires. Bien qu'aucune ne satisfasse pleinement nos besoins, certaines apportent des avancées très intéressantes dans le traitement du problème de capacité. Les méthodes les plus précises se basent sur des modèles et algorithmes d'optimisation mathématique. Plus précisément, les approches présentées par Libardo et al. [41], Delorme et al. [20, 21], Gandibleux et al. [32, 33] et Lusby et al. [46] pour des études à échelle microscopique ont en commun l'utilisation de la programmation linéaire en nombres entiers (PLNE), technique issue du domaine de la recherche opérationnelle.

Forts de ces constats, nous utilisons également la PLNE comme outil de modélisation et de résolution du problème de saturation. Ce second chapitre présente donc un aperçu de la PLNE en introduisant le vocabulaire et les notations nécessaires. Résoudre un problème de PLNE non trivial se fait souvent en ayant recourt à ce qu'on appelle sa relaxation continue : celle-ci est définie et un aperçu de ses propriétés est donné. Un algorithme de résolution classique pour de tels problèmes, l'algorithme dit du simplexe, est ensuite résumé et un aperçu des fondements théoriques sur lesquels ils se base est donné. Nous nous penchons ensuite sur les notions principales à connaître autour des techniques de résolution exacte de problèmes de PLNE, et plus précisément sur l'encadrement de la valeur optimale d'un problème de PLNE à l'aide de diverses méthodes, visant *in fine* à déterminer cette valeur optimale.

La modélisation PLNE que nous utilisons pour le problème de saturation – le *Set Packing Problem* (SPP) – est ensuite détaillée et expliquée, notamment en proposant une présentation d'études théoriques précédemment réalisées et portant sur sa structure. Une fois les bases essentielles concernant le SPP posées, les données ferroviaires (demande de trafic, infrastructure, etc.) sont formalisées, ce qui nous permet ensuite de lier ces données à une formulation SPP. Plus exactement, deux formulations issues de la littérature sont présentées et leurs qualités respectives en vue de leur résolution sont discutées. Nous choisissons finalement une formulation SPP, dite par disjonction de ressources, au vu des qualités qu'elle possède.

Certaines propriétés de la formulation SPP dépendent directement des instances traitées. De ce fait, la méthode de résolution ne peut être conçue pertinemment qu'en ayant connaissance des instances, et plus précisément des propriétés numériques des instances SPP qu'elles engendrent. Ainsi, le chapitre se termine par la présentation des instances que nous traitons, d'abord en termes ferroviaires puis sous l'angle purement mathématique. La génération des instances que nous traitons est réalisée en faisant

varier trois paramètres précis, permettant à la fois d'obtenir une large variété d'instances tout en se dotant de repères pour l'analyse des résultats.

3.1 Contexte de la programmation linéaire en nombres entiers

3.1.1 Vocabulaire et notations

La programmation linéaire (PL) est une discipline de la recherche opérationnelle permettant la résolution de problèmes d'optimisation sous contraintes. Un programme linéaire permet de modéliser ces problèmes par une formulation possédant les éléments caractéristiques suivants :

- des *variables de décision* (ou simplement variables) représentant des quantités que l'on cherche à déterminer ;
- des *contraintes* portant sur ces variables, modélisant les limites imposées à ces quantités ;
- une *fonction-objectif* (ou simplement objectif), représentant la quantité que l'on cherche à optimiser (i.e. à minimiser ou maximiser), exprimée également en fonction des variables de décision.

Dans un programme linéaire, les contraintes et la fonction-objectif sont linéaires en fonction des variables. Les variables prennent leur valeur dans un intervalle de $\mathbb{R}$. On parle donc également de problèmes en *variables continues*. Un problème de PL $\mathcal{P}$ à $n \in \mathbb{N}$ variables $(x_j)_{1 \leq j \leq n}$ et $m \in \mathbb{N}$ contraintes peut être formalisé de façon générique comme suit :

$$\mathcal{P} = \left[\begin{array}{lll} \max z(x) = & \displaystyle\sum_{1 \leq j \leq n} \rho_j x_j & \\ \text{s.c.} & \displaystyle\sum_{1 \leq j \leq n} a_{ij} x_j = b_i & , \ 1 \leq i \leq m \\ & x_j \in \mathbb{R} & , \ 1 \leq j \leq n \end{array} \right] \tag{3.1}$$

où les $\rho_1, \ ..., \ \rho_n$ sont des réels appelés *poids* ou encore *coûts* respectifs des variables $x_1, \ ..., \ x_n$ et où $a_{ij} \in \mathbb{R}$ est le coefficient de la variable x_j dans la $i^{\text{ème}}$ contrainte. Il est également courant de trouver une notation matricielle équivalente :

$$\mathcal{P} = \left[\begin{array}{ll} \max z(x) = & \rho x \\ \text{s.c.} & Ax = b \\ & x \in \mathbb{R}^n \end{array} \right] \tag{3.2}$$

où ρ est un vecteur-colonne composé des n éléments respectifs $\rho_1, \ ..., \ \rho_n$, x est un vecteur-ligne composé respectivement des n variables $x_1, \ ..., \ x_n$, b est un vecteur-colonne composé des m éléments respectifs $b_1, \ ..., \ b_m$ et A est une matrice de m lignes et n colonnes appelée *matrice des contraintes*, dont chaque élément aux coordonnées (i, j) possède la valeur a_{ij}. Une colonne de A représente les coefficients d'une variable dans chacune des contraintes, et une ligne représente ceux de toutes les variables d'une contrainte. Ainsi, on trouve parfois dans la terminologie associée à la programmation linéaire les termes respectifs de *ligne* et *colonne* en lieu et place de contrainte et variable.

Il est bien entendu courant de trouver des formulations de PL qui semblent *a priori* ne pas correspondre à la formulation 3.1 du fait des différences suivantes :

- la fonction-objectif peut être une minimisation plutôt qu'une maximisation ;
- les contraintes ne sont pas nécessairement exprimées par des égalités.

Les notations 3.1 et 3.2 peuvent cependant être tout de même considérées comme génériques. En effet, minimiser une quantité est d'une part simplement équivalent à maximiser l'opposé de cette quantité. D'autre part, des inégalités dans les contraintes peuvent être ramenées à des égalités *via* l'introduction

de *variables d'écart* dans la formulation. Nous ne détaillons pas ici le processus d'introduction des variables d'écart, qui est un mécanisme inclus dans les solveurs de PL et donc considéré comme inhérent à la résolution de ces problèmes. Les formulations présentées par la suite pourront donc comporter indifféremment des égalités ou inégalités sans que cela implique une quelconque modification de l'approche de résolution.

Une *solution réalisable* pour $\mathcal{P}$ est une affectation de x respectant l'ensemble des m contraintes de ce problème. Nous nommerons *solution optimale* x^* pour $\mathcal{P}$ est une solution réalisable telle que $z(x)$ atteigne sa valeur maximale pour $x = x^*$, c'est-à-dire telle que $z(x) \leq z(x^*)$ pour toute solution réalisable x. *Résoudre* le problème $\mathcal{P}$ consiste à en trouver une solution optimale, qui n'est par ailleurs pas nécessairement unique. La distinction peut être faite entre une *résolution exacte*, dont le but consiste effectivement à trouver une ou plusieurs solutions dont l'optimalité est prouvée, et une *résolution approchée* dont le but est de trouver des solutions se rapprochant le plus possible des solutions optimales (en terme de valeur de fonction-objectif) sans pour autant prouver que les solutions trouvées sont optimales.

Il est finalement important d'introduire la notion de *problème dual* à un problème de PL. Le problème dual, ou simplement dual, noté $\mathcal{D}$, du problème $\mathcal{P}$, est décrit par l'expression 3.3 :

$$\mathcal{D} = \left[\begin{array}{lll} \min z^d(x) = & \sum\limits_{1 \leq i \leq m} b_i y_i & \\ \text{s.c.} & \sum\limits_{1 \leq i \leq m} a_{ij} y_j = \rho_j & , \ 1 \leq j \leq n \\ & y_i \in \mathbb{R} & , \ 1 \leq i \leq m \end{array} \right]. \tag{3.3}$$

Le problème $\mathcal{P}$ est alors désigné comme le *primal* de $\mathcal{D}$. Le dual d'une formulation linéaire est fortement lié à celle-ci, puisque les vecteurs b, ρ ainsi que la matrice A y sont conservés. Plus généralement, les règles suivantes s'appliquent pour la formulation du problème dual :

- si le primal est un problème de maximisation, le dual est un problème de minimisation, et vice-versa ;
- à chaque variable du primal correspond une contrainte du dual, et vice-versa ;
- les coefficients de la fonction-objectif du primal deviennent les second-membres des contraintes du dual, et vice-versa.

Le primal et son dual sont en outre liés par un ensemble de propriétés particulièrement intéressantes dans la perspective de la résolution des problèmes de PL, issues de la théorie de la dualité et notamment détaillées par Chvàtal [12]. Certaines de ces propriétés sont exploitées par les contributions que nous proposons au chapitre 4 et seront de ce fait énoncées lors de leur utilisation dans le cadre de la méthode de résolution que nous mettons en place.

En *programmation linéaire en nombres entiers* (PLNE), chaque variable doit prendre une valeur entière. De façon cohérente avec la formulation 3.1, un problème de PLNE peut donc être formalisé de façon générique par le problème $\mathcal{P}_E$ tel que décrit par l'équation 3.4 :

$$\mathcal{P}_E = \left[\begin{array}{lll} \max z(x) = & \sum\limits_{1 \leq j \leq n} \rho_j x_j & \\ \text{s.c.} & \sum\limits_{1 \leq j \leq n} a_{ij} x_j = b_i & , \ 1 \leq i \leq m \\ & x_j \in \mathbb{Z} & , \ 1 \leq j \leq n \end{array} \right]. \tag{3.4}$$

La contrainte d'intégralité implique l'impossibilité d'utiliser directement l'ensemble des propriétés des problèmes de PL et les algorithmes de résolution qui en découlent, notamment du fait que l'ensemble des solutions réalisables passe d'une structure continue à une structure discrète. Une solution optimale à un problème de PL n'est en effet dans le cas général pas réalisable pour la version PLNE du problème, l'intégralité des variables n'étant évidemment pas garantie *a priori* par un problème de PL. Toutefois, les algorithmes usuels de résolution des problèmes de PLNE sont construits autour de la résolution de la *relaxation continue* de ces problèmes : il s'agit de la formulation du problème de PLNE au sein duquel les contraintes d'intégralité sont omises (ou *relâchées*). La relaxation continue d'un problème de PLNE est donc un problème de PL, pour lequel une méthode de résolution classique consiste à utiliser un algorithme dit *algorithme du simplexe*. Il s'agit d'une brique technique essentielle pour un grand nombre de méthodes de résolution de PL et de PLNE.

3.1.2 Principe de l'algorithme du simplexe

L'algorithme du simplexe (aussi appelé simplement « simplexe » et proposé dans sa forme initiale par Dantzig et al. [16, 17]) est dédié à la résolution exacte des problèmes de PL. Il tire son efficacité de l'exploitation de deux propriétés essentielles de ces problèmes :

– l'ensemble des solutions réalisables d'un problème de PL forme un polyèdre convexe ;
– une solution optimale d'un PL se trouve à un des sommets du polyèdre convexe qui lui est associé.

Bien que nous n'entrions pas ici dans les détails algorithmiques du simplexe, il est nécessaire d'en connaître les principes généraux et d'avoir un aperçu des éléments qu'il manipule. Les détails calculatoires et théorèmes fondamentaux sont présentés de façon complète par Chvàtal [12].

Pour un problème de PL donné, le principe du simplexe consiste à améliorer itérativement une solution réalisable jusqu'à obtention d'une solution optimale. D'une solution réalisable initiale donnée, on cherche à déterminer une autre solution réalisable dont la valeur de l'objectif soit meilleure. Cette opération est répétée jusqu'à ce qu'il ne soit plus possible d'améliorer la solution courante, signifiant alors que celle-ci est optimale.

Une solution réalisable est associée à un sous-ensemble de variables appelé *base* vérifiant les propriétés suivantes :

– la restriction de A aux variables de la base (ou variables en base) est une matrice inversible ;
– les variables hors base sont fixées à la valeur extrême qu'elles sont autorisées à prendre dans la définition du problème (il est en pratique courant que cette valeur soit 0, lorsque les variables sont définies comme devant être positives).

L'amélioration itérative de la solution est réalisée en faisant rentrer en base une variable hors base dont le passage à une valeur non extrême est susceptible d'améliorer l'objectif. En contrepartie, une variable sort de la base. Cette opération est appelée *pivot*, et est est réalisée à chaque itération du simplexe. Il est possible de montrer qu'une solution ainsi calculée correspond à un sommet du polyèdre de l'ensemble des solutions réalisables. Le simplexe se « déplace » donc de sommet en sommet sur ce polyèdre.

La détermination des variables entrante et sortante à chaque pivot repose sur des critères mathématiques dont une explication complète est donnée par Chvàtal [12] :

– la variable entrante est choisie selon la valeur de son *coût réduit*, celui-ci devant être strictement positif (respectivement négatif) pour que la variable associée soit susceptible d'améliorer la valeur de l'objectif d'un problème de maximisation (respectivement de minimisation) ;
– la variable sortante est celle qui limite le plus l'augmentation de cette dernière du fait des relations entre les variables définies par les contraintes, s'assurant ainsi que l'ensemble des contraintes reste respecté.

Lorsqu'il n'existe aucune variable hors base possédant un coût réduit indiquant qu'elle est susceptible d'améliorer l'objectif, la base courante correspond à une solution optimale et est qualifiée de *base optimale*. Nous explicitons dans les chapitres 4 et 6 une méthode de calcul des coûts réduits dans le cadre particulier d'un algorithme de génération de colonnes que nous présentons. Nous ne donnons pas ici le détail des calculs matriciels permettant la détermination des coûts réduits, le choix de la variable entrante ainsi que celui de la variable sortante. Dans le cadre du simplexe, nous supposons enfin que la connaissance des coûts réduits de l'ensemble des variables du problème de PL considéré est inhérente à l'opération de pivot, ce qui correspond concrètement à une fonctionnalité offerte par la large majorité des logiciels de résolution de PL. Nous supposons également lorsque cela est nécessaire que le calcul d'indices de variables entrantes ou sortantes est obtenu grâce à l'existence de fonctions dédiées.

Pour résumer, l'algorithme du simplexe adopte le fonctionnement suivant afin de parvenir à une solution optimale pour un problème de PL :

1. déterminer une base réalisable initiale et calculer la valeur associée pour l'objectif ;
2. rechercher les variables dont l'entrée en base peut améliorer l'objectif ;
3. – si de telles variables sont trouvées, en sélectionner une et effectuer le pivot permettant de la faire entrer en base ;
   – dans le cas contraire, terminer l'algorithme car la solution optimale a été trouvée ;
4. réitérer à l'étape 2.

Si la complexité de l'algorithme du simplexe peut être exponentielle dans des cas défavorables (voir Klee et Minty [35]), il se révèle en pratique efficace pour la résolution d'un grand nombre de problèmes de PL, ce qui en fait un algorithme très couramment utilisé. Il est par ailleurs implémenté dans plusieurs logiciels d'optimisation libres (Coin-OR, glpk) comme commerciaux (Cplex, XpressMP). Il est à ce titre un élément constitutif central de méthodes de résolution plus évoluées, que ce soit pour la résolution de problèmes de PL ou de PLNE. Son existence est considérée comme un acquis lors de la présentation de nos contributions algorithmiques aux chapitres 4 et 6.

3.1.3 Recherche d'une solution entière optimale

L'algorithme du simplexe n'est pas réutilisable tel quel pour résoudre les problèmes de PLNE, le caractère discret de l'ensemble des solutions réalisables faisant perdre à celui-ci la structure de polyèdre convexe. L'impossibilité de se fonder sur cette propriété rend les problèmes de PLNE nettement plus difficiles à résoudre en terme de temps de calcul. Plus précisément, alors que la PL est de complexité polynomiale, la PLNE induit une complexité exponentielle.

Une stratégie essentielle dans la recherche d'une solution optimale à un problème de PLNE consiste, faute de pouvoir calculer celle-ci aisément, à encadrer sa valeur entre des *bornes* inférieure et supérieure [1] en utilisant des algorithmes considérés comme peu coûteux. Pour un problème de maximisation tel que $\mathcal{P}_E$ défini par l'équation 3.4, il est possible de faire ces simples constats :

1. toute solution réalisable de $\mathcal{P}_E$ est une borne inférieure pour une solution optimale de $\mathcal{P}_E$, c'est-à-dire que la valeur de la fonction-objectif pour cette solution est inférieure à la valeur de la fonction-objectif pour une solution optimale ;

1. Les termes de « bornes » supérieure et inférieure sont usuellement employés – y compris dans le présent manuscrit – pour mentionner l'encadrement de la valeur optimale d'un problème de PLNE, mais des traductions plus précises des termes anglais *upper bound* et *lower bound* seraient respectivement *majorant* et *minorant*.

2. toute solution optimale pour une relaxation de $\mathcal{P}_E$ (et en particulier pour la relaxation continue) est une borne supérieure pour une solution optimale de $\mathcal{P}_E$, c'est-à-dire que la valeur de la fonction-objectif pour cette solution est supérieure à la valeur de la fonction-objectif pour une solution optimale pour $\mathcal{P}_E$.

Ainsi, de nombreuses méthodes de résolution de tels problèmes adoptent une stratégie consistant à calculer des bornes supérieure par relaxation et inférieure par calcul de solution réalisable, puis à réduire l'écart progressivement. Si les bornes inférieure et supérieure sont de même valeur, cela signifie que la valeur optimale de la fonction-objectif est égale à la valeur des bornes. La conception d'algorithmes permettant le calcul en temps raisonnable de bornes de bonne qualité, c'est-à-dire les plus proches possible de la valeur optimale de la fonction-objectif, est donc un enjeux crucial de la résolution des problèmes de PLNE.

Du fait des qualités de l'algorithme du simplexe, faire appel à la relaxation continue pour déterminer une borne supérieure est souvent une approche pertinente. Une borne inférieure peut quant à elle être par exemple déterminée par un algorithme heuristique. Nemhauser et Wolsey [57] présentent les principales méthodes pouvant être mises en œuvre pour la résolution de problèmes de PLNE, telles que d'une part les algorithmes par *séparation et évaluation* [2] effectuant une énumération implicite de toutes les solutions du problème et d'autre part la génération progressive de contraintes appelées *inégalités valides* au sein de la relaxation continue. Une inégalité valide est une contrainte qui ne modifie pas l'ensemble des solutions réalisables du problème de PLNE mais qui restreint celui de sa relaxation continue, visant par ce biais à en améliorer la borne qui en découle.

L'exemple 3.1 définit une formulation triviale de PLNE et nous permet d'illustrer l'utilisation des bornes et des inégalités valides pour déterminer la solution optimale d'un problème de PLNE.

Exemple 3.1. Formulation de PLNE.

$$\mathcal{P}_{E_1} = \left[\begin{array}{llll} \min & 2x_1 & +x_2 & \\ s.c. & 2x_1 & & \geq 1 \\ & x_1 & +3x_2 & \geq 2 \\ & x_1 & , x_2 & \in \mathbb{N} \end{array}\right]. \tag{3.5}$$

Une solution réalisable pour $\mathcal{P}_{E_1}$ est $x_1 = x_2 = 1$, la valeur de l'objectif associée étant 3 : il n'est *a priori* pas possible de savoir s'il s'agit d'une solution optimale, mais cela constitue une borne supérieure pour $\mathcal{P}_{E_1}$. La relaxation continue de $\mathcal{P}_{E_1}$ possède une solution optimale $x_1 = x_2 = 0,5$, d'objectif égal à $1,5$. Les coefficients dans l'objectif étant entiers, on déduit qu'une borne inférieure pour $\mathcal{P}_{E_1}$ est 2. La figure 3.1 donne une représentation graphique des valeurs pouvant être affectées aux variables de $\mathcal{P}_{E_1}$: la surface grise est l'ensemble des points dont les coordonnées correspondent à des solutions réalisables pour sa relaxation continue, formant un polyèdre convexe borné par les contraintes de la formulation. On y visualise également, en noir, les solutions réalisables telles que x_1 et x_2 prennent une valeur entière. La solution optimale continue, représentée en rouge, correspond bien à un sommet de ce polyèdre.

Ajoutons maintenant la contrainte suivante à la formulation :

$$x_1 + x_2 \geq 2. \tag{3.6}$$

2. Adaptation de la terminologie anglaise *branch and bound*.

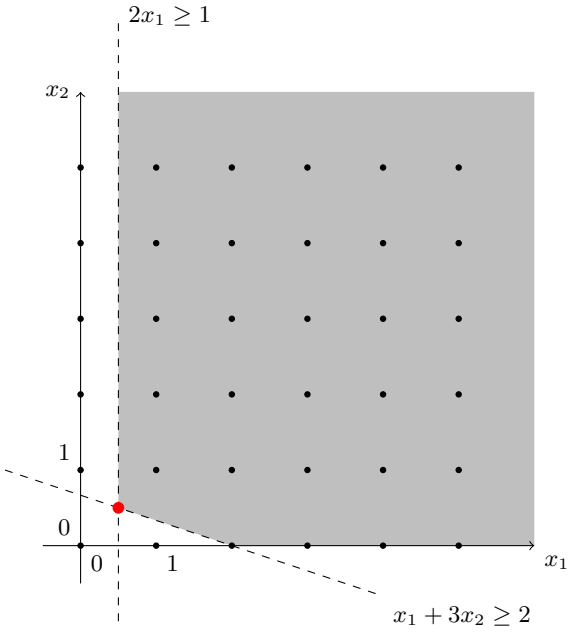

Figure 3.1 – Représentation de l'espace des variables du problème $\mathcal{P}_{E_1}$.

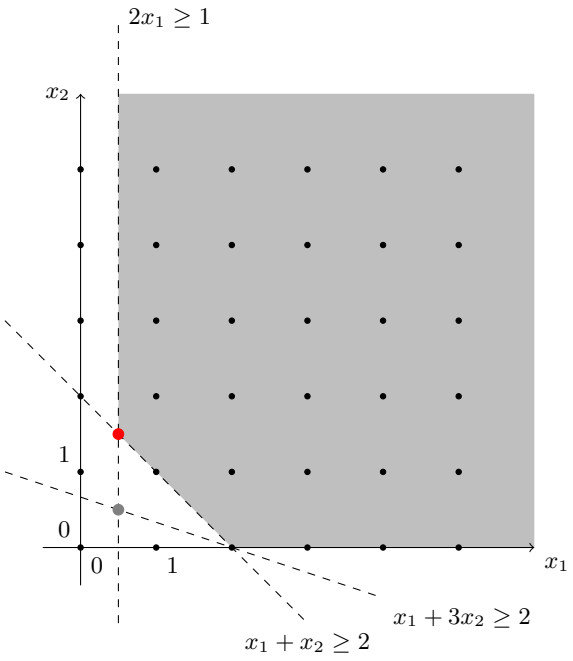

Figure 3.2 – Espace des variables du problème $\mathcal{P}_{E_1}$ après ajout de la contrainte supplémentaire.

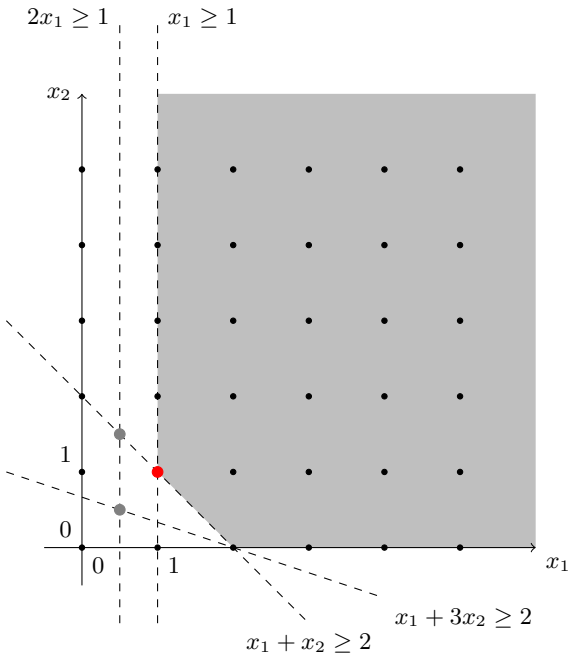

Figure 3.3 – Enveloppe convexe de $\mathcal{P}_{E_1}$: la solution optimale continue est une solution entière.

La figure 3.2 permet de visualiser l'impact de cette nouvelle contrainte sur l'ensemble des solutions réalisables. Celui-ci reste inchangé pour $\mathcal{P}_{E_1}$, en particulier l'affectation $x_1 = x_2 = 1$. Il s'agit donc d'une formulation équivalente du point de vue PLNE. La solution $x_1 = x_2 = 0,5$ n'est en revanche plus réalisable pour sa relaxation continue. L'optimum de la relaxation continue est obtenu pour $x_1 = 0,5$ et $x_2 = 1,5$, soit un objectif valant $2,5$. Une nouvelle borne inférieure pour $\mathcal{P}_{E_1}$ est donc 3, nous permettant de conclure que la solution $x_1 = x_2 = 1$ d'objectif égal à 3 est une solution optimale pour $\mathcal{P}_{E_1}$. Dans les situations les plus avantageuses, il est possible de générer les inégalités valides délimitant *l'enveloppe convexe* de $\mathcal{P}_{E_1}$, auquel cas le sommet du polyèdre convexe correspondant à une solution optimale pour la relaxation continue est également une solution optimale pour $\mathcal{P}_{E_1}$, permettant le calcul de la solution optimale entière avec un algorithme de résolution de PL tel que le simplexe. Dans le cas simple du problème $\mathcal{P}_{E_1}$, l'ajout de la contrainte $x_1 \geq 1$ suffit à obtenir l'enveloppe convexe de la formulation, tel que représenté par la figure 3.3.

Nemhauser et Wolsey [57] présentent plusieurs techniques de génération d'inégalités valides spécifiques à certains types de formulation de PLNE, pouvant donc être utilisées pour en améliorer la qualité de relaxation continue et resserrer les bornes.

Fermer l'écart entre les bornes n'est toutefois dans certains cas pas possible en temps raisonnable. Ceci peut être causé par la taille du problème considéré (nombre élevé de variables et/ou de contraintes) ou encore par sa structure (mauvaise qualité de relaxation continue et difficulté à trouver des inégalités valides). Il est donc également courant d'arrêter la résolution lorsque l'écart entre les bornes est jugé suffisamment petit ou lorsqu'un temps de résolution limite est atteint. Finalement, des algorithmes approchés tels les heuristiques et métaheuristiques sont conçus dans le but de trouver rapidement des solutions de bonne qualité tout en se passant d'en prouver l'optimalité. La qualité de tels algorithmes est souvent montrée empiriquement en effectuant de nombreuses résolutions, notamment sur des instances où les solutions optimales sont déjà connues. Lorsque la valeur de l'objectif à l'optimal n'est pas connue *a priori*, la qualité de la solution trouvée par un tel algorithme peut également être estimée par l'écart entre cette solution et une borne fournie par une autre méthode. Nous utiliserons cette approche dans le cadre de la résolution du modèle PLNE associé au problème de saturation, à savoir la formulation de *Set Packing Problem*.

3.2 Modélisation Set Packing par disjonction de ressources

3.2.1 Présentation du problème de Set Packing

Le *Set Packing Problem* (SPP) est un problème classique d'optimisation combinatoire. Étant donné un ensemble fini d'éléments $J = \{1, ..., n\}$ possédant des poids respectifs ρ_1, ..., ρ_n, et m sous-ensembles de J notés $G_1, ..., G_m$, un *packing* est un sous-ensemble d'éléments $P \subseteq J$ tel que $|G_i \cap P| \leq 1,\ 1 \leq i \leq m$. En d'autres termes, un packing est un ensemble composé d'éléments de J deux-à-deux disjoints au sens des sous-ensembles G_i. Ces éléments étant définis, deux questions peuvent être posées, correspondant aux deux variantes du SPP, à savoir :

- un problème de décision : existe-t-il un packing de poids total $K \in \mathbb{R}$?
- un problème d'optimisation : quel est le packing de poids maximal qu'il est possible de trouver ?

Pour modéliser le problème de saturation, nous nous focalisons sur le problème d'optimisation, c'est donc à cette variante que nous faisons référence en mentionnant le SPP. Il est possible de distinguer deux cas, selon les valeurs prises par les poids des éléments :

- si tous les poids ρ_j sont de même valeur, il s'agit d'un cas particulier appelé *Unicost Set Packing Problem* (USPP) où la recherche d'un packing de poids maximal est équivalent à chercher un packing contenant le maximum d'éléments ;
- s'il existe au moins deux éléments de poids différents, on parle de *Weighted Set Packing Problem* (WSPP).

Considérons n variables binaires $x_1, \ ..., \ x_n$, définies, pour tout j tel que $1 \leq j \leq n$, comme suit :

$$x_j \ = \ \begin{cases} 1 & \text{si l'élément } j \text{ est inclu dans le packing ;} \\ 0 & \text{sinon.} \end{cases} \tag{3.7}$$

Il est alors possible de formuler le SPP par un problème de PLNE, décrit par la formulation 3.8 :

$$\left[\begin{array}{lll} \max & \displaystyle\sum_{j \in J} \rho_j x_j & \\ \text{s.c.} & \displaystyle\sum_{j \in G_i} x_j \leq 1 & , 1 \leq i \leq m \\ & x_j \in \{0,1\} & , 1 \leq j \leq n \end{array} \right] . \tag{3.8}$$

On y retrouve bien les caractéristiques du SPP :

- la fonction-objectif, que l'on cherche à maximiser, somme le poids des éléments dont la variable associée vaut 1, ce qui correspond bien à maximiser le poids total du packing ;
- les m contraintes linéaires, aussi appelées *contraintes disjonctives* ou encore *contraintes de packing* dans le cas du SPP, s'assurent bien de la disjonction des éléments au sein de chaque sous-ensemble G_i.

Pour illustrer, considérons le SPP défini par l'exemple 3.2.

Exemple 3.2. Formulation SPP.
Soit une instance de SPP possédant les caractéristiques suivantes :

- *un ensemble d'éléments* $J = \{1, 2, 3, 4, 5, 6, 7\}$*, tous possédant un poids unitaire ;*
- *les sous-ensembles de* J *suivants :*
  - $G_1 = \{1, 2, 3\}$ *;*
  - $G_2 = \{2, 4, 5\}$ *;*
  - $G_3 = \{4, 6\}$ *;*
  - $G_4 = \{1, 3, 7\}$ *;*
  - $G_5 = \{2, 7\}$.

Les contraintes de ce SPP peuvent être aisément représentées graphiquement, comme cela est montré par la figure 3.4 : les six éléments y figurent en gris, ainsi que les contraintes disjonctives entre ces éléments, illustrées par les ellipses colorées. La formulation 3.9 décrit la représentation de cette même instance sous forme de problème de PLNE, les couleurs des contraintes correspondant respectivement à

celles des ellipses de la figure 3.4 :

$$\left[\begin{array}{llllllll}
\max & x_1 & +x_2 & +x_3 & +x_4 & +x_5 & +x_6 & +x_7 & \\
\text{s.c.} & x_1 & +x_2 & +x_3 & & & & & \leq 1 \\
& & x_2 & & +x_4 & +x_5 & & & \leq 1 \\
& & & & x_4 & & +x_6 & & \leq 1 \\
& x_1 & & +x_3 & & & & +x_7 & \leq 1 \\
& & x_2 & & & & & +x_7 & \leq 1 \\
& x_1 & , x_2 & , x_3 & , x_4 & , x_5 & , x_6 & , x_7 & \in \{0,1\}
\end{array}\right]. \tag{3.9}$$

Une observation rapide de la figure 3.4 permet de conclure qu'un packing peut ici contenir au maximum 3 éléments, ce qui est donc la valeur optimale de la fonction-objectif de la formulation 3.9. Les ensembles d'éléments $\{1, 5, 6\}$, $\{3, 5, 6\}$ et $\{5, 6, 7\}$ sont des exemples de packings optimaux.

Le SPP est donc un problème dont la compréhension est relativement aisée, notamment grâce à la possibilité d'en effectuer une visualisation graphique. Sa formulation sous forme de problème de PLNE est également simple, puisque toutes les contraintes du problème s'expriment sous la même forme. La matrice des contraintes est en outre exclusivement composée de coefficients binaires : le coefficient aux coordonnées (i, j) dans celle-ci vaut 1 (respectivement 0) si et seulement si $j \in G_i$ (respectivement $j \notin G_i$). La résolution du SPP pose cependant en général d'importantes difficultés combinatoires, impliquant la nécessité de s'intéresser plus particulièrement à ce type de matrice. Il a été de ce fait l'objet d'études théoriques cherchant à y trouver des structures particulières pouvant conférer des propriétés avantageuses à un problème de SPP.

3.2.2 Structure des matrices de Set Packing

3.2.2.1 Études et exploitations des propriétés structurelles

Un certain nombre d'études ont été réalisées sur les propriétés qu'il est possible de trouver au sein des matrices à coefficients binaires dans les problèmes de PLNE. Certaines propriétés peuvent être exploitées pour faciliter la résolution du problème d'optimisation associé. Les deux applications principales de ces études sont :

- une amélioration de la qualité de la borne issue de la relaxation continue du problème ;
- l'identification de sous-structures de la matrice des contraintes étant la cause de la présence de valeurs fractionnaires dans la solution optimale de la relaxation continue.

Les études théoriques liées à la formulation SPP sont relativement anciennes. Berge [5] s'intéresse à la caractérisation des matrices à coefficients binaires permettant d'affirmer l'existence d'une solution optimale entière – c'est-à-dire au sein de laquelle toutes les variables prennent une valeur entière – dans une formulation *a priori* continue. Il utilise pour ce faire la notion de *matrice équilibrée* [3]. Une matrice est dite équilibrée si elle ne possède pas de sous-matrice carrée de taille impaire dont la somme de chaque ligne et chaque colonne vaut 2. Si une matrice ne contient pas de telle sous-matrice, alors le programme linéaire associé possède une solution optimale entière. S'il est intéressant de noter cette caractérisation,

3. Traduction de l'anglais *balanced matrix*.

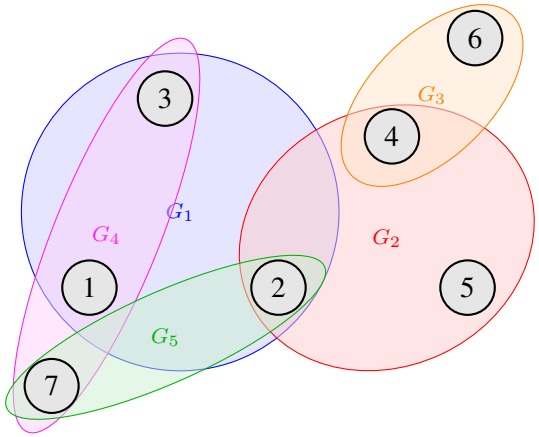

Figure 3.4 – Représentation graphique des contraintes d'un SPP.

il est toutefois en pratique peu réaliste de faire l'hypothèse que la matrice d'un problème quelconque vérifie naturellement cette propriété.

Padberg [59] s'intéresse plus particulièrement au SPP, et identifie des propriétés moins contraignantes permettant l'obtention d'une solution optimale entière par une résolution de la relaxation continue. Il introduit pour cela les matrices dites parfaites et montre que les matrices équilibrées en sont un cas particulier plus contraint. De façon similaire à Berge [5], il caractérise les sous-matrices ne devant pas être contenues dans une matrice afin que cette dernière soit parfaite.

S'appuyant sur le travail de Berge [5] et Padberg [59], Ryan et Foster [65] proposent une méthode de résolution pour un problème d'ordonnancement de bus. En se basant sur des observations particulières au problème traité, les auteurs réduisent la combinatoire du problème en ne prenant en compte qu'un sous-ensemble des variables dont le choix permet d'obtenir une matrice équilibrée. Ce problème restreint a l'avantage de pouvoir être résolu à l'optimalité par un algorithme de PL peu coûteux, tout en fournissant une solution optimale entière. Les contraintes sur la matrice garantissant l'intégralité de la solution sont par la suite relâchées, ce qui permet la considération de nouvelles variables, la formulation résultante étant résolue par un algorithme de séparation et évaluation. Les auteurs bénéficient ici de propriétés intrinsèques au problème traité, leur permettant l'obtention d'une solution de bonne qualité lorsque la formulation est restreinte aux variables permettant de conserver une matrice équilibrée.

3.2.2.2 Problème équivalent de Node Packing et graphe associé

Padberg [58] exploite la théorie des graphes pour caractériser certaines faces de l'enveloppe convexe du SPP, correspondant donc à des inégalités valides pour ce problème. Il utilise pour cela une forme particulière de SPP au sein de laquelle chaque contrainte ne lie que deux variables, appelée *Node Packing Problem* (NPP) et possédant un graphe associé. À toute formulation SPP correspond un graphe NPP, dont l'exploitation peut permettre d'identifier des inégalités valides et ainsi améliorer la qualité de relaxation continue. Ceci peut être illustré en reprenant l'exemple de SPP formulé par l'expression 3.9. La formulation NPP équivalente utilise les mêmes variables et objectif mais exprime les disjonctions en se

limitant à des paires d'éléments. Une telle transformation aboutit à la formulation 3.10 :

$$\left[\begin{array}{llllllll}
\max & x_1 & +x_2 & +x_3 & +x_4 & +x_5 & +x_6 & +x_7 & \\
\text{s.c.} & x_1 & +x_2 & & & & & & \leq 1 \\
& x_1 & & +x_3 & & & & & \leq 1 \\
& & x_2 & +x_3 & & & & & \leq 1 \\
& & x_2 & & +x_4 & & & & \leq 1 \\
& & x_2 & & & +x_5 & & & \leq 1 \\
& & & & x_4 & +x_5 & & & \leq 1 \\
& & & & x_4 & & +x_6 & & \leq 1 \\
& x_1 & & +x_3 & & & & & \leq 1 \\
& x_1 & & & & & & +x_7 & \leq 1 \\
& & & x_3 & & & & +x_7 & \leq 1 \\
& & x_2 & & & & & +x_7 & \leq 1 \\
& x_1 & ,x_2 & ,x_3 & ,x_4 & ,x_5 & ,x_6 & ,x_7 & \in \{0,1\}
\end{array}\right]. \tag{3.10}$$

Les couleurs montrent l'équivalence des disjonctions par paire avec les contraintes de la formulation 3.9. La figure 3.5 montre le graphe NPP associé à cette nouvelle formulation. Il possède un nœud par élément et chaque contrainte disjonctive est modélisée par une arête. Du point de vue de la résolution du problème de PLNE, la formalisation NPP ne possède *a priori* que des inconvénients par rapport à la formulation SPP :

- 11 contraintes sont nécessaires à la formulation NPP pour exprimer un problème formulé auparavant avec 5 contraintes, augmentant de façon significative la taille de la formulation ;
- la qualité de relaxation continue est dégradée (on peut vérifier que la borne supérieure calculée par résolution de la relaxation continue vaut 3 pour la formulation SPP, et $3,5$ pour la formulation NPP).

En revanche, le modèle de graphe qui y est associé peut se révéler utile pour améliorer la relaxation continue du SPP. Une approche classique, présentée par Padberg [58], consiste à rechercher dans ce graphe les ensembles de sommets tous reliés deux-à-deux par une arête, que l'on appelle *cliques*. Une clique à laquelle il n'est plus possible d'ajouter un sommet supplémentaire au vu des arêtes du graphe est appelée *clique maximale*. Par exemple, les cliques maximales du graphe 3.5 sont $\mathcal{C}_1 = \{1,2,3,7\}$, $\mathcal{C}_2 = \{2,4,5\}$ et $\mathcal{C}_3 = \{4,6\}$. Ainsi, les contraintes des formulations 3.9 et 3.10 peuvent être avantageusement et intégralement remplacées par les contraintes correspondant respectivement à $\mathcal{C}_1$, $\mathcal{C}_2$ et $\mathcal{C}_3$. Il en résulte une formulation plus petite (le nombre de lignes de la matrice des contraintes est finalement réduit à 3) et de meilleure qualité de relaxation continue, comme mis en valeur par Padberg [58].

Plus généralement, il est pertinent de privilégier les formulations SPP dont les contraintes correspondent à des cliques les plus grandes possible dans le graphe NPP associé. Ce constat est couramment pris en considération dans les méthodes de résolution des formulations SPP. En effet, il est en pratique peu aisé de garantir des propriétés strictes comme définies par Berge [5] et Padberg [59], alors que l'ap-

proche par la théorie des graphes fournit un moyen d'améliorer la qualité de relaxation continue de façon progressive. Cependant, si, comme nous l'avons illustré, exprimer la formulation NPP à partir d'une formulation SPP est immédiat, la recherche de cliques maximales dans un graphe peut être un problème algorithmiquement coûteux. Ainsi, la réalisation de choix judicieux de modélisation du problème réel et la conception d'algorithmes efficaces recherchant des cliques dans le graphe d'un SPP sont des techniques allant couramment de paire lors de la résolution de telles formulations.

3.2.3 Modélisation du problème de saturation

Les objets génériques que sont les éléments j et les ensembles G_i dans le SPP doivent désormais être associés aux données ferroviaires afin de modéliser le problème de saturation. Nous formalisons donc dans un premier temps l'ensemble des données ferroviaires. Il existe par la suite dans la littérature deux approches principales menant à une formulation SPP pour la saturation. Nous donnons pour chacune un aperçu de ses propriétés et de la façon dont celles-ci ont été exploitées pour la mise en œuvre d'algorithmes de résolution par leurs auteurs respectifs. Comme nous venons de le voir, le choix de la modélisation mathématique ne peut être décorrélé de la façon de résoudre le problème, et il est donc important de replacer ce choix dans la perspective de la résolution.

3.2.3.1 Formalisation des données

Nous avons vu qu'une infrastructure est composée de zones de détection, chacune de ces zones ne pouvant être occupée que par un seul train à chaque instant. Il s'agit donc de ressources unaires permettant le respect des espacements de sécurité. Nous notons leur ensemble Z.

Outre la discrétisation de l'espace assuré par la présence des zones de détection, nous considérons également une discrétisation temporelle en *périodes*, groupées dans un ensemble noté P et numérotées de 1 à $|P|$. Cet ensemble couvre la totalité de l'intervalle temporel considéré par l'instance. La finesse de discrétisation temporelle doit être adaptée aux instances du problème de saturation traitées. Typiquement, une période peut correspondre à une durée d'une à plusieurs secondes, selon la précision avec laquelle la marche des trains dans l'infrastructure est décrite par l'instance considérée.

La demande de circulation consiste en un ensemble de trains noté T. Il regroupe tous les trains de la demande quel que soit leur spécificité ou leur type (fret, TGV, intercité, etc.). Nous considérons que chaque train $t \in T$ possède une date nominale d'entrée sur l'infrastructure, représentée par une période $p_t^e \in P$. En outre, nous souhaitons donner une marge de tolérance quant à la date d'entrée du train en lui autorisant un décalage maximal $\overline{d_t} \in \mathbb{N}$, représentant le nombre maximal de périodes au-delà de p_t^e pendant lesquelles le train est autorisé à entrer sur l'infrastructure. Le train t est donc autorisé à rentrer dans l'infrastructure à n'importe quelle période $p \in P$ telle que $p_t^e \leq p \leq p_t^e + \overline{d_t}$. La restriction de $\overline{d_t}$ à une valeur positive est choisie par soucis de simplicité, une valeur négative étant simplement équivalente à l'affectation d'une valeur plus petite à p_t^e.

La terminologie utilisée pour décrire la course des trains à travers une infrastructure varie sensiblement dans la littérature. Les termes « itinéraire », « route » et « chemin » sont systématiquement privilégiés mais sont associés à des définitions légèrement variables. De façon cohérente avec Lusby et al. [46, 47, 48], nous utilisons les définitions suivantes :

– un *itinéraire* désigne la description purement spatiale du parcours d'un train dans l'infrastructure, il peut donc être simplement modélisé par une séquence ordonnée de zones ;

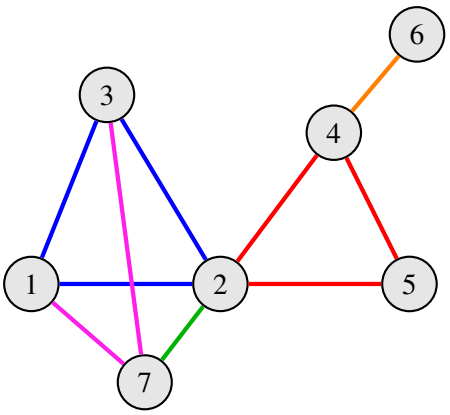

Figure 3.5 – Représentation graphique des contraintes d'un NPP.

– un *chemin* est un itinéraire accompagné de l'information temporelle permettant de décrire complètement la marche d'un train à travers l'infrastructure, c'est-à-dire les périodes de début et de fin d'occupation de chacune des zones de l'itinéraire ;
– le *routage* d'un train consiste en l'affectation d'un chemin à un train au sein de l'infrastructure ;
– un routage peut plus généralement désigner l'affectation d'un chemin à un ensemble de trains.

Bien entendu, la formulation du problème n'aura d'intérêt que si les chemins pris en compte obéissent aux règles imposées par le système de signalisation. Nous supposons que les données de l'instance contiennent les informations nécessaires à la connaissance de tels itinéraires et chemins.

Un nœud ferroviaire pouvant se situer à l'intersection de plusieurs voies, il peut être possible d'y pénétrer comme d'en sortir par plusieurs points, un point d'entrée/sortie étant relié à une voie du réseau au sein duquel le nœud est présent. Il est en outre possible qu'il existe plusieurs itinéraires permettant d'aller d'un point d'entrée à un point de sortie donnés. Ainsi, l'ensemble des itinéraires pour un train $t \in T$ donné est noté R_t. Par la suite, l'ensemble des zones d'un itinéraire $r \in R_t$ est noté Z_r. Enfin, l'ensemble des chemins disponibles pour t est noté C_t, plusieurs chemins différents pouvant exister pour un même itinéraire du fait des variations possibles de la vitesse et/ou de la période d'entrée du train dans l'infrastructure.

Finalement, il est dit qu'il existe un *conflit* entre deux chemins si ceux-ci impliquent l'occupation d'au moins une zone au même moment. Il n'est donc pas possible de router deux trains distincts sur des chemins en conflit. Formellement, le problème de saturation consiste donc à trouver un routage maximisant le nombre de trains distincts routés sur des chemins donc aucune paire ne crée de conflit.

Par la suite, par soucis de cohérence et pour en permettre une meilleure comparaison, les modélisations existant dans la littérature sont adaptées aux notations données ici.

3.2.3.2 Modèle Node Packing pour le problème de faisabilité

L'utilisation du SPP dans le cadre d'études de capacité d'infrastructures ferroviaires a été introduite par Zwaneveld et al. [78, 79] et Kroon et al. [37] dans le cadre du problème de faisabilité pour le projet néerlandais STATIONS. Plus précisément, la modélisation initialement proposée par Zwaneveld et al. [78, 79] est essentiellement basée sur une formulation NPP, que nous présentons ici.

Zwaneveld et al. [78, 79] modélisent directement la notion de conflit qui peut exister entre deux chemins. La détection des conflits est réalisée avant la formulation de l'instance sous forme de NPP, de façon à pouvoir générer les contraintes de packing représentant ces conflits. La présence de conflits est, comme nous l'avons défini plus haut, liée au fait que des chemins distincts peuvent occuper certaines zones au même moment. Formellement, les auteurs considèrent, pour chaque couple de train $(t_1, t_2) \in T \times T$, l'ensemble $F_{t_1 t_2}$ des couples de chemins $(c_1, c_2) \in C_{t_1} \times C_{t_2}$ tels que c_1 et c_2 soient compatibles, c'est-à-dire qu'il n'existe pas de conflit entre eux.

Une variable binaire est ensuite définie par l'expression 3.11 pour chaque $t \in T$ et $c \in C_t$:

$$x_{t,c} = \begin{cases} 1 & \text{si le train } t \text{ emprunte le chemin } c\,; \\ 0 & \text{sinon.} \end{cases} \tag{3.11}$$

Une affectation de l'ensemble des variables peut donc représenter l'affectation d'un chemin à travers l'infrastructure à certains trains. Pour que cette affectation corresponde à un routage effectivement réalisable, il est nécessaire de définir les contraintes excluant les solutions ne respectant pas les conditions requises. En premier lieu, chaque train $t \in T$ ne doit pouvoir être routé qu'une seule fois sur l'ensemble des chemins qui lui sont disponibles. Ceci est exprimé par l'ensemble de contraintes 3.12, que nous

appelons *contraintes d'unicité* :

$$x_{t,c} + x_{t,c'} \leq 1 \quad \forall t \in T, \forall (c, c') \in C_t^2, c \neq c'. \tag{3.12}$$

Deuxièmement, les conflits entre les chemins associés à des trains distincts doivent également être modélisés, de façon à éviter la sélection de chemins conflictuels dans la solution. C'est la connaissance de l'ensemble des couples de chemins compatibles qui permet cette modélisation, aboutissant à l'ensemble de contraintes 3.13, que nous nommons *contraintes de conflits* :

$$\begin{aligned} x_{t_1,c_1} + x_{t_2,c_2} \leq 1 \quad & \forall (t_1, t_2) \in T^2, t_1 \neq t_2, \\ & \forall (c_1, c_2) \in C_{t_1} \times C_{t_2}, (c_1, c_2) \notin F_{t_1 t_2}. \end{aligned} \tag{3.13}$$

Finalement, Zwaneveld et al. [78, 79] donnent la possibilité d'associer un poids arbitraire $\rho_{t,c}$ à chacune des variables $x_{t,c}$, permettant si nécessaire de favoriser ou de désavantager certains chemins. La fonction-objectif du problème correspond dès lors à la fonction-objectif standard d'une formulation SPP, et est donnée par la formulation 3.14 :

$$\max \sum_{t \in T, c \in C_t} \rho_{t,c} x_{t,c}. \tag{3.14}$$

La formulation NPP complète est ainsi donnée par l'expression 3.15 :

$$\left[\begin{array}{lll} \max \displaystyle\sum_{t \in T, c \in C_t} \rho_{t,c} x_{t,c} & & \\ x_{t,c} + x_{t,c'} & \leq 1 & \forall t \in T, \forall (c, c') \in C_t^2, c \neq c' \\ x_{t_1,c_1} + x_{t_2,c_2} & \leq 1 & \forall (t_1, t_2) \in T^2, t_1 \neq t_2, \\ & & \forall (c_1, c_2) \in C_{t_1} \times C_{t_2}, (c_1, c_2) \notin F_{t_1 t_2} \\ x_{t,c} \in \{0, 1\} & & \forall t \in T, \forall c \in C_t \end{array} \right]. \tag{3.15}$$

Il s'agit bien d'une formulation exclusivement NPP, dont nous avons vu précédemment qu'elle n'est *a priori* pas avantageuse du point de vue de la qualité de relaxation continue, cette dernière étant un outil important pour la résolution entière.

Les auteurs effectuent en premier lieu une série de pré-traitements portant à la fois sur les données et sur le graphe NPP afin de réduire le nombre de variables du problème. Par la suite, la formulation est résolue à l'optimalité par un algorithme de séparation et évaluation, au sein duquel une recherche d'inégalités valides est effectuée [4]. Les auteurs exploitent les données spécifiques au problème ferroviaire pour générer plus rapidement ces inégalités valides. Trois approches sont utilisées en ce sens :

– sans coût algorithmique significatif, l'ensemble des contraintes d'unicité 3.12 peut être remplacé par des contraintes où exactement une clique par train est générée, puisque tous les chemins distincts d'un même train sont deux-à-deux incompatibles :

$$\sum_{c \in C_t} x_{t,c} \leq 1 \quad \forall t \in T \quad ; \tag{3.16}$$

4. Cette combinaison de méthodes porte le nom anglais de *branch and cut*.

– considérant chaque clique générée pour chaque train $t \in T$, l'étape suivante consiste à y rechercher un sous-ensemble de chemins étant tous incompatibles avec le chemin $c' \in C_{t'}$ d'un autre train $t' \neq t$, ce qui peut être formulé par l'expression 3.17 :

$$x_{t',c'} + \sum_{c \notin F_t^{t'c'}} x_{t,c} \leq 1 \quad \forall (t,t') \in T^2, t \neq t', \forall c' \in C_{t'} \tag{3.17}$$

où $F_t^{t'c'}$ est l'ensemble des chemins de t compatibles avec c' ;
– cette seconde approche est ensuite généralisée en considérant plusieurs chemins pour t' plutôt que le seul $c' \in C_{t'}$, et l'union des ensembles de chemins de t compatibles avec ces chemins de t' à la place du seul $F_t^{t'c'}$.

Cette approche itérative permet aux auteurs de trouver des cliques dont ils font augmenter la taille progressivement. Toutefois, plus la taille des cliques augmente, plus il est difficile algorithmiquement de les trouver en temps raisonnable, c'est pourquoi une limite de temps est imposée pour la recherche de telles inégalités valides. Associé à cette génération de coupes, l'algorithme utilise des heuristiques visant à calculer des solutions entières, et donc des bornes inférieures, de bonne qualité.

Cette contribution, hormis son apport dans le domaine ferroviaire, met en valeur qu'en pratique il peut être difficile d'appliquer de façon efficace la recherche de cliques telles qu'étudiées théoriquement par Padberg [58], de la même manière qu'il n'est pas évident de s'assurer de la présence de certaines propriétés d'intégralité au sein de la matrice telles que mises en valeur par Berge [5] et Padberg [59]. À travers la stratégie employée par Zwaneveld et al. [78, 79] pour la génération d'inégalités valides, on remarque par contre la pertinence de l'utilisation de données spécifiques au problème considéré pour améliorer les performances de la résolution.

3.2.3.3 Utilisation du Node Packing pour la saturation

Delorme [20] s'inspire de cette formulation pour traiter le problème de saturation par une formulation SPP et expérimente en premier lieu une approche de résolution comparable : celle-ci est composée de pré-traitements incluant également la génération de cliques et la suppression de certaines variables et contraintes. La résolution, confiée au logiciel commercial Cplex, montre un temps de résolution de plus de 50 000 secondes sur certaines instances compte tenu de la combinatoire importante. Nous avons en effet vu que si les problèmes de saturation et de faisabilité pouvaient être modélisés par des formulations SPP identiques, les degrés de liberté inhérents au problème de saturation impliquent une combinatoire nettement plus importante.

Afin de pallier à ce verrou, Delorme [20, 21] propose la métaheuristique GRASP [5] pour le SPP, dont le choix est justifié par de bonnes performances constatées sur des problèmes de PLNE similaires. Son fonctionnement peut être succinctement résumé de telle sorte :

1. recherche d'une solution réalisable combinant une heuristique de recherche gloutonne et une recherche aléatoire, dont l'importance de l'une par rapport à l'autre est pondérable par un paramètre $\alpha \in [0,1]$;
2. amélioration cette solution par une heuristique de recherche locale ;
3. réitération du processus jusqu'à ce qu'un critère d'arrêt ait été atteint (Delorme [20] utilise une limite du nombre d'itérations de GRASP égale à 200).

5. Acronyme anglais pour *Greedy Randomized Adaptive Search Procedure*.

Deux améliorations sont en outre utilisées, à savoir un mécanisme de sélection automatique de la valeur de α appelé « reactive GRASP », et une procédure dite de « path relinking » recherchant de nouvelles solutions par combinaison d'autres solution.

Une autre métaheuristique, ACO [6], est également proposée pour résoudre le SPP modélisant le problème de saturation. Son principe est initialement suggéré par Colorni et al. [14] dans le cadre de la résolution de problèmes de voyageur de commerce, en s'inspirant de la façon dont les fourmis trouvent le chemin le plus court entre deux points grâce à l'intelligence collective induit par le dépôt de phéromones. L'adaptation d'ACO au SPP est proposée par Gandibleux et al. [30, 31, 32] et améliore les résultats fournis par GRASP.

Ces constats montrent l'intérêt des métaheuristiques pour résoudre de façon approchée des problèmes de PLNE lorsque les algorithmes exacts buttent sur une combinatoire trop importante. Bien qu'elle ne soit pas conçue pour garantir l'optimalité, la bonne qualité d'une métaheuristique peut être empiriquement montrée par les constats suivants :

- l'écart entre la valeur de l'objectif des meilleures solutions connues et celle des solutions calculées par la métaheuristique est faible, permettant de s'assurer de la capacité de l'algorithme à trouver des solutions de bonne qualité ;
- l'algorithme possède une bonne régularité dans la qualité des solutions trouvées sur un large ensemble de solutions, permettant de le considérer comme fiable.

Ces arguments sont notamment avancés pour GRASP et ACO par leurs contributeurs respectifs, cette dernière étant donc l'algorithme donnant les résultats les plus prometteurs pour la résolution de larges formulations de SPP. Nous en faisons une présentation plus détaillée au chapitre suivant, celle-ci étant partie intégrante de l'algorithme de résolution que nous proposons.

3.2.3.4 Disjonction de ressources par formulation Set Packing

Dans le cadre de l'étude d'un nœud ferroviaire allemand, Velasquez et al. [77] et Lusby et al. [46, 47] proposent une autre formulation SPP pour modéliser puis résoudre le problème de faisabilité. Cette formulation possède des propriétés intéressantes dans la perspective de sa résolution. La formulation que nous utilisons pour modéliser et résoudre le problème de saturation s'en inspire fortement. Nous introduisons donc ici la formulation que nous utilisons et présentons ses avantages principaux à travers les arguments avancés par Velasquez et al. [77] et Lusby et al. [46, 47].

Considérons tout d'abord l'ensemble de l'espace-temps $U = Z \times P$, contenant donc l'ensemble des couples formés d'une zone et d'une période. Chaque zone peut être occupée par un unique train au maximum à toute période, tout couple $(s, p) \in Z \times P$ est donc une ressource pouvant être utilisée par un seul train au maximum. L'ensemble U est donc un ensemble de ressources spatio-temporelles unaires.

Par la suite, nous avons vu que la spécification de chaque chemin inclut le temps de début et de fin d'occupation de chaque zone de l'itinéraire correspondant. Cette information donne l'ensemble des ressources de U utilisées par chaque chemin. Pour tout $t \in T$, $c \in C_t$ et $u \in U$, nous définissons le paramètre $\delta_{c,u}$ de façon définie par l'expression 3.18

$$\delta_{c,u} = \begin{cases} 1 & \text{si } c \text{ utilise la ressources } u\,; \\ 0 & \text{sinon.} \end{cases} \tag{3.18}$$

Chaque ressource $u \in U$ étant unaire, tous les chemins utilisant cette même ressource sont deux-à-deux incompatibles. Ils peuvent donc tous figurer au sein d'une même contrainte de packing. L'ensemble de

6. Acronyme anglais pour *Ant Colony Optimization*, ou optimisation par colonies de fourmis.

ces contraintes est donné par l'expression 3.19 :

$$\sum_{t \in T, c \in C_t} \delta_{c,u} x_{t,c} \leq 1 \quad \forall u \in U. \tag{3.19}$$

Cet ensemble modélise la totalité des conflits entre les chemins. De ce fait, si toutes les incompatibilités entre les chemins distincts sont issus de conflits d'occupation des zones, il est possible de remplacer l'ensemble des contraintes de conflits de la formulation 3.15 par ce nouvel ensemble de contraintes, que nous nommons *contraintes de ressource*. De plus, les cliques formulées par l'ensemble de contraintes 3.16 étant déduites sans calcul, il est judicieux de les utiliser directement dans la formulation. Cela nous permet de construire la formulation SPP que nous utilisons pour la résolution du problème de saturation, donnée par l'expression 3.20 :

$$\left[\begin{array}{lll} \max & \displaystyle\sum_{t \in T, c \in C_t} \rho_{t,c} x_{t,c} & \\ & \displaystyle\sum_{c \in C_t} x_{t,c} \leq 1 & \forall t \in T \\ & \displaystyle\sum_{t \in T, c \in C_t} \delta_{c,u} x_{t,c} \leq 1 & \forall u \in U \\ & x_{t,c} \in \{0,1\} & \forall t \in T, \forall c \in C_t \end{array}\right]. \tag{3.20}$$

Un exemple de matrice des contraintes est donné par le tableau 3.1, les éléments prenant la valeur 0 étant représentés par des cases vides. Les chemins de trois trains t_1, t_2 et t_3 y sont illustrés, le premier possédant quatre chemins possibles et les deux seconds possédant chacun trois chemins. On y retrouve les deux ensembles distincts de contraintes : d'une part les contraintes d'unicité, chacune contenant exactement l'ensemble des chemins associés à un unique train, et d'autre part les contraintes de ressource, chacune contenant les chemins occupant la ressource associée à cette contrainte. Il est également intéressant d'effectuer une lecture en colonne de la matrice : hormis l'unique contrainte d'unicité qu'elle occupe, la position des coefficients dans une colonne correspond à la description exacte d'un chemin, en listant les ressources qu'il occupe. Par exemple, le premier chemin du train t_1 utilise la zone s_1 aux périodes p_1 et p_2, puis la zone s_2 aux périodes p_2 et p_3, et ainsi de suite.

Cette formulation permet d'éviter le calcul de l'ensemble des conflits deux-à-deux lors de l'insertion d'un nouveau chemin : ce dernier détermine la place des coefficients de la nouvelle colonne dans la matrice des contraintes, modélisant les conflits directement. Cela signifie également que le nombre de contraintes ne dépend pas du nombre de chemins de chaque train. Par opposition, la modélisation proposée par Zwaneveld et al. [78, 79] nécessite un recalcul des conflits à chaque insertion d'un nouveau chemin, ce qui est consommateur de temps de calcul et fait augmenter le nombre de contraintes.

La taille de la formulation dépend en revanche fortement de la taille de U, c'est-à-dire du nombre de zones sur l'infrastructure ainsi que du nombre de périodes temporelles considérées. En particulier, une finesse élevée de la discrétisation de l'horizon temporel en périodes peut occasionner une forte augmentation du nombre de contraintes.

Enfin, un argument majeur de Velasquez et al. [77] et Lusby et al. [46, 47] est la qualité de relaxation continue de cette formulation. Les contraintes de ressource induisent en effet la présence de cliques regroupant tout ensemble de chemins utilisant la même ressource, sans nécessiter de pré-traitement. Cela évite donc l'utilisation de temps de calcul pour l'amélioration de la relaxation continue de la formulation.

La méthode de résolution choisie par Velasquez et al. [77] et Lusby et al. [46, 47] est principalement basée sur un algorithme de génération de colonnes, procédure de résolution exacte de problèmes de PL dont l'utilisation peut s'avérer pertinente lorsqu'il n'est pas possible d'énumérer entièrement toutes les variables d'une formulation ou lorsque le problème contient un très grand nombre de variables. Nous détaillons son principe de fonctionnement au chapitre suivant, une telle procédure étant également partie intégrante de la solution algorithmique que nous proposons.

Du fait de ses avantages, nous privilégions cette formulation. Il est finalement nécessaire de préciser les données à notre disposition et les instances ferroviaires qui en découlent, ainsi que leur structure une fois formulées par un SPP. Ces caractéristiques ont en effet un impact sur l'approche à mettre en œuvre pour la résolution.

3.3 Présentation des instances

3.3.1 Infrastructure

Toutes les instances à notre disposition concernent la jonction ferroviaire de Pierrefitte-Gonesse, située au nord de Paris, et qui s'étend sur une envergure d'environ 10 km. Ce nœud, également sujet d'étude par Delorme [20], possède un rôle clé dans la circulation du trafic entre Paris et le nord de la France, puisqu'il relie :

- du côté parisien, la gare du Nord et la Grande ceinture [7] ;
- de l'autre côté, d'une part la LGV Nord en direction de Lille et Bruxelles et d'autre part la ligne classique permettant la desserte de la proche province, comme Chantilly puis Soissons, Laon, Amiens, etc.

La jonction accueille de ce fait un trafic à la fois hétérogène et dense, composé de :

- plusieurs types de TGV circulant entre la gare du Nord et la LGV Nord, à savoir des TGV de la SNCF vers Lille, des Eurostar vers Londres et des Thalys vers Bruxelles, Cologne et Amsterdam ;
- trains de voyageurs classiques de type « intercité » (trains régionaux, trains Corail) entre la gare du Nord et diverses destinations sur la ligne de Chantilly ;
- trafic de fret, entre la Grande ceinture et la ligne de Chantilly.

La circulation est en outre complexifiée par la présence d'un nombre important de voies banalisées dans le nœud, qui impliquent qu'un train circulant dans un sens peut empêcher la circulation d'un autre circulant dans l'autre sens. L'ensemble de ces éléments font de ce nœud un point de passage critique du réseau et un cas d'étude particulièrement intéressant.

La jonction est composée d'un total de 87 zones de détection. Sa topologie est schématisée par la figure 3.6, le sens de circulation des voies étant indiqué par les flèches qui y figurent, permettant notamment de visualiser le nombre important de voies banalisées. Les figures 3.7 et 3.8 schématisent respectivement un itinéraire de train de marchandise et de TGV. Elles permettent de visualiser clairement qu'il y a des zones communes utilisées par des itinéraires distincts, impliquant des incompatibilités entre la circulation des trains de différents types.

Finalement, les données de l'infrastructure spécifient un nombre d'itinéraires fixés alloué à chacun des types de trafic, selon le sens. Le tableau 3.2 donne le nombre d'itinéraires disponibles selon type et le sens du trafic. Ces itinéraires sont ensuite déclinés en chemins selon les retards autorisés pour chaque train.

7. Ligne formant une boucle à l'extérieur du boulevard périphérique de Paris, en grande partie dédiée au fret mais permettant également le passage de RER et TGV sur certains segments.

		t_1				t_2			t_3		
Unicité		1	1	1	1						
						1	1	1			
									1	1	1
Ressources	(s_1, p_1)	1		1							
	(s_1, p_2)	1	1			1			1		
	(s_1, p_3)		1			1	1			1	
	$\vdots$				1		1				1
	(s_2, p_1)				1	1					
	(s_2, p_2)	1			1	1	1	1			
	(s_2, p_3)	1	1				1	1	1		
	$\vdots$		1	1				1		1	

Table 3.1 – Structure d'une matrice de SPP représentant une instance du problème de saturation.

Type de trafic	Sens	#itinéraires
TGV	Paris → LGV Nord	6
	LGV Nord → Paris	5
Intercité	Paris → Chantilly	8
	Chantilly → Paris	5
Fret	Grande Ceinture → Chantilly	4
	Chantilly → Grande Ceinture	3

Table 3.2 – Itinéraires disponibles dans la jonction de Pierrefitte-Gonesse.

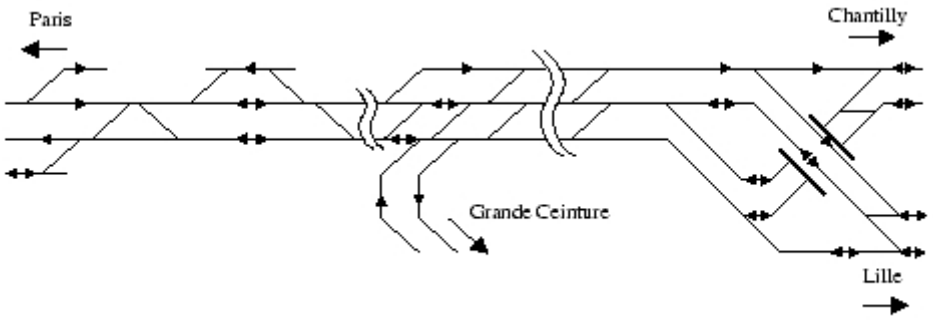

Figure 3.6 – Schéma du nœud de Pierrefitte-Gonesse.

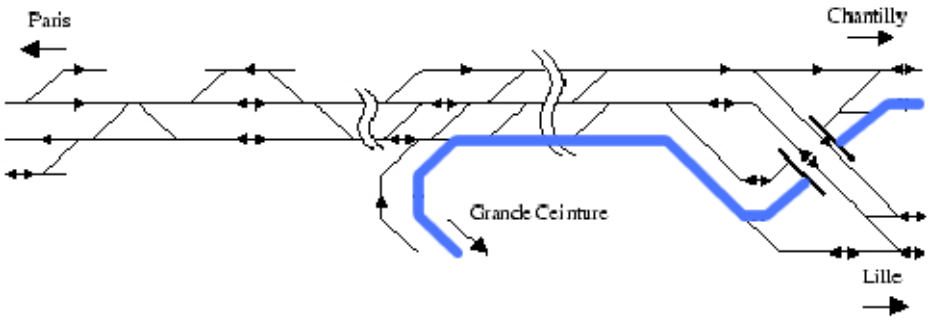

Figure 3.7 – Exemple d'itinéraire d'un train de marchandises.

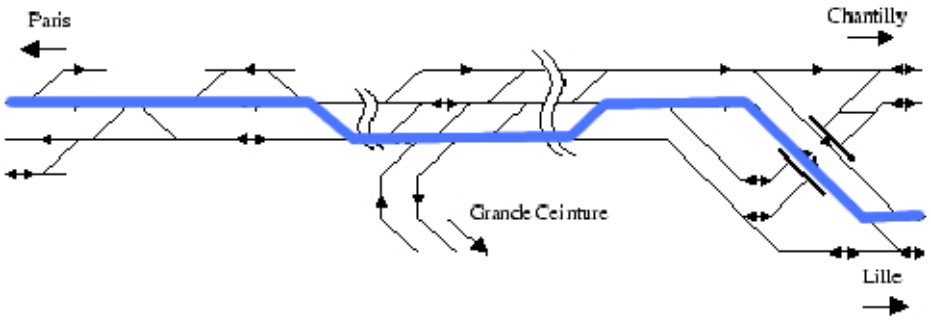

Figure 3.8 – Exemple d'itinéraire d'un TGV.

3.3.2 Variations de la demande de trafic

La demande de trafic est composée d'un ensemble de trains de plusieurs types, à savoir des TGV, des trains intercité, et du fret. Ainsi, faire varier le nombre de trains de la demande permet de créer des instances variables du problème de saturation. En outre, le type d'un train ne constitue en lui-même pas une information pertinente dans notre cas, mais c'est plutôt l'implication du type du train sur les itinéraires à la disposition de celui-ci qui est pertinent. Plus précisément, le type et la direction de chaque train détermine ses itinéraires possibles dans l'infrastructure, dont le nombre est décrit par le tableau 3.2. Chacun de ces itinéraires est décrit par les données en termes de séquence de zones utilisées.

Dans le cadre des données dont nous disposons, il est considéré qu'un itinéraire ne peut être emprunté qu'à la vitesse maximale autorisée tout au long de celui-ci : il s'agit d'un parcours en voie libre. En d'autres termes, aucune variation de vitesse possible pour un itinéraire donné n'est spécifiée. La variation temporelle menant à la création de plusieurs chemins par itinéraire ne repose donc que sur le décalage qu'un train est autorisé à prendre par rapport à sa date d'entrée nominale. À chaque itinéraire correspond donc une unique date de début et de fin d'occupation pour chacune de ses zones, relativement à la date d'entrée sur l'infrastructure d'un train utilisant cet itinéraire. En outre, les données ne spécifient pas de priorité de certains trains ou chemins par rapport à d'autres.

La spécification temporelle de l'occupation des zones de chaque itinéraire est précise à la seconde près. En conséquence, les données nécessitent une discrétisation de l'horizon temporel en périodes d'une seconde. La longueur totale de l'horizon temporel d'une instance est définie par la différence de temps entre la sortie la plus tardive qu'un train est susceptible d'effectuer (en considérant le retard maximal que celui-ci est autorisé à avoir), et la première entrée susceptible d'être effectuée.

Finalement, outre le retard maximal qu'un train peut être autorisé à prendre, il est possible de faire varier la *granularité du retard*, c'est-à-dire le nombre de valeurs de retard que le train peut prendre entre sa date d'entée nominale et son retard maximal. Par exemple, pour un train autorisé à prendre un retard maximal d'une minute, il est possible d'essayer de le faire rentrer dans l'infrastructure toutes les 10 secondes (7 valeurs de retards distinctes autorisées : $\{0, 10, 20, 30, 40, 50, 60\}$), toutes les 20 secondes (4 valeurs autorisées), toutes les 30 secondes, etc. Bien entendu, la valeur de la granularité n'a de sens que si elle n'excède pas celle du retard maximal autorisé.

Pour résumer, nous faisons varier les trois paramètres suivants afin de créer différentes instances :
- la composition de l'ensemble de trains T ;
- la valeur de retard maximal octroyé aux trains d'une instance, notée $\overline{d}$;
- la granularité de retard, notée g.

Nous faisons ainsi varier ces paramètres dans les ensembles de valeurs suivants afin de créer un jeu de 45 instances :
- le nombre de trains varie entre les valeurs 53, 66 et 104, et chaque instance contient les trois types de trains ;
- la valeur de retard $\overline{d}$ varie dans l'ensemble $\{30, 60, 90\}$;
- la granularité g varie dans l'ensemble $\{1, 2, 5, 10, 15\}$.

Le tableau 3.3 donne le détail du nombre de trains par type et sens de circulation pour chacune des valeurs de $|T|$.

Notons que les instances traitées par Delorme et al. [20, 21] et Gandibleux et al. [32] concernent également Pierrefitte-Gonesse, et étudient en conséquence des schémas de trafic similaires pour le problème de saturation. Toutefois, à notre connaissance, la granularité de retard la plus fine ayant été traitée jusqu'ici est de 15 secondes. Ainsi, outre les contributions algorithmiques que nous voulons apporter pour améliorer la résolution même du problème, nous essayons d'améliorer la précision de l'estimation

		$\|T\|$		
Type de trafic	Sens	53	66	104
TGV	Paris → LGV Nord	9	11	18
	LGV Nord → Paris	12	11	23
Intercité	Paris → Chantilly	8	11	16
	Chantilly → Paris	11	13	23
Fret	Grande Ceinture → Chantilly	6	13	9
	Chantilly → Grande Ceinture	7	7	15

Table 3.3 – Composition de la demande (sens et type de trafic).

de capacité, en autorisant l'utilisation de chemins non considérés par une granularité plus élevée. De plus, le fait que nous utilisions ici des paramètres précis pour créer les instances peut permettre l'observation séparée de l'influence de chacun de ces paramètres sur le comportement de l'algorithme de résolution.

3.3.3 Propriétés numériques

Le tableau 3.4 donne finalement la taille des formulations SPP correspondant à chacune des 45 instances. Nous nous intéressons plus précisément au nombre de variables – c'est-à-dire le nombre total de chemin distincts, tous trains confondus – et au nombre de contraintes de ressource, puisque le nombre de contraintes d'unicité est égal au nombre de trains de l'instance. Nous remarquons d'ailleurs que les contraintes d'unicité sont en nombre négligeable par rapport au nombre de contraintes de ressource, et que c'est bien ce dernier qui est le facteur déterminant dans le nombre de contraintes de la formulation. Le nombre de variables varie entre 876 et 52 689, ce qui représente un large intervalle de valeurs qui nous permettra de tester le comportement de l'algorithme de résolution sur des formulations balayant des tailles étant considérées comme petites jusqu'à des tailles particulièrement grandes et donc *a priori* difficiles à résoudre. Pour comparaison, les instances liées au problème de saturation traitées par Delorme [20] et Gandibleux et al. [30] ne dépassent pas 2 500 variables. Du fait de l'absence de priorités de certains trains ou chemins envers d'autres, l'ensemble des instances SPP sont de type USPP. Enfin, nous avons des instances présentant le même nombre de variables pour des valeurs de paramètres différentes, ce qui peut nous permettre d'identifier d'éventuelles conséquences de différence de structure de la matrice SPP pour une taille de problème comparable.

Ces instances étant formulées sous forme de SPP, il est désormais possible de développer un algorithme de résolution de PLNE afin de résoudre le problème de saturation. Le chapitre suivant développe l'algorithme de résolution, se basant sur l'analyse des méthodes existantes, notamment par la réutilisation du SPP et l'étude de l'adéquation de l'algorithme de résolution proposé par Lusby et al. [46, 47, 48].

$\overline{d}$	g	$\lvert T\rvert$					
		53		66		104	
		$\lvert U\rvert$	n	$\lvert U\rvert$	n	$\lvert U\rvert$	n
	15		876		1059		1737
	10		1168		1412		2316
30	5	120 582	2044	351 828	2471	197 142	4053
	2		4672		5648		9264
	1		9052		10943		17949
	15		1460		1765		2895
	10		2044		2471		4053
60	5	123 192	3796	354 438	4589	199 752	7527
	2		9052		10943		17949
	1		17812		21533		35319
	15		2044		2471		4053
	10		2920		3530		5790
90	5	125 802	5548	357 048	6707	202 362	11001
	2		13432		16238		26634
	1		26572		32123		52689

Table 3.4 – Nombre de contraintes de ressource ($|U|$) et de colonnes (n) selon les instances.

CHAPITRE 4

Un algorithme hybride de résolution

Les chapitres précédents nous ont permis d'une part de présenter le problème de capacité d'infrastructure dans son ensemble, et plus particulièrement celui de saturation, et d'autre part de présenter le modèle PLNE choisi pour résoudre le problème de saturation. Ce choix a été réalisé en se basant sur des considérations telles que ses avantages en termes de qualité de relaxation continue ou la facilité d'y ajouter des variables sans recalcul de conflits.

Il convient désormais de développer une méthode de résolution adaptée à cette formulation, et plus particulièrement aux problèmes soulevés par les instances que nous considérons, à savoir principalement un grand nombre de variables et de contraintes. Ce chapitre présente dans un premier temps une discussion autour de méthodes existantes et semblant pertinentes à première vue. La faisabilité d'une réapplication de ces méthodes dans notre cas est étudiée, montrant les éléments pertinents qu'elles fournissent ainsi que les verrous restant à traiter.

La partie suivante de ce chapitre est donc consacrée au développement d'un algorithme de résolution adapté. Nous y combinons plusieurs techniques de résolution, issues à la fois de la famille des algorithmes de résolution exacte des problèmes de PL et de celle de la résolution approchée des problèmes de PLNE, formant ainsi un algorithme hybride. La majeure partie du développement concerne une procédure dite de génération de colonnes, qui est adaptée à la résolution de formulations contenant un nombre important de variables. La génération de colonnes est un cadre de résolution générique, qui nécessite des adaptations spécifiques au problème traité. Les adaptations que nous faisons sont détaillées et l'ensemble des considérations visant à en augmenter l'efficacité sont expliquées.

Nous proposons également une méthode d'agrégation dynamique de contraintes visant à pallier au grand nombre de contraintes présentes dans la formulation. Cette méthode consiste à éliminer les redondances entre contraintes au sein de la formulation. Tirant parti ici aussi de certaines spécificités liées au problème ferroviaire, un algorithme d'agrégation de contraintes est proposé. Cette méthode d'agrégation est intégrée à la génération de colonnes, nécessitant des adaptations qui sont également présentées.

L'algorithme interagit finalement avec la métaheuristique ACO, de façon à obtenir une solution entière au problème. Cette interaction, également détaillée, réside à la fois dans les données fournies par la génération de colonnes à ACO et dans le fait que la génération de colonnes fournit une borne supérieure pouvant permettre l'estimation de la qualité de la solution retournée par ACO.

4.1 Étude de réutilisabilité des méthodes existantes

Dans le chapitre précédent, nous avons constaté l'existence de plusieurs méthodes de résolution dédiées à des formulations SPP pour les problèmes de faisabilité et de saturation.

Nous avons tout d'abord mentionné trois méthodes développées autour de la résolution des formulations basées sur le NPP :

- un algorithme de résolution exacte basé sur une procédure par séparation et évaluation, proposé par Zwaneveld et al. [78, 79] dans le cadre du problème de faisabilité ;
- la métaheuristique GRASP, proposée par Delorme et al. [20, 21] pour résoudre le problème de saturation ;
- la métaheuristique ACO, proposée par Gandibleux et al. [30, 31, 32], également pour le problème de saturation.

En second lieu, nous avons présenté la formulation SPP que nous utilisons pour résoudre le problème de saturation, inspirée d'une formulation similaire proposée par Velasquez et al. [77] et Lusby et al. [46, 47, 48]. Les auteurs présentent également un algorithme de résolution majoritairement basée sur une procédure de génération de colonnes, dont l'utilisation peut s'avérer pertinente pour les problèmes possédant un grand nombre de variables.

D'un point de vue théorique, toutes ces méthodes sont réapplicables à la formulation SPP que nous avons choisie. En effet, les deux métaheuristiques peuvent être utilisées pour résoudre n'importe quelle formulation SPP. En outre, l'algorithme de résolution exacte proposé par Zwaneveld et al. [78, 79] utilise des données spécifiques au problème que nous utilisons également dans notre modèle. Enfin, de par la similarité des modélisations employées avec Velasquez et al. [77] et Lusby et al. [46, 47, 48], l'approche par génération de colonnes consiste *a priori* également en une piste intéressante.

D'un point de vue pratique, les résultats donnés par Delorme et al. [20, 21] montrent que la résolution du problème de saturation par une méthode exacte analogue à celle proposée par Zwaneveld et al. [78, 79] implique des temps de résolution rédhibitoires, des résultats de tests faisant état de durées dépassant les 50 000 secondes, pour des instances ne dépassant pas 2 500 variables. Étant donné la taille des instances dont nous disposons (jusqu'à plus de 50 000 variables), nous estimons que l'utilisation d'un algorithme équivalent à celui proposé par Zwaneveld et al. [78, 79] pour notre formulation SPP n'est pas un choix pertinent. De plus, des tests préliminaires d'utilisation d'algorithmes par séparation et évaluation fournis par la bibliothèque logicielle libre Coin-OR ainsi que par le logiciel commercial Cplex ont confirmé cette tendance, nous poussant à explorer d'autres pistes.

Par la suite, du fait des résultats comparatifs présentés par Gandibleux et al. [30, 31, 32] montrant la capacité d'ACO à trouver des solutions de meilleure qualité que GRASP, il nous paraît naturel de privilégier l'utilisation d'ACO. De plus, une implémentation de cette métaheuristique dans la plateforme logicielle RECIFE PC est à notre disposition, facilitant ainsi des tests utilisant directement les données ferroviaires.

En résumé, nous focalisons notre étude de l'existant sur deux approches algorithmiques :

- la métaheuristique ACO pour le SPP telle que développée par Gandibleux et al. [30, 31, 32] ;
- la génération de colonnes, proposée par Lusby et al. [46, 47, 48].

Nous présentons le fonctionnement de ces deux méthodes, puis étudions la mesure dans laquelle il est envisageable de les réappliquer au modèle et aux données dont nous sommes en présence.

4.1.1 Métaheuristique d'optimisation par colonie de fourmis

4.1.1.1 Présentation

ACO est une métaheuristique initialement suggérée par Colorni et al. [14] dont le fonctionnement se base sur l'analogie avec laquelle les colonies de fourmis développent une intelligence collective *via* le dépôt de phéromones, leur permettant notamment de trouver le chemin le plus court entre deux points. Ces phéromones forment une « trace » le long des chemins explorés par les fourmis. La concentration de phéromones par unité de surface augmentant plus rapidement sur le chemin le plus court du fait des plus nombreux aller-retours réalisés par les individus l'empruntant fait peu à peu converger l'ensemble de la colonie vers ce chemin. Il y a donc à la fois un phénomène d'exploration de nouveaux chemins et un phénomène d'exploitation de l'information acquise collectivement.

Étant donnée une formulation SPP à n variables, la version d'ACO proposée par Gandibleux et al. [30, 31, 32] pour le SPP implémente la trace de phéromones par un n-vecteur. Toute variable x_j possède ainsi un unique niveau de phéromone associé dans l'intervalle $[0, 1]$. Plus la valeur de phéromone est proche de 1, plus la variable associée est privilégiée dans la recherche d'une solution, c'est-à-dire qu'une préférence plus forte lui sera accordée dans la décision de lui attribuer la valeur 1, lui faisant ainsi contribuer à augmenter la valeur de la fonction-objectif. Initialement, tous les niveaux de phéromone sont à 1, signifiant qu'aucune variable n'est privilégiée.

ACO est un algorithme itératif, au sein duquel chaque fourmi calcule exactement une solution réalisable à chaque itération, le nombre de fourmis étant fixé arbitrairement à l'initialisation de l'algorithme. L'ensemble de l'algorithme peut être résumé de la manière suivante :

1. construire une solution initiale par un algorithme heuristique ;
2. initialiser les phéromones ;
3. calculer une solution réalisable pour chaque fourmi en exploitant la valeur des phéromones ;
4. mettre à jour les phéromones en fonction de la meilleure solution qui a été trouvée à cette itération ;
5. réitérer à l'étape 3, sauf si un critère d'arrêt est atteint.

Le critère d'arrêt de l'algorithme consiste à estimer si la valeur des phéromones est stabilisée depuis un nombre minimal d'itérations. L'algorithme ne s'arrête cependant pas dès la première stabilisation : à l'issue des deux premières stabilisations au moins, une perturbation est effectuée puis l'algorithme est relancé, ceci ayant pour objectif d'échapper aux extrema locaux.

À la fin d'une itération, deux opérations sont réalisées sur les phéromones : toutes les valeurs sont multipliées par un coefficient « d'évaporation » situé dans l'intervalle $]0, 1[$, puis les phéromones correspondant aux variables ayant la valeur 1 dans la meilleure solution de l'itération sont augmentée d'une valeur de « dépôt ». La recherche de solutions réalisables à l'étape 3 combine l'exploitation des phéromones avec une exploration aléatoires des variables pour diversifier la recherche de solutions.

Cette implémentation d'ACO repose sur la génération d'une *matrice d'exclusivité* de taille $n \times n$ lors de son initialisation. Cette matrice modélise simplement l'ensemble des conflits deux-à-deux entre variables, et permet à l'algorithme de déterminer en temps constant si deux variables sont incompatibles pendant son exécution. La génération et le stockage de cette matrice sont fortement consommateurs respectivement de temps et de mémoire, en particulier pour les instances de SPP de grande taille.

4.1.1.2 Réutilisabilité

L'analyse des résultats comparatifs par Gandibleux et al. [30] mettent en valeur la capacité d'ACO à trouver des solutions de bonne qualité, de façon régulière et avec un temps de résolution raisonnable. Cet

algorithme est en outre spécifiquement présenté comme méthode privilégiée dans le cadre du problème de saturation par Gandibleux et al. [32], et est implémenté au sein de la plateforme de calcul de capacité d'infrastructures ferroviaires RECIFE PC.

Il a cependant été constaté que les instances les plus grandes (en terme de nombre de variables) considérées dans le présent travail nécessitent un temps de résolution dépassant les 100 000 secondes avec ACO, ce qui, dans l'hypothèse de l'utilisation du logiciel RECIFE PC, apparaît comme difficilement acceptable. Après analyse du comportement de l'algorithme, il est apparu que la phase de création de la matrice d'exclusivité, était majoritairement responsable de ce temps de résolution. La présence de cette matrice ayant un rôle crucial dans le fonctionnement de l'algorithme, il ne semble *a priori* pas envisageable de réduire ce temps aisément.

Ainsi, bien qu'ACO représente le meilleur compromis entre la qualité des solutions obtenues et le temps de résolution nécessaire, ses performances restent insuffisantes pour certaines de nos instances. De plus, il a été remarqué une consommation de mémoire particulièrement importante sur les instances de grande taille, impliquant l'arrêt de l'algorithme sur l'instance la plus grande. Ce second problème peut s'avérer particulièrement gênant pour un algorithme intégré dans un logiciel d'aide à la décision.

Enfin, si la régularité d'ACO en terme de qualité de solution a effectivement été montrée empiriquement par Gandibleux et al. [30, 31, 32], il peut être souhaitable de se doter d'un moyen d'estimer la qualité d'une solution donnée par ACO de façon simultanée avec l'obtention de cette solution. Bien entendu, ceci n'est pas spécifique à ACO, mais simplement au fait qu'un algorithme approché n'est pas en mesure de prouver l'optimalité de la solution retournée. En d'autres termes, il peut être souhaitable d'adjoindre une méthode fournissant une borne supérieure.

4.1.2 Génération de colonnes

4.1.2.1 Principe général

La génération de colonnes est un cadre algorithmique générique visant à résoudre les formulations de PL. Son utilisation peut être particulièrement pertinente dans les deux cas de figure suivants :

- la formulation considérée contient un grand nombre de variables, et la solution optimale peut être atteinte sans avoir à considérer la totalité d'entre elles ;
- l'ensemble des variables n'est pas fourni explicitement par les données, mais la définition de celles-ci répond à un ensemble de contraintes définissant un problème qu'il est nécessaire de résoudre pour générer des variables valides.

Une procédure de génération de colonnes contient en outre les éléments caractéristiques suivants :

- un *problème maître*, qui désigne la formulation de PL que l'on cherche à résoudre ;
- un *problème maître restreint* (PMR), qui désigne cette même formulation, restreinte à un sous-ensemble de variables, telle qu'elle prise en compte pendant l'exécution de la génération de colonnes ;
- un *sous-problème*, qui est un problème de décision visant à trouver (ou générer) une ou plusieurs variables (ou colonnes) à ajouter au PMR et dont l'insertion est susceptible d'améliorer l'objectif de ce dernier.

La génération de colonnes est un algorithme itératif, dont une itération correspond à la résolution successive du PMR et du sous-problème. Le déroulement de l'algorithme de génération de colonnes peut ainsi être résumé de la sorte :

1. résoudre le PMR à l'optimalité avec un algorithme usuel de PL (typiquement, l'algorithme du simplexe) ;

2. résoudre le sous-problème de façon à trouver de nouvelles variables valides et dont l'insertion est susceptible d'améliorer l'objectif du PMR ;
3. si de telles variables sont trouvées, en sélectionner un sous-ensemble, les insérer dans le PMR puis réitérer à l'étape 1 ; sinon, arrêter l'algorithme et retourner la dernière solution optimale du PMR.

Le principe général de la génération de colonnes consiste donc à améliorer itérativement la valeur de l'objectif du PMR par insertion progressive de colonnes dans la formulation, jusqu'à obtention de la valeur optimale.

La résolution du sous-problème à l'étape 2 doit aboutir à l'obtention de nouvelles variables, celles-ci devant obéir à deux contraintes :

- représenter des objets valides pour le problème considéré (dans notre cas, cela serait par exemple un train et un chemin valides) ;
- être susceptibles d'améliorer l'objectif du PMR.

Ce second point est estimé par la valeur du coût réduit associé à chacune de ces variables : si le PMR est un problème de maximisation (respectivement minimisation), le coût réduit doit être strictement positif (respectivement négatif) afin que l'insertion de la colonne soit susceptible d'améliorer l'objectif du PMR.

Calculer le coût réduit d'une variable non présente dans le PMR se fait en connaissant la solution optimale du dual du PMR et les coefficients de cette variable dans la matrice du PMR. Nous considérons que la solution optimale duale est fournie par le simplexe simultanément à la solution optimale primale. Un coût réduit indiquant qu'une nouvelle colonne est susceptible d'améliorer l'objectif du PMR correspond à une contrainte violée dans le dual par la solution optimale duale courante. La formule du coût réduit peut donc être déduite de l'expression du dual du problème maître.

Si la résolution du sous-problème ne trouve pas de variable à insérer dans la formulation, c'est-à-dire respectant les deux critères énoncés ci-dessus, deux cas de figure sont possibles :

- si le sous-problème est capable d'énumérer l'ensemble des variables et qu'aucune d'entre elles ne possède un coût réduit adéquat, la dernière solution optimale du PMR correspond à l'optimal pour le problème maître complet ;
- si le sous-problème n'offre pas cette garantie, l'optimalité de la dernière solution optimale du PMR n'est pas prouvée.

En outre, tout problème de PL et son dual possèdent la même valeur de fonction-objectif à l'optimum. Ainsi, s'il est possible de calculer une solution réalisable pour le problème dual complet (appelée *borne duale*) pendant l'exécution de la génération de colonnes, tester l'égalité entre la solution optimale courante au PMR et la borne duale fournit un critère d'arrêt supplémentaire. Il peut également être pertinent d'arrêter l'algorithme lorsque l'écart entre les bornes primale et duale passe sous un seuil donné.

En pratique, il est courant que le sous-problème soit considéré comme un problème d'optimisation, où l'on cherche des colonnes au coût réduit dont la valeur absolue est la plus élevée possible. En outre, le sous-problème est extrêmement spécifique au problème concret considéré, puisque les variables générées doivent être valides au regard de celui-ci.

La génération de colonnes implique donc de multiples exécutions à la fois du simplexe et de la résolution du sous-problème. Il est donc nécessaire d'estimer, selon le cas, s'il est effectivement moins coûteux en terme de temps de calcul d'effectuer plusieurs résolutions successives de PMR et de sous-problème plutôt que de ne réaliser qu'une seule résolution de la formulation complète. La connaissance préalable de la structure de certaines solutions au problème considéré peut notamment aider à répondre à cette question.

Pour terminer, il est important d'identifier les deux leviers sur lesquels il est possible d'agir pour réduire le temps de calcul d'une génération de colonnes :

– réduire le temps de calcul de chaque itération, c'est-à-dire le temps passé dans la résolution du PMR ou du sous-problème ;
– réduire le nombre total d'itérations nécessaires pour atteindre la solution optimale, c'est-à-dire le nombre de fois où un PMR et un sous-problème sont résolus.

Le premier objectif peut être poursuivi en cherchant à résoudre de façon plus performante le PMR ou le sous-problème. Il peut être possible d'exploiter certaines spécificités du problème pour en accélérer la résolution. Réduire le nombre d'itérations peut être réalisé en tentant de sélectionner, après la résolution du sous-problème, un sous-ensemble de colonnes que l'on estime être les plus pertinentes pour se diriger vers la solution optimale. Cette notion peut faire appel ici encore à des données spécifiques au problème.

4.1.2.2 Application au Set Packing modélisant la faisabilité

Lusby et al. [46, 47, 48] utilisent un algorithme de génération de colonnes dont le problème maître est la relaxation continue du SPP modélisant le problème de faisabilité. Les caractéristiques des données auxquelles sont confrontés les auteurs sont les suivantes :

– le nombre de trains est faible, à savoir de l'ordre de la dizaine pour les résultats présentés par les auteurs ;
– le nombre de valeurs de retard possibles pouvant être subi par chaque train à l'entrée de l'infrastructure est également faible et ne dépasse pas 5 ;
– une finesse de discrétisation temporelle de 15 secondes est suffisante ;
– les auteurs ont a leur disposition les propriétés dynamiques des trains et les limitations de vitesse sur l'infrastructure, ce qui permet de considérer, pour chaque train et pour chaque itinéraire, un grand nombre de courbes de vitesse possibles permettant au train d'aller de son point d'entrée à son point de sortie.

Ce dernier point implique un très grand nombre de chemins possibles pour chaque train. S'il existe théoriquement une infinité de variations de vitesse possibles, les hypothèses prises par les auteurs restreignent toutefois les courbes de vitesses disponibles pour chaque train à un ensemble fini. Lusby et al. [46] indiquent qu'elles induisent un total de plus de 700 000 chemins, justifiant l'utilisation de la génération de colonnes. Il est en outre aisé de se rendre compte qu'une solution optimale à la formulation SPP n'utilisera qu'une proportion infime des variables, puisqu'au plus une variable par train peut prendre la valeur 1. Les auteurs mentionnent enfin la présence d'environ 3 800 contraintes dans la formulation SPP.

Le sous-problème consiste à générer des chemins réalisables en faisant varier la vitesse de chaque train sur chacun de ses itinéraires possibles. Il est intéressant de se pencher sur la façon dont sa résolution est réalisée au vu des spécificités des données.

Par hypothèse, sur chaque zone, un train ne peut avoir qu'une accélération constante, qu'elle soit positive, négative ou nulle. Les auteurs créent une structure d'arbre pour chaque itinéraire et retard possibles pour chaque train. Ceci est illustré par la figure 4.1 : la racine correspond à la date d'entrée du train sur la première zone de son itinéraire. Les trois arcs sortant de ce nœud indiquent les trois possibilités de temps passé sur cette zone en fonction de l'action choisie (accélération, décélération, vitesse constante). Les trois nœuds au deuxième niveau de l'arbre correspondent aux trois possibilités de date à laquelle le train entre sur la deuxième zone de son itinéraire, et ainsi de suite jusqu'à la dernière zone.

Un chemin de la racine à une feuille de l'arbre représente une variable du problème maître, car elle définit l'ensemble des périodes d'occupation de chacune des zones de l'itinéraire considéré. Ainsi, en réalisant un parcours de l'arbre depuis la racine jusqu'à chacune des feuilles, les coefficients des colonnes dans la matrice des contraintes sont connus et les coûts réduits associés peuvent être calculés.

Les auteurs font état d'un temps de calcul de l'ordre de la seconde pour les arbres les plus grands (jusqu'à 250 000 chemins), et d'un temps de résolution total du problème inférieur à 5 secondes.

Nous avons donc ici une procédure de génération de colonnes performante, dont l'utilisation est justifiée par le grand nombre de variables. Nous y trouvons des éléments similaires avec les formulations SPP induites par nos données, à savoir :

- un grand nombre de chemins possibles pour chaque train ;
- par voie de conséquence, une faible proportion des variables nécessaires à l'obtention de la solution optimale.

Cependant, les données fournies permettant de définir les chemins contiennent une différence essentielle : la vitesse des trains est variable, ce qui est permis par la connaissance de la dynamique des trains. Dans notre cas, la vitesse sur chaque itinéraire est fixe, mais la résolution du problème de saturation implique une grande liberté de variation des dates d'entrée. Ceci modifie donc substantiellement la nature du sous-problème, dont la modélisation et la résolution ne sont pas réutilisables. Ceci ne permet en outre *a priori* pas de profiter des mêmes performances, inhérentes au parcours performant des arbres.

Enfin, les auteurs ont la possibilité d'utiliser une discrétisation temporelle de 15 secondes, ce qui n'est pas permis par nos instances, qui imposent une discrétisation 15 fois plus fine et en conséquence un important nombre de contraintes, à savoir compris entre 120 000 et 350 000.

Ainsi, si cette approche par génération de colonnes apporte des éléments pertinents, il apparaît aussi qu'elle ne répond pas complètement au problème auquel nous faisons face.

4.1.3 Synthèse

Nous avons vu plusieurs innovations algorithmiques apportant chacune des éléments pour résoudre des formulations SPP modélisant les problèmes de faisabilité ou de saturation. En particulier, la méta-heuristique ACO proposée par Gandibleux et al. [30, 31, 32] et la génération de colonnes proposée par Lusby et al. [46, 47, 48] montrent des performances intéressantes, même si la première butte sur les instances possédant un grand nombre de variables et la seconde n'est pas complètement adaptée aux caractéristiques de nos instances.

Nous nous penchons donc sur le développement d'un algorithme permettant d'améliorer la résolution du problème de saturation pour les instances dont nous disposons, en profitant de certaines innovations apportées par les travaux présentés précédemment tout en les adaptant et en les combinant avec des contributions algorithmiques originales. Les contributions que nous apportons sont les suivantes :

- une utilisation de la génération de colonnes avec un sous-problème adapté à nos données et un algorithme de résolution efficace ;
- un ensemble de stratégies d'insertion de colonnes prenant en compte à la fois des données génériques et spécifiques au problème afin de sélectionner les colonnes à insérer avec plus de pertinence à chaque itération ;
- des algorithmes d'agrégation de contraintes permettant de réduire le nombre de contraintes dans la formulation et accélérer la résolution du PMR ;
- enfin, une hybridation de l'algorithme avec ACO, où la génération de colonnes nous permet à la fois une exécution d'ACO sur un ensemble de variables plus petit et l'utilisation par ACO de données issues de la génération de colonnes.

Ainsi, nous commençons par présenter la procédure de génération de colonnes adaptée à notre problème dans la section 4.2. Nous détaillons ensuite dans la section 4.3 comment nous réduisons le nombre de contraintes de la formulation grâce à des techniques d'agrégation, profitant notamment de données spécifiques au problème. L'agrégation de contraintes est combinée à la procédure de génération de colonnes.

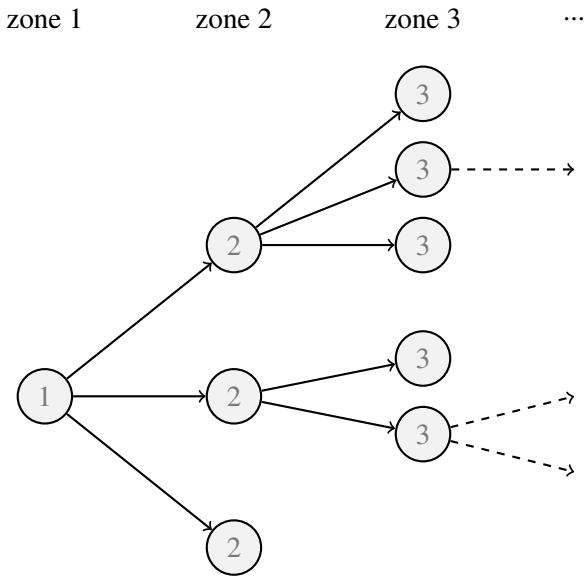

Figure 4.1 – Arbre représentant les chemins possibles d'un train pour un itinéraire et une date d'entrée fixés.

Nous voyons enfin à la section 4.4 comment nous combinons la génération de colonnes avec ACO afin d'obtenir une solution entière.

4.2 Génération de colonnes pour le problème de saturation

Détailler l'algorithme de génération de colonnes que nous utilisons implique en premier lieu d'en donner le problème maître et le sous-problème. Nous revenons donc brièvement sur le problème maître, qui est la relaxation continue de la formulation SPP 3.20 et présentons le dual de celui-ci, dont l'expression permet de formuler l'expression du coût réduit d'une colonne. Nous formalisons ensuite le sous-problème et la procédure de résolution que nous lui appliquons. Nous terminons par un aperçu de plusieurs critères à la fois génériques et spécifiques au problème pouvant être pris en compte lors de la sélection des colonnes à l'issue de la résolution du sous-problème. Ces travaux ont notamment été présentés de façon détaillée par Merel et al. [52], contribution donnant un aperçu global de la méthode de résolution proposée pour le problème de saturation tout en se focalisant particulièrement sur la génération de colonnes.

4.2.1 Problème maître et problème dual

La formulation 3.20 donne le SPP qui modélise notre problème, qui est *a fortiori* un problème de PLNE. Le problème maître de notre algorithme de génération de colonne est la relaxation continue de cette formulation. L'objectif de la génération de colonne est donc de résoudre à l'optimalité cette relaxation continue.

Une façon de formaliser l'expression du coût réduit dans le cadre de la génération de colonnes peut se faire en exprimant le dual du problème maître. Retenons les considérations suivantes, issues de la théorie de la dualité et essentielles dans le cadre de la génération de colonnes :

- rechercher une colonne susceptible d'améliorer l'objectif du PMR est équivalent à rechercher une contrainte du dual du problème maître complet violée par la solution optimale au dual du PMR ;
- une solution réalisable du problème dual complet est une borne supérieure pour la valeur optimale de l'objectif du problème maître complet ;
- lorsque le problème maître est résolu à l'optimalité, la valeur de sa fonction-objectif est égale à la valeur de la fonction-objectif de son dual à l'optimalité.

La formulation duale de notre problème est décrite par l'expression 4.1 :

$$\left[\begin{array}{lllll} \min & \displaystyle\sum_{t\in T} v_t + \sum_{u\in U} w_u & & & \\ \text{s.c.} & v_t + \displaystyle\sum_{u\in U} \delta_{c,u} w_u & \geq \rho_{t,c} & \forall t \in T, \forall c \in C_t \\ & v_t & \geq 0 & \forall t \in T \\ & w_u & \geq 0 & \forall u \in U \end{array}\right] \tag{4.1}$$

où les variables v_t et w_u sont des variables duales, que nous nommons respectivement *variables de train* et *variables de ressource*.

Nous avons vu que la connaissance de l'ensemble des coefficients $\delta_{c,u}$ pour un chemin $c \in C_t$ associé à un train $t \in T$ décrit intégralement l'utilisation des ressources que celui-ci implique, et conséquemment décrit l'intégralité de la colonne correspondant à ce chemin dans la matrice des contraintes du problème

maître. La définition de la formulation duale montre que la connaissance de ces mêmes coefficients décrit intégralement, de façon analogue, une contrainte dans le problème dual.

De ce fait, nous pouvons rechercher une colonne à ajouter au problème maître en recherchant une contrainte de sa formulation duale violée par la solution optimale duale du PMR, puisque cette solution est fournie par le simplexe lors de la résolution du PMR.

4.2.2 Sous-problème et borne duale

Notons la solution optimale au dual du PMR (v^*, w^*), où v^* est un $|T|$-vecteur et w^* est un $|U|$-vecteur, dont les composantes respectives sont les valeurs des variables v_t et w_u pour cette solution. Le coût réduit pour une colonne du problème maître, c'est-à-dire d'un chemin $c \in C_t$ pour un train $t \in T$, est noté $\pi_{t,c}$, et est donné par la formule 4.2 :

$$\pi_{t,c} = \rho_{t,c} - v_t^* - \sum_{u \in U} \delta_{c,u} w_u^*. \tag{4.2}$$

Ainsi, nous recherchons des contraintes duales violées par cette solution. Le sous-problème consiste donc à rechercher l'ensemble des couples (t, c) tels que $t \in T$, $c \in C_t$ et $\pi_{t,c} > 0$. En d'autres termes, ceci revient à chercher, parmi les chemins disponibles pour chacun des trains de l'instance, ceux dont la colonne associée possède un coût réduit positif.

En outre, le problème maître étant un problème de maximisation, la connaissance d'une borne supérieure peut être un indicateur d'avancement ou servir de critère d'arrêt lors de la résolution. Nous cherchons donc une affectation des variables duales $v_t, \forall t \in T$ et $w_u, \forall u \in U$ telle que l'ensemble des contraintes de la formulation 4.1 soient respectées. Une façon naïve de construire une solution duale serait d'affecter la valeur $\max_{t \in T, c \in C_t}(\rho_{t,c})$ à chaque variable v_t, et 0 à chaque variable w_u. Cependant, l'intérêt de la borne duale est de posséder la valeur la plus basse possible, sans pour autant nécessiter un temps de calcul supplémentaire conséquent. Ici, il est possible d'exploiter la connaissance de la solution (v^*, w^*) afin d'améliorer, à chaque itération, la valeur de cette borne supérieure.

Nous proposons une formule permettant de calculer une borne duale à chaque itération de la génération de colonnes, une fois que le PMR est résolu. Nous définissions le vecteur $\hat{\pi} = (\hat{\pi}_t)_{t \in T}$ tel qu'exprimé par la formule 4.3 :

$$\hat{\pi}_t = \max_{c \in C_t} (\pi_{t,c}) \,, \ \forall t \in T. \tag{4.3}$$

Par conséquent, nous avons donc, pour tout $t \in T$ et $c \in C_t$:

$$\begin{aligned} \rho_{t,c} - v_t^* - \sum_{u \in U} \delta_{c,u} w_u^* &\leq \hat{\pi}_t \\ \text{i.e.} \quad v_t^* + \hat{\pi}_t + \sum_{u \in U} \delta_{c,u} w_u^* &\geq \rho_{t,c}. \end{aligned}$$

Ceci signifie que la solution composée de $(v^* + \hat{\pi}, w^*)$ vérifie l'ensemble des contraintes du problème dual. Il s'agit donc d'une solution réalisable, dont la valeur de la fonction-objectif associée consiste en une borne supérieure pour le problème maître. Ainsi, à chaque itération de colonne, une nouvelle borne supérieure $\overline{z}$ peut être calculée par la formulation 4.4 :

$$\overline{z} \ = \ \sum_{t \in T} (v_t^* + \hat{\pi}_t) + \sum_{u \in U} w_u^*. \tag{4.4}$$

Notons que le calcul de cette borne duale suppose de connaître, pour chaque train $t \in T$, la valeur de $\hat{\pi}_t$. Bien que cela semble nécessiter à première vue le parcours de l'ensemble des chemins pour chaque train, nous voyons ci-après que le calcul de ces vecteurs est inhérent à la résolution du sous-problème que nous mettons en place, et peut être donc connu sans effort algorithme supplémentaire.

4.2.3 Résolution du sous-problème

Étant donné une solution optimale au dual du PMR (v^*, w^*), le sous-problème vise à rechercher l'ensemble des couples train-chemin possédant un coût réduit strictement positif. Formellement, pour chaque train $t \in T$, nous recherchons un vecteur $(\delta_{c,u})_{u \in U}$ décrivant un chemin $c \in C_t$ tel que $\pi_{t,c} > 0$.

Bien que, comme le montre le tableau 3.4, leur nombre soit particulièrement élevé, tous les chemins disponibles pour chaque train sont implicitement connus et peuvent de ce fait être énumérés, et le coût réduit qui y est associé peut être calculé. Il est de ce fait possible d'envisager une approche naïve, consistant à effectuer le calcul de $\pi_{t,c}$ autant de fois qu'il y a de chemins disponibles dans une instance. En particulier, le calcul de $\sum_{u \in U} \delta_{c,u} w^*_u$ nécessite un nombre d'itérations au moins égal au nombre de ressources de U utilisées par le chemin considéré. En pratique, des tests préliminaires ont montré que la conjugaison du grand nombre de chemins et du nombre important de ressources utilisées par chacun des chemins rendait ce calcul coûteux et non-négligeable par rapport au temps de résolution total de la génération de colonnes.

Une approche plus subtile peut être considérée pour accélérer la résolution du sous-problème tout en conservant le calcul de l'ensemble des coûts réduits. Nous expliquons ici le raisonnement permettant de réduire le nombre d'itérations pour résoudre le sous-problème. Considérons deux chemins c et c_{+1} associés à un même itinéraire, qui ne diffèrent l'un de l'autre que par une unique période temporelle, c'est-à-dire où les entrées et sorties sur les zones de c_{+1} sont réalisées exactement une période plus tard que sur c. Ceci est un cas de figure typique pour un train pouvant emprunter un itinéraire donné avec plusieurs valeurs de retard successives.

Formellement, nous pouvons écrire $c = (r, p)$ où r est l'itinéraire qui est associé à c et $p \in \{p^e_t, \ ..., \ (p^e_t + \overline{d_t})\}$ est la période d'entrée du train pour ce chemin. Conséquemment, c_{+1} peut s'écrire $c_{+1} = (r, p+1)$. Pour $s \in Z_r$, nous posons $p^i_{s,c}$ et $p^o_{s,c}$ les périodes auxquelles le train respectivement rentre et sort de la zone s lorsqu'il utilise le chemin c. Entre c et c_{+1}, l'utilisation de la zone s est décalée d'exactement une période temporelle, ce qui peut s'écrire $p^i_{s,c_{+1}} = p^i_{s,c} + 1$ pour la date d'entrée et $p^o_{s,c_{+1}} = p^o_{s,c} + 1$ pour la date de sortie.

En reprenant l'expression 4.2 donnant la formule du coût réduit $\pi_{t,c}$, et en rappelant que toute ressource $u \in U$ peut être écrite sous la forme d'un couple zone-période (s, p), nous pouvons ainsi formaliser la différence entre $\pi_{t,c_{+1}}$ et $\pi_{t,c}$:

$$\begin{aligned}
\pi_{t,c_{+1}} - \pi_{t,c} &= \sum_{(s,p) \in U} \delta_{c,u} w^*_{(s,p)} - \sum_{(s,p) \in U} \delta_{c_{+1},u} w^*_{(s,p)} \\
&= \sum_{s \in Z_r} \left(\sum_{p=p^i_{s,c}}^{p^o_{s,c}} w^*_{(s,p)} - \sum_{p=p^i_{s,c}+1}^{p^o_{s,c}+1} w^*_{(s,p)} \right) \\
&= \sum_{s \in Z_r} \left(w^*_{(s,p^i_{s,c})} - w^*_{(s,p^o_{s,c}+1)} \right)
\end{aligned}$$

ce qui est finalement équivalent à l'expression suivante :

$$\pi_{t,c_{+1}} = \sum_{s \in Z_r} \left(w^*_{(s,p^i_{s,c})} - w^*_{(s,p^o_{s,c}+1)} \right) + \pi_{t,c}. \tag{4.5}$$

L'expression 4.5 montre qu'il est possible de calculer $\pi_{t,c_{+1}}$ à partir $\pi_{t,c}$ en itérant uniquement sur les zones de Z_r, plutôt que sur l'ensemble des ressources de c_{+1}. Le degré de liberté en terme de retard autorisé pour chaque train étant une cause principale du grand nombre de variables, l'utilisation de cette méthode est particulièrement pertinente. Ainsi, un nombre d'itérations égal au nombre de zones du chemin peut être considéré comme négligeable par rapport au nombre total de ressources spatio-temporelles utilisées par ce chemin, rendant le gain d'autant plus significatif. Cette approche est donc utilisée pour calculer les coûts réduits lors de la génération de colonnes.

Afin d'illustrer l'avantage de cette approche, nous nous penchons sur un itinéraire présent au sein de la jonction de Pierrefitte-Gonesse, possédant les caractéristiques suivantes :

– il comprend 32 zones de détection ;
– le temps d'occupation d'une zone varie entre 40 et 121 secondes selon les zones ;
– un total de 2 366 ressources spatio-temporelles sont utilisées par tout chemin utilisant cet itinéraire.

Considérons un train possédant un retard maximal autorisé de 5 périodes et pouvant circuler sur cet itinéraire. La première approche de résolution du sous-problème nécessite d'itérer 6 fois sur l'ensemble des 2 366 ressources afin de calculer tous les coûts réduits, soit un total de 14 196 itérations. La seconde méthode nécessite d'itérer une fois sur l'ensemble des ressources (pour le chemin correspondant à un délai nul), après quoi cinq itérations sur chacune des 32 zones sont requises, soit un total de 2 526 itérations, et un gain d'un facteur proche du nombre de décalages autorisés. En pratique, nous considérons des décalages temporels compris entre 30 et 90 secondes, impliquant un potentiel d'amélioration nettement plus élevé.

Finalement, l'algorithme 4.1 applique ce principe pour résoudre le sous-problème. Étant donné un train t et un de ses itinéraires $r \in R_t$, le coût réduit pour un retard nul est calculé selon la formule standard exprimée par l'équation 4.2, à la ligne 4 de l'algorithme. Les coûts réduits pour les chemins associés au même train, au même itinéraire et pour des décalages supérieurs sont ensuite calculés itérativement comme cela est décrit par la formule 4.5, à la ligne 8. Le vecteur $\hat{\pi}$ est calculé au même moment (lignes 5 et 9), autorisant ainsi le calcul de la borne duale sans effort calculatoire supplémentaire. L'algorithme itérant sur tous les trains et tous les chemins possibles pour chaque train, l'ensemble des coûts réduits est bien calculé à chaque résolution du sous-problème.

Nous sommes donc capables de réaliser une itération complète de génération de colonnes, à savoir :

1. résoudre le problème maître de façon classique par un algorithme du simplexe ;
2. résoudre le sous-problème grâce à la connaissance de la solution duale optimale du PMR, par l'algorithme 4.1.

L'ensemble des variables candidates dont le coût réduit est strictement positif peuvent ainsi être identifiées à chaque itération. Dans le cas où aucune ne répond à ce critère, cela signifie que la solution optimale au problème maître a été trouvée. Dans le cas contraire, il est nécessaire de sélectionner certaines de ces variables pour les ajouter au PMR. Cette sélection, bien que pouvant se faire sur l'unique critère de la valeur du coût réduit, peut bénéficier de la considération d'autres critères, et ainsi permettre la mise en place de stratégies d'insertion de colonnes.

Entrées : solution optimale duale du PMR (v^*, w^*).

$\hat{\pi} \leftarrow (-\infty,\ ...,\ -\infty)$;

pour chaque $t \in T,\ r \in R_t$ **faire**

`// Calcul pour un délai nul`

$c \leftarrow (r, p_t^e)$;

$\pi_{t,c} \leftarrow \rho_{t,c} - v_t^* - \sum_{u \in U} \delta_{c,u} w_u^*$;

$\hat{\pi}_t \leftarrow \texttt{max}\,(\hat{\pi}_t,\, \pi_{t,c})$;

pour $d \leftarrow 1$ **a** $\overline{d_t}$ **faire**

`// Calcul pour les valeurs suivantes de délai`

$c_{+1} \leftarrow (r, p_t^e + d)$;

$\pi_{t,c_{+1}} \leftarrow \pi_{t,c} + \sum_{z \in Z_r} \left(w^*_{(z,p^i_{z,c})} - w^*_{(z,p^o_{z,c}+1)} \right)$;

$\hat{\pi}_t \leftarrow \texttt{max}\,(\hat{\pi}_t,\, \pi_{t,c_{+1}})$;

$c \leftarrow c_{+1}$;

fin

fin

Sorties : coûts réduits $(\pi_{t,c})_{c \in C_t, t \in T}$.

Algorithme 4.1: Calcul de l'ensemble des coûts réduits.

4.2.4 Stratégies d'insertion

Résoudre le sous-problème mène très souvent à trouver plus d'une unique colonne valide possédant un coût réduit strictement positif. Les expériences ont montré qu'insérer toutes les colonnes possédant un tel coût réduit à chaque itération fait grandir la taille du PMR bien trop rapidement, faisant perdre l'intérêt de la génération de colonnes par l'augmentation trop rapide du temps de résolution du PMR. Un compromis doit de ce fait être trouvé pour éviter ce phénomène tout en sélectionnant les colonnes pertinemment de façon à atteindre la solution optimale en un petit nombre d'itérations.

Un grand nombre de stratégies peuvent être conçues afin de sélectionner quelles colonnes ayant un coût réduit strictement positif doivent être insérées. Comme il en existe un nombre potentiellement illimité, nous nous focalisons ici sur certains critères clés pouvant influencer la sélection dans notre cas. Ces paramètres peuvent être bien entendu être combinés pour former ainsi des stratégies particulières.

4.2.4.1 Coût réduit

Dans le contexte très général de la génération de colonnes, la valeur du coût réduit est souvent un critère prédominant pour décider si une colonne doit être insérée ou non. En pratique, on ne cherche pas seulement les colonnes ayant un coût réduit positif, mais plutôt les coûts réduits les plus élevés. Il est ainsi possible de définir un seuil de telle sorte que les colonnes possédant un coût réduit inférieur à celui-ci ne soient pas sélectionnées. L'intérêt de ce critère réside dans le fait que les variables avec le coût réduit le plus élevé sont théoriquement les plus à même d'améliorer la valeur de la fonction-objectif.

4.2.4.2 Nombre de colonnes

Une manière simple de contrôler l'augmentation de la taille du PMR, et donc le temps de calcul nécessaire à sa résolution, consiste à limiter le nombre de colonnes à insérer à chaque itération. Une approche classique consiste à sélectionner les $l > 0$ meilleure colonnes, classées selon un autre critère, typiquement le coût réduit. Cette approche peut être utile pour éviter d'insérer en une itération un nombre trop important de colonnes ayant toutes une bonne valeur de coût réduit dont la valeur similaire est due au fait que ces colonnes sont très proches en termes des coefficients de la matrice des contraintes qu'elles occupent.

4.2.4.3 Diversification des trains et des itinéraires

Dans le contexte plus particulier du problème de saturation, nous savons qu'un train ne peut être routé qu'une fois. Nous savons en outre que deux chemins distincts pour le même train peuvent être extrêmement proches en termes d'utilisation de ressources, auquel cas, si les deux ont un coût réduit élevé, il n'est pas nécessairement pertinent de les insérer tous les deux. Une approche peut donc consister à forcer le fait que pas plus d'une colonne par train ou par couple train-itinéraire ne soit insérée à chaque itération.

L'idée principale menant à ce type d'approche est que la valeur des coûts réduits tend à privilégier les chemins utilisant des ressources dont les contraintes correspondantes sont faiblement saturées par la solution optimale au PMR. En conséquence, deux chemins très similaires (par exemple, avec une simple période temporelle de décalage) peuvent avoir tous les deux valeurs de coût réduit élevées. Cependant, leur similarité signifie également que ces chemins sont en conflits sur un très grand nombre de ressources. En forçant la diversification des itinéraires sélectionnés, une forme d'anticipation sur les conflits au sein du PMR est réalisée.

Un exemple de stratégies utilisant ces critères de diversification serait de sélectionner le chemin possédant le coût réduit le plus haut pour chaque train et chaque itinéraire. Nous pouvons noter en outre qu'il s'agit là de la stratégie employée par Lusby et al. [48] dans le cadre de l'algorithme de génération de colonnes qu'ils proposent.

4.2.4.4 Limitation de l'utilisation des ressources

Finalement, dans cette même perspective d'anticiper les conflits dans le PMR, une approche possible consiste à fixer une limite entière $L_u > 0$ de telle sorte que chaque ressource $u \in U$ ne soit pas concernée par plus de L_u nouvelles colonnes à une itération donnée. Ici, plutôt que de forcer une diversification en termes de trains, cela tente de forcer une diversification en termes de ressources utilisées.

4.3 Agrégation de contraintes

Les instances que nous considérons nécessitent une discrétisation temporelle fine, entraînant la présence d'un grand nombre de contraintes dans la formulation du fait du grand nombre de périodes temporelles présentes dans le modèle. Les instances décrites dans le tableau 3.4 montrent en effet un nombre de contraintes compris entre environ 120 000 et 350 000, ce qui implique une augmentation significative du temps de résolution du PMR.

Nous proposons donc des algorithmes visant à réduire ce nombre par suppression des redondances au sein de l'ensemble des contraintes. La conception de ces algorithmes est basée sur des observations liées à la structure du problème, et profitent dans leur recherche de certaines considérations spécifiques. Nous expliquons tout d'abord le principe général de l'agrégation de contraintes puis donnons des précisions relatives au SPP. Nous présentons enfin l'algorithme d'agrégation développé et détaillons la façon dont celui-ci est intégré à la procédure de génération de colonnes.

4.3.1 Principe général

L'idée de regrouper les contraintes dans une matrice à coefficients binaires est initialement suggérée par Ryan et Foster [65] dans le contexte du problème de *Set Partitioning*. Leur approche consiste à regrouper les contraintes de façon à donner à la matrice une structure particulière visant à respecter les propriétés décrites par Padberg [59] permettant de s'assurer que la solution optimale à la relaxation continue soit une solution entière. Les groupements de contraintes sont vus comme un pré-traitement à la résolution même de la formulation, et exploitent des données spécifiques au problème. De plus, les auteurs bénéficient, de par la structure du problème traité, de la possibilité d'atteindre des solutions entières de bonne qualité en restreignant les variables considérées à des variables dont les coefficients dans la matrice des contraintes ne casse pas ces groupes.

La terminologie de l'agrégation de contraintes dans le cadre de la génération de colonnes est introduite par Elhallaoui et al. [28], également dans le cadre de problème de *Set Partitioning*. Dans leur cas, la motivation principale est la réduction de la taille de la formulation du problème maître dans le but de réduire le temps passé à résoudre le PMR. Contrairement à Ryan et Foster [65], l'idée est simplement de regrouper les contraintes redondantes plutôt que de chercher à garantir une structure particulière à la matrice des contraintes. Les auteurs présentent toutefois, comme résultat additionnel, des résultats bénéfiques de leur approche sur les propriétés de la matrice. La méthode proposée par Elhallaoui et al. [28] est nommée « agrégation dynamique de contraintes » (DCA [1]), pour souligner le fait qu'elle n'est pas

1. *Dynamic Constraint Aggregation*, en anglais.

seulement utilisée comme pré-traitement mais est également exécutée entre deux itérations consécutives de la génération de colonnes.

4.3.2 Application au Set Packing pour le problème de saturation

Dans le cas du problème de saturation modélisé par la formulation SPP 3.20, il est possible de percevoir intuitivement la présence de contraintes fortement similaires. Étant donné que tout chemin emprunte chaque zone pendant une durée strictement supérieure à une seconde, il est possible d'anticiper la présence d'un certain nombre de redondances, particulièrement au sein des contraintes de ressource. En effet, il est aisément imaginable que la plupart des chemins occupant une zone s à la période p l'occupent également à la période $p+1$, seuls les chemins pour lesquels la période p est la dernière période d'occupation de s ou pour lesquels la période $p+1$ est la première période d'occupation de s faisant exception. Il résulte de cette observation que les contraintes associées aux ressources respectives (s, p) et $(s, p+1)$ contiennent probablement des ensembles de variables fortement similaires.

Ces observations nous incitent à anticiper la présence de deux types de relations pouvant apparaître entre les contraintes. La première est la relation de redondance, telle qu'utilisée par Elhallaoui et al. [28]. Deux contraintes de packing sont *redondantes* si elles contiennent exactement les mêmes variables. Dans une représentation graphique du SPP telle que donnée en exemple par la figure 3.4, deux contraintes redondantes seraient représentées par deux ensembles contenant exactement les mêmes éléments. La seconde relation, non valable dans le cas du *Set Partitioning*, est la relation de *dominance* entre les contraintes. Une première contrainte domine une seconde contrainte si toutes les variables de la seconde sont strictement inclues dans la première. Ceci serait visible sur la représentation graphique du SPP par une relation d'inclusion stricte entre deux ensembles.

Lorsque deux contraintes sont redondantes, une des deux peut être supprimée de la formulation sans que cela ne modifie l'espace des solutions du problème de PLNE et de sa relaxation continue. *A fortiori*, cela ne modifie ni les solutions réalisables ni les solutions optimales. Lorsqu'une contrainte est dominée, elle peut également être supprimée de la formulation, toujours sans modifier l'ensemble des solutions. Un algorithme supprimant les contraintes redondantes et dominées dans le SPP modélisant le problème de saturation est proposé par Merel et al. [54], sous forme de pré-traitement pur. Dans le cas présent, nous mettons en place une méthode supprimant les contraintes redondantes et dominées au sein de la formulation du PMR, intégrée dans l'algorithme de génération de colonnes de façon comparable à ce qui est proposé par Elhallaoui et al. [28]. Par soucis de cohérence, notre approche sera également nommée sous la terminologie d'agrégation dynamique de contraintes.

4.3.3 Interaction avec la génération de colonnes

L'objectif de l'agrégation de contraintes est l'accélération du temps de résolution du PMR à chaque itération. De ce fait, l'agrégation doit être effectuée une fois que de nouvelles colonnes ont été ajoutées au PMR, et avant une nouvelle résolution de celui-ci. Nous appelons problème maître restreint agrégé (PMRA) le PMR duquel des contraintes redondantes et/ou dominées ont été supprimées par un algorithme d'agrégation.

Dans une procédure de génération de colonnes, le PMR est modifié à chaque itération du fait des nouvelles colonnes ajoutées. Dans un contexte classique de génération de colonnes, ces nouvelles colonnes ajoutent des éléments à différentes lignes de la matrice des contraintes, c'est-à-dire au sein de certaines contraintes de packing. Toutefois, dans le PMRA, une nouvelle variable peut posséder un coefficient présent dans une contrainte qui a été précédemment supprimée par la procédure d'agrégation, du fait

qu'elle était redondante ou dominée. Conséquemment, cette contrainte peut ne plus être dominée ou redondante, et doit de ce fait de nouveau figurer dans le PMRA. Cette observation doit donc être prise en compte par la conception de l'algorithme de génération de colonnes.

Elhallaoui et al. [28] utilisent un critère afin de décider si le PMRA doit être ré-agrégé après l'insertion de nouvelles colonnes. En effet, l'algorithme d'agrégation de contraintes ne possède pas nécessairement un temps d'exécution négligeable, et il peut ne pas être rentable de l'exécuter à chaque itération. Dans notre cas, des tests ont toutefois montré qu'il était profitable de réaliser une agrégation à chaque itération. Ceci s'explique notamment par notre capacité à profiter de données spécifiques au problème pour agréger un nombre important de contraintes rapidement.

Ainsi, notre algorithme d'agrégation dynamique de contraintes s'intègre de la manière suivante au sein d'une itération de génération de colonnes :

1. résoudre le PMRA :
2. résoudre le sous-problème et sélectionner un ensemble $\mathcal{X}$ de colonnes à insérer dans le PMRA selon une stratégie quelconque (nous supposons que $\mathcal{X} \neq \emptyset$, l'algorithme de génération de colonnes terminant ici dans le cas contraire) ;
3. insérer les contraintes au sein desquelles les variables de $\mathcal{X}$ sont présentes et qui ont été supprimée par la procédure d'agrégation à une itération précédente ;
4. insérer les colonnes de $\mathcal{X}$ dans le PMRA ;
5. ré-agréger le PMRA ;
6. réitérer.

Seules les étapes 3 et 5 représentent finalement une variation par rapport à la procédure de génération de colonnes standard.

4.3.4 Algorithme d'agrégation

Le cadre de la génération de colonnes nous permet d'exploiter des propriétés intéressantes pour éviter les itérations inutiles dans la recherche de redondances. Considérons une contrainte quelconque, notée S_0, qui ne soit ni redondante, ni dominée au terme d'une itération, et S_1 une autre contrainte quelconque dans la même formulation de PL. Par définition, il existe au moins une variable de S_0 qui ne soit pas présente dans S_1. Étant donné que cette variable ne sera pas supprimée dans les futures itérations, S_0 ne sera pas dominée par S_1 dans toutes les itérations suivantes.

Plus généralement, si une contrainte n'est ni dominée ni redondante à la $k^{\text{ème}}$ itération, elle ne l'est pas non plus à aucune des itérations suivantes de génération de colonnes. Conséquemment, elle doit être présente de façon inconditionnelle dans le PMRA à toutes les itération $k' \geq k$.

Il est important de noter que cette proposition repose sur le fait qu'aucune variable n'est supprimée de la formulation pendant le processus de résolution. Ceci permet de réduire significativement le nombre de comparaisons réalisées deux-à-deux entre les contraintes pour détecter les relation de redondance et de dominance. En d'autres termes, un nombre croissant de contraintes peuvent être laissées sans vérification de redondance ou dominance au fur et à mesure que le PMRA grandit.

L'algorithme 4.2 décrit la procédure d'agrégation. Un ensemble de contraintes $\mathcal{S}$ à agréger est donné en entrée. L'ensemble $\mathcal{A}$ de contraintes agrégées est initialement vide (ligne 1). Chaque contrainte candidate S de $\mathcal{S}$ (définie à la ligne 4) est ajoutée à $\mathcal{A}$ à condition qu'elle ne crée pas de relation de dominance ou de redondance. Si S est déjà connue pour satisfaire cette condition depuis une itération précédente, elle est ajoutée à la ligne 7 du fait de la proposition que nous avons formulée au-dessus. Comme les contraintes candidates sont triées par taille décroissante, la recherche de contraintes redondantes ou dominées

Entrées : $\mathcal{S}$: ensemble de contraintes à agréger.

Données : $\mathcal{N}$: ensemble de contraintes connues comme non redondantes/dominées.

$\mathcal{A} \leftarrow \emptyset$;
Réordonner $\mathcal{S}$ par taille décroissante de contraintes ;
pour $k \leftarrow 1$ **a** $|\mathcal{S}|$ **faire**
    $S \leftarrow$ `element(`$\mathcal{S}$, k`)` ; `// Prendre le` $k^{\text{ème}}$ `élément de` $\mathcal{S}$
    $unique_S \leftarrow$ **vrai** ;
    si $S \in \mathcal{N}$ **alors**
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{S\}$
    sinon
        si $\exists S^r \in \mathcal{A}$, $S^r = S$ **alors**
            $unique_S \leftarrow$ **faux** ;
            $unique_{S^r} \leftarrow$ **faux** ;
        sinon
            si $\nexists S^d \in \mathcal{A}$, $S \subset S^d$ **alors** $\mathcal{A} \leftarrow \mathcal{A} \cup \{S\}$;
        fin
    fin
fin
pour chaque $S \in \mathcal{A}$ **faire**
    si $unique_S$ **alors** $\mathcal{N} \leftarrow \mathcal{N} \cup \{S\}$;
fin

Sorties : $\mathcal{A}$: ensemble agrégé.

Algorithme 4.2: Algorithme d'agrégation de contraintes.

peut être limitée à $\mathcal{A}$ à la place de $\mathcal{S}$. Si une telle contrainte n'est pas trouvée, S est ajoutée à $\mathcal{A}$ à la ligne 13. Les contraintes de $\mathcal{A}$ ne possédant aucune contrainte redondante sont finalement marquées comme étant ni redondantes ni dominées à la ligne 18.

La recherche de redondance ou de dominance réalisée à la ligne 13 afin de déterminer s'il faut insérer la contrainte S dans le PMRA tire partie d'éléments spécifiques au problème dans le but d'accélérer son exécution. Ces éléments servent à orienter la recherche de façon heuristique vers des ensembles de contraintes au sein desquels il est davantage probable de trouver une contrainte redondante avec S ou dominant celle-ci :

- suivant le raisonnement selon lequel l'ensemble des chemins utilisant une même zone varie probablement très peu pour deux périodes consécutives, et si la contrainte S est une contrainte de ressource, la recherche est en premier lieu effectuée sur l'ensemble des contraintes de ressource associées à la même zone, pour les autres périodes de l'horizon temporel ;
- de façon symétrique, la recherche est effectuée ensuite sur les contraintes de ressource associées à la même période temporelle, mais sur l'ensemble des autres zones de l'infrastructure.

Si ces deux heuristiques de recherche prioritaire ne parviennent pas à trouver de contrainte redondante ou dominant S, la recherche est ensuite effectuée normalement sur le reste des contraintes de $\mathcal{A}$. Un ensemble d'autres techniques mineures, décrites par Merel et al. [50, 54], sont mises en place pour réduire davantage le temps de calcul nécessaire à la recherche de contraintes redondantes ou dominantes, telles que l'utilisation de la possibilité d'ordonner les variables au sein des contraintes ou encore la rapidité de certains appels de fonctions système.

4.3.5 Désagrégation de la solution duale pour le calcul des coûts réduits

La solution optimale au PMRA est aussi optimale pour le PMR non agrégé, et leur valeur de fonction-objectif à l'optimal coïncident. Ceci est donc valable également pour leurs formulations duales respectives. Toutefois, certaines contraintes primales étant absentes du PMRA, la valeur de leur variable duale correspondante n'est pas disponible à l'issue sa résolution. Ceci pose un problème pour résoudre le sous-problème, puisque la connaissance d'une solution duale au PMR complet est nécessaire pour calculer les coûts réduits.

Il est donc nécessaire de calculer une solution optimale au dual du PMR à partir de la solution duale partielle connue, issue des contraintes primales conservées dans le PMRA. Ce processus est nommé sous le terme de désagrégation.

Une première approche simple consiste à affecter la valeur 0 à toute variable duale associée à une contrainte du PMR non présente dans le PMRA. Il est aisé de vérifier qu'une telle solution ne modifie pas l'objectif du problème dual exprimé par la formulation 4.1, et que la faisabilité de la solution duale est conservée.

Cette solution n'est toutefois pas satisfaisante, notamment car deux contraintes identiques sont évidemment sujettes à la même saturation dans la solution primale. Toutefois, si l'une de ces deux contraintes figure dans le PMRA et que l'autre ni figure pas, la formule du coût réduit 4.2 favorise par la suite les chemins utilisant les ressources correspondant à celle qui a été supprimée. Ceci est un contre-sens au vu de l'utilisation des ressources représentées par ces contraintes : il n'y a aucune raison de favoriser l'une ou l'autre de ces ressources alors que leur saturation à l'issue de la résolution du PMRA est la même. Nous devons donc répartir l'occupation dans les contraintes redondantes à travers les variables duales qui leur sont associées.

Considérons $i \in 1 \,..\, (|T| + |U|)$ un indice quelconque de contrainte. Notons y_i sa variable duale correspondante. Nous cherchons à attribuer une valeur y_i^* à l'ensemble de ces variables duales de façon

à créer une solution optimale au dual du PMR qui effectue la répartition décrite ci-dessus. Plusieurs cas de figure sont possibles :

- si y_i figure dans le dual du PMRA (car sa contrainte primale correspondante n'a pas été supprimée), notons y_i^A sa valeur dans la solution optimale duale du PMRA, considérons $\mathscr{R}_i$ l'ensemble contenant i et tous les indices de contraintes redondantes avec i puis nous appliquons la formule suivante :
$$y_i^* = \frac{y_i^A}{|\mathscr{R}_i|} \,, \forall i \in \mathscr{R}_i \,; \tag{4.6}$$
- si y_i ne figure pas dans le dual du PMRA mais que sa contrainte correspondante est redondante avec une contrainte figurant dans le PMRA, la valeur de y_i^* est définie grâce au cas précédent ;
- enfin, si y_i ne figure pas dans le dual du PMRA car sa contrainte correspondante est dominée au sens strict, y_i^* est fixée à la valeur 0.

Il est aisé de vérifier que la valeur de l'objectif n'est pas modifiée et que les contraintes duales sont respectées par la solution $(y_i^*)_{i\in\{1\ ,...,\ (|T|+|U|)\}}$. Nous avons donc une solution optimale pour le dual du PMR et une répartition équitable des valeurs duales au sein de chaque ensemble de contraintes redondantes, ce qui permet finalement un calcul satisfaisant des coûts réduits.

4.4 Hybridation avec la métaheuristique

Nous avons conçu une procédure de génération de colonnes, dont le sous-problème est résolu par une méthode adaptée, à laquelle nous avons ajouté une méthode d'agrégation dynamique de contraintes. La génération de colonnes fournit une solution optimale à la relaxation continue du SPP. Afin de résoudre le problème de saturation, il est nécessaire de calculer une solution entière de bonne qualité, sinon optimale, à la formulation SPP.

Outre la valeur optimale de la fonction-objectif, la génération de colonnes fournit un sous-ensemble restreint de variables qui à lui seul permet l'obtention de la solution optimale continue. Nous avons également vu qu'ACO était une solution pertinente pour résoudre le SPP pour des tailles raisonnables. L'idée sous-jacente à l'hybridation est donc d'utiliser le problème restreint aux variables fournies par la génération de colonnes comme formulation résolue par ACO.

Il s'agit donc d'une hybridation entre une technique issue des méthodes de résolution exacte dans le cadre de la PL avec une métaheuristique. De nombreuses approches hybridant ces deux types de méthodes sont présentes dans la littérature. Nous en effectuons donc une présentation générale. Nous nous intéressons ensuite plus particulièrement aux hybridations réalisées autour de la génération de colonnes, et formalisons enfin l'interaction réalisée entre notre procédure de génération de colonnes et ACO pour le SPP. L'hybridation présentée ici est également détaillée par Merel et al. [51].

4.4.1 Principe de l'hybridation

Les métaheuristiques et les algorithmes exacts utilisés dans le cadre de résolution de problème de PLNE possèdent des avantages qui leurs sont propres et qui peuvent se révéler complémentaires, comme cela est mis en avant par Dumitrescu et Stützle [25]. Les méthodes de résolution associées à la PL et à la PLNE fournissent souvent des bornes qui peuvent être de bonne qualité, à travers la résolution de la relaxation continue et d'éventuelles améliorations telles que la génération d'inégalités valides. Par exemple, Puchinger et al. [61] proposent un algorithme mémétique qui exploite les solutions entières trouvées par une procédure de *branch-and-cut* exécutée simultanément, ainsi que les solutions duale fournies par la résolution de sa relaxation continue. D'un autre côté, les métaheuristiques sont souvent

considérées comme un moyen pertinent de trouver des solutions de bonne qualité tout en maintenant un temps de calcul réduit. Elles ne sont cependant pas conçues pour prouver l'optimalité d'une solution. Les deux classes de méthodes peuvent donc se révéler complémentaires.

Les méthodes hybrides peuvent être classées selon la manière dont interagissent l'algorithme exact et la métaheuristique. Puchinger [60] suggère une classification entre les hybridations collaboratives et intégratives. Une hybridation collaborative signifie que l'algorithme exact et la métaheuristique sont exécutées séparément, que ce soit séquentiellement ou en parallèle. Les combinaisons intégratives consistent à incorporer un algorithme exact au sein d'une métaheuristique, ou vice-versa.

Un large aperçu d'exemples d'hybridations est proposé par Blum et al. [6], tel que l'hybridation d'une algorithme ACO avec des techniques de programmation par contraintes. Les auteurs donnent également des exemples supplémentaires d'interaction entre les méthodes par séparation et évaluation et des métaheuristiques.

4.4.2 Les hybridations liées à la génération de colonnes

Dans le cadre général de la génération de colonnes, résoudre le sous-problème en temps raisonnable est un défi essentiel pour obtenir un algorithme global possédant des performances correctes. Il n'est toutefois pas nécessairement aisé de le résoudre.

De ce fait, des hybridations intégratives ont été proposées au sein desquelles la métaheuristique est l'algorithme esclave d'un algorithme de génération de colonnes, et est responsable de la recherche de nouvelles colonnes. De telles approches incluent le travail de Filho et Lorena [29], proposant un algorithme génétique, ainsi que le travail de Puchinger [60] qui propose un algorithme évolutionnaire. Clements et al. [13] proposent une hybridation collaborative séquentielle au sein de laquelle une heuristique est utilisée pour générer une formulation restreinte d'un problème d'ordonnancement de production, formulation résolue par la suite par une procédure basée sur la génération de colonnes. Plus récemment, Sadki-Fenzar [67] propose des hybridations intégratives sequentielles entre la génération de colonnes et une heuristique gloutonne pour la résolution de problèmes de couverture en nombres entiers.

Bien que ces approches possèdent des différences, l'algorithme approché est chargé dans chaque cas de générer les variables pour la méthode exacte. Les rôles sont donc inversés par rapport à l'approche que nous adoptons, puisque nous souhaitons utiliser la génération de colonnes pour fournir une formulation à la métaheuristique ACO.

4.4.3 Exploitation de la génération de colonnes par la métaheuristique

L'hybridation que nous proposons est simple et rentre dans la catégorie des hybridation collaboratives séquentielles, telles que classifiées par Puchinger [60]. La génération de colonnes est utilisée pour fournir les informations suivantes à ACO :

- la valeur optimale de la fonction-objectif pour la relaxation continue ;
- l'ensemble des variables générées au cours de l'ensemble des itérations de l'algorithme ;
- l'affectation d'une valeur fractionnaire à chacune de ces variables.

La combinaison de deux algorithmes consiste simplement à utiliser deux de ces informations pour initialiser ACO. D'une part, à la place de la formulation entière, un SPP restreint aux variables générées est fourni en entrée à ACO. De plus, nous avons vu qu'ACO utilise une matrice de phéromones afin de guider le processus de résolution. Ceci nous permet, plutôt que d'initialiser toutes les phéromones à une valeur par défaut de 1, d'utiliser la valeur des variables dans la solution optimale continue comme

vecteur d'initialisation des phéromones. Par ce biais, les variables dont la valeur est plus élevée dans la solution optimale continue seront privilégiées dans la recherche de solution par ACO.

Enfin, la solution optimale continue est une borne supérieure pour la formulation SPP. À ce titre, elle peut être un outil d'estimation de la qualité de la solution donnée par ACO.

4.5 Résumé de la méthode

Nous avons présenté les trois éléments de la méthode mise en place :

1. la procédure de génération de colonnes, dont le sous-problème spécifique à nos données est résolu par un algorithme adapté, et pour laquelle des critères adaptés de sélection des colonnes à coût réduit positif sont mis en place ;
2. l'agrégation dynamique des contraintes redondantes et dominées, qui se matérialise par une suppression des ces contraintes du PMR par le biais d'algorithmes exploitant également certaines données du problème ;
3. l'interaction avec ACO, à travers le fait de fournir à celle-ci uniquement l'ensemble des colonnes générées et d'initialiser sa matrice de phéromones avec la solution optimale continue.

L'algorithme 4.3 propose une vision d'ensemble de la méthode mise en place pour résoudre le problème de saturation modélisé par un SPP. Considérant comme données d'entrée l'ensemble des variables du problème, c'est-à-dire la définition de tous les chemins, il fournit en sortie une solution entière réalisable au problème de SPP tel que défini par la formulation 3.20, une solution optimale continue ainsi que la borne supérieure issue de cette solution. Pour davantage de clarté, cette présentation de l'algorithme sépare volontairement l'ensemble des colonnes de l'ensemble de ses lignes, qui sont respectivement notées $\mathcal{C}$ et $\mathcal{L}$. Il est aisé de distinguer au sein de cet algorithme les trois composantes caractéristiques que nous avons présentées.

La procédure de génération de colonnes est exécutée entre les lignes 8 et 27, et est arrêtée si l'ensemble de colonnes à coût réduit positif est vide (ligne 24) ou si les bornes inférieure et supérieure sont égales (ligne 15). Une formulation initiale de petite taille lui est fournie aux lignes 1 et 2 en considérant uniquement les variables associées à un chemin correspondant à un itinéraire emprunté avec un délai nul. L'algorithme du simplexe, fournissant une solution optimale continue primale et duale pour le PMRA, est exécuté à la ligne 9. Le calcul de l'ensemble des coûts réduits est réalisé à la ligne 11 conformément à l'algorithme 4.1, permettant de déterminer l'ensemble des colonnes candidates pouvant être ajoutées dans le PMRA (ligne 17) puis, si celui-ci n'est pas vide, la sélection puis l'insertion de certaines d'entre elles, selon la stratégie choisie, aux lignes 19, 20 et 21.

L'agrégation dynamique des contraintes, conformément à ce qui est présenté par l'algorithme 4.2, est en premier lieu réalisée à la phase d'initialisation (ligne 3). Elle est également exécutée à la ligne 22 après l'insertion de nouvelles colonnes, puisque celles-ci peuvent avoir nécessité l'ajout dans le PMRA de nouvelles contraintes par destruction de certaines relations de redondance ou dominance. Enfin, la désagrégation, dont le principe est donné par la formule 4.6, est réalisée à la ligne 10 et est suivie par le calcul des coûts réduits (ligne 11).

Pour terminer, l'hybridation avec ACO se à la ligne 29. Ceci permet finalement d'obtenir une solution entière réalisable, qui correspond à une solution au problème de saturation.

Cet algorithme fait l'objet d'une implémentation et d'une intégration au sein de RECIFE PC, une plateforme logicielle dédiée au calcul de capacité d'infrastructures ferroviaires à échelle microscopique. Il a été testé sur l'ensemble des 45 instances présentées au chapitre 2. Le chapitre suivant présente donc

l'ensemble des résultats, comparés à ceux obtenus en utilisant la méthode ayant, à notre connaissance, donné les meilleurs résultats jusqu'ici, à savoir ACO. Un aperçu de RECIFE PC et en particulier du composant dédié à la résolution du problème de saturation est également proposé.

Entrées : $\mathcal{C}$: ensemble des couples (t, c) valides ($\forall t \in T, c \in C_t$), $\mathcal{L}$: ensemble des contraintes.

$\mathcal{C}_{pmra} \leftarrow$ variables de $\mathcal{C}$ correspondant à un délai nul pour tous les trains et itinéraires ;

$\mathcal{L}_{pmra} \leftarrow$ contraintes de $\mathcal{L}$ couvertes par les variables de $\mathcal{C}_{pmra}$;

$\mathcal{L}_{pmra} \leftarrow$ `agréger(`$\mathcal{L}_{pmra}$`)` ;

$(\overline{x}_{t,c})_{(t,c)\in\mathcal{C}} \leftarrow (x^*_{t,c})_{(t,c)\in\mathcal{C}} \leftarrow (0,\ ...,\ 0)$;

$\overline{z} \leftarrow +\infty$;

$\underline{z} \leftarrow 0$;

$fin \leftarrow$ **faux** ;

tant que $fin =$ **faux faire**

  $(\{\overline{x}_{t,c}\}_{(t,c)\in\mathcal{C}_{pmra}}, \{\overline{v}_t\}_{t\in\mathcal{L}_{pmra}}, \{\overline{w}_u\}_{u\in\mathcal{L}_{pmra}}) \leftarrow$ `simplexe(`$\mathcal{C}_{pmra}$, $\mathcal{L}_{pmra}$`)` ;

  $(\overline{v}, \overline{w}) \leftarrow$ `désagréger(`$\mathcal{L}_{pmra}$`)` ;

  $(\pi_{t,c})_{(t,c)\in\mathcal{C}} \leftarrow$ `coutsréduits(`$\overline{v}$, $\overline{w}$`)` ;

  $\underline{z} \leftarrow \sum_{(t,c)\in\mathcal{C}_{pmra}} (\rho_{t,c}\overline{x}_{t,c})$;

  $\overline{z} \leftarrow$ `borneduale(`$\overline{v}$, $\overline{w}$, π`)` ;

  si $\overline{z} = \underline{z}$ **alors**

    $fin \leftarrow$ **vrai** ;

  sinon

    $\mathcal{C}_{candidats} \leftarrow \{(t, c) \in \mathcal{C},\ (t, c) \notin \mathcal{C}_{pmra},\ \pi_{t,c} > 0\}$;

    si $\mathcal{C}_{candidats} \neq \emptyset$ **alors**

      $\mathcal{C}_{sel} \leftarrow$ `sélection(`$\mathcal{C}_{candidats}$`)` ;

      $\mathcal{C}_{pmra} \leftarrow \mathcal{C}_{pmra} \cup \mathcal{C}_{sel}$;

      $\mathcal{L}_{pmra} \leftarrow \mathcal{L}_{pmra} \cup$ {lignes couvertes par $\mathcal{C}_{sel}$} ;

      $\mathcal{L}_{pmra} \leftarrow$ `agréger(`$\mathcal{L}_{pmra}$`)` ;

    sinon

      $fin \leftarrow$ **vrai** ;

    fin

  fin

fin

$\overline{z}^* \leftarrow \lfloor \underline{z} \rfloor$;

$(x^*_{t,c})_{(t,c)\in\mathcal{C}_{pmra}} \leftarrow$ `aco(`$\mathcal{C}_{pmra}$, $\mathcal{L}_{pmra}$, $\overline{x}$`)` ;

$(x^*_{t,c})_{(t,c)\notin\mathcal{C}_{pmra}} \leftarrow 0$;

Sorties : solution entière réalisable $(x^*_{t,c})_{(t,c)\in\mathcal{C}}$, solution optimale continue $(\overline{x}_{t,c})_{(t,c)\in\mathcal{C}}$, borne supérieure $\overline{z}^*$.

Algorithme 4.3: Résolution du SPP modélisant le problème de saturation.

CHAPITRE 5

Résultats expérimentaux et plateforme logicielle RECIFE PC

L'ensemble de la méthode de résolution que nous proposons pour le problème de saturation, et en particulier pour répondre aux difficultés posées par nos instances, a été présentée. Il s'agit d'une combinaison d'une procédure de génération de colonnes et de la métaheuristique ACO. La génération de colonnes joue un rôle de pré-traitement visant à sélectionner un sous-ensemble réduit de variables à fournir en entrée à ACO. Diverses améliorations intégrées à la génération de colonnes ont en outre été présentées, la principale d'entre elles résidant dans l'agrégation dynamique de contraintes.

Ce chapitre présente les résultats expérimentaux de ce nouvel algorithme avec les données auxquelles nous sommes confrontées. ACO ayant à notre connaissance donné les meilleurs résultats pour résoudre des formulations SPP du problème de saturation jusqu'ici, nous l'utilisons comme point de comparaison. L'algorithme que nous proposons est comparé à ACO selon trois axes essentiels : le temps de calcul, la qualité des solutions fournies (en terme de valeur de l'objectif) et enfin l'utilisation mémoire. Nous y voyons notamment une amélioration significative des temps de calcul ainsi que de l'utilisation mémoire, et des qualités de solutions comparables.

L'algorithme proposé au chapitre précédent est en outre une méthode composite : ses deux composants sont ACO et la génération de colonnes, cette dernière étant elle-même composée d'un ensemble de contributions individuelles, telles que l'agrégation de contraintes. Nous présentons donc également un aperçu des effets de ces contributions sur le processus global de résolution.

Du fait des résultats positifs obtenus, la contribution algorithmique est implémentée au sein de la plateforme logicielle RECIFE PC, dédiée aux études de capacité d'infrastructures à échelle microscopique. La plateforme logicielle contient un ensemble de modules destinés à l'estimation de capacité, qui sont passés succinctement en revue. Le module de saturation est présenté plus en détails, et nous développons un certain nombre d'améliorations techniques apportées lors de l'intégration du nouvel algorithme.

Nous terminons par un bilan des résultats ainsi qu'une observation critique de la structure des solutions retournées par le module de saturation, caractérisée par un déséquilibre significatif entre les types de trains routés sur l'infrastructure. L'ensemble des raisons menant à ce déséquilibre est finalement explicité, permettant ainsi d'ouvrir la réflexion à une solution palliant à ce problème.

5.1 Résultats expérimentaux comparatifs

5.1.1 Protocole d'expérimentation

Nous comparons notre méthode à la métaheuristique ACO seule, exécutée sur les instances que nous avons présentées. Cette comparaison est justifiée par le fait qu'ACO est jusqu'ici l'algorithme qui a montré les meilleures performances pour les formulations importantes de SPP, et plus particulièrement pour résoudre le problème de saturation. Conformément aux points problématiques que nous avons mis en avant concernant ACO sur les instances considérées, nous effectuons les comparaisons sur les axes suivants :

- temps de calcul CPU nécessaire à l'obtention d'une solution entière ;
- qualité de solution, c'est-à-dire la valeur de la fonction-objectif ;
- utilisation mémoire.

Le temps de résolution est bien entendu d'un point crucial de comparaison, puisque nous avons vu qu'ACO seule nécessitait une durée pouvant être considérée comme particulièrement handicapante dans le cadre d'un logiciel d'aide à la décision. Pour nos expérimentations, le temps d'exécution des algorithmes est limité à 100 000 secondes.

La mesure de qualité de solution est aussi un résultat important à comparer, puisque la précision de l'estimation de la saturation dépend notamment de la qualité des solutions. Nous devons plus précisément vérifier que le fait d'exécuter ACO sur un nombre de colonnes restreint n'est pas nuisible à la qualité des solutions. Cette vérification nous permet conséquemment d'estimer la pertinence des variables choisies par la génération de colonnes du point de vue de la résolution en nombre entiers. Nous donnons également la valeur de la borne supérieure calculée par la génération de colonnes et examinons dans quelle mesure celle-ci permet de tirer des conclusions, sur la qualité de la relaxation continue et de la solution entière.

La consommation mémoire est également un point important, puisque nous avons vu que l'exécution d'ACO pouvait être, dans le pire des cas, interrompue du fait d'une consommation trop importante. Ici, nous souhaitons que la génération de colonnes, par la sélection d'un sous-ensemble de colonnes de taille limitée, contribue à réduire la consommation mémoire.

Afin d'éviter les confusions, nous utiliserons la terminologie suivante dans cette section relative à la comparaison des résultats :

- l'algorithme ACO seul exécuté pour résoudre une instance du problème de saturation sera désigné par « ACO » ;
- l'algorithme que nous proposons, combinant la génération de colonnes et ACO, sera désigné par « GC-ACO ».

Pour les deux méthodes, les temps de résolution sont une moyenne calculée sur 10 exécutions sur chaque instance. Bien que l'algorithme ACO ne soit pas déterministe du fait qu'il fasse appel à une recherche aléatoire de solutions, nous avons mesuré des coefficients de variation n'excédant pas 3% en ce qui concerne sont temps de résolution, rendant en conséquence la valeur moyenne pertinente.

La résolution du PMRA par le simplexe au sein de la génération de colonnes est réalisée par la bibliothèque de PL libre Coin-OR Clp, dans sa version 1.09. Le simplexe qu'elle implémente fournit l'ensemble des fonctionnalités dont nous avons supposé l'existence aux chapitres 3 et 4 (solution duale, coût réduit des variables de la formulation, etc.). Tous les algorithmes sont exécutés sur un système Linux, sur une machine équipée d'un microprocesseur Intel Core2 Duo cadencé à 2.60GHz et possédant 2Gio de mémoire vive.

5.1.2 Temps de calcul

Le tableau 5.1 reprend le nombre de variables (noté n) de chaque instance et présente le temps de résolution de chaque méthode pour l'ensemble des instances. Les temps de résolution d'ACO et de GC-ACO sont dans les colonnes respectives « aco » et « cgaco ». Lorsqu'une méthode présente un meilleur temps de résolution pour une instance donnée, son temps de résolution est en gras. L'indication *temps* signifie que l'algorithme n'a pas retourné de solution dans le temps imparti (100 000 secondes), et l'indication *mem.* signifie que l'algorithme a été interrompu par le système du fait d'une utilisation de la mémoire vive trop importante.

La comparaison des temps de résolution montre une nette tendance en faveur de GC-ACO, la comparaison donnant globalement les résultats suivants :

- 3 instances, figurant parmi les 5 plus petites, sont résolues plus rapidement pas ACO, GC-ACO mettant dans le pire des cas (pour l'instance la plus petite) 65% de temps en plus ;
- 37 instances sont résolues plus rapidement par GC-ACO ;
- 5 instances sont résolues par GC-ACO dans le temps imparti sans l'être par ACO.

Cette tendance est d'autant plus nette avec l'augmentation de la taille des formulations. Jusqu'à environ 2 000 variables, le gain de temps reste raisonnable notamment du fait qu'ACO ne nécessite pas un temps prohibitif. Au-delà de 5 000 variables, le gain est très net et varie entre 60% et 90%. L'amélioration est d'autant plus marquée que le nombre de variables est élevé du fait d'une valeur de granularité faible, plus précisément telle que $g \leq 5$. En terme de temps de résolution, la tendance est donc très nettement en faveur de GC-ACO.

Les propriétés des instances affectent les deux algorithmes d'une façon différente. ACO est essentiellement affecté par le nombre de variables, bien que pour deux instances possédant le même nombre de variables, une valeur plus grande pour $\overline{d}$ semble faire augmenter légèrement le temps de résolution. Au contraire, les performances de GC-ACO sont plus sensibles envers les paramètres ferroviaires. En effet, l'augmentation de la taille de l'instance par une augmentation de $\overline{d}$ entraîne une augmentation significative du temps de résolution, alors que la diminution de g (entraînant également une augmentation du nombre de variables) a un impact plus limité. Cette observation est bien illustrée par la colonne correspondant aux instances à 66 trains, pour $\overline{d} = 30$ et $g = 1$ d'une part et pour $\overline{d} = 60$ et $g = 2$ d'autre part : les deux instances possèdent 10 943 variables, mais il y a un facteur multiplicateur égal à 3.4 entre leurs temps de résolution respectifs avec GC-ACO. Enfin, l'augmentation du nombre de trains pour une taille comparable d'instance ne montre pas d'impact clairement identifiable sur le temps de résolution.

Ces différences de comportement s'expliquent donc nécessairement par la variation de la structure de la matrice des contraintes à nombre de lignes et de colonnes très proches. Nous avons vu qu'ACO repose sur une matrice NPP décrivant l'ensemble des conflits deux-à-deux, matrice qui est donc faiblement dépendante de la structure des contraintes dans la matrice SPP initiale, et notamment de sa qualité de relaxation continue. En revanche, la procédure de génération de colonnes exécutée dans GC-ACO utilise directement la matrice SPP, permettant d'identifier plusieurs facteurs susceptibles d'impacter la résolution :

- un nombre de variables augmentant du fait d'une augmentation de $\overline{d}$ plutôt que d'une diminution de g peut entraîner moins d'apparition de contraintes redondantes, réduisant ainsi l'impact de l'algorithme d'agrégation et augmentant le temps de résolution du PMRA en conséquence ;
- une qualité de relaxation continue plus faible implique une valeur de solution optimale continue plus haute, pouvant nécessiter davantage d'itérations de génération de colonnes et un temps de résolution plus important pour atteindre l'optimal ;

– davantage d'itérations de la génération de colonnes peut également mener à un temps d'exécution plus long de l'algorithme ACO suivant la procédure de génération de colonnes, du fait qu'un nombre plus important de variables doivent être prises en compte.

Nous avons donc, d'une façon globale, obtenu une réduction nette du temps du résolution, réduction s'exprimant de façon plus marquée sur les instances plus difficiles, qui sont justement celles sur lesquelles le temps de ACO est particulièrement handicapant.

5.1.3 Qualité de la solution entière et de la borne supérieure

Les comparaisons en termes de fonction-objectif sont montrées par le tableau 5.2, qui donne les meilleures valeurs de fonction-objectif trouvées par chacun des algorithmes ainsi que la valeur de borne supérieure donnée par la résolution de la relaxation continue. Il est possible d'y observer ceci :

– les deux méthodes se montrent capables de retourner la même valeur de fonction-objectif dans 31 cas, c'est-à-dire que la meilleure solution retournée par les deux algorithmes sont égales ;
– GC-ACO se montre moins performant qu'ACO dans 8 cas, parmi lesquels l'écart est d'une unité dans 6 cas et de deux unités dans 2 cas ;
– GC-ACO se montre plus performant qu'ACO dans 2 cas, où l'écart est d'une unité ;
– dans 4 cas correspondant aux 4 instances contenant le plus grand nombre de variables, ACO ne retourne aucune solution en temps imparti, soit du fait d'un excès de consommation mémoire, soit du dépassement de la limite des 100 000 secondes, alors que GC-ACO retourne effectivement une solution.

Pour une majorité des instances, la valeur d'objectif trouvée est donc la même pour les deux algorithmes ou bien GC-ACO surpasse ACO. Ce constat montre que l'adjonction de la génération de colonnes comme pré-traitement à la métaheuristique dans le but d'identifier un sous-ensemble de variables prometteuses est une approche pertinente. Elle permet donc, au vu des d'affirmer que le gain en terme de temps de calcul ne se fait pas au prix d'un sacrifice significatif en terme de fonction-objectif. L'impact ne semble toutefois pas inexistant non plus, au vu des 8 instances en faveur de ACO.

Cette application montre en outre, dans le contexte des hybridations de méthodes exactes et approchées, que la génération de colonnes peut être un pré-traitement pertinent à une métaheuristique ACO pour fournir à celle-ci un ensemble de variables à même de conduire à une solution entière de bonne qualité.

En ce qui concerne la relaxation continue, l'écart entre la valeur de la solution optimale continue et la valeur de la solution entière est hétérogène. Il est toutefois possible d'effectuer plusieurs observations. Pour les plus petites instances, l'écart n'est que d'une unité, mettant ainsi en valeur à la fois la bonne qualité de solution entière obtenue ainsi que la bonne qualité de la relaxation continue. Ceci permet notamment de venir étayer les mesures de Gandibleux et al. [30, 31, 32] montrant la stabilité d'ACO et sa capacité à trouver des solutions de bonne qualité.

L'écart se dégrade toutefois avec l'augmentation de la taille des instances, et plus particulièrement pour de grandes valeurs de $|T|$ et $\overline{d}$. Cette observation ne permet pas en elle-même de conclure formellement sur la cause d'un tel écart, à savoir si cela est causé par une incapacité d'ACO à trouver une meilleure solution ou par une dégradation de la borne supérieure. Il paraît toutefois fort probable que cette seconde raison en soit la cause, conjecture que nous pouvons justifier par les points suivants :

– la régularité de la métaheuristique au niveau de la qualité des solutions a été montrée empiriquement, à la fois par Gandibleux et al. [30, 31, 32] sur des instances du problème de saturation et par les petites instances dont nous disposons ici, rendant peu vraisemblable une perte de qualité ponctuelle pour certaines instances particulières ;

$\overline{d}$	g	$\lvert T\rvert$								
		53			66			104		
		n	aco	cgaco	n	aco	cgaco	n	aco	cgaco
	15	876	**17**	28	1059	**26**	44	1737	**86**	103
	10	1168	37	**33**	1412	56	**52**	2316	181	**120**
30	5	2044	159	**51**	2471	250	**76**	4053	922	**186**
	2	4672	1546	**66**	5648	2355	**120**	9264	7304	**289**
	1	9052	7371	**132**	10943	11435	**207**	17949	35539	**481**
	15	1460	81	**49**	1765	127	**72**	2895	420	**188**
	10	2044	207	**69**	2471	317	**116**	4053	1104	**319**
60	5	3796	1147	**156**	4589	1789	**319**	7527	5879	**909**
	2	9052	10152	**436**	10943	14804	**703**	17949	45387	**3373**
	1	17812	41821	**1067**	21533	60965	**1602**	35319	*mem.*	**11011**
	15	2044	226	**94**	2471	347	**137**	4053	1420	**432**
	10	2920	543	**168**	3530	829	**258**	5790	2784	**925**
90	5	5548	3179	**535**	6707	4954	**1081**	11001	15111	**4105**
	2	13432	21440	**2501**	16238	30534	**3876**	26634	*temps*	**26695**
	1	26572	*temps*	**8019**	32123	*temps*	**13215**	52689	*mem.*	**89753**

Table 5.1 – Nombre de variables et temps de résolution pour toutes les instances.

– qu'elle utilise la formulation complète ou la formulation restreinte à l'issue de la génération de colonnes, la métaheuristique retourne des valeurs de fonction-objectif très proches pour une instance donnée.

D'un point de vue ferroviaire, le tableau 5.2 montre enfin que pour une taille d'offre $|T|$ et un délai maximal $\overline{d}$ fixés, le nombre de train routés par l'algorithme varie au plus d'une unité avec la variation de g. Dans tous les cas, ce train supplémentaire n'est routé qu'à condition que $g \leq 10$. Ainsi, si le nombre de chemins disponibles augmente fortement lorsque g diminue, il semble que ceci n'ait en réalité que peu d'impact sur la mesure de saturation. L'augmentation du retard maximal a un impact sensiblement plus important sur la possibilité de router davantage de trains, comme montré par les résultats obtenus pour le $|T| = 66$, où l'écart atteint 3 unités.

5.1.4 Utilisation mémoire

Le stockage de la matrice d'exclusivité par la métaheuristique ACO peut être extrêmement consommateur de mémoire. En effet, une formulation de plus de 50 000 variables entraîne l'impossibilité d'exécuter l'algorithme, et ce par épuisement de la mémoire vive disponible, à savoir 2Gio. Ce phénomène est un problème critique, d'autant plus qu'il est susceptible de se produire sur des systèmes relativement récents. De plus, la matrice d'exclusivité étant stockée pendant l'ensemble du processus de résolution de la métaheuristique, cette quantité de mémoire est utilisée pendant une durée non négligeable.

Au contraire, GC-ACO évite la manipulation d'une telle matrice du fait de la sélection d'une petite proportion des variables. Ainsi, nos mesures ont montré que le pic d'utilisation le plus important en terme de consommation mémoire sur l'ensemble des instances était à 300Mio, intervenant pendant la résolution du PMRA par Coin-OR Clp. Par la suite, la résolution par la métaheuristique des formulations restreintes n'implique à aucun moment le dépassement de cette valeur pour les instances que nous utilisons.

Sur l'axe de l'utilisation de la mémoire, GC-ACO apporte donc une amélioration appréciable qui peut être bénéfique du point de vue de l'utilisabilité.

5.2 Impact des composants de l'algorithme

La section précédente a montré clairement la pertinence de l'algorithme GC-ACO dans son ensemble. La procédure de génération de colonnes et ses éléments constitutifs présentés au chapitre 4 constituent l'outil essentiel permettant ces gains. De ce fait, cette section fournit des données et éléments d'analyse supplémentaires visant à mieux cerner l'impact de ces éléments sur la performance de la procédure de génération de colonnes.

5.2.1 Répartition du temps de calcul

5.2.1.1 Partage du temps de calcul entre génération de colonnes et colonie de fourmis

Afin de mieux visualiser le comportement de GC-ACO, le tableau 5.3 présente la proportion du temps de résolution occupée par la procédure de génération de colonnes, le pourcentage restant étant bien entendu utilisé par la métaheuristique exécutée sur les variables générées.

La génération de colonnes occupe clairement la majeure partie du temps de calcul, nous permettant de tirer deux conclusions. D'une part, nous avons vu un gain de temps de résolution significatif pour les instances les plus difficiles, ce qui signifie que la procédure de génération de colonnes remplit correctement son rôle de pré-traitement pour la métaheuristique, permettant à cette dernière de conserver un

$\overline{d}$	g	$\lvert T\rvert$								
		53			66			104		
		borne	aco	cgaco	borne	aco	cgaco	borne	aco	cgaco
	15	28	27	27	37	34	34	54	53	53
	10	28	27	27	37	34	34	55	53	53
30	5	28	27	27	37	34	34	55	53	53
	2	28	27	27	37	34	34	55	53	53
	1	28	27	27	37	33	34	55	53	53
	15	31	28	28	41	36	36	61	54	54
	10	32	28	28	41	36	36	61	54	53
60	5	32	28	28	41	36	36	62	54	52
	2	32	28	28	42	36	35	62	54	52
	1	32	28	28	42	36	35	62	54	54
	15	33	28	28	42	36	36	63	54	54
	10	33	28	28	43	36	36	64	54	53
90	5	34	28	28	43	37	36	65	54	53
	2	34	27	28	44	37	37	65	-	54
	1	34	-	28	44	-	36		-	54

Table 5.2 – Meilleures solutions entières et bornes supérieures par relaxation continue.

$\overline{d}$	g	$\lvert T\rvert$								
		53			66			104		
		durée gc.	taux gén.	#it.	durée cg.	taux gén.	#it.	durée. cg.	taux gén.	#it.
	15	92%	57%	10	89%	59%	10	85%	58%	11
	10	96%	48%	11	96%	50%	11	82%	49%	12
30	5	86%	36%	15	81%	36%	14	74%	37%	15
	2	73%	21%	21	70%	22%	21	67%	21%	21
	1	68%	15%	32	68%	15%	28	64%	13%	25
	15	88%	44%	13	85%	47%	13	78%	48%	14
	10	84%	39%	16	79%	43%	17	76%	43%	19
60	5	78%	30%	23	75%	34%	28	76%	34%	26
	2	73%	20%	37	73%	20%	36	71%	24%	45
	1	72%	14%	54	72%	13%	50	73%	18%	69
	15	82%	44%	18	80%	45%	19	78%	46%	19
	10	80%	40%	24	76%	41%	23	77%	44%	26
90	5	80%	31%	34	76%	34%	40	81%	35%	39
	2	77%	22%	61	79%	22%	59	83%	27%	74
	1	79%	16%	90	82%	16%	89	85%	18%	112

Table 5.3 – Proportion de temps CPU passé dans la génération de colonnes, taux de colonnes générées et nombre d'itérations.

temps de résolution très court. D'autre part, cela met en avant le fait qu'il est essentiel de concevoir une procédure de génération de colonnes efficace. Dans notre cas, les améliorations apportées à la génération de colonnes telles que l'agrégation de contraintes et l'identification de stratégies d'insertion de colonnes jouent un rôle non négligeable.

Ces résultats confirment en outre qu'il est pertinent de profiter des données spécifiques au problème pour concevoir des algorithmes performants, approche déjà mise en valeur notamment par Zwaneveld et al. [78, 79] et Ryan et Foster [65]. La génération de colonnes ajoute donc à la métaheuristique une prise en compte de données supplémentaires bénéfique, conformément à ce qui est mis en avant par Dumitrescu et Stützle [25]. Ces données sont d'une part liées aux propriétés de la matrice de la formulation et d'autre part liées spécifiquement au problème de saturation.

5.2.1.2 Répartition du temps au sein de la génération de colonnes

Au sein de la procédure de génération de colonnes, nous pouvons également examiner plus précisément comment se répartit le temps entre les différentes composantes. Les graphes 5.1, 5.2 et 5.3 présentent, pour l'ensemble des instances, la proportion de temps de calcul passé respectivement à résoudre le PMRA, à résoudre le sous-problème et à agréger les contraintes. Le taux restant, qui n'est pas coloré sur les graphes, représente la proportion de temps passée à réaliser des opérations annexes comme le maintien des structures de données, représentant par exemple la matrice de la formulation pour Coin-OR ou encore les relations de dominance et de redondance entre contraintes. Tous ces taux sont exprimés en fonction du temps de calcul de la procédure de génération de colonnes uniquement. Les remarques suivantes peuvent être faites :

- pour les instances les plus petites (vérifiant $\overline{d} \in \{30, 60\}$ et $g \in \{10, 15\}$), environ la moitié du temps est passé à réaliser les opérations de maintien des structures de données plutôt qu'à exécuter les algorithmes de résolution à proprement parler ;
- le temps est équitablement réparti entre le simplexe, le sous-problème et l'agrégation dans environ la moitié des cas, correspondant aux instances les plus petites pour chacune des trois demandes ;
- lorsqu'un déséquilibre apparaît au niveau de la répartition, celui-ci est en défaveur de l'algorithme du simplexe, dont la durée cumulée devient nettement supérieure à la durée de chacun des deux autres composants.

Le premier point est explicable par le fait que la facilité des petites instances du point de vue mathématique rend les algorithmes de résolution aussi peu coûteux que les opérations annexes, coût étant alors principalement engendré par le simple et nécessaire parcours de certaines structures de données.

Le second et le troisième points montrent que l'accroissement du temps de résolution pris par la génération de colonnes est principalement dû à la résolution du PMRA, du fait de sa taille inévitablement croissante. En outre, la forte diminution de la proportion de temps passé à agréger les contraintes dans les cas où $g \in \{1, 2\}$ comporte une autre cause : la diminution de la granularité entraîne une augmentation des redondances entre contraintes, et plus particulièrement entre contraintes de ressource à zones identiques et à périodes temporelles proches. Les heuristiques de recherche prioritaire que nous avons décrites au chapitre 4 identifient en conséquence un grand nombre de relations de dominance ou redondance tout en effectuant un parcours très restreint, et donc peu coûteux, des contraintes. Cette explication est plus généralement confirmée par la diminution de la proportion de temps consommée par l'agrégation lorsque g diminue, à $|T|$ et $\overline{d}$ fixés. Un examen plus détaillé du comportent de l'agrégation au fur et à mesure d'une résolution de la génération de colonnes nous permet par la suite d'apporter des éclairages supplémentaires sur le rôle joué par celle-ci.

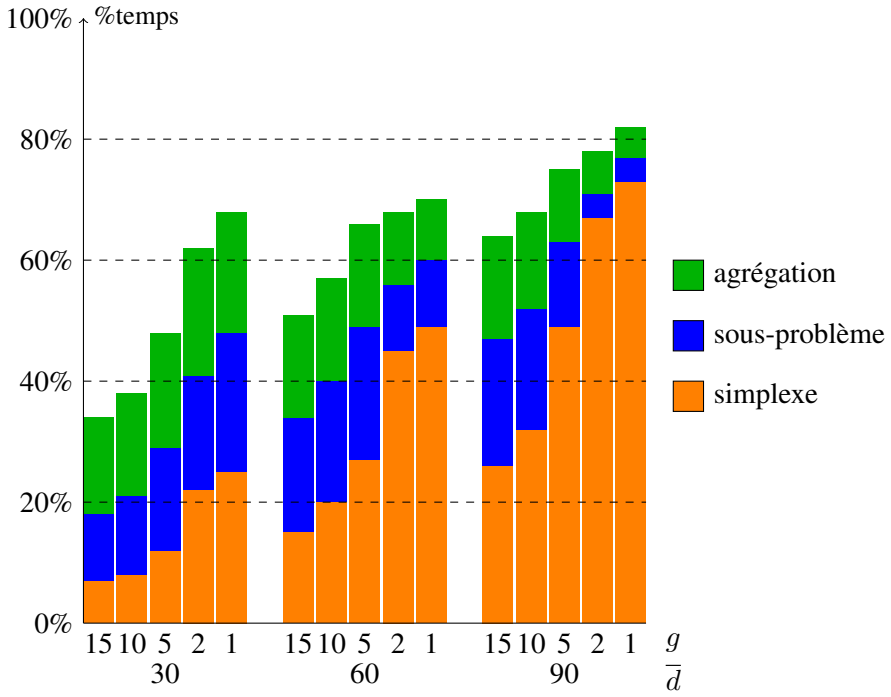

Figure 5.1 – Répartition du temps de calcul de la procédure de génération de colonnes (53 trains).

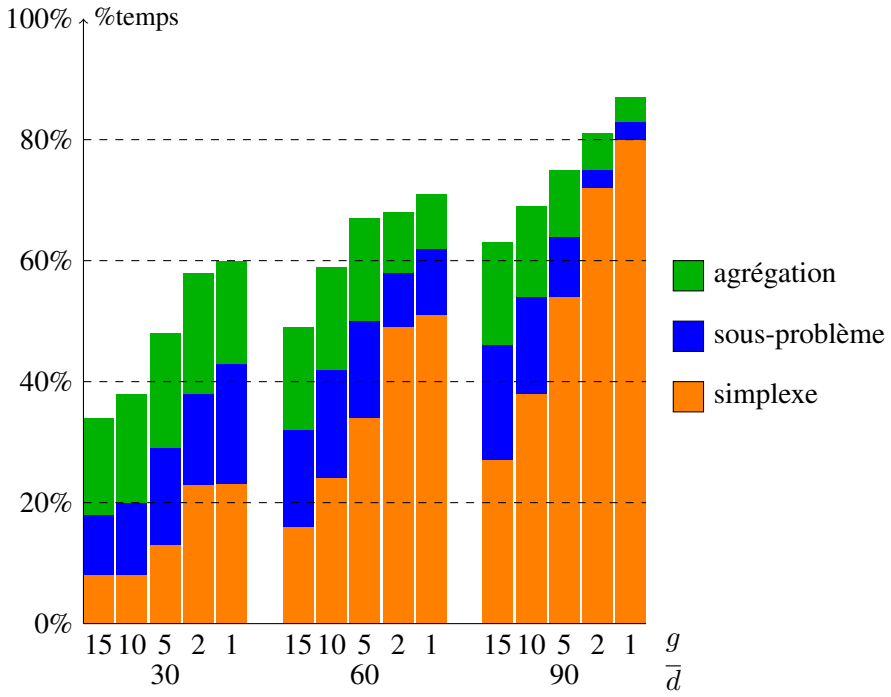

Figure 5.2 – Répartition du temps de calcul de la procédure de génération de colonnes (66 trains).

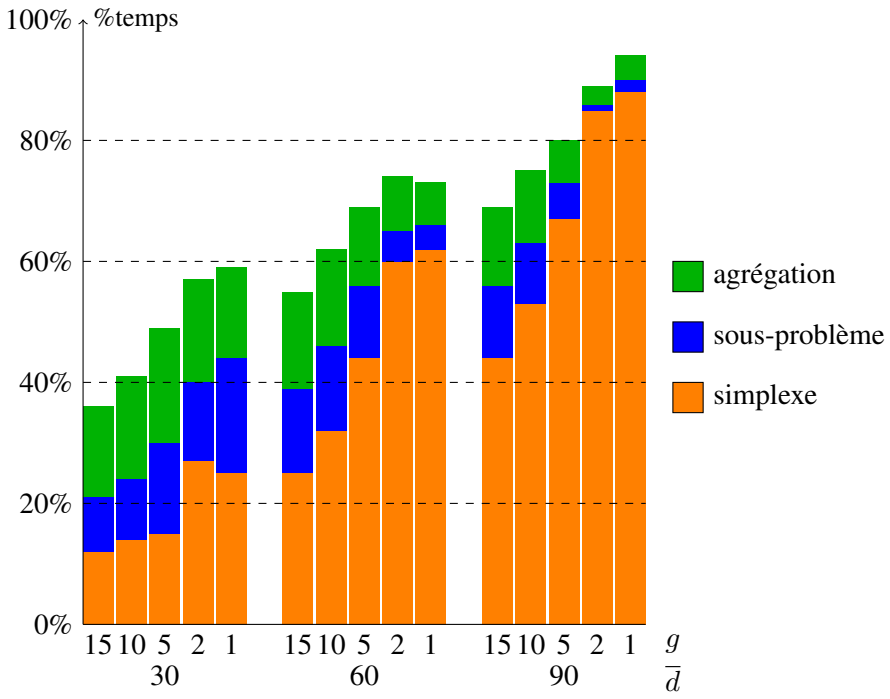

Figure 5.3 – Répartition du temps de calcul de la procédure de génération de colonnes (104 trains).

5.2.2 Impact de l'agrégation de contraintes sur la formulation

5.2.2.1 Nombre de contraintes à l'issue de la génération de colonnes

Le tableau 5.4 montre l'impact de l'agrégation dynamique de contraintes sur le nombre de lignes présentes dans la formulation à l'issue de la procédure de génération de colonnes, en comparaison du nombre total de contraintes couvertes par toutes les variables présentes dans le PMRA à l'issue de cette même génération de colonnes. Les contraintes contenant zéro ou une variable ne sont pas prises en compte, étant sans impact sur la formulation, trivialement mises de côté et n'ayant de ce fait aucun intérêt en regard de l'agrégation.

Nous pouvons y observer clairement un effet conséquent de l'agrégation sur la taille de la formulation : une réduction de plus de 95% est observée dans les cas les plus favorables, qui correspondent aux plus petites instances. Cette réduction s'amenuise jusqu'à atteindre 46% pour les formulation de plus grande taille.

Ce comportement confirme en premier lieu les anticipations réalisées sur la structure de la matrice des contraintes, à savoir la présence d'un grand nombre de redondances dans les contraintes du fait d'utilisation de ressources très similaires par différents chemins.

D'autre part, l'observation de la répartition du temps de calcul entre les différents composants de la génération de colonnes, montrée par les graphes 5.1, 5.2 et 5.3, a mis en valeur que la proportion de temps passée à agréger les contraintes diminuait significativement avec l'augmentation de la taille des instances. La baisse de performance en terme de nombre de contraintes supprimées est donc compensée par une baisse d'impact en terme de temps de calcul au sein de la génération de colonnes.

5.2.2.2 Comportement de l'algorithme au cours de la résolution

Il est possible d'analyser plus finement l'effet de l'agrégation en s'intéressant au comportement de celle-ci à chacune des itérations de la génération de colonnes. Le graphe 5.4 donne l'évolution du nombre de contraintes de la formulation avant agrégation (courbe bleue) et le nombre de contraintes supprimées (courbe rouge), à chaque itération de la génération de colonnes, pour l'instance correspondant aux paramètres $|T| = 104$, $\overline{d} = 60$ et $g = 5$. L'écart entre les deux courbes correspond de ce fait au nombre de contraintes présentes dans le PMRA à chaque itération après agrégation des contraintes. L'évolution pour cette instance est un comportement représentatif de ce qui a été constaté sur l'ensemble des instances.

Nous distinguons deux comportements globaux :

- le nombre de contraintes supprimées suit une évolution globalement décroissante ;
- la courbe de l'agrégation suit approximativement la forme de celle du nombre de contraintes dans la formulation avant agrégation.

Le premier point s'explique par le fait que l'augmentation progressive du nombre de variables dans la formulation implique une dispersion croissante de l'utilisation des ressources par les chemins correspondant à ces variables. Cette dispersion est d'autant plus cohérente qu'une augmentation de la valeur de la fonction-objectif privilégie l'utilisation de ressources peu saturées, faisant perdre aux contraintes associées leur relation de dominance ou de redondance. Le second point s'explique par le fait qu'un nombre de contraintes élevé dû à l'ajout de nouvelles variables n'échappe pas à la présence d'un grand nombre de relations de redondance ou dominance.

D'autres comportements plus locaux peuvent être remarqués, comme une diminution subite du nombre de contraintes avant agrégation à certaines itérations, immédiatement contrebalancée dans l'itération

$\overline{d}$	g	$\lvert T\rvert$					
		53		66		104	
		utilisées	agrégées	utilisées	agrégées	utilisées	agrégées
	15	90765	3238	115803	3815	164813	6584
	10	91115	3605	116609	4668	165065	8134
30	5	90720	5024	116941	6212	166335	10857
	2	90430	7017	116052	8975	165071	14516
	1	90964	9213	116731	10794	165781	16896
	15	95045	5219	125536	6334	172569	11168
	10	95611	6510	124238	8570	173012	15086
60	5	95923	10190	123957	13869	173638	24126
	2	95996	16256	124046	20072	174102	41317
	1	96221	22235	124879	26314	173722	57864
	15	99980	7764	126762	9462	178005	16746
	10	99908	10267	126857	12579	177507	22981
90	5	99358	16341	127963	22173	178196	38967
	2	99245	29633	126737	36184	178353	70471
	1	99334	40754	128963	49670	178112	95876

Table 5.4 – Nombre total de contraintes couvertes (utilisées) et présentes dans le PMRA (agrégées) à l'issue de la génération de colonnes.

suivante. Ceci est le cas par exemple à l'itération 6 et, de façon particulièrement évidente, à l'itération 25. Ce phénomène est causé par une séquence d'opérations au sein de la génération de colonnes :

1. l'algorithme d'agrégation réduit l'ensemble des contraintes, une fois les variables sélectionnées insérées ;
2. le PMRA est ainsi résolu ;
3. une solution duale au PMRA est calculée et le sous-problème est résolu ;
4. les variables sélectionnées après résolution du sous-problème diversifient peu les ressources utilisées par rapport aux variables déjà présentes dans le PMRA ;
5. conséquemment, peu de relations de dominance ou de redondance sont cassées par l'ajout de ces nouvelles variables ;
6. un nombre faible de contraintes supplémentaires doit de ce fait être ajouté au PMRA, la majorité étant en grande partie déjà présente du fait de la conservation des relations de dominance et de redondance ;
7. lors de l'itération suivante, le nombre de contraintes dans la formulation avant agrégation apparaît donc plus faible que ce qu'il était avant agrégation à l'itération précédente.

Ces considérations doivent être liées avec les contributions d'Elhallaoui et al. [27] qui, dans le cadre de l'agrégation dynamique de contraintes pour le cas du *Set Partitioning*, mesurent dans quelle proportion l'ajout d'une variable casse les relations de redondance existantes entre les contraintes. Cette mesure est utilisée de façon à privilégier les variables cassant le moins possible ces relations. Les auteurs évitent ainsi une croissance trop importante du nombre de contraintes dans la formulation au cours de la génération de colonnes. Cette approche peut par ailleurs être vue comme le pendant dynamique de la restriction du choix des variables effectué par Ryan et Foster [65].

De bons résultats sont rapportés par Elhallaoui et al. [27] en combinant ce critère avec un seuil de valeur du coût réduit. Dans notre cas, des expérimentations préliminaires ont montré que trouver des colonnes ne modifiant pas l'ensemble agrégé des contraintes était quasiment impossible. Ceci peut être expliqué par la détection additionnelle des relations de dominance, qui rend plus difficile le fait de trouver des variables qui respectent à la fois les relations de redondance et de dominance qui existent. Les autres raisons envisageables incluent également une structure différente du problème maître du fait des spécificités du problème.

Notons enfin que l'ajout de la considération des relations de dominance possède un impact non négligeable sur la taille du PMRA. En effet, si lors des premières itérations la très grande majorité des contraintes sont supprimées du fait d'une relation de redondance stricte, cette tendance s'affaiblit progressivement au cours de la résolution, pour finalement atteindre une proportion similaire. La diminution du nombre de relations de redondance stricte se fait donc au profit de relations de dominance, dont l'utilisation pour agréger la formulation se révèle conséquemment pertinente.

5.2.3 Impact de différentes stratégies d'insertion des colonnes

Des tests comparatifs ont été effectués en phase préliminaire afin de déterminer dans quelle mesure le choix de stratégies d'insertion de colonnes avait un impact sur la génération de colonnes. Le graphe 5.5 illustre ceci à travers la présentation de différents temps de calcul de la génération de colonnes pour une instance, avec 4 stratégies différentes :

– la première, notée A, consiste à limiter à chaque itération l'insertion d'une unique colonne par train, cette colonne étant celle possédant le coût réduit strictement positif le plus grand parmi toutes les colonnes candidates associées à ce train ;

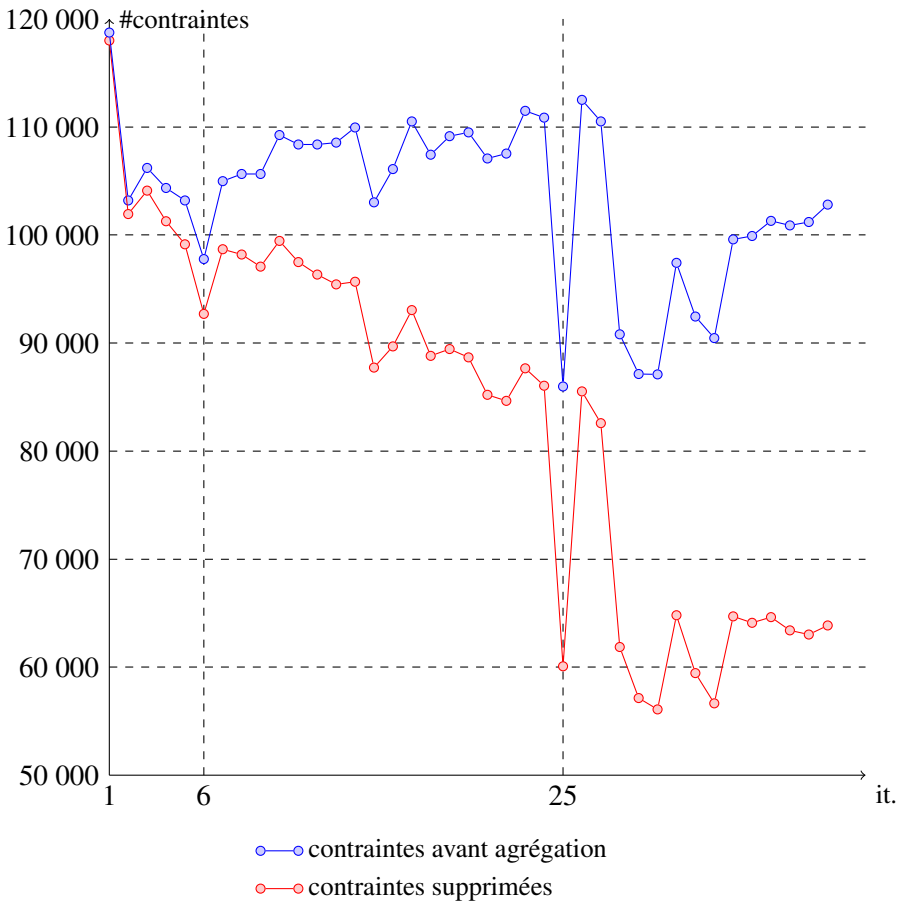

Figure 5.4 – Nombre de contraintes supprimées par itération pour $|T| = 104$, $\overline{d} = 90$ et $g = 5$.

– la seconde, notée B, consiste simplement à insérer les 50 colonnes au coût réduit strictement positif le plus grand (si moins de 50 colonnes ont un coût réduit strictement positif, la totalité est alors insérée) ;
– la troisième, notée C, consiste à insérer, pour chaque train, la colonne possédant le coût réduit strictement positif le plus élevé, tout en limitant chaque ressource de U à être couverte au maximum par 5 colonnes nouvellement ajoutées à la formulation ;
– enfin, la stratégie BC, par ailleurs utilisée pour résoudre l'ensemble des 45 instances, combine les deux précédentes : si la stratégie C sélectionne au moins 50 colonnes, l'ensemble qu'elle a sélectionné est inséré ; sinon, la stratégie B est utilisée.

Bien entendu, un très grand nombre de telles stratégies peut être conçu. Celles-ci, basées sur des approches heuristiques, sont évaluées à travers des mesures empiriques. Nous cherchons ici, sans présenter de résultats exhaustifs, à mettre en valeur l'impact de différentes stratégies dont aucune ne semble *a priori* aberrante dans une perspective de résolution efficace.

Ainsi, le graphe 5.5 montre que le temps de calcul pour la stratégie A est environ 50% supérieur à celui de BC. Il montre en outre, à travers les stratégies B, C et BC, l'intérêt de combiner des données spécifiques au problème permettant d'adapter la sélection des colonnes plus finement. En effet, la seule prise en compte du coût réduit dans la stratégie B est moins pertinent que la stratégie BC. Cependant, la stratégie C, qui limite la sélection de colonnes par la prise en compte de données spécifiques, est également moins bonne que la stratégie B. Leur combinaison est toutefois la stratégie qui se comporte le mieux. L'exploitation de données spécifiques dans la stratégie C est bien entendue corrélée avec les deux types de contraintes linéaires que sont les contraintes d'unicité et de ressource. Cette corrélation donne à cette stratégie un apport pertinent. Cet exemple simple montre finalement que :

– une forme d'anticipation sur les conflits présents dans le PMRA en limitant le nombre de colonnes insérées *via* des données spécifiques est bénéfique ;
– cette limitation doit être mise en balance avec la nécessité d'insérer un nombre suffisant de colonnes, sélectionnées par le critère essentiel qu'est la valeur du coût réduit, afin d'éviter un trop grand nombre d'itérations.

Une vue du nombre de colonnes générées à chaque itération par chaque stratégie est proposée par le graphe 5.6, venant préciser ces conclusions. L'excès de limitation menant à une augmentation du nombre d'itérations pour les stratégies A et C est clairement visible. À l'inverse, la stratégie B sélectionne 50 colonnes pendant la majorité des itérations, mais la seule utilisation du coût réduit comme critère de sélection entraîne finalement la nécessité de réaliser davantage d'itérations que la stratégie BC. Une autre raison à ce surplus d'itérations peut être le fait que la limite fixe égale à 50 se révèle trop faible dans les premières itérations. Ces deux dernières stratégies ont en commun une décroissance très rapide du nombre de colonnes générées sur les dernières itérations.

5.3 Le logiciel RECIFE PC

RECIFE est un projet de recherche initié à l'origine par la Région Nord-Pas-de-Calais, pour lequel des recherches ont été conduites par un ensemble de partenaires incluant l'IFSTTAR, la SNCF, l'Université de Valenciennes, l'École des Mines de Saint-Étienne et l'Université de Nantes. Il a mené au développement de plateformes logicielles incluant les fonctionnalités suivantes :

– des méthodes présentées par Rodriguez [63] visant à réordonnancer le trafic en cas de perturbation et utilisant des modélisations et techniques de résolution issues de la programmation par contraintes ;

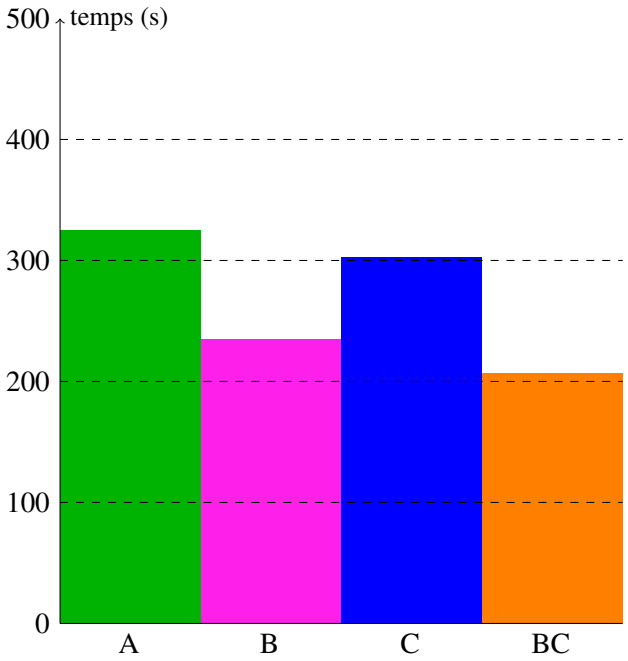

Figure 5.5 – Temps de résolution total pour $|T| = 66$, $\overline{d} = 30$ et $g = 1$ selon différentes stratégies d'insertion de colonnes.

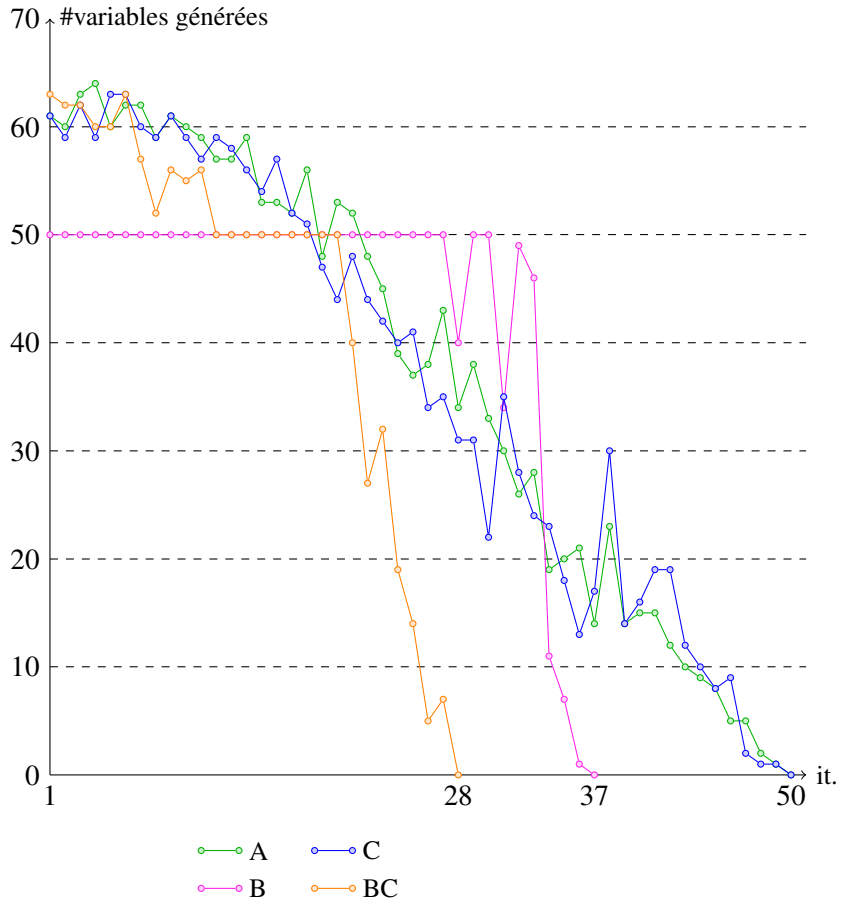

Figure 5.6 – Nombre de colonnes générées par itération, pour l'instance $|T| = 66$, $\overline{d} = 30$ et $g = 1$ selon différentes stratégies d'insertion de colonnes.

– des algorithmes d'optimisation tels que GRASP, ACO et désormais GC-ACO, dont le but est la résolution du problème de saturation à échelle microscopique selon une offre de trafic donnée ;
– des algorithmes d'évaluation de robustesse des grilles horaires saturées, dont une présentation spécifique est donnée par Delorme et al. [22] ;
– un ensemble d'outils de visualisation de la circulation au sein d'une infrastructure, incluant des diagrammes de Gantt et des graphes permettant de mesurer l'impact de différents délais sur une grille horaire.

L'ensemble des développements logiciels réalisés dans le cadre de RECIFE sont présentés sous divers angles, notamment par Delorme [20], Rodriguez [62, 63], Gandibleux et al. [33] et Merel et al. [53], et ont bénéficié d'un certain nombre d'évolutions depuis leur création. De ces développement, deux logiciels ont été produits :

– le logiciel RECIFE lui-même, implémentant les algorithmes de réordonnancement de trafic visant à minimiser le retard en cas de perturbation ;
– un autre logiciel, RECIFE PC, implémentant la résolution du problème de saturation, un module d'étude de robustesse des grilles horaires ainsi qu'un générateur de scénarios et des outils de visualisation.

Du fait de l'orientation du présent travail, nous nous focalisons sur le logiciel RECIFE PC, dont nous donnons une présentation succincte des différents modules. Nous fournissons des précisions supplémentaires sur le module de saturation, dont le fonctionnement a profité de la contribution apportée par l'algorithme GC-ACO, l'intégration de celui-ci ayant amené diverses améliorations pour la plateforme logicielle dans son ensemble.

5.3.1 Génération de scénarios

Le module de génération de scénarios permet comme son nom l'indique de produire des instances de problèmes liées au calcul de la capacité d'infrastructures. Les instances ainsi créées peuvent contenir un nombre variable de trains de chaque type (TGV, intercités, fret), les tolérances de retard peuvent être réglées et certains trains ou certains itinéraires peuvent être pondérés.

Les instances produites par le générateur sont les données d'entrée du module de saturation.

5.3.2 Saturation

Le module de saturation est dédié, comme son nom l'indique, à la résolution du problème de saturation. Il peut également être utilisé pour la résolution des problèmes connexes comme la faisabilité et l'optimisation des préférences.

Lors de l'implémentation de GC-ACO au sein de ce module, il a été décidé de ne pas supprimer ACO, permettant ainsi à l'utilisateur d'avoir le choix de l'algorithme de résolution. Cette décision a impliqué d'importants travaux d'implémentation permettant de désolidariser l'interface graphique de la partie algorithmique. Nous obtenons une structure logicielle qui, outre le fait de proposer un algorithme de résolution supplémentaire, possède une modularité accrue et de ce fait une meilleure évolutivité. Cette caractéristique se révèle importante puisqu'elle permet l'ajout plus aisé d'algorithmes supplémentaires de résolution dans le futur.

La résolution de la saturation pouvant être longue, le logiciel permet la visualisation de l'avancement de l'algorithme par des courbes de progression, qui affichent des informations différentes selon l'algorithme utilisé :

- s'il s'agit d'ACO, trois courbes sont affichées et correspondent aux meilleures solutions courantes trouvées par ACO, dont le détail est donné par Gandibleux et al. [33] ;
- dans le cas de GC-ACO, nous proposons également la visualisation de trois courbes de progression, permettant d'observer :
  - la progression de la valeur optimale du PMRA ;
  - la progression de la valeur borne duale telle que définie par l'équation 4.4 ;
  - la meilleure solution entière connue.

Même si le temps de résolution n'est bien entendu pas prévisible, les courbes de la solution optimale du PMRA et de la borne duale peuvent permettre à l'utilisateur d'estimer le temps restant de par la visualisation du comportement de l'algorithme. Un exemple d'une telle visualisation est présenté par la figure 5.7 et montre respectivement la solution primale, la borne duale et la meilleure solution entière connue en trait pointillé noir, trait plein bleu et trait épais rouge. Du point de vue de l'évolutivité, l'affichage des courbes peut être adapté à tout algorithme de résolution fournissant des données adéquates à l'interface.

En outre, le logiciel utilise le *multithreading*, permettant l'exécution du processus de résolution de façon indépendante de l'interface graphique. Nous y avons intégré, dans le cadre de GC-ACO, un mécanisme de communication des résultats intermédiaires pour la mise à jour de la courbe par le biais de *callbacks*, *via* la définition de structures de données également réutilisables.

En résumé, le travail d'implémentation réalisé au sein du module de saturation a permis l'adjonction de GC-ACO tout en conservant le fonctionnement d'ACO et en améliorant la flexibilité de l'interface pour de futures évolutions.

5.3.3 Robustesse

Le module de robustesse a pour objectif l'estimation de la robustesse d'une grille horaire générée par le module de saturation. Une solution retournée par ce dernier est en effet un routage d'un ensemble de trains, c'est-à-dire l'affectation d'un chemin à un certain nombre de trains de la demande à travers l'infrastructure. Le module de robustesse simule un retard, appelé retard primaire, sur l'un des trains de cette grille horaire et mesure l'impact sur l'ensemble des autres trains de l'horaire, à savoir les retards dits secondaires. Cette mesure est réalisée en propageant le retard à travers la grille horaire, sans réordonnancer les trains mais de façon à conserver le respect des contraintes de sécurité, c'est-à-dire le fait que chaque zone ne peut être occupée que par un train à la fois. L'algorithme de calcul est basé sur une modélisation sous forme de graphe et la résolution d'un problème de plus court chemin, dont les détails sont donnés par Delorme et al. [22].

Au-delà de la capacité algorithmique à propager les retards, le module de robustesse fournit deux outils de visualisation permettant de comparer le comportement de plusieurs grilles horaires en termes de robustesse. L'utilisateur peut dans un premier temps appliquer la propagation de retards à un ensemble de valeurs de retards primaires. Par la suite, il peut identifier les grilles horaires les plus à même de résister à un retard selon la valeur de celui-ci, et effectuer une comparaison sur plusieurs valeurs de retard à la fois. Il peut identifier les meilleures solutions de compromis, c'est-à-dire pour lesquelles il n'existe pas de solution meilleure pour toutes les valeurs de retard considérées.

5.3.4 Visualisation des solutions

Pour terminer, RECIFE PC intègre plusieurs outils de visualisation de circulation, dont une partie est issue des développements réalisés sur le logiciel RECIFE par l'IFSTTAR. Ces outils correspondent aux différents types de visualisation qui peuvent être employées dans le domaine ferroviaire, et incluent :

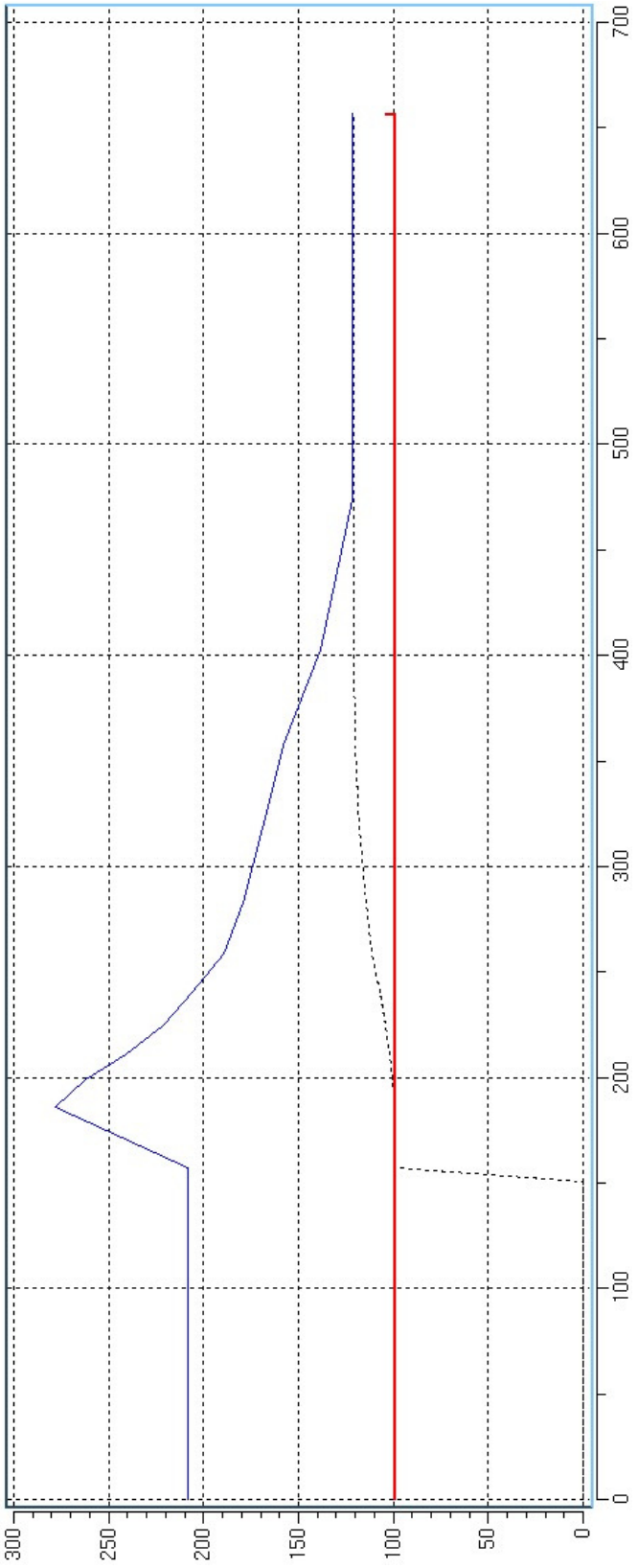

Figure 5.7 – Visualisation de ACO et de la borne supérieure.

– un diagramme de Gantt permettant la visualisation de l'occupation des zones de détection et détaillant les temps de réservation, d'occupation et de dégagement ;
– un diagramme espace-temps permettant la visualisation des sillons sur l'infrastructure ;
– une visualisation interactive du schéma de l'infrastructure permettant d'observer l'évolution de la circulation des trains avec le temps.

Elles complètent de ce fait la plateforme par une restitution des données sous une forme destinée à être exploitable par l'utilisateur final.

La figure 5.8 montre la visualisation sous forme de diagramme de Gantt, où il est possible de distinguer le temps de réservation (marqué par la lettre « r »), le temps d'occupation (marqué par le numéro du train) et le temps de dégagement (marque par la lettre « c ») de chaque train sur chaque zone. On y distingue aussi clairement le résultat du cantonnement, remarquable par le fait que plusieurs zones sont parfois réservées simultanément puis dégagées successivement.

5.4 Bilan et critique

5.4.1 Bilan des contributions

La plateforme RECIFE PC intègre désormais deux algorithmes de résolution pour le problème de saturation, et nous avons vu que l'implémentation du nouvel algorithme, GC-ACO, a été l'occasion d'apporter plusieurs améliorations substantielles à la plateforme, la principale étant une meilleure évolutivité. Nous avons également présenté un ensemble de résultats permettant de comparer ces deux algorithmes. Nous avons vu que les deux algorithmes se comportent différemment, selon les paramètres des instances. Nous synthétisons donc ici leur comportement et en tirons succinctement les cas de figure où l'utilisation de GC-ACO est bénéfique.

Pour résoudre une instance impliquant un grand nombre de variables sur un système où la quantité de mémoire vive est limitée, GC-ACO est nettement préférable à ACO, du fait de sa consommation mémoire plus faible et, bien entendu, du temps de résolution plus faible. Plus particulièrement, les tests ont montré que le temps de résolution devient nettement plus faible avec GC-ACO lorsque le nombre de variables dépasse 2 000, l'algorithme devenant alors le choix le plus pertinent pour la résolution. Ceci est finalement appuyé par les résultats obtenus en terme de qualité de solution, laquelle n'est dans la plupart des cas pas sacrifiée en contrepartie du gain de temps.

De plus, GC-ACO fournit une borne supérieure par la solution continue donnée par la génération de colonnes, ce qui est une information supplémentaire donnée simultanément au résultat. Toutefois, nous avons également vu qu'il existe une forte marge d'amélioration en ce qui concerne la qualité de la borne supérieure, et ce principalement pour les instances les plus grandes.

Nous avons donc, par le développement de cet algorithme et son intégration à RECIFE PC, contribué nettement à la résolution du problème de saturation, lui-même étant un problème sous-jacent à l'estimation de la capacité des infrastructures ferroviaires.

5.4.2 Critique des solutions obtenues

Outre les performances encourageantes mesurées pour la résolution du problème de saturation, nous étudions finalement dans quelle mesure les solutions fournies par la résolution du problème de saturation sont valides dans le contexte plus large de l'estimation de capacité.

Nous considérons pour ce faire l'exemple une solution pour l'instance définie par $|T| = 66$, $\overline{d} = 60$ et $g = 2$. Le détail des trains routés pour cette solution apparaît dans le tableau 5.5. Il est possible d'y

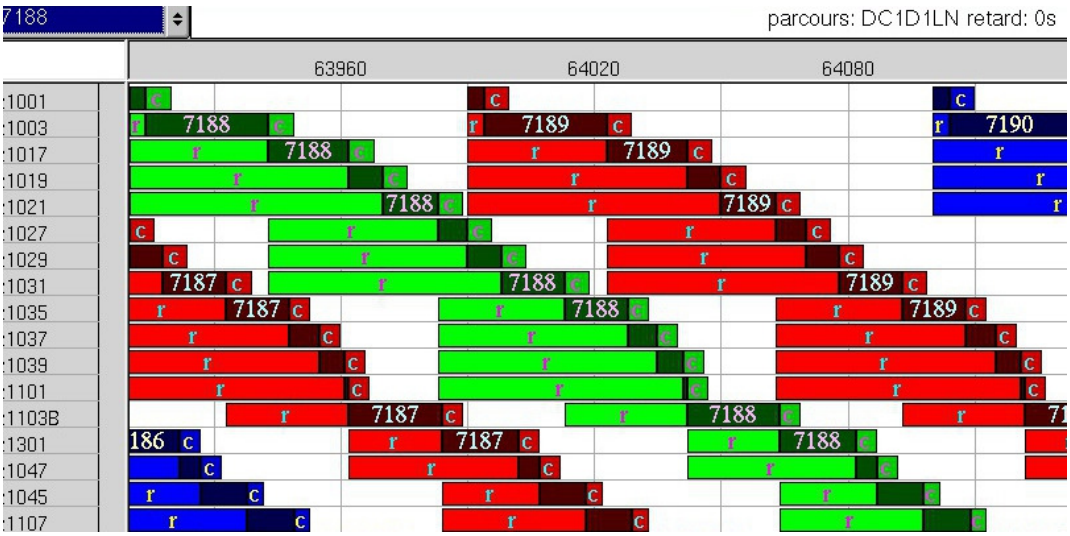

Figure 5.8 – Diagramme de Gantt d'une solution dans RECIFE PC.

observer clairement des disparités importantes au niveau des trains routés, selon leur type ou leur sens. Nous voyons par exemple que si une forte proportion des TGV de la demande est routée, très peu de trains intercité le sont. De plus, au sein des trains de fret, il y a également une forte disparité selon le sens de circulation. De ce fait, la répartition des trains routés dans une solution ne reflète pas la structure de la demande de trafic.

Ce phénomène s'explique par la combinaison de plusieurs raisons, liées à la fois aux données du problème et à la façon dont celui-ci est modélisé :

- le problème d'optimisation considéré possède un unique objectif, à savoir la maximisation du nombre total de trains routés à travers l'infrastructure ;
- comme cela a pu être remarqué sur la figure 3.6, l'infrastructure contient un grand nombre de voies banalisées, impliquant de fortes incompatibilités entre trafics d'un même type de train allant en sens opposé ;
- certains itinéraires associés à des types et des sens particuliers utilisent davantage de zones de détection que d'autres ;
- router le maximum de trains implique implicitement de favoriser les itinéraires utilisant le moins de ressources spatio-temporelles, c'est-à-dire les trains à circulation plus rapide sur des itinéraires plus courts, entraînant par ce biais moins de conflits avec d'autres trains ;
- la circulation d'une série de trains du même type dans le même sens est une combinaison qui minimise les conflits tout en permettant de router un grand nombre de trains.

Le problème principal que nous observons dans cette solution peut être résumé par un manque d'équité entre les différentes catégories de trains présentes (définies par leur type et leur sens de circulation) au sein de la solution. C'est bien le tout premier point – l'expression de la fonction-objectif – qui est la cause initiale dans l'obtention d'une telle structure de solution, l'ensemble des autres raisons étant simplement liées aux données, qu'il n'est pas envisageable de modifier, et au processus d'optimisation résultant naturellement de la conjonction des données et de la fonction-objectif. Ce manque d'équité n'est donc pas particulier à l'algorithme de résolution, celui-ci se contentant de résoudre la formulation proposée.

Une étape suivante en vue d'approcher l'estimation de capacité d'une valeur réaliste par une amélioration de l'équité paraît donc pertinente. Elle peut être réalisée en affinant le modèle d'optimisation par la prise en compte d'un objectif supplémentaire, qui vise à garantir un équilibre correct entre différentes catégories de trains. Le chapitre suivant tente de modéliser cette problématique et de proposer des pistes algorithmiques en vue de sa résolution.

Type de trafic	Sens	Demande initiale	Routés	Taux
TGV	Paris → LGV Nord	11	11	100%
	LGV Nord → Paris	11	8	73%
Intercité	Paris → Chantilly	11	3	27%
	Chantilly → Paris	13	2	15%
Fret	Grande Ceinture → Chantilly	13	9	69%
	Chantilly → Grande Ceinture	7	2	29%

Table 5.5 – Taux de routage par type de train et par sens pour $|T| = 66$, $\overline{d} = 30$ et $g = 2$.

CHAPITRE 6

Modélisation biobjectif pour un critère d'équité

Comme en témoignent les résultats présentés au chapitre précédent, les performances de la résolution du problème de saturation ont été améliorées avec succès. La combinaison d'un algorithme de génération de colonnes amélioré par diverses contributions avec la métaheuristique ACO a en effet permis de significatifs gains de temps de calcul, tout en conservant une qualité de solution similaire. Nous avons cependant remarqué, dans la structure des solutions, de nets déséquilibres entre les types de trains routés, privant cette estimation de capacité d'un certain niveau de réalisme, la cause de ce déséquilibre prenant son origine dans l'expression de la fonction-objectif.

Ce dernier chapitre propose de ce fait une approche prospective visant à modifier la formulation mathématique. La notion d'optimisation multiobjectif est tout d'abord introduite, puis nous nous focalisons plus spécifiquement sur l'optimisation des problèmes de PL biobjectifs. Nous introduisons une formalisation de ceux-ci, un ensemble de notations et de vocabulaire associé, et enfin le simplexe paramétrique, qui est une méthode de résolution classique pour une telle formulation. Celui-ci est basé sur des exécutions successives du simplexe mono-objectif résolvant des sommes pondérées des deux objectifs considérés.

Deux propositions d'objectifs visant à garantir l'équité sont ensuite formulées. La première s'inspire de travaux réalisés par Soomer et Franx [72] dans le cadre de problèmes d'atterrissage d'aéronefs au sein desquels les coûts supplémentaires subits par les compagnies aériennes du fait d'un retard à l'atterrissage doivent être répartis équitablement entre celles-ci. La seconde, plus simple, cherche à réutiliser le travail fourni lors de la conception de l'algorithme GC-ACO en exploitant la solution continue et en calculant de nouvelles pondérations pour les variables afin de compenser les déséquilibres.

Pour terminer, privilégiant le second modèle pour sa simplicité d'adaptation par rapport au travail effectué préalablement, nous suggérons une approche algorithmique simple visant à combiner le principe de la génération de colonnes avec celui du simplexe paramétrique. Celle-ci n'a pas mené à une implémentation fonctionnelle du fait de difficultés d'ordre technique, mais cette suggestion, ainsi que les deux modèles proposés, peuvent toutefois constituer une ouverture intéressante à la fois dans la perspective du problème de capacité d'infrastructures ferroviaires et dans le domaine de l'optimisation multiobjectif.

6.1 Aperçu de l'approche

6.1.1 Problématique

La procédure de génération de colonnes mise en place au chapitre 4 permet la résolution du problème de saturation en temps raisonnable. Nous avons vu qu'une solution au problème de saturation représente, par définition, une estimation de la capacité de l'infrastructure considérée. La structure du problème de saturation implique cependant, comme nous l'avons vu, de forts déséquilibres dans les grilles horaires retournées par l'algorithme. Ces déséquilibres consistent plus précisément en un problème d'équité visible selon deux axes. D'une part, le nombre de trains routés est, pour certains types (TGV, intercité, fret), très peu représentatif de la structure initiale de la demande. Le tableau 5.5 montre par exemple que que 19 TGV sur 22 sont routés contre seulement 5 sur 24 pour les trains Intercité. D'autre part, pour un même type de train, il y a également d'importantes disparités selon le sens de circulation, phénomène montré de façon évidente par l'exemple des trains de fret dans l'exemple présenté par le tableau 5.5.

Plus généralement, si la résolution du problème de saturation donne le nombre maximal de trains qu'il est possible de router sur une infrastructure selon une demande de circulation donnée, le respect de la structure de la demande n'est absolument pas garanti par la solution permettant d'obtenir ce maximum. Ainsi, dans le large cadre de l'estimation de capacité des infrastructures ferroviaires, améliorer le réalisme de l'estimation passe par la nécessité de pallier à ces déséquilibres, ou, autrement dit, améliorer l'*équité* entre différentes *classes* de trains, qui sont ici définies par leur type et leur sens de circulation, au sein de la structure d'une solution.

De bonnes performances ayant été obtenues pour la résolution du problème de saturation, nous cherchons maintenant à complexifier la modélisation du problème afin d'y prendre en compte l'équité. À notre connaissance, intégrer cette notion au sein d'un modèle d'optimisation pour le calcul de capacité n'est pas envisagé dans la littérature.

6.1.2 Mise en œuvre

Afin de mettre en place la modélisation de ce nouveau critère tout en prenant en compte le travail précédemment réalisé, il apparaît nécessaire de se poser les questions suivantes :

1. comment quantifier formellement l'équité d'une solution ?
2. comment intégrer cette quantification à la formulation courante ?
3. comment optimiser cette quantité de façon simultanée à la résolution du problème de saturation ?

Intuitivement, et au vu des problèmes identifiés, l'équité peut ici être vue comme la nécessité de limiter l'écart, en termes de nombre de trains routés, existant entre les différentes classes de trains de la demande de circulation à l'issue de la résolution. Il est donc impératif de faire apparaître les classes définies par la demande dans la mesure d'équité, puis dans un second temps de mesurer l'écart entre les classes et enfin de chercher à réduire cet écart. C'est cette approche que nous adoptons pour quantifier l'équité.

La mesure d'équité, selon la façon dont elle est formalisée, peut impliquer des modifications d'importance variable dans le modèle PLNE mis en place pour le problème de saturation. En effet, intégrer une nouvelle mesure peut impliquer la nécessité d'ajouter de nouvelles variables et contraintes à la formulation. Ceci peut par la suite non seulement faire augmenter la taille de la formulation mais également lui faire perdre sa structure de SPP, rendant plus difficile la réexploitation de l'algorithme de génération de colonnes précédemment mis au point. Nous présentons de ce fait deux propositions : l'une est inspirée de travaux réalisés dans le cadre de l'ordonnancement d'atterrissage d'aéronefs présentés par Soomer et

Franx [72], et l'autre est une proposition que nous formulons visant à conserver intégralement la structure SPP de la formulation et envisager ainsi une mise en place plus simple pour la résolution.

En ce qui concerne la résolution, nous cherchons en premier lieu à réutiliser la procédure de génération de colonnes que nous avons développée, et ce afin de profiter de ses performances en termes de temps de calcul. Bien que cela soit également dépendant de la formalisation choisie pour quantifier l'équité, le caractère modulaire de la génération de colonnes permet d'envisager d'en modifier ou d'en adapter certaines composantes, tout en limitant des modifications structurelles importantes de l'algorithme. Ainsi, la génération de colonnes, en tant que cadre algorithmique générique, peut permettre d'envisager la résolution d'une formulation PL sensiblement différente au prix de modifications limitées.

Il existe enfin dans le domaine de l'optimisation des méthodes dites d'*optimisation multiobjectif*, dédiées à la modélisation et à la résolution de problèmes de PL et de PLNE possédant plusieurs fonction-objectifs. Nous proposons donc d'intégrer la quantification de l'équité comme seconde fonction-objectif, menant à la formalisation de modèles biobjectifs et à l'utilisation de méthodes dédiées à de tels modèles.

En résumé, nous choisissons de conserver la procédure de génération de colonnes et d'utiliser un modèle biobjectif. De ce fait, nous limitons l'approche à la résolution de la relaxation continue de la formulation, pouvant servir de base pour une future résolution biobjectif en nombres entiers. Nous suggérons une méthode visant à combiner génération de colonnes et résolution biobjectif. Une telle combinaison n'a à notre connaissance pas fait l'objet d'études approfondies dans la littérature existante.

6.2 Notions d'optimisation biobjectif

La présente section résume les points essentiels qu'il est important de connaître lorsque l'on s'intéresse à des problèmes d'optimisation multiobjectifs. Il s'agit en effet d'un domaine au sein duquel un formalisme et un vocabulaire particuliers existent, se différenciant partiellement du formalisme existant dans le cas mono-objectif tel que nous l'avons décrit au chapitre 3. Seuls les éléments pertinents au vu du cas que nous traitons sont présentés ici, une présentation complète de l'optimisation multiobjectif (modèles, formalisme et méthodes de résolution) étant donnée par Ehrgott [26].

Par soucis de simplicité et de façon cohérente avec le problème ferroviaire traité, nous nous plaçons dans le cas précis des problèmes d'optimisation linéaires biobjectifs. Les notions génériques, qui sont valables pour le cas plus général de l'optimisation multiobjectif, sont tout d'abord présentées, puis un algorithme usuel de résolution, quant à lui spécifique au cas biobjectif, est résumé, à savoir l'algorithme du simplexe paramétrique.

6.2.1 Formalisation et vocabulaire

Un problème de PL biobjectif s'écrit simplement en considérant une deuxième fonction-objectif linéaire, celle-ci possédant des coefficients *a priori* indépendants de la première. Ainsi, en reprenant la formalisation générique employée dans le cas mono-objectif telle que montrée par la formulation 3.1, nous exprimons une formulation générique de problème de PL biobjectif de la façon décrite par

l'expression 6.1 :

$$\mathcal{P}^2 = \left[\begin{array}{lll} \max z^1(x) = & \sum\limits_{1\leq j\leq n} \rho_j x_j & \\ \max z^2(x) = & \sum\limits_{1\leq j\leq n} \phi_j x_j & \\ \text{s.c.} & \sum\limits_{1\leq j\leq n} a_{ij} x_j = b_i & , \ 1 \leq i \leq m \\ & x_j \in \mathbb{R} & , \ 1 \leq j \leq n \end{array} \right] \tag{6.1}$$

où ρ et ϕ sont des n-vecteurs représentant respectivement les coefficients des variables pour le premier et le second objectif, les expressions respectives de ceux-ci étant données par z^1 et z^2. Il est essentiel de noter que les objectifs ne sont en aucun cas ordonnés, l'indexage de ceux-ci ne sous-tendant donc aucune priorité de l'un envers l'autre. Résoudre un problème biobjectif implique donc d'optimiser simultanément les deux fonction-objectifs. D'une façon générale, il est plus intéressant d'employer une approche biobjectif lorsque les deux fonction-objectifs sont corrélées négativement, c'est-à-dire que l'optimisation selon un objectif a tendance à défavoriser l'autre. En effet, dans le cas où les objectifs sont corrélés positivement, il peut être suffisant de réaliser une optimisation selon un unique objectif.

La présence de deux objectifs implique également des différences essentielles dans la façon de comparer des solutions entre elles. Afin d'introduire ces différences, considérons deux solutions réalisables quelconques x^1 et x^2 pour la formulation 6.1. La question que nous nous posons est de savoir laquelle de x^1 ou de x^2 est une meilleure solution au regard de la valeur de chacune selon les deux objectifs. Dans le cas mono-objectif, la comparaison de la valeur de la seule fonction-objectif pour les deux solutions est immédiate : les valeurs peuvent être égales, auquel cas aucune des deux solutions n'est meilleure, ou distinctes, auquel cas l'une des deux est meilleure. Les possibilités à considérer dans le cas biobjectif sont notoirement différentes et nécessitent d'introduire un vocabulaire et une notation appropriés, tels que présentés par Ehrgott [26]. Ce vocabulaire regroupe les notions de *dominance faible*, de *dominance* et de *dominance forte*, dont nous donnons les définitions respectives ici relativement à la notation que nous avons mise en place.

Définition 6.1. Dominance faible.
Nous considérons que x^1 domine faiblement x^2, et notons $x^1 \leqq x^2$, si et seulement si les relations suivantes sont respectées :

$$z^1(x^1) \leq z^1(x^2) \quad \textit{et} \quad z^2(x^1) \leq z^2(x^2).$$

Définition 6.2. Dominance.
Nous considérons que x^1 domine x^2, et notons $x^1 \leq x^2$, si et seulement si les relations suivantes sont respectées :

$$\begin{array}{c} z^1(x^1) \leq z^1(x^2) \quad \textit{et} \quad z^2(x^1) < z^2(x^2) \\ \textit{ou} \\ z^1(x^1) < z^1(x^2) \quad \textit{et} \quad z^2(x^1) \leq z^2(x^2). \end{array}$$

Définition 6.3. Dominance stricte.
Nous considérons que x^1 domine strictement x^2, et notons $x^1 < x^2$, si et seulement si les relations suivantes sont respectées :

$$z^1(x^1) < z^1(x^2) \quad \textit{et} \quad z^2(x^1) < z^2(x^2).$$

Ces trois définitions permettent de distinguer les différentes nuances de cas où x^1 est une solution considérée comme meilleure que x^2. Il est cependant également possible de rencontrer le cas de figure suivant :

$$z^1(x^1) < z^1(x^2) \quad \text{et} \quad z^2(x^1) > z^2(x^2)$$

signifiant que x^1 est meilleure que x^2 relativement au premier objectif mais que x^2 est meilleure que x^1 pour le second objectif. Dans ce cas, il n'existe aucune relation de dominance entre x^1 et x^2 au sens où cela est donné par les définitions 6.1, 6.2 et 6.3. Plus généralement, il existe des solutions réalisables qui ne sont dominées (respectivement faiblement, strictement) par aucune autre solution. En d'autres termes, cela signifie qu'il n'existe pas de solution réalisable qui puisse être considérée comme meilleure, mais qu'il peut exister d'autres solutions réalisables non comparables. Il est donc possible – et courant en pratique – qu'il existe un ensemble de solutions réalisables non comparables entre elles mais qui ne sont également ni dominées faiblement, ni dominées, ni dominées strictement par les autres solutions réalisables au problème considéré. De telles solutions sont désignées, selon la perspective, par deux termes distincts :

- dans l'espace des variables, une telle solution est appelée *solution efficace* ;
- du point de vue des valeurs des fonction-objectifs, la dénomination de *point non dominé* est utilisée, un unique point non dominé pouvant théoriquement correspondre à plusieurs solutions efficaces distinctes.

Ainsi, alors que la résolution d'un problème de PL mono-objectif consiste à rechercher une ou plusieurs solutions optimales (ou de bonne qualité si l'optimalité exacte n'est pas recherchée), la résolution d'un problème de PL biobjectif consiste quant à elle à rechercher un ensemble de solutions efficaces. Du point de vue d'un décideur, l'ensemble des solutions efficaces représente un ensemble de solutions de compromis, le choix final d'une solution particulière revenant à ce même décideur. Bien que plus coûteux algorithmiquement que la résolution optimale d'un problème mono-objectif, il existe pour le cas biobjectif un algorithme utilisé usuellement permettant de trouver l'intégralité des solutions efficaces d'un problème : le simplexe paramétrique.

6.2.2 Algorithme du simplexe paramétrique

L'algorithme du simplexe paramétrique permet de calculer l'ensemble des solutions efficaces d'un problème biobjectif. Il repose sur l'utilisation sous-jacente de l'algorithme du simplexe tel que nous le connaissons pour le cas mono-objectif. Nous ne rentrons pas ici dans l'ensemble des détails techniques et fondements théoriques de l'algorithme mais nous focalisons sur le principe général et les points permettant d'en envisager combinaison avec la génération de colonnes. Les détails théoriques sur le simplexe paramétrique sont donnés par Ehrgott [26].

Le simplexe paramétrique repose sur des fondements démontrant que l'ensemble des solutions efficaces d'un problème de PL biobjectif peut être calculé par la résolution d'un ensemble de problèmes mono-objectifs dont la fonction-objectif est une somme pondérée de deux objectifs de la formulation initiale. Cette propriété est également exprimée en qualifiant de *solutions supportées* l'ensemble des solutions efficaces du problème biobjectif, une solution efficace supportée étant une solution efficace calculable par résolution dudit problème mono-objectif pondéré. Formellement, il s'agit de résoudre une

séquence de formulations mono-objectifs telles que formalisées par l'expression 6.2 :

$$\mathcal{P}(\lambda) = \left[\begin{array}{lll} \max z_\lambda(x) = & \lambda z^1(x) + (1-\lambda) z^2(x) & \\ \text{s.c.} & \sum_{1 \leq j \leq n} a_{ij} x_j = b_i & , \ 1 \leq i \leq m \\ & x_j \in \mathbb{R} & , \ 1 \leq j \leq n \end{array} \right] \tag{6.2}$$

où λ est le paramètre dont l'algorithme tire son nom, variant dans l'intervalle $[0, 1]$. La variation de λ dans cet intervalle entraîne une modification des coefficients des variables dans l'objectif, pouvant impliquer une modification de la valeur des variables à l'optimalité. C'est par ce mécanisme que l'ensemble des solutions efficaces peut être trouvé.

Il est en outre montré qu'il est suffisant de parcourir un nombre fini de valeurs pour λ afin de déterminer la totalité des solutions efficaces (voir par exemple Ehrgott [26] pour la démonstration). Ici aussi, ceci est intuitivement compréhensible par le fait qu'une modification extrêmement faible de λ peut ne pas impliquer de modification de la valeur des variables à l'optimalité, signifiant qu'une solution est optimale pour un intervalle de valeurs de λ.

La séquence des valeurs de λ est calculée itérativement en fonction de la valeur des coûts réduits des variables selon chacun des deux objectifs, et nous donnons ici le raisonnement permettant d'obtenir ces valeurs successives. Notons $(\pi_j^1)_{1 \leq j \leq n}$ et $(\pi_j^2)_{1 \leq j \leq n}$ les n-vecteurs de coûts réduits associés respectivement au premier et au second objectif. Le vecteur de coûts réduits π pour la somme pondérée de la formulation 6.2 est alors donné par l'expression suivante :

$$\pi(\lambda) = \lambda \pi^1 + (1-\lambda)\pi^2.$$

Supposons que la formulation 6.2 soit résolue à l'optimalité pour une valeur $\hat{\lambda} \in [0, 1]$. Nous cherchons à déterminer la valeur de λ faisant perdre son optimalité à la solution courante, impliquant ainsi le recalcul d'une solution optimale et permettant de trouver une solution efficace supplémentaire au problème biobjectif. Nous avons par définition $\pi_j(\hat{\lambda}) \leq 0$, pour tout j tel que $1 \leq j \leq n$. Deux cas sont alors possibles :

- si $\pi_j^2 \leq 0$ pour $1 \leq j \leq n$, la solution est également optimale pour tout $\lambda < \hat{\lambda}$;
- s'il existe un indice j de variable hors-base tel que $\pi_j^2 > 0$, alors les deux assertions suivantes sont nécessairement vérifiées :
  - $\pi_j^1 \leq 0$;
  - la solution courante n'est plus optimale pour $\lambda \leq \frac{-\pi_j^2}{\pi_j^1 - \pi_j^2}$.

Ces conclusions permettent d'aboutir à la formule utilisée pour la mise à jour de λ à partir de la solution optimale obtenue pour $\hat{\lambda}$. Il s'agit de la valeur λ' en-dessous de laquelle l'optimalité est perdue, définie par l'équation 6.3 :

$$\lambda' = \max_{j \in \mathcal{J}} \frac{-\pi_j^2}{\pi_j^1 - \pi_j^2}. \tag{6.3}$$

où $\mathcal{J}$ est l'ensemble des indices j des variables hors base dans la solution optimale à $\mathcal{P}(\lambda)$, tels que $\pi_j^2 > 0$ et $\pi_j^1 \leq 0$.

Ceci définit donc une façon de faire décroître λ afin de rechercher une nouvelle solution efficace. Ainsi, le simplexe paramétrique peut être vu comme possédant deux phases :

1. résolution du problème restreint au premier objectif, ce qui correspond au cas $\lambda = 1$;

2. résolutions des problèmes mono-objectifs pour les valeurs décroissantes successives de λ, jusqu'à l'obtention d'une solution étant optimale pour $\lambda = 0$.

L'algorithme 6.1 décrit la seconde phase de la résolution du simplexe paramétrique. Celle-ci, exploitant la solution optimale pour $\lambda = 1$, calcule successivement puis fournit en sortie la séquence des valeurs de λ ainsi que chaque solution optimale associée à chacune de ces valeurs. Le calcul de λ est réalisé à la ligne 2, la sélection des variables entrante et sortante respectivement aux lignes 3 et 4 puis l'opération de pivot à la ligne 5. La variable sortante est sélectionnée par la méthode classique de pivot de l'algorithme du simplexe. Chaque opération de pivot détermine une nouvelle solution efficace associée à la valeur courante de λ. La séquence des solutions et valeurs de λ sont mémorisées pour être retournées à la fin de l'algorithme, mémorisation que nous ne représentons pas explicitement afin de simplifier la présentation de l'algorithme. La ligne 1 impose la réitération de l'algorithme tant que l'ensemble $\mathcal{J}$ n'est pas vide, conformément aux situations que nous avons listées ci-dessus. S'il est vide, la dernière solution optimale trouvée est optimale jusqu'à $\lambda = 0$ et l'ensemble des solutions efficaces a été trouvé.

Il est important, pour terminer, de retenir deux points relatifs au coût de l'algorithme du simplexe paramétrique en terme de temps de calcul :

- une résolution complète du problème selon un objectif doit être réalisée en premier lieu par l'exécution du simplexe classique ;
- à cette résolution s'ajoute un nombre de pivots égal au nombre de solutions efficaces au problème biobjectif.

Ces deux phases peuvent donc chacune représenter, selon les caractéristiques de la formulation traitée, un coût non négligeable en terme de temps de calcul.

6.3 Modélisation et intégration de l'équité

Comme nous l'avons vu, la modélisation biobjectif et la technique de résolution associée qu'est le simplexe paramétrique sont des outils idéaux pour calculer un ensemble de solutions étant les meilleurs compromis entre les deux objectifs, cet ensemble de solutions étant l'ensemble des solutions efficaces.

Nous nous intéressons donc désormais à la formulation de l'équité sous forme d'un second objectif intégrable dans un problème de PL, pour le cas du problème de capacité d'infrastructures ferroviaires. Même si la notion d'équité est intuitivement aisée à concevoir, sa définition est très variable selon le domaine d'étude. Ainsi, il est possible d'en trouver des définitions dans les domaines de la théorie des jeux (voir Barbanel et Brams [3]), des problèmes de partage de bande passante en télécommunications (voir Massoulié et Roberts [49]) ou encore de l'ordonnancement (voir Sabin et al. [66]). Dans notre cas, le problème présent au sein de la structure des solutions a été clairement identifié, et il est peu probable qu'il existe une unique réponse à celui-ci en terme de formalisation puis d'optimisation d'un critère d'équité. Nous explorons donc, à travers les deux modélisations que nous proposons, des compromis possibles entre précision de la formulation et facilité d'adaptation avec la méthode de résolution déjà en place.

La première est un modèle inspiré de travaux réalisés par Soomer et Franx [72] dans le cadre de problèmes de trafic aérien et étant explicitement défini par les auteurs comme un critère d'équité. Le contexte aérien et l'approche des auteurs sont succinctement présentés, puis l'adaptation au cas ferroviaire est détaillée. La seconde approche est une modification simple de la formulation SPP pour le problème de saturation qui vise, plutôt qu'insérer un objectif nécessitant un nouveau formalisme, à exploiter l'information fournie par la procédure de génération de colonnes dans le but de compenser les déséquilibres détectés.

Entrées : problème $\mathcal{P}(\lambda)$ issu de la formulation 6.1.

Données : $\mathcal{B}$: base optimale pour $\lambda = 1$; $(\pi_j^1)_{1 \leq j \leq n}$ et $(\pi_j^2)_{1 \leq j \leq n}$: vecteurs de coûts réduits respectivement selon z^1 et z^2.

1 **tant que** $\mathcal{J} = \{j \notin \mathcal{B},\ \pi_j^1 \leq 0,\ \pi_j^2 > 0\} \neq \emptyset$ **faire**

2 $\quad \lambda \leftarrow \max_{j \in \mathcal{J}} \frac{-\pi_j^2}{\pi_j^1 - \pi_j^2}$;

3 $\quad j_e \leftarrow \text{argmax}_{j \in \mathcal{J}} \left(\frac{-\pi_j^2}{\pi_j^1 - \pi_j^2}\right)$;

4 $\quad j_s \leftarrow$ `indicesortie(`$\mathcal{B}$, j_e`)` ;

5 $\quad (\mathcal{B}, \pi^1, \pi^2) \leftarrow$ `pivot(`$\mathcal{B}$, j_e, j_s`)` ;

6 **fin**

Sorties : séquence de valeurs pour λ ; solution optimale pour chacune de ces valeurs.

Algorithme 6.1: Calcul des solutions efficaces par le simplexe paramétrique.

6.3.1 Critère explicite d'équité

6.3.1.1 Développement dans le cadre du trafic aérien

Soomer et Franx [72] mettent en lumière, dans le cadre du trafic aérien, un problème de répartition des coûts entre les différentes compagnies du fait des retards pouvant être imposés à l'atterrissage des aéronefs. Ces retards sont imposés par les instances de contrôle aérien à un niveau tactique, c'est-à-dire dans les heures précédent l'atterrissage. Du fait des nombreux aléas impactant de façon plus ou moins importante le trafic aérien, il est nécessaire pour les contrôleurs de prendre en compte l'heure d'arrivée réelle estimée des aéronefs afin d'établir une planification des atterrissage. L'approche classique pour ce faire consiste à réordonnancer l'atterrissage des appareils en minimisant la somme totale des délais générés par rapport aux heures d'atterrissage initialement prévues. Les auteurs montrent que l'utilisation de ce seul critère entraîne d'importantes disparités entre les compagnies aériennes en terme de coûts liés au délai par rapport à l'horaire prévu. En effet, chaque aéronef devant attendre un délai supplémentaire pour son atterrissage entraîne un surcoût à sa compagnie lié à différents facteurs (consommation de kérosène, salaires des équipages, pénalités éventuelles, etc.).

En conséquence, après avoir suggéré un ensemble de critères visant à normaliser les fonctions de coût devant être fournies par les compagnies aériennes (voir Soomer et Franx [71]), les auteurs s'intéressent à réduire les disparités entre compagnies en proposant des indicateurs qui agrègent les données par compagnie. Trois indicateurs sont proposés, présentés comme trois variations possibles de quantification de l'équité :

- l'équité absolue, correspondant à la façon la plus naturelle d'envisager celle-ci, consiste à mesurer puis à minimiser l'écart entre le coût moyen par aéronef entre les compagnies ;
- l'équité relative consiste à mesurer le taux d'amélioration en terme de coût, pour chaque compagnie, de la nouvelle planification par rapport au coût par compagnie issu d'une planification existante, puis à équilibrer ce taux entre les compagnies ;
- l'équité mesurée par le retard consiste finalement à ne pas prendre en compte le coût mais directement le décalage temporel total par compagnie par rapport à une planification de référence, puis à équilibrer ce décalage entre les compagnies.

Par la suite, les auteurs détaillent l'intégration de ces critères au sein du problème d'optimisation visant initialement à minimiser la somme totale des délais, intégration réalisée en ajoutant le critère d'équité désiré à la fonction-objectif avec une pondération prédéterminée de façon empirique. Du point de vue de l'intégration du critère à la formulation, il ne s'agit donc pas d'une approche biobjectif.

Dans notre cas, nous cherchons à adapter certaines des notions d'équité proposées par Soomer et Franx [72] au cas de l'estimation de capacité d'infrastructures ferroviaires, et ce à travers un modèle biobjectif.

6.3.1.2 Adaptation au cas ferroviaire

Il apparaît assez clairement que l'agrégation par compagnie dans le cas aérien trouve un parallèle dans les classes de trains dont nous sommes en présence *via* les types de trains et sens de circulation. En outre, l'approche que nous avons pour l'équité, à savoir limiter les disparités entre compagnies en termes de nombre de trains routés par rapport à la demande initiale, se rapproche nettement de l'équité absolue définie par Soomer et Franx [72]. L'équité relative peut toutefois également être modélisée simplement, même si le fait qu'elle repose sur la connaissance d'une grille horaire de référence déjà existante rend son utilité dans notre cas sujette à interrogation. Enfin, bien qu'il soit possible d'imaginer des cas où

l'adaptation de mesure d'équité par le retard soit pertinente, il est plus intéressant dans notre cas de s'intéresser directement au poids. Nous formalisons donc ici les mesures d'équité absolue et relative.

Notons $\mathcal{F}$ l'ensemble des classes de trains, formant une partition de l'ensemble des trains T. Alors que dans le cas aérien le coût considéré dans le calcul d'équité est une fonction du retard par rapport à un horaire prévu, nous utilisons ici directement une valeur de poids associé à chaque chemin. Notons finalement que dans le cas aérien, un coût faible par compagnie est recherché, alors qu'ici un poids plus élevé pour une classe donnée est bien entendu avantageux, signifiant que celle-ci a une place plus grande qui lui est attribuée sur l'infrastructure.

Définissons, pour toute classe $F \in \mathcal{F}$, Γ^a_F le poids total de tous les chemins possibles associés à l'ensemble des trains de la classe :

$$\Gamma^a_F = \sum_{t \in F, c \in C_t} \rho_{t,c}.$$

Posons par la suite E^a_F, qui représente la mesure d'équité absolue pour la classe F. Elle consiste à mesurer le rapport entre le poids total des chemins (partiellement) activés par une solution continue x et le poids total des chemins d'une classe Γ^a_F. La formule est donnée par l'équation 6.4 :

$$E^a_F(x) = \frac{\sum_{t \in F, c \in C_t} \rho_{t,c} x_{t,c}}{\Gamma^a_F}. \tag{6.4}$$

La mesure d'équité relative, dont le principe consiste à mesurer l'amélioration du coût total pour une classe par rapport à une solution de référence déjà connue $\hat{x}$, peut également être définie aisément. Posons dans ce but Γ^r_F le poids total des chemins activés par la solution de référence $\hat{x}$:

$$\Gamma^r_F = \sum_{t \in F, c \in C_t} \rho_{t,c} \hat{x}_{t,c}.$$

Ceci permet finalement de donner la formule pour l'équité relative pour une classe $F \in \mathcal{F}$, décrite par l'expression 6.5 :

$$E^r_F(x) = \frac{\sum_{t \in F, c \in C_t} \rho_{t,c} x_{t,c}}{\Gamma^r_F}. \tag{6.5}$$

De façon plus générique, nous notons $E_F(x)$ une mesure d'équité pour une classe $F \in \mathcal{F}$, que celle-ci représente la mesure absolue ou relative. Cette notation générique nous permet par la suite d'exprimer les formules visant à intégrer indifféremment l'une ou l'autre des mesures au sein de la formulation PL. De la même façon, nous noterons les quantités Γ^a_F et Γ^r_F de façon générique par Γ_F.

Le but poursuivi est d'obtenir des valeurs $E_F(x)$ qui soient égales ou, du moins, les plus proches possibles, pour l'ensemble des classes, garantissant ainsi une équité dans la répartition des poids. Cela revient donc à minimiser l'écart entre chaque couple de valeur d'équité pour des classes distinctes. Un moyen de formaliser un objectif d'optimisation permettant de poursuivre ce but consiste à maximiser la valeur d'équité la plus faible parmi l'ensemble des classes. L'expression 6.6 formalise un tel objectif :

$$\max_{x \in \{0,1\}^n} \min_{F \in \mathcal{F}} \{E_F(x)\}. \tag{6.6}$$

Bien entendu, un tel objectif n'est pas une expression linéaire, et n'est à ce titre pas intégrable tel quel dans la formulation du problème de saturation. Ceci se fait simplement en intégrant une nouvelle variable e_{min} et en ajoutant les contraintes décrites par l'expression 6.7 à la formulation linéaire :

$$e_{min} \leq E_F(x), \ \forall F \in \mathcal{F} \tag{6.7}$$

Automatiquement, l'ajout de telles contraintes implique que la relation $e_{min} \leq \min_{F \in \mathcal{F}}\{E_F(x)\}$ est vérifiée. Par la suite, il convient de maximiser cette quantité, ce qui constitue l'objectif d'équité, qui est donné par l'équation 6.8 :

$$\max e_{min}. \tag{6.8}$$

La relaxation continue de la formulation SPP modélisant le problème de saturation devient alors une formulation de PL biobjectif telle que décrite par l'expression 6.9, possédant donc une variable et un ensemble de contraintes supplémentaires :

$$\left[\begin{array}{lll} \max & \displaystyle\sum_{t\in T, c\in C_t} \rho_{t,c} x_{t,c} & \\ \max & e_{min} & \\ \text{s.c.} & \displaystyle\sum_{c\in C_t} x_{t,c} & \leq 1 \quad ,\forall t \in T \\ & \displaystyle\sum_{t\in T, c\in C_t} \delta_{c,u} x_{t,c} & \leq 1 \quad ,\forall u \in U \\ & e_{min} - \dfrac{1}{\Gamma_F} \displaystyle\sum_{t\in T, c\in C_t} \sigma_{F,t}\rho_{t,c} x_{t,c} & \leq 0 \quad ,\forall F \in \mathcal{F} \\ & x_{t,c} & \geq 0 \quad ,\forall t \in T, \forall c \in C_t \end{array}\right] \tag{6.9}$$

où $\sigma_{F,t}$ est un paramètre dont l'introduction permet par la suite de faciliter l'expression du problème dual, et dont la signification est donnée par l'équation 6.10 :

$$\sigma_{F,t} = \begin{cases} 1 & \text{si } t \in F\,; \\ 0 & \text{sinon.} \end{cases} \tag{6.10}$$

Nous obtenons donc bien une formulation de PL biobjectif intégrant explicitement un objectif visant à assurer l'équité entre les différentes classes de trains, ces dernières pouvant être définies bien entendu en fonction des types et sens de circulation des trains. Ce modèle permet en outre d'intégrer, au choix, l'équité relative ou absolue.

6.3.2 Génération dynamique de pondérations

L'approche précédente possède l'avantage d'exprimer de façon explicite un objectif d'équité au sein du problème d'optimisation. Il est de plus théoriquement possible de résoudre ce problème tel quel grâce à une méthode telle que le simplexe paramétrique. Nous avons toutefois vu que les caractéristiques des instances auxquelles nous sommes confrontés, notamment du fait du grand nombre de variables et de contraintes, font de la procédure de génération de colonnes un élément clé de la résolution pour garantir un temps de calcul raisonnable. De ce fait, il nous paraît intéressant de proposer également une formulation aux différences minimes par rapport à la formulation résolue par la génération de colonnes, même si cela ne permet pas une formalisation à proprement parler d'un critère donnant une garantie théorique d'équité.

L'approche présentée ici consiste donc à conserver intégralement la structure de SPP de la matrice des contraintes, sans réaliser d'ajout de variable ou de contrainte. Un second objectif est toutefois ajouté, dont la seule différence avec le premier objectif réside donc dans la pondération des variables, dont le

but est de compenser les déséquilibres générés par la résolution de la relaxation continue du problème de saturation.

Pour ce faire, le résultat fourni par la procédure de génération de colonnes est exploité en agrégeant les poids dans la solution continue par classe de train. Un vecteur ϕ de nouveaux poids est calculé de façon dynamique selon la solution optimale continue fournie par la génération de colonnes. Tous les chemins associés à une classe donnée possèdent alors le même poids. Cette procédure mène donc au problème de PL biobjectif donné par l'expression 6.11 :

$$\left[\begin{array}{lll} \max & \displaystyle\sum_{t\in T, c\in C_t} \rho_{t,c} x_{t,c} & \\ \max & \displaystyle\sum_{t\in T, c\in C_t} \phi_{t,c} x_{t,c} & \\ \text{s.c.} & \displaystyle\sum_{c\in C_t} x_{t,c} & \leq 1 \quad ,\forall t \in T \\ & \displaystyle\sum_{t\in T, c\in C_t} \delta_{c,m} x_{t,c} & \leq 1 \quad ,\forall m \in M \\ & x_{t,c} & \geq 0 \quad ,\forall t \in T, \forall c \in C_t \end{array}\right]. \tag{6.11}$$

où les $\phi_{t,c}$ sont calculés de façon à compenser les déséquilibres entre classes créés par la première solution. Notons x^* la solution optimale selon le premier objectif, il est alors simplement possible d'accorder un poids aux chemins des trains d'une classe de façon inversement proportionnelle au poids total accordé à cette même classe dans la solution x^*, définissant ainsi chaque $\phi_{t,c}$ tel que décrit par l'équation 6.12 :

$$\phi_{t,c} = \frac{\displaystyle\sum_{t\in F, c\in C_t} \rho_{t,c}}{\displaystyle\sum_{t\in F, c\in C_t} \rho_{t,c} x^*_{t,c}},\ \forall F \in \mathcal{F},\ \forall t \in F,\ \forall c \in C_t. \tag{6.12}$$

Par ce biais, les classes de trains les plus défavorisées dans la première solution sont favorisées par le second vecteur de poids.

Bien entendu, il n'y a pas ici d'objectif explicite d'équité, il n'est donc théoriquement pas garanti d'atteindre, avec ce modèle, des solutions efficaces continues garantissant l'équité entre les classes. Cependant, la mise en place d'un tel modèle répond à deux attentes :

- explorer la combinaison d'une résolution biobjectif avec la génération de colonnes tout en conservant la validité de l'ensemble des améliorations apportées à cette dernière qui reposent entre autres sur la structure SPP de la matrice des contraintes ;
- étudier dans quelle mesure la résolution biobjectif favorise la prise en compte de nouvelles colonnes, jusqu'alors laissées de côté par la résolution selon le premier objectif, lors des pivots de la deuxième phase du simplexe paramétrique.

Ainsi, nous explorons par la suite les possibilités envisageables pour mettre en place une telle combinaison et présentons les questionnements auxquels se confronte la conception d'une solution algorithmique complète.

6.4 Approche pour la résolution

Nous proposons dans cette dernière section une approche algorithmique cherchant à combiner l'algorithme du simplexe paramétrique et la génération de colonnes. Le but d'une telle combinaison est d'être capable de résoudre les formulations de PL biobjectifs dont sont sommes en présence, et qui sont notamment rendues difficiles par la présence d'un grand nombre de variables et de contraintes.

Nous identifions tout d'abord les composants des deux algorithmes permettant d'envisager leur combinaison, puis l'approche que nous choisissons et enfin les difficultés identifiées dans la perspective de réaliser cette combinaison. Nous explicitons ensuite les formulations duales et expressions des coûts réduits des variables pour les deux formulations biobjectifs proposées, la présence de deux objectifs impliquant l'existence de deux formulations duales et de deux expressions de coûts réduits. Nous donnons finalement le détail de la procédure algorithmique que nous avons implémentée puis évoquons les résultats préliminaires obtenus, en quantité toutefois limitée du fait de difficultés liées à l'implémentation, que nous expliquons également.

6.4.1 Intégration du simplexe paramétrique et de la génération de colonnes

6.4.1.1 Interaction entre les algorithmes

La procédure de génération de colonnes peut être vue comme une généralisation du simplexe mono-objectif, en cela qu'elle est destinée à résoudre une formulation de PL en prenant en compte le coût réduit de variables non présentes initialement dans la formulation. Sa procédure est en outre composée d'exécutions successives du simplexe, visant à résoudre le PMR. Le simplexe paramétrique intègre quant à lui également une exécution du simplexe mono-objectif classique dans sa première phase. Ceci nous permet d'identifier les points d'interaction les plus évidents pouvant exister entre une procédure de génération de colonnes et la résolution paramétrique, qui peuvent donc être, au choix :

- chaque résolution du PMR au sein de la génération de colonnes peut être suivie d'une deuxième phase du simplexe paramétrique ;
- l'intégralité de la procédure de génération de colonnes peut être vue comme la première phase du simplexe paramétrique puisqu'elle permet la résolution à l'optimalité de la formulation restreinte à un objectif.

Il apparaît que la seconde approche est nettement plus cohérente. En effet, celle-ci permet, sans modification, d'utiliser la génération de colonnes pour réaliser la première phase du simplexe paramétrique en fournissant la première solution efficace au problème biobjectif, à savoir la solution pour $\lambda = 1$. Toutefois, dans la perspective d'obtenir l'ensemble des solutions efficaces, un certain nombre de difficultés additionnelles doivent être mises en lumière.

6.4.1.2 Difficultés anticipées

La génération de colonnes, bien que fournissant la solution optimale continue selon un objectif, s'achève avec une formulation incomplète, c'est-à-dire un sous-ensemble des variables du problème. Plus précisément, son utilisation signifie justement qu'il n'est pas souhaitable (ou pas possible) de prendre toutes les variables en compte au sein d'un algorithme de résolution.

La seconde phase du simplexe paramétrique nécessite le calcul du coût réduit de toutes les variables relativement aux deux objectifs. Classiquement, chaque opération de pivot réalisée pendant la seconde phase du simplexe paramétrique induit un recalcul de la totalité des coûts réduits des variables pour les deux objectifs. L'ensemble de ces coûts réduits est en outre nécessaire pour calculer les valeurs

successives du paramètre λ et sélectionner le prochain pivot à réaliser. Par extension, la connaissance de la totalité des coûts réduits selon les deux objectifs est donc indispensable à la détermination des solutions efficaces.

De ce fait, l'absence d'une grande partie des variables de la formulation, du fait de l'utilisation de la génération de colonnes comme première phase, est problématique pour la deuxième phase du simplexe paramétrique. Ceci peut être d'autant plus marqué dans le cas d'objectifs négativement corrélés, puisqu'il est alors probable que les variables générées lors de la première phase soient particulièrement peu pertinentes en regard du second objectif. Les solutions trouvées par la deuxième phase du simplexe paramétrique seront en conséquence efficaces pour le problème restreint aux variables générées, mais seront probablement dominées pour la formulation complète.

Ainsi, dans la perspective d'obtenir l'ensemble des solutions efficaces, il est nécessaire de calculer également le coût réduit selon les deux objectifs de l'ensemble des variables ne figurant pas dans la formulation à l'issue de la génération de colonnes. Ceci nécessite donc, de façon analogue à ce qui est réalisé dans le cadre de la génération de colonnes par la résolution du sous-problème, de calculer les coûts réduits des variables non encore présentes dans la formulation. Ceci revient donc à résoudre deux sous-problèmes à chaque itération de la seconde phase du simplexe paramétrique, à savoir un pour chaque objectif. Nous voyons donc ici que les performances de l'algorithme peuvent être aisément dégradées si au moins l'un des deux sous-problèmes est coûteux à résoudre en terme de temps de calcul.

Enfin, tout comme la génération de colonnes peut nécessiter un grand nombre d'itérations pour atteindre la solution optimale, le simplexe paramétrique peut également nécessiter un grand nombre de pivots afin de trouver l'ensemble des solutions efficaces. Toutefois, si le nombre d'itérations dans une procédure de génération de colonnes peut être influencé par la qualité de la sélection des colonnes à chaque itération, la recherche de l'ensemble de toutes les solutions efficaces de la formulation biobjectif nécessite quant à elle dans tous les cas exactement une itération par solution efficace. Il n'y a de ce fait *a priori* pas de levier permettant de réduire le nombre d'itérations tout en recherchant l'ensemble des solutions efficaces.

Une piste à explorer afin de pallier à ce problème peut consister à sacrifier l'exactitude de l'ensemble des solutions efficaces en ne cherchant le nouveau pivot qu'au sein d'un sous-ensemble de toutes les variables :

- soit en n'effectuant la recherche qu'au sein des variables générées, ce qui peut toutefois laisser anticiper des résultats de qualité moyenne si les objectifs sont négativement corrélés, tel que nous l'avons expliqué au-dessus ;
- soit en limitant la recherche à un sous-ensemble de toutes les colonnes possibles, sans pour autant se cantonner à l'ensemble généré lors de la première phase.

Toutefois, selon la formulation, il peut être possible de profiter de sous-problèmes pouvant être résolus aisément de façon complète. Ceci dépend bien entendu de la structure de la formulation.

Il est donc nécessaire de s'intéresser au calcul des coûts réduits selon chacun des objectifs afin d'évaluer la difficulté de résolution des sous-problèmes associés et leur impact sur l'ensemble du problème. En comparant leurs expressions respectives pour les deux formulations proposées, il est possible d'identifier les opportunités algorithmiques à mettre en place.

6.4.2 Formulations duales et coût réduits

6.4.2.1 Équité explicite

En restreignant successivement la formulation biobjectif 6.9 à chacun de ses deux objectifs, il est possible d'écrire deux formulation duales. La formulation duale correspondant au primal restreint au premier objectif, celui du SPP issu du problème de saturation, est donnée par l'expression 6.13 :

$$\left[\begin{array}{lll} \min & \sum_{t\in T} v_t + \sum_{u\in U} w_u & \\ \text{s.c.} & v_t + \sum_{m\in M} \delta_{c,u} w_u - \sum_{F\in\mathcal{F}} \frac{1}{\Gamma_F}\sigma_{F,t}\rho_{t,c}y_F & \geq \rho_{t,c} \quad ,\forall t\in T, c\in C_t \\ & \sum_{F\in\mathcal{F}} y_F & = 0 \\ & v_t & \geq 0 \quad ,\forall t\in T \\ & w_u & \geq 0 \quad ,\forall u\in U \\ & y_F & \geq 0 \quad ,\forall F\in\mathcal{F} \end{array}\right] \tag{6.13}$$

La formulation duale du problème restreint au second objectif est quant à elle décrite par l'expression 6.14 :

$$\left[\begin{array}{lll} \min & \sum_{t\in T} v_t + \sum_{u\in U} w_u & \\ \text{s.c.} & v_t + \sum_{u\in U} \delta_{c,u} w_u - \sum_{F\in\mathcal{F}} \frac{1}{\Gamma_F}\sigma_{F,t}\rho_{t,c}y_F & \geq 0 \quad ,\forall t\in T, c\in C_t \\ & \sum_{F\in\mathcal{F}} y_F & = 1 \\ & v_t & \geq 0 \quad ,\forall t\in T \\ & w_u & \geq 0 \quad ,\forall u\in U \\ & y_F & \geq 0 \quad ,\forall F\in\mathcal{F} \end{array}\right]. \tag{6.14}$$

L'ajout de colonnes dans le primal peut se faire par rapport à l'un ou à l'autre des objectifs. Ainsi, il est ici aussi possible d'exprimer les coûts réduits par la recherche d'une contrainte violée dans le dual correspondant à la formulation restreinte au premier ou au second objectif. Dans le cas du premier objectif, il s'agit d'un couple $(t,c),\ t\in T, c\in C_t$ qui est recherché, de façon à vérifier l'expression 6.15 :

$$v_t^* + \sum_{u\in U} \delta_{c,u} w_u^* - \sum_{F\in\mathcal{F}} \frac{\sigma_{F,t}\rho_{t,c}y_F^*}{\Gamma_F} < \rho_{t,c} \tag{6.15}$$

où l'égalité $y_F^* = 0,\ \forall F\in\mathcal{F}$ est nécessairement vérifiée, impliquant finalement que la formule du coût réduit pour le premier objectif reste inchangée par rapport à la formule utilisée pour la résolution de la formulation mono-objectif, correspondant donc à celle qui est donné par l'expression 4.2. Pour davantage de clarté au regard des deux objectifs, nous notons ici ce coût réduit $\pi^1_{t,c}$.

Insérer une colonne permettant d'améliorer le second objectif se fait en cherchant également un couple de la forme (t,c), violant cette fois une contrainte dans la formulation duale par rapport au problème

primal restreint au second objectif. Le coût réduit $\pi^2_{t,c}$ pour le second objectif est alors donné par l'équation 6.16 :

$$\pi^2_{t,c} = \sum_{F \in \mathcal{F}} \frac{\sigma_{F,t}\rho_{t,c}y^*_F}{\Gamma_F} - v^*_t - \sum_{u \in U} \delta_{c,u} w^*_u \tag{6.16}$$

celui-ci devant être strictement positif pour que la colonne correspondante soit susceptible d'améliorer la valeur de la seconde fonction-objectif.

Ainsi, si la formule du coût réduit reste logiquement inchangée pour le premier objectif, celle du second objectif comporte une nouvelle composante qui dépend de l'ensemble des variables duales y_F associées au nouveau jeu de contraintes présentes dans la formulation primale.

6.4.2.2 Pondération dynamique

Au sein de la seconde formulation proposée, l'objectif ajouté à la formulation est extrêmement similaire au premier. Contrairement au cas précédent, aucune variable et aucune contrainte ne sont ajoutées à la formulation, et uniquement les coefficients des variables déjà considérées sont modifiés.

De ce fait, il est possible de formuler directement le coût réduit associé à un couple (t, c) selon le second objectif. L'expression est donnée par l'équation 6.17 :

$$\pi^2_{t,c} = \phi_{t,c} - v^*_t - \sum_{u \in U} \delta_{c,u} w^*_u. \tag{6.17}$$

Il s'agit bien entendu d'une expression tout à fait similaire à celle du coût réduit pour le premier objectif, donné par l'expression 4.2, puisque l'expression des coûts réduits diffère alors par une simple constante :

$$\pi^2_{t,c} = \pi^1_{t,c} + (\phi_{t,c} - \rho_{t,c}).$$

6.4.3 Mise en place

Nous avons mis en évidence, d'une part, certaines difficultés inhérentes à l'obtention de l'ensemble des solutions efficaces par la combinaison du simplexe paramétrique avec la génération de colonnes. Sans être identiques, ces difficultés sont comparables à celles qui sont rencontrées lors de la conception d'un algorithme de génération de colonnes. Elles sont liées à deux causes principales :

- le temps de calcul qui peut être nécessaire pour résoudre les sous-problèmes liés à chacun des objectifs (dans la génération de colonnes classique, il n'y a bien entendu qu'un seul sous-problème à résoudre) ;
- l'amplification de ce phénomène du fait du caractère itératif du simplexe paramétrique.

Nous avons vu deux formulations biobjectifs : l'une fournit un critère explicite d'équité et à ce titre garantit théoriquement l'obtention de solutions optimisant cette quantité, alors que l'autre ne fournit pas cette garantie mais évite de modifier la structure de la formulation considérée initialement. En particulier, la formulation des coûts réduits selon le second objectif est quasiment similaire à celle du premier objectif, puisqu'elles ne diffèrent que par une constante.

La procédure de résolution du sous-problème implémentée dans le cadre de la génération de colonnes se montrant simple et performante, sa réutilisation dans le cadre de la résolution biobjectif paraît donc avantageuse du point de vue de la simplicité de mise en place. Dans une perspective de mise en place d'un premier algorithme combinant génération de colonnes et résolution biobjectif, nous faisons donc le choix de réutiliser la procédure de résolution du sous-problème. Nous nous concentrons de ce fait sur la

deuxième formulation biobjectif proposée. Le calcul des coûts réduits selon le second objectif peut donc se faire simultanément au calcul des coûts réduits selon le premier objectif.

Conformément au principe du simplexe paramétrique, la résolution biobjectif consiste donc finalement, ici, à résoudre une série de problèmes de PL mono-objectifs paramétrés par $\lambda \in [0,1]$. Leur expression est donnée par la formulation 6.18 :

$$\left[\begin{array}{lll} \max & \sum_{t\in T, c\in C_t} (\lambda\rho_{t,c} + (1-\lambda)\phi_{t,c})x_{t,c} & \\ \text{s.c.} & \sum_{c\in C_t} x_{t,c} & \leq 1 \quad ,\forall t \in T \\ & \sum_{t\in T, c\in C_t} \delta_{c,u} x_{t,c} & \leq 1 \quad ,\forall u \in U \\ & x_{t,c} & \geq 0 \quad ,\forall t \in T, \forall c \in C_t \end{array}\right]. \tag{6.18}$$

Le calcul des valeurs successives du paramètre λ nécessite par la suite la connaissance des coûts réduits selon les deux objectifs pour l'ensemble des variables, et en particulier pour celles qui sont hors-base, puisque c'est parmi celles-ci qu'une variable entrante sera sélectionnée pour retrouver l'optimalité après la sélection de la nouvelle valeur pour λ. Pour l'ensemble des colonnes hors base dont il est nécessaire d'inspecter la valeur de coût réduit, deux cas de figure sont possibles :

- la colonne est déjà présente dans la formulation, auquel cas sont coût réduit est disponible suite à l'opération de pivot précédente ;
- la colonne n'a pas encore été insérée dans la formulation, auquel cas son coût réduit ne peut être obtenu que par résolution du sous-problème.

Le premier cas n'implique pas de modification du simplexe paramétrique, et rentre donc dans le cadre de la procédure classique présentée par l'algorithme 6.1. Le second nécessite en revanche le calcul des coûts réduits de la colonne concernée *via* la résolution des sous-problèmes.

L'algorithme 6.2 synthétise l'approche utilisée pour combiner la génération de colonnes, telle que nous l'avons développée au chapitre 4, avec une résolution biobjectif reprenant le principe du simplexe paramétrique. Outre la procédure de génération de colonnes en lieu et place de la première phase du simplexe paramétrique (ligne 1), l'algorithme intègre une seconde forme de génération de colonnes au cours de la seconde phase du simplexe paramétrique, celle-ci étant démarrée après le calcul du second jeu de poids des variables à la ligne 2 en utilisant la formule 6.12. Les lignes 3, 13 et 9 permettent la prise en compte des colonnes ne figurant pas dans la formulation, d'une part en calculant leur coût réduit et d'autre part en insérant si nécessaire une variable devant entrer en base. La prise en compte de l'agrégation de contraintes effectuées au sein de la génération de colonnes se fait à la ligne 10, puisque les éventuelles contraintes ayant été supprimées par agrégation sont ajoutées à la formulation si cela est requis par la nouvelle variable. Globalement, l'algorithme est conçu, de façon similaire au simplexe paramétrique, pour retourner finalement l'ensemble des solutions efficaces du problème de PL biobjectif défini par la formulation 6.11.

Cette procédure algorithmique a donné lieu à une implémentation visant une intégration de la résolution biobjectif avec la génération de colonne, elle-même reposant sur la bibliothèque logicielle Coin-OR pour l'exécution du simplexe et l'interfaçage avec les données que celui-ci fournit. L'interfaçage avec Coin-OR se fait à plusieurs niveaux :

- la résolution du simplexe mono-objectif au sein de la génération de colonnes et la récupération d'informations telles que la solution duale au PMRA issues de cette résolution ainsi que les coûts réduits des variables de la formulation (ligne 1) ;

Entrées : $\mathcal{C}$, $\mathcal{L}$: ensembles respectifs des variables et contraintes définissant la formulation 3.20.

$(\mathcal{B}, \overline{x}, \overline{v}, \overline{w}, \mathcal{C}_{pmra}, \mathcal{L}_{pmra}) \leftarrow$ `gc` ($\mathcal{C}$, $\mathcal{L}$) ;

$\phi \leftarrow$ `poidsequité` ($\overline{x}$, ρ, $\mathcal{F}$) ;

$(\pi^1, \pi^2) \leftarrow$ `coutsréduits` ($\overline{v}$, $\overline{w}$) ;

tant que $\mathcal{J} = \{j \notin \mathcal{B},\ \pi_j^1 \leq 0,\ \pi_j^2 > 0\} \neq \emptyset$ **faire**

  $\lambda \leftarrow \max_{j \in \mathcal{J}} \frac{-\pi_j^2}{\pi_j^1 - \pi_j^2}$;

  $j_e \leftarrow \text{argmax}_{j \in \mathcal{J}} \left(\frac{-\pi_j^2}{\pi_j^1 - \pi_j^2}\right)$;

  $j_s \leftarrow$ `indicesortie` ($\mathcal{B}$, j_e) ;

  si $j_e \notin \mathcal{C}_{pmra}$ **alors**

    $\mathcal{C}_{pmra} \leftarrow \mathcal{C}_{pmra} \cup \{j_e\}$;

    $\mathcal{L}_{pmra} \leftarrow \mathcal{L}_{pmra} \cup \{$contraintes couvertes par $j_e\}$;

  fin

  $(\mathcal{B}, \pi^1, \pi^2) \leftarrow$ `pivot` ($\mathcal{B}$, j_e, j_s) ;

  $(\pi^1, \pi^2) \leftarrow$ `coutsréduits` ($\overline{v}$, $\overline{w}$) ;

fin

Sorties : séquence de valeurs pour λ ; solution optimale pour chacune de ces valeurs.

Algorithme 6.2: Combinaison de la génération de colonnes et du simplexe paramétrique.

– l'insertion de nouvelles colonnes et contraintes au cours de chacune des deux phases de l'algorithme (lignes 1, 10 et 9) ;
– l'exécution d'opérations de pivot individuelles au cours de la seconde phase (ligne 12).

La réalisation des pivots individuels se révèle délicate à mettre en œuvre, en particulier en conjonction avec la récupération de la valeur des coûts réduits selon les deux objectifs, la bibliothèque Coin-OR Clp n'étant pas conçue pour la résolution multiobjectif. Il suit de ces difficultés d'implémentation qu'il n'a pas été possible de mener à bien une résolution biobjectif complète. Toutefois, plusieurs itérations de la seconde phase de l'algorithme ont été effectuées avec succès sur plusieurs instances, nous permettant d'observer le phénomène de décroissante de λ.

Une implémentation pleinement fonctionnelle de cet algorithme figure donc plutôt au sein des perspectives les plus immédiates de ce travail, ainsi que la réalisation de tests complets permettant d'évaluer plusieurs points principaux pour cet algorithme, à savoir :

– la performance globale de l'algorithme en terme de temps de calcul ainsi que la proportion de temps consommée par chacune des deux phases ;
– la pertinence de la mesure d'équité au sein des solutions continues obtenues suite à la résolution de la formulation traitée ici ;
– le nombre de solutions efficaces obtenues ;
– le nombre de nouvelles variables générées lors de la seconde phase et la mesure dans laquelle elles diffèrent des variables issues de la première phase.

Par la suite, du fait des garanties théoriques fournies par le modèle 6.9, qui inclut l'optimisation d'un critère d'équité explicitement, il apparaît particulièrement intéressant de mettre en place une procédure de résolution pour ce modèle. Celle-ci peut bien entendu réutiliser le principe développé au sein de l'algorithme 6.2, l'adaptation majeure devant être réalisée concernant le sous-problème relatif au second objectif.

Pour terminer, ces modèles ouvrent également la voie à l'étude d'un algorithme de résolution biobjectif en nombres entiers, puisque le but final est l'obtention d'un ensemble de grilles horaires de compromis, représentées par des solutions efficaces dont les variables sont à valeur entière. Des méthodes génériques en ce sens existent dans la littérature et peuvent de ce fait servir servir de base à une méthode adaptée aux modélisations biobjectifs que nous avons proposées. Un ensemble de voies intéressantes liées à l'estimation de la capacité d'infrastructures ferroviaire sont donc ouvertes par les présents travaux.

CHAPITRE 7

Conclusion et perspectives

Nous avons présenté dans ce mémoire les avancées d'ordres algorithmique et mathématique que nous avons apportées pour résoudre les problématiques liées à l'estimation de la capacité d'infrastructures ferroviaires. Comme nous l'avons développé au chapitre 2, l'estimation de capacité est un problème crucial dans la gestion prévisionnelle du trafic ferroviaire, et il est de plus particulièrement d'actualité en Europe. Nous avons vu l'existence de diverses plateformes logicielles liées à l'étude de capacité, répondant souvent à des problématiques spécifiques ne correspondant pas exactement aux besoins que nous formulons, à savoir la résolution du problème de saturation à échelle microscopique. La plateforme RECIFE PC, dédiée à de tels calculs, intégrait jusqu'ici une métaheuristique, ACO, dont les performances se sont toutefois révélées insuffisantes sur certaines instances auxquelles nous nous sommes intéressés.

Le processus de modélisation du problème, de conception d'un algorithme de résolution puis de test et de validation de cet algorithme a été présenté tout au long des chapitres 3, 4 et 5. Le chapitre 6 a finalement apporté une contribution en terme de modélisation dans le but de pallier à un problème identifié au sein des solutions générées par l'algorithme présenté aux chapitres précédents.

À travers la conception de la méthode basée sur la génération de colonnes ainsi que par les propositions formulées suite à la critique des solutions obtenues par cette méthode, nous avons apporté des contributions selon deux axes : le domaine ferroviaire d'une part, et le développement de méthodes algorithmiques d'autre part.

Contributions du point de vue ferroviaire

D'un point de vue ferroviaire, l'apport de notre contribution consiste d'abord en une modélisation sous forme de problème de PLNE pour le problème de saturation. Elle est inspirée de travaux réalisés précédemment dans le cadre du problème de faisabilité et diffère de ce fait des modélisations utilisées jusqu'ici pour le problème de saturation. Son utilisation est justifiée par une étude de ses propriétés mathématiques avantageuses, à savoir :

- la qualité de sa relaxation continue, celle-ci se révélant meilleure que la modélisation NPP sans nécessiter de pré-traitement ;
- son nombre de contraintes indépendant du nombre de chemins considérés.

Par la suite, la majeure partie de notre contribution, *via* la méthode algorithmique présentée, vient apporter plusieurs améliorations à l'existant du point de vue des résultats obtenus :

- un temps de résolution réduit substantiellement, permettant notamment d'envisager de considérer des instances de taille nettement plus grande, élargissant ainsi les perspectives en termes de don-

nées ferroviaires pouvant être considérées : un nombre de trains plus élevé et/ou un degré de liberté accru en terme de retards possibles ;
- une consommation mémoire plus faible de l'algorithme, ce qui est bénéfique dans le cadre de l'intégration de l'algorithme à une plateforme logicielle ;
- le calcul d'une borne supérieure pour la solution qui, bien qu'elle soit de qualité variable, fournit dans certains cas un indicateur pertinent au décideur et plus généralement ouvre la voie à l'utilisation de méthodes standard de PL pour améliorer cette borne, puisque celle-ci est calculée par la relaxation continue de la formulation SPP.

Nous avons en outre mis en lumière le problème d'équité apparaissant au sein des solutions produites par la résolution du problème de saturation et avons proposé des modèles visant à y pallier. Ces apports, même s'ils n'aboutissent pas présentement à des résultats chiffrés, constituent une contribution du point de vue ferroviaire en ébauchant une méthode de résolution pouvant à terme permettre à un décideur l'obtention d'un panel de solutions de compromis au sein desquelles l'équité est prise en compte.

Enfin, d'un point de vue pratique, la plateforme logicielle RECIFE PC se retrouve améliorée par le biais de l'intégration du nouvel algorithme : il en ressort un logiciel plus performant pour la résolution de la saturation et à la modularité accrue dans une perspective de futurs ajouts de nouveaux algorithmes de résolution. L'intégration réutilise en effet l'outil de visualisation de la progression de l'algorithme déjà en place pour ACO, tout en modifiant les données affichées afin de les rendre plus pertinentes avec l'implémentation de la génération de colonnes. Du point de vue de la modularité, une décorrélation plus importante de l'interface et de la partie algorithmique a été réalisée, dans la perspective de faciliter le futur travail d'intégration d'éventuels algorithmes supplémentaires.

Contributions algorithmiques

Des contributions ont été apportées d'un point de vue algorithmique, en particulier du point de vue de la PL. En combinant plusieurs techniques, nous avons développé un algorithme de génération de colonnes au sein duquel nous avons soigné les performances de trois composants :
- la résolution du sous-problème se sert des similarités entre chemins correspondant à des décalages temporels consécutifs pour un même itinéraire afin de calculer l'ensemble des coûts réduits plus efficacement ;
- l'identification de critères pertinents en regard du problème considéré pour insérer des colonnes dans la formulation de façon intelligente, et ainsi se diriger le plus rapidement possible vers la solution optimale sans pour autant faire augmenter trop rapidement la taille du PMRA ;
- le calcul d'une borne duale à chaque itération de la génération de colonnes par le calcul d'une solution réalisable au dual de la formulation complète.

Nous avons également montré la pertinence de réaliser une agrégation dynamique de contraintes ne se réduisant pas aux contraintes redondantes, mais recherchant également les contraintes dominées. La mise en place et l'exécution de ce principe générique au sein de la génération de colonnes se montre ainsi rentable en terme de temps de calcul, notamment grâce à l'exploitation, ici aussi, de données spécifiques au problème. En outre, au-delà du seul algorithme d'agrégation, nous avons vu une méthode de désagrégation de la solution duale permettant le calcul des coûts réduits de façon cohérente avec la saturation des contraintes par la solution optimale au PMRA. Enfin, dans la large thématique des hybridations d'algorithmes exacts et approchés, l'algorithme conçu montre qu'un algorithme exact, en l'occurrence une génération de colonnes, peut se révéler bénéfique comme pré-traitement à un algorithme approché, ici ACO, pour lui fournir des informations pertinentes tout en réduisant le temps global de résolution.

Le caractère modulaire de la génération de colonnes nous a finalement permis, à travers la nécessité d'intégrer l'équité à l'estimation de capacité, d'en présenter une tentative de combinaison avec le simplexe paramétrique. Par ce biais, il peut être envisagé de résoudre des problèmes biobjectifs contenant un nombre de variables *a priori* rédhibitoire pour le simplexe paramétrique classique. Bien que l'implémentation de ce travail ait été confrontée à des difficultés d'ordre technique ne permettant pas d'obtenir des résultats expérimentaux, nous avons cherché à mettre en lumière les composants pouvant impacter fortement cet algorithme, notamment la nécessité de résoudre un sous-problème pour chaque objectif. Ainsi, la formalisation algorithmique présentée dans le chapitre 6 est susceptible d'ouvrir des perspectives à de nouvelles contributions dans le domaine de l'optimisation multiobjectif.

Perspectives

En premier lieu, les deux modélisations biobjectifs et l'algorithme suggérés visant à pallier au problème d'équité peuvent ouvrir la voie à de nouvelles contributions, celles-ci pouvant être envisagées selon deux axes.

Premièrement, de façon spécifique à la capacité des infrastructures ferroviaires, une implémentation fonctionnelle de l'algorithme biobjectif que nous proposons pour améliorer l'équité des solutions semble être une perspective immédiate. Par la suite, il peut être particulièrement intéressant de se pencher sur la résolution du modèle biobjectif intégrant le critère d'équité explicite que avons proposé. De tels travaux peuvent buter sur les problèmes que nous avons identifiés, ce qui peut donner l'opportunité d'explorer des pistes permettant une résolution biobjectif performante. De façon similaire à l'approche que nous avons eue, ces pistes sont à rechercher à la fois dans des principes techniques génériques issus du domaine de l'optimisation ainsi que dans l'exploitation d'éléments spécifiques. Enfin, il reste nécessaire de proposer une solution algorithmique permettant l'obtention de solutions entières efficaces. Il peut être dans un premier temps envisageable d'étudier les solutions fournies par ACO suite à la résolution biobjectif de la relaxation continue. Toutefois, le fait que cette implémentation de la métaheuristique ne soit *a priori* pas conçue pour la résolution biobjectif impliquera probablement la nécessité de développer de nouvelles méthodes.

En second lieu, la combinaison entre génération de colonnes et simplexe paramétrique est une approche qui semble être peu abordée au sein de la littérature existante. Le principe algorithmique que nous proposons peut être généralisé afin d'être examiné sous l'angle d'un « algorithme de génération de colonnes biobjectif », sa validité en tant que tel devant alors être étudiée. Ainsi, la conception plus générique d'un algorithme de génération de colonnes permettant la résolution de problèmes de PL biobjectifs de grande taille est une piste qui semble intéressante à explorer du point de vue plus générique du domaine de l'optimisation.

Outre le critère d'équité et l'aspect biobjectif de la résolution qui en découle, d'autres perspectives algorithmiques peuvent être envisagées. Du point de vue de la seule résolution de la relaxation continue du problème de saturation, le caractère générique et modulaire de la génération de colonnes donne la possibilité d'implémenter des techniques variées dans le but d'améliorer le temps de résolution ou encore la qualité de relaxation continue. Il est par exemple possible d'envisager une recherche d'inégalités valides soit au cours de la procédure de génération de colonnes soit en post-traitement de celle-ci, puisque nous avons vu que la qualité de la relaxation continue semble pouvoir être nettement améliorée, ce qui permettrait une réduction de l'écart et une meilleure estimation de la qualité de la solution entière.

Dans la plus large perspective du calcul de capacité d'infrastructures ferroviaires, il reste nécessaire d'intégrer les différents problèmes sous-jacents, que sont notamment l'optimisation des préférences et

la robustesse, dans un algorithme résolution les prenant en compte, dans la mesure du possible, simultanément. Même si ces problèmes sont clairement formalisés et modélisés, la résolution multiobjectif de façon efficace reste une piste particulièrement importante à explorer.

Bibliographie

[1] M. Abril, F. Barber, L. Ingolotti, M.A. Salido, P. Tormos, and A. Lova. An assessment of railway capacity. *Transportation Research Part E : Logistics and Transportation Review*, 44(5) :774–806, 2008.

[2] M. Abril, M.A. Salido, F. Barber, L. Ingolotti, A. Lova, and P. Tormos. A Heuristic Technique for the Capacity Assessment of Periodic Train. *Frontiers in Artificial Intelligence and Applications*, 131 :339–346, 2005.

[3] J.B. Barbanel and S.J. Brams. Cake division with minimal cuts : envy-free procedures for three persons, four persons, and beyond. *Mathematical Social Sciences*, 48(3) :251–269, 2004.

[4] F. Barber, L. Ingolotti, M.A. Salido, M. Abril, P. Tormos, and A. Lova. A decision support system for railway timetabling (MOM) : the Spanish case. In *Computers in Railways X. The Tenth International Conference, Prague, Czech Republic*, pages 235–244, 2006.

[5] C. Berge. Balanced matrices. *Mathematical Programming*, 2(1) :19–31, 1972.

[6] C. Blum, M.J.B. Aguilera, A. Roli, and M. Sampels, editors. *Hybrid Metaheuristics : An Emerging Approach to Optimization*, volume 114 of *Studies in Computational Intelligence*. Springer, 2008.

[7] R. Borndörfer and T. Schlechte. Models for railway track allocation. In *7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, Sevilla, Spain*, pages 62–78, 2007.

[8] D.M. Burkolter. *Capacity of Railways in Station Areas using Petri Nets*. PhD thesis, Swiss Federal Institute of Technology, 2005.

[9] V. Cacchiani, A. Caprara, and P. Toth. A column generation approach to train timetabling on a corridor. *4OR : A Quarterly Journal of Operations Research*, 6(2) :125–142, 2008.

[10] A. Caprara, M. Fischetti, and P. Toth. Modeling and Solving the Train Timetabling Problem. *Operations Research*, 50(5) :851–861, 2002.

[11] A. Caprara, M. Monaci, P. Toth, M. Fischetti, and P.L. Guida. Solution of real-world train timetabling problems. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences, Maui, United States of America*, pages 1057–1066, 2001.

[12] V. Chvátal. *Linear programming*. W.H. Freeman, 1983.

[13] D.P. Clements, J.M. Crawford, D.E. Joslin, G.L. Nemhauser, M.E. Puttlitz, and M.W.P. Savelsbergh. Heuristic Optimization : A hybrid AI/OR approach. In *Workshop on Industrial Constraint-Directed Scheduling*, 1997.

[14] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed Optimization by Ant Colonies. In *First European Conference on Artificial Life (ECAL), Paris*, pages 134–142, 1991.

[15] G. Confessore, G. Liotta, P. Cicini, F. Rondinone, and P. De Luca. A simulation-based approach for estimating the commercial capacity of railways. In *Proceedings of the 2009 Winter Simulation Conference (WSC), Austin, United States of America*, pages 2542–2552, 2010.

[16] G.B. Dantzig. *Maximization of a linear function of variables subject to linear inequalities*, chapter 4. John Wiley and Sons, New York, 1951.

[17] G.B. Dantzig, A. Orden, and P. Wolfe. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2) :183–195, 1955.

[18] A. D'Ariano, F. Corman, and I.A. Hansen. Railway traffic optimization by advanced scheduling and rerouting algorithms. In *Eighth World Congress on Railway Research (WCRR 2008), Séoul, Corée*, mai 2008.

[19] A. D'Ariano, D. Pacciarelli, and M. Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2) :643–657, 2007.

[20] X. Delorme. *Modélisation et résolution de problèmes liés à l'exploitation d'infrastructures ferroviaires*. PhD thesis, Université de Valenciennes et du Hainaut-Cambrésis, 2003.

[21] X. Delorme, X. Gandibleux, and J. Rodriguez. GRASP for set packing problems. *European Journal of Operational Research*, 153(3) :564–580, 2004.

[22] X. Delorme, X. Gandibleux, and J. Rodriguez. Stability evaluation of a railway timetable at station level. *European Journal of Operational Research*, 195(3) :780–790, 2009.

[23] Conseil des communautés européennes. Directive 91/440/CEE relative au développement des chemins de fer communautaires. *Journal officiel*, L237, 29 juillet 1991.

[24] Conseil des communautés européennes. Directive 2001/14/CE concernant la répartition des capacités d'infrastructure ferroviaire, la tarification de l'infrastructure ferroviaire et la certification en matière de sécurité. *Journal officiel*, L75, 26 février 2001.

[25] I. Dumitrescu and T. Stützle. Combinations of Local Search and Exact Algorithms. In *Applications of Evolutionary Computing*, volume 2611 of *Lecture Notes in Computer Science*, pages 57–68. 2003.

[26] M. Ehrgott. *Multicriteria optimization*, volume 491. Springer Verlag, 2005.

[27] I. Elhallaoui, A. Metrane, F. Soumis, and G. Desaulniers. Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming*, 123(2) :345–370, 2010.

[28] I. Elhallaoui, D. Villeneuve, F. Soumis, and G. Desaulniers. Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research*, 53(4) :632–645, 2005.

[29] G.R. Filho and L.A.N. Lorena. Constructive Genetic Algorithm and Column Generation : an Application to Graph Coloring. In *Fifth Conference of The Association of Asian-Pacific Operations Research Societies Within IFORS, Singapore*, juillet 2000.

[30] X. Gandibleux, X. Delorme, and V. T'Kindt. An Ant Colony Optimisation Algorithm for the Set Packing Problem. *Ant Colony Optimization and Swarm Intelligence : 4th International Workshop, ANTS 2004, Bruxelles, Belgique*, 2004.

[31] X. Gandibleux, J. Jorge, S. Angibaud, X. Delorme, and J. Rodriguez. An ant colony optimization inspired algorithm for the set packing problem with application to railway infrastructure. In *Proceedings of the Sixth Metaheuristics International Conference (MIC2005), Vienne, Autriche*, pages 390–396, 2005.

[32] X. Gandibleux, J. Jorge, X. Delorme, and J. Rodriguez. An ant algorithm for measuring and optimizing the capacity of a railway infrastructure. In N. Monmarché, F. Guinand, and P. Siarry, editors, *Artificial Ants*, volume 1, chapter 9. ISTE Ltd and John Wiley & Sons Inc, 2010.

[33] X. Gandibleux, P. Riteau, and X. Delorme. RECIFE : A MCDSS for Railway Capacity Evaluation. *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems, Lecture Notes in Economics and Mathematical Systems*, 634(1) :93–103, 2010.

[34] P. Hachemane. *Évaluation de la capacité de réseaux ferroviaires*. PhD thesis, École Polytechnique Fédérale de Lausanne (Suisse), 1997.

[35] V. Klee and G.J. Minty. How good is the simplex algorithm ? *Proceedings of the Third Symposium on Inequalities, Los Angeles, États-Unis d'Amérique*, pages 159–175, 1972.

[36] E. Kontaxi and S. Ricci. Calculation of railway network capacity : comparing methodologies for lines and nodes. In *Fourth International Seminar on Railway Operations Modelling and Analysis (RailRome 2011), Rome, Italie*, février 2011.

[37] L.G. Kroon, H.E. Romeijn, and P.J. Zwaneveld. Routing trains through railway stations : complexity issues. *European Journal of Operational Research*, 98(3) :485–498, 1997.

[38] A. Kuckelberg, E. Wendler, and T. Gröger. A UIC 406-compliant, practically relevant capacity-consumption evaluation algorithm. In *Fourth International Seminar on Railway Operations Modelling and Analysis (RailRome 2011), Rome, Italie*, février 2011.

[39] V. Labouisse and H. Djellab. DEMIURGE : A tool for the optimisation and the capacity assessment for railway infrastructure. In *Fifth World Congress on Railway Research (WCRR 2011), Cologne, Allemagne*, novembre 2001.

[40] A. Landex, A. H. Kaas, B. Schittenhelm, and J. Schneider-Tilli. Evaluation of railway capacity. In *Proceedings of Trafficdays*, 2006.

[41] A. Libardo, P. Pellegrini, and G. Salerno. Capacity in Railway Junctions and Optimal Rouge Management. In *Fourth International Seminar on Railway Operations Modelling and Analysis (RailRome 2011), Rome, Italie*, février 2011.

[42] C. Liebchen. *Periodic Timetable Optimization in Public Transport*. PhD thesis, Technische Universität Berlin, 2006.

[43] T. Lindner. Applicability of the analytical compression for evaluating node capacity. In *Fourth International Seminar of Railway Operations Modelling and Analysis (RailRome 2011), Rome, Italie*, février 2011.

[44] L. Lucchini, A. Curchod, and R. Rivier. Transalpine rail network : A capacity assessment model (capres). In *First Swiss Transport Research Conference, Monte Verità, Suisse*, 2001.

[45] R. Lusby, J. Larsen, M. Ehrgott, and D. Ryan. Railway Track Allocation : Models and Methods. *OR Spectrum*, pages 1–41, 2009.

[46] R. Lusby, J. Larsen, D. Ryan, and M. Ehrgott. Routing Trains Through Railway Junctions : A New Set-Packing Approach. *Transportation Science*, 45(2) :228–245, 2011.

[47] R.M. Lusby. *Optimization Methods for Routing Trains Through Railway Junctions*. PhD thesis, The University of Auckland, 2008.

[48] R.M. Lusby, J. Larsen, M. Ehrgott, and D. Ryan. A set packing inspired method for real-time junction train routing. Technical report, DTU, 2009.

[49] L. Massoulié and J. Roberts. Bandwidth sharing : objectives and algorithms. *IEEE/ACM Transactions on Networking*, 10(3) :320–328, 2002.

[50] A. Merel. Résolution exacte du problème de capacité d'infrastructures ferroviaires. Master's thesis, École des Mines de Nantes, Nantes, juillet 2008.

[51] A. Merel, X. Gandibleux, and S. Demassey. A Collaborative Combination between Column Generation and Ant Colony Optimization for Solving Set Packing Problems. In *Proceedings of the Ninth Metaheuristics International Conference (MIC2011), Udine, Italie*, juillet 2011.

[52] A. Merel, X. Gandibleux, and S. Demassey. Assessing Railway Infrastructure Capacity by Solving the Saturation Problem with an Improved Column Generation Algorithm. In *Fourth International Seminar on Railway Operations Modelling and Analysis (RailRome 2011), Rome, Italie*, février 2011.

[53] A. Merel, X. Gandibleux, and S. Demassey. Towards a Realistic Evaluation of Railway Infrastructure Capacity. In *Ninth World Congress on Railway Research (WCRR 2011), Lille*, mai 2011.

[54] A. Merel, X. Gandibleux, S. Demassey, and R. Lusby. An improved Upper Bound for the Railway Infrastructure Capacity Problem on the Pierrefitte-Gonesse Junction. In *Dixième congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, Nancy*, pages 62–76, 2009.

[55] T. Nakamura, C. Hirai, and Nishioka Y. A Practical Rescheduling Algorithm Using Three Predetermined Factors. In *Fourth International Seminar on Railway Operations Modelling and Analysis (RailRome 2011), Rome, Italie*, février 2011.

[56] A. Nash and D. Huerlimann. Railroad simulation using OpenTrack. In *Ninth International Conference on Computers in Railways, Dresde, Allemagne*, pages 45–54, 2004.

[57] G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*. Wiley (New York), 1999.

[58] M.W. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5(1) :199–215, 1973.

[59] M.W. Padberg. Perfect zero–one matrices. *Mathematical Programming*, 6(1) :180–196, 1974.

[60] J. Puchinger. *Combining Metaheuristics and Integer Programming for Solving Cutting and Packing Problems*. PhD thesis, Vienna University of Technology, janvier 2006.

[61] J. Puchinger, G.R. Raidl, and M. Gruber. Cooperating Memetic and Branch-and-Cut Algorithms for Solving the Multidimensional Knapsack Problem. In *Proceedings of the Sixth Metaheuristics International Conference (MIC2005)*, pages 775–780, 2005.

[62] J. Rodriguez. Projet RECIFE : « Analyse de la capacité ferroviaire ». Technical report, Institut National de Recherche sur les Transports et leur Sécurité, 2001.

[63] J. Rodriguez. A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B*, 41(2) :231–245, 2007.

[64] J. Rodriguez, X. Delorme, X. Gandibleux, G. Marlière, R. Bartusiak, F. Degoutin, and S. Sobieraj. RECIFE : Models and tools for analyzing rail capacity. *Recherche Transports Sécurité*, 95 :129–146, 2007.

[65] D.M. Ryan and B.A. Foster. An integer programming approach to scheduling. *Computer scheduling of public transport urban passenger vehicle and crew scheduling*, pages 269–280, 1981.

[66] G. Sabin, G. Kochhar, and P. Sadayappan. Job fairness in non-preemptive job scheduling. In *International Conference on Parallel Processing (ICPP)*, pages 186–194. IEEE, 2004.

[67] J. Sadki-Fenzar. *Problèmes de couverture en nombres entiers : génération de colonnes, heuristiques d'approximation garantie et schémas hybrides. Applications en transport ferroviaire et en planification de production*. PhD thesis, Université Paris 13, juillet 2011.

[68] M.A. Salido, M. Abril, F. Barber, L. Ingolotti, P. Tormos, and A. Lova. Topological Constraints in Periodic Train Scheduling. *Frontiers in Artificial Intelligence and Applications*, 117 :11–20, 2005.

[69] T. Schlechte. Railway Track Allocation – Simulation and Optimization. In *Fourth International Seminar on Railway Operations Modelling and Analysis (RailRome 2011), Rome, Italie*, février 2011.

[70] P. Serafini and W. Ukovich. A Mathematical Model for Periodic Scheduling Problems. *SIAM Journal on Discrete Mathematics*, 2(4) :550–581, 1989.

[71] M.J. Soomer and G.J. Franx. Scheduling aircraft landings using airlines' preferences. *European Journal of Operational Research*, 190(1) :277–291, 2008.

[72] M.J. Soomer and G.M. Koole. Fairness in the Aircraft Landing Problem. Technical report, Vrije Universiteit Amsterdam, Amsterdam, Pays-Bas, juin 2008.

[73] UIC. *UIC leaflet code 405-1 : Method to be Used for the Determination of the Capacity of Lines*. International Union of Railways, 1983.

[74] UIC. *UIC leaflet code 405 OR : Links between Railway Infrastructure Capacity and the Quality of Operations*. International Union of Railways, 1996.

[75] UIC. *UIC leaflet code 406 : Capacity*. International Union of Railways, juin 2004.

[76] R.-J. van Egmond. *Railway Capacity Assessment, an Algebraic Approach*. Number S99/2 in Train Studies in Transportation Science Series. The Netherlands TRAILS Research School, Delft, 1999.

[77] R. Velásquez, M. Ehrgott, D. Ryan, and A. Schöbel. A Set-packing Approach to Routing Trains Through Railway Stations. In *40th Annual Conference of the Operational Research Society of New Zealand, Wellington, New Zealand*, pages 305–314, 2005.

[78] P.J. Zwaneveld, L.G. Kroon, H.E. Romeijn, M. Salomon, S. Dauzère-Pérès, S.P.M. van Hoesel, and H.W. Ambergen. Routing trains through railway stations : Model formulation and algorithms. *Transportation science*, 30(3) :181–194, 1996.

[79] P.J. Zwaneveld, L.G. Kroon, and S.P.M. van Hoesel. Routing trains through a railway station based on a node packing model. *European Journal of Operational Research*, 128(1) :14–33, 2001.

Liste des Algorithmes

Liste des tableaux

Liste des figures

Table des matières

Évaluation biobjectif de la capacité d'infrastructures ferroviaires par génération de colonnes hybride

Aurélien MEREL

Résumé

Ce mémoire propose de s'intéresser aux problématiques liées à la capacité des infrastructures ferroviaires. Évaluer la capacité d'une infrastructure revient à estimer le trafic maximal qu'elle peut accueillir pendant une période donnée, en regard d'une demande de circulation donnée. L'estimation de capacité est partie intégrante de la gestion prévisionnelle du trafic ferroviaire, et se révèle une donnée essentielle pour les gestionnaires d'infrastructure et les exploitants ferroviaires, d'autant plus étant donné le contexte européen de libéralisation du marché. Calculer la capacité peut être réalisé par la résolution d'un problème sous-jacent appelé problème de saturation, consistant à insérer le maximum de circulation possible sur une infrastructure et soulevant d'importantes difficultés combinatoires : des temps de calcul rédhibitoires apparaissent pour certains jeux de données en utilisant les méthodes existantes. Une nouvelle méthode est donc proposée, basée sur une modélisation sous forme de programme linéaire en nombres entiers. Une contribution algorithmique est présentée, se basant sur plusieurs techniques : génération de colonnes, agrégation dynamique de contraintes et hybridation d'algorithmes exacts et approchés. Les tests comparatifs montrent une réduction significative du temps de résolution grâce à notre contribution, que nous implémentons dans un logiciel d'analyse de capacité, RECIFE PC. Une analyse des solutions nous conduit finalement à affiner le modèle mathématique en proposant deux formulations biobjectifs et une suggestion algorithmique visant à tirer profit de la génération de colonnes tout en réalisant une résolution biobjectif.

Mots-clés : recherche opérationnelle ; capacité d'infrastructure ferroviaire ; optimisation combinatoire ; génération de colonnes ; optimisation multiobjectif.

Abstract

The present thesis is focused on problematics related to railway infrastructure capacity. Evaluating an infrastructure's capacity can be seen as assessing the maximum amount of traffic that can pass through it within a given time period, depending on some circulation demand. Capacity assessment is an essential part of previsional planning of the railway traffic and is consequently an essential piece of data for infrastructure managers as well as for train operators. This is even more important in the current context of the European railway market being open to free competition. Capacity assessment can be achieved by solving an underlying problem, the saturation problem, which entails inserting as much traffic as possible inside an infrastructure and yields significant combinatorial difficulties. Prohibitive computation times are yielded by some data sets when using existing computation algorithms. A new solution method is consequently proposed, based on an integer linear programming modelling of the problem. An original algorithmic contribution is presented in order to solve this formulation by using several techniques, including column generation, dynamic constraint aggregation and hybridizations of exact and approximate solution algorithms. Comparative tests show a substantial reduction of the solving time thanks to our contribution, which we consequently implement into a capacity analysis software, RECIFE PC. An analysis of the solutions finally leads us to refine the mathematical model by proposing two biobjective formulations and an algorithmic suggestion aiming at benefiting from the column generation procedure while doing a biobjective resolution.

Keywords: operations research; railway infrastructure capacity; combinatorial optimization; column generation; multiobjective optimization.