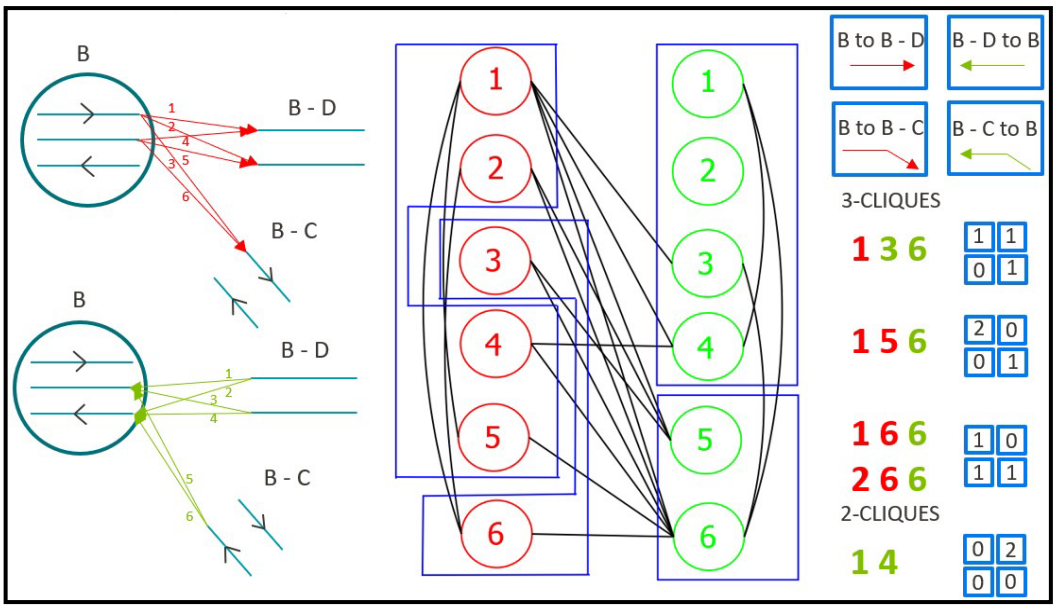
Ordonnancement de sillons ferroviaires périodiques avec affectation de voies à l'échelle mésoscopique

Guillaume Joubert

Par **Guillaume JOUBERT**

Periodic train timetabling with mesoscopic tracks assignment

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 20 novembre 2023
Spécialité : Informatique : Unité de recherche Heudyasic (UMR-7253)

D2775

Thèse CIFRE

présentée en vue d'obtenir le grade de

Docteur de l'UTC

Spécialité : Informatique

Champ disciplinaire : Recherche Opérationnelle

par **Guillaume Joubert**

Periodic train timetabling with mesoscopic tracks assignment

soutenue le 20 novembre 2023

Composition du jury :

Mme. Feng Chu	Professeure des Universités	Rapportrice
M. François Clautiaux	Professeur des Universités	Rapporteur
Mme. Tifenn Rault	Maîtresse de Conférences	Examinatrice
M. Aziz Moukrim	Professeur des Universités	Examinateur, Président du jury
M. Dritan Nace	Professeur des Universités	Directeur de thèse
M. Antoine Jouglet	Professeur des Universités	Directeur de thèse

Membre invitée :

Mme. Marion Postec	SNCF Réseau | DGEX DIGIT	Co-encadrante de thèse

préparée en collaboration avec le laboratoire Heudiasyc, UMR 7253

déliverée par l'Université de Technologie de Compiègne

Ecole Doctorale 71 - Sciences pour l'ingénieur

Remerciements

Comme vous pouvez vous en douter, cette thèse ne se résume pas exclusivement à l'ouvrage que vous avez sous les yeux. Au cours de ces années, j'ai eu la chance de rencontrer beaucoup de personnes passionnées, tant par la richesse des expertises dans les métiers du ferroviaire en entreprise que par les systèmes complexes dans tous leurs états lors de mes passages au laboratoire. Au départ, trouver tout à fait ma place, mon rythme et mon utilité dans ces environnements a été un défi à part entière. Au bout du compte, je m'essaie à quelques mots en témoignage de ma reconnaissance envers ces personnes avec qui j'ai beaucoup apprécié travailler et qui ont eu un rôle dans mon évolution vers la personne que je suis devenu aujourd'hui.

Je remercie Feng Chu et François Clautiaux pour avoir accepté de rapporter cette thèse et pour les précieux conseils qu'ils m'ont transmis. Je remercie Tifenn Rault pour avoir examiné ma thèse, ainsi que Aziz Moukrim pour son rôle d'examinateur et de président de mon jury dont je suis honoré.

J'adresse mes remerciements tout particuliers à Dritan Nace et à Antoine Jouglet. Dritan et Antoine ont non seulement été des contributeurs de premier plan dans mon éducation en Informatique et en Recherche Opérationnelle, mais ils m'ont aussi guidé sans relâche dans mon parcours de jeune chercheur pour mener à bien l'ensemble des étapes qui ont permis l'aboutissement de cette thèse. Je retiendrai le dévouement dont ils ont fait preuve à mon égard.

Je remercie chaleureusement Marion Postec pour avoir cru en mes compétences et contribué à leur développement. J'ai rejoint Marion chez SNCF Réseau pour mon projet de fin d'études de Master de Recherche, quand nous avons décidé de poursuivre en thèse. Son suivi régulier, son écoute et nos échanges d'idées m'ont permis d'en apprendre beaucoup sur le fonctionnement de l'entreprise et de rencontrer des experts métier qui nous ont permis de mener à bien nos projets.

Je remercie Bertrand Houzel pour m'avoir permis d'effectuer ma thèse à la DGEX So-

lutions, pour la liberté qu'il m'a laissé dans la conduite de mes travaux, et pour avoir fait en sorte que je puisse encadrer deux stagiaires assistant-ingénieurs dans son service. Je remercie Clément Raoux pour le suivi administratif de ma thèse à la DGEX Solutions et pour m'avoir intégré comme un membre de ses équipes. Je remercie les chargés d'études horaires de DGEX INCA qui ont eu tant de patience et d'intérêt à m'expliquer les rouages du tracé horaire à la SNCF et à échanger régulièrement sur l'avancer de nos travaux. Je remercie Frédéric Fevre, Loic Hamelin et Giuliana Barbarino pour avoir rendu possible une passation de ces travaux de thèse afin de les réutiliser dans le cadre du projet Open Source Railway Designer.

Les travaux de cette thèse ont été grandement facilités par les efforts de Eugène Pin et de Damien Berthelon, qui ont l'un après l'autre pris le pari de venir prêter main forte à un thésard pendant six mois. Eugène a fait naître l'interface graphique de l'outil OptiSillons, Damien y a apporté des possibilités d'interaction complémentaires avec des gestes métier. En plus de faciliter la visualisation et l'interprétation de mes résultats, leurs travaux ont permis de faciliter les échanges avec mon encadrement, avec des spécialistes de l'horaire dans l'entreprise ayant eu la patience de s'intéresser au projet, et avec la communauté scientifique. Je leur en suis reconnaissant.

Je tiens à remercier Jacques Carlier, David Savourey et Ronan Bocquillon pour avoir périodiquement animé mon Comité de Suivi Individuel Scientifique de thèse. Leurs recommandations ont été inestimables lors de ces jalons. Je remercie par ailleurs Mylène Masson, Bérengère Guermonprez, Benjamin Quost et Yves Grandvalet qui se sont continuellement assurés du bon déroulement de ma thèse, et notamment pendant ma période de rédaction.

Je remercie à nouveau grandement Ronan Bocquillon, cette fois-ci pour m'avoir invité à venir travailler avec lui, à Tours, dans le cadre d'une mobilité doctorale. Le fruit de nos réflexions a littéralement insufflé une nouvelle dynamique pour ma thèse. Cette mobilité a été possible grâce au concours des Actions Jeunes Chercheurs du Groupement de recherche Recherche Opérationnelle et Décision du CNRS dirigées par Ayşe N. Arslan que je remercie. Je suis aussi reconnaissant envers Yannick Kergosien et toute l'équipe ROOT du LIFAT pour leur accueil chaleureux et pour les discussions autour de ma thèse.

Je remercie les enseignants-chercheurs du Master Parisien de Recherche Opérationnelle pour m'avoir accepté en qualité d'auditeur libre dans leurs cours pendant ma première année de thèse. Je garde également un excellent souvenir de trois Ecoles pour jeunes chercheurs organisées par le GdR ROD. L'ensemble de ces événements a eu un impact positif sur mon exploration de ce vaste domaine de recherche.

J'ai eu le privilège, pendant ma thèse, d'avoir eu l'opportunité de consacrer quelques heures hebdomadaires à l'enseignement en travaux dirigés de divers cours en Informatique à l'Université de Technologie de Compiègne, allant des fondements scientifiques de l'informatique, en passant par la programmation orientée objet, la Recherche Opérationnelle et la sûreté de fonctionnement. J'exprime ma gratitude envers Dritan Nace, Antoine Jouglet, Walter Schön et Mohamed Sallak pour leur confiance et leur soutien pour mener à bien ces missions. Je remercie l'entreprise de m'avoir permis cette flexibilité. Je suis également redevable à Benoit Cantais, qui a été mon chargé de travaux dirigés en Informatique à plusieurs reprises pendant mes études, et qui est sans aucun doute l'une des inspirations qui m'a donné l'envie d'enseigner à mon tour.

Je remercie Franck Kamenga, Marie Milliet de Faverges et Sélim Cornet pour m'avoir présenté leur sujet de recherche pendant leur thèse à la SNCF et pour m'avoir ainsi permis de me projeter dans le démarrage de ma thèse. Je remercie les personnes qui m'ont introduit à l'Association des Doctorant.e.s du Groupe SNCF, devenu Club des Doctorant.e.s, et avec qui j'ai pris plaisir à animer cette communauté au sein du bureau. J'ai donc une pensée particulière pour Capucine-Marin, Rémi, Mégane, Orphée, Jean, Tess et Louis. Je salue les doctorants du Groupe que j'ai eu le plaisir de cottoyer. Je remercie la Direction Technologies, Innovations et Projets Groupe de la SNCF et notamment François Ramond, Christelle Lerin, Françoise Dubois et Allan Armougum pour l'organisation d'événements scientifiques auxquels j'ai toujours été convié : je pense aux rencontres trimestrielles des experts et chercheurs de la SNCF, aux Villages des Thèses, à la formation et au concours Ma Thèse en 180 secondes.

J'adresse mes remerciements à Gary Burkhart, sans qui mes premiers pas dans la communication et la rédaction en anglais scientifique auraient été périlleux. You were right Gary, conferences are a very nice experience ! Special thanks are due to Anita Schöbel, Michael Kümmling and Valentina Cacchiani, who kindly took the time to answer my questions during my research and thus helped me realising the strength of the international scientific community.

J'exprime toute ma reconnaissance à Dominique Kreikenmayer et à son équipe pour l'organisation du programme de mentorat de Sorbonne Université, ainsi qu'à Philippe Reymond, qui a été mon mentor à l'occasion de la première édition de ce programme et m'a accompagné pendant un an sur ma fin de thèse. Merci Philippe pour ton écoute, ton soutien et ton partage d'expérience qui m'ont aidé à garder le cap dans des moments particulièrement charnières.

Je remercie l'ensemble des personnels du laboratoire Heudiasyc. Merci Philippe pour m'avoir accepté au sein du laboratoire. Merci Nathalie, Sabine et Brigitte, pour votre implication permanante pour les départs en missions. Merci Gildas pour ton appui technique. Merci Tom et Stefano pour ces discussions sur la PPC, les problèmes SAT et les réseaux de neurones. Merci Marinela pour avoir été un exemple pour moi, ta soutenance de thèse est la première à laquelle j'ai eu la chance d'assister. Merci Nader, Michaël et Qing pour m'avoir supporté pendant ces années dans notre bureau. Merci à tous pour m'avoir honoré à deux reprises du prix du meilleur poster de la journée du laboratoire. Enfin, je remercie les Pifometrics pour l'aventure artistique que nous avons partagée à l'occasion des 40 ans du laboratoire.

Je présente mes remerciements les plus amicaux à Isabel Barros Garcia. Isabel, tu as été une binôme en or, il y a eu tant de découvertes avec toi : programmation de nos algos d'arc-consistance, rédaction de nos premiers résumés pour la ROADEF, premières expériences d'enseignement à l'UTC, excursion en Ecole du GdR ROD. Par-dessus tout, merci pour ton éternel soutien lors de mes passages à Compiègne.

Je remercie les collègues de la DGEX Solutions pour l'ambiance de travail conviviale que nous avons partagée au quotidien. A l'extérieur, certains d'entre vous n'ont pas hésité à me suivre dans des paris fous. Merci à Firas avec qui nous avons sillonné l'Ile-de-France à la conquête du classement 30/5. Merci à Arthur et à Pierre, avec qui nous avons pris des départs sur la plus belle course parisienne.

Je remercie chaleureusement l'Association de Tennis Paris 13, et notamment Maryelle, Stéphane, Thierry, Pascal, Fabia et Karim. Vous m'avez fait passer trois années extraordinaires, sur les courts comme parmi le bureau du Conseil d'Administration.

Je suis reconnaissant envers mes amis qui ont toujours fait leur possible pour me témoigner leur soutien quand je pouvais les croiser. Du fond du coeur merci à vous. En particulier, je tiens à remercier Lucas Warin pour sa présence lors de ma période de rédaction.

Je remercie infiniment ma mère Muriel, mon père Joël, ma soeur Manon, mes cousins Paul, Claire, Caroline, Laurent, Eglantina, et mes anciens. A tout moment au cours de ma thèse, j'ai pu trouver en votre compagnie un refuge. Vous n'avez cessé de croire en moi et de me donner la force d'avancer. Merci pour votre soutien inconditionnel jusqu'à la soutenance.

J'exprime en dernier lieu ma gratitude envers Pauline. Merci ne pourrait suffire pour tout ce que toi et tes proches m'apportez.

Résumé

Le Gestionnaire d'Infrastructure SNCF Réseau produit annuellement un ordonnancement de sillons pour permettre la circulation de trains répondant à une demande des Autorités Organisatrices de Transports. Cette thèse s'inscrit dans une démarche d'aide à la décision pour les chargés d'études horaires produisant ces sillons. Nous traitons une phase de structuration : il s'agit de trouver, s'il existe, un ordonnancement périodique répondant à une demande de sillons et respectant les contraintes de circulation. Nous proposons deux méthodes pour résoudre ce problème.

La première méthode se base sur un modèle en Programmation Linéaire en Nombres Entiers inspiré de la littérature et décomposé pour faciliter sa résolution. La grille horaire est décidée, suivie de l'affectation de voies. Une fonction d'évaluation de conflits entre sillons sans connaître l'affectation de voies permet de construire une grille horaire garantissant une affectation de voies compatibles. Nous résolvons les conflits des instances difficiles avec une heuristique constructive et une recherche tabou.

La deuxième méthode s'appuie sur un modèle en Programmation Par Contraintes optimisant le temps de retournement des rames en gares terminus. Nous concevons des algorithmes de filtrage améliorant la propagation des contraintes périodiques de précédence et disjonctives. Nous proposons des bornes inférieures basées sur la contribution de sous-ensembles de sillons à la fonction objectif pour améliorer la preuve d'optimalité. Nous présentons une procédure de Branch-and-Check accélérerant la résolution.

Nous testons ces approches sur des instances fictives du problème et sur l'Etoile de Savoie autour de Chambéry.

Mots-clés : Grilles horaires périodiques, Affectation de voies à l'échelle mésoscopique, Programmation Linéaire en Nombres Entiers, Heuristique de recherche tabou, Programmation par Contraintes, Procédure de Branch-and-Check.

Abstract

The Infrastructure Manager SNCF Réseau produces train timetables on a yearly basis, to allow trains circulation fulfilling a mobility demand from the Transport Organisation Authority : this is a periodic slot scheduling task. This PhD project aims at providing a decision-aid tool to the timetable planners. We focus on a capacity structuration phase: we look for a periodic scheduling fulfilling a slots demand and satisfying the safety constraints arising from rail operations. We propose two methods to solve this problem.

The first method is based on an Integer Linear Programming model inspired from the literature and that we decomposed to ease the solution search. The timetable determination is then made before the tracks assignment. A conflict evaluation function that does not require explicit tracks assignment knowledge is used to obtain conflict-free timetables for which a tracks assignment is reachable. We solve the conflicts of challenging instances with a constructive heuristic and a tabu search.

The second method relies on a Constraint Programming model that we use to optimise the turnover time of rolling stock at terminus stations. We conceive filtering algorithms to improve algorithms that maintain the arc-consistency of precedence and disjunctive constraints of the problem, all having a periodic aspect. We propose lower bounds based on the cost contribution of slots subsets to the objective function in order to improve the optimality proof. Finally, we present branching strategies and a Branch-and-Check procedure to improve the resolution performance.

We test these approaches on fictive instances of the problem and on the Savoie area around Chambéry.

Keywords: Periodic train timetabling, Mesoscopic tracks assignment, Integer Linear Programming, Tabu search heuristic, Constraint Programming, Branch-and-Check procedure.

Contents

List of Figures

List of Tables

Résumé substantiel

Ces quelques paragraphes ont pour objet de rassembler les idées principales de ce manuscrit de thèse sur l'ordonnancement de sillons ferroviaires périodiques avec affectation de voies à l'échelle mésoscopique, afin d'en permettre l'accès aux lecteurs francophones. Ces travaux s'intéressent à la conception de modèles mathématiques et d'algorithmes permettant de calculer des horaires de trains pouvant se répéter dans le temps selon une certaine fréquence ainsi qu'une affectation de voies pour ces trains sur le réseau ferré. Pour permettre à un train de circuler, il est nécessaire de lui réserver des intervalles de temps tout au long de son parcours sur l'infrastructure : on parle de sillon. Dans la littérature scientifique, une majorité des modèles existants utilise une représentation de l'infrastructure ferroviaire à l'échelle macroscopique, à l'échelle microscopique, ou en faisant interagir ces deux niveaux de détail. Le premier est traditionnellement adapté à l'optimisation d'horaires tactiques, plusieurs années à semaines avant l'exploitation ferroviaire. Le second permet à la fois une simulation détaillée du trafic ferroviaire, et une optimisation de décisions de replanification en temps réel afin de limiter les retards en cas de perturbation. Ces modèles peuvent toutefois montrer certaines limites pour l'optimisation d'horaires tactiques. En effet, à l'échelle macroscopique, le plan de voies modélisant l'infrastructure ferroviaire est souvent simplifié, notamment dans la prise en compte des itinéraires complexes reliant les voies en lignes aux voies à quai en gares. Ces approximations peuvent conduire à des difficultés d'implémentation des horaires tactiques en pratique, nécessitant un travail d'ajustement de l'horaire et une supervision plus délicate en opérationnel. D'un autre côté, à l'échelle microscopique, l'infrastructure est représentée avec une granularité suffisamment fine pour modéliser le fonctionnement dynamique de la signalisation ferroviaire et l'occupation des zones de protection élémentaires par chaque essieu d'une rame grâce au shuntage des circuits de voie. Une optimisation d'horaires tactiques à l'échelle microscopique de l'infrastructure pourrait donc être implémentée en pratique en éliminant les inconvénients liés à l'utilisation de l'échelle macroscopique. Néanmoins, le niveau de détail requis pour les modèles microscopiques aboutit à des modèles de plus grandes tailles en terme de nombre de variables et de contraintes pour la représentation de la même instance d'un problème, comparé à un modèle à l'échelle macroscopique. Par conséquent, il n'est pas envisageable d'utiliser des modèles microscopiques pour produire des horaires

tactiques sur des périmètres géographiques trop étendus. Quelques travaux de la décennie écoulée ont introduit ou utilisé la modélisation de l'infrastructure ferroviaire à l'échelle mésoscopique. Ce niveau de détail permet de prendre en compte la compatibilité des itinéraires entre les voies en lignes et les voies à quai en gares, sans toutefois aller jusqu'au détail de la signalisation ferroviaire. L'échelle mésoscopique est donc un compromis intéressant entre les échelles macroscopique et microscopique, la première permettant une prise en compte plus réaliste des conflits ferroviaires que la deuxième, tout en gardant des tailles de modèles toujours assez avantageux comparé à la troisième. Dans les travaux de cette thèse, nous nous inspirons des travaux de planification horaire utilisant des modèles mésoscopiques, nous étendons ces modèles avec une classification de contraintes métiers de SNCF Réseau, et nous proposons des méthodes originales pour résoudre ces problèmes. Nous allons maintenant explorer plus en détail le contenu de cette thèse, à savoir un chapitre introductif, un état de l'art sur la planification ferroviaire et la Recherche Opérationnelle, une description des problèmes traités ainsi que la structure des données et les instances à résoudre, les deux familles d'approches de résolution basées sur un programme linéaire en nombre entiers et un modèle de programmation par contraintes, ainsi que quelques commentaires sur le prototype de planification interactive OptiSillons.

Le mode de transport ferroviaire a des atouts considérables en terme de sécurité, décarbonation et de confort. Chaque jour, ce sont 5 millions de passagers et 250000 tonnes de marchandises qui sont transportés par 15000 trains sur les 28000 kilomètres de voies sur le réseau ferré en France, en majeure partie électrifié. Et cette base est sur une pente ascendante avec le souhait d'un doublement du trafic ferroviaire d'ici aux années 2030. Maintes étapes sont nécessaires à la possibilité de faire rouler un train. Il faut tout d'abord investir dans le développement et la régénération du réseau. A partir du réseau existant et projeté, des études socio-économiques permettent de prévoir la demande de mobilités futures et de planifier les lignes à opérer sur le réseau. C'est ensuite que vient l'étape de création de plans de transports sous forme de grilles horaires pour couvrir l'ensemble des lignes planifiées, en respectant toutes les contraintes liées à l'utilisation de l'infrastructure par les trains. Cette thèse s'intéresse à ce problème, qui consiste à concevoir des méthodes permettant d'optimiser la création de plans de transport périodiques pour des lignes ferroviaires à opérer en heures de pointe dans le contexte d'un trafic mixte, c'est-à-dire avec différents types de trains de passagers et des trains de fret. A partir de ces horaires, il est nécessaire de planifier le matériel roulant à affecter sur ces trains, ainsi que les personnels à bord, en escale et dans les centres opérationnels de gestion des circulations. Dans ces centres opérationnels, des décisions de replanification du trafic ferroviaire sont prises en temps-réel pour limiter les retards en cas de perturbations. Afin de limiter les risques de pannes, la maintenance préventive du matériel roulant est aussi planifié en amont.

Les premiers outils d'aide à la conception pour les horairistes de la SNCF permettaient d'effectuer des calculs de marches à partir de caractéristiques précises du matériel roulant, et de déterminer les périodes de traction optimales afin de minimiser la consommation énergétique du trajet. Les systèmes informatiques se sont améliorés, permettant peu à peu aux horairistes de concevoir des plans de transport faisant interagir plusieurs train et de visualiser les conflits potentiels grâce à une modélisation fine des systèmes de signalisation utilisés sur l'infrastructure. Plus récemment, l'essor de la recherche opérationnelle a permis l'apparition d'outils d'aide à la décision pour les problèmes de planification ferroviaire. A la SNCF, ces outils permettent de déterminer les voies à quai dans les gares, de réaliser des études de robustesse, et de planifier la maintenance des rames et de l'infrastructure, la supply-chain permettant de réaliser ces travaux de maintenance, les circulations de trains de fret, les horaires des agents, l'affectation des passagers à bord des trains, la replanification en temps-réel et le contrôle optimal de conduite sur les lignes hyper denses. Bien que des travaux de recherche sur la conception des lignes ferroviaires et de leurs horaires aient été conduits au sein de l'entreprise, notons que la conception de grilles horaires périodiques est effectuée grâce à l'expertise des chargés d'étude horaires de SNCF Réseau en utilisant des outils d'aide à la conception permettant la détection de conflit entre les circulations planifiées. La motivation des travaux de cette thèse est née de l'idée de fournir un outil d'aide à la décision aux chargés d'études horaires pour obtenir des suggestions de plans de transports optimisés et ainsi permettre un processus interactif de création de l'horaire à partir de ces suggestions.

La planification du système ferroviaire est décomposée en plusieurs problèmes classifiés selon des horizons décisionnels hiérarchiques. Au niveau stratégique, correspondant à des décisions sur le long terme, nous retrouvons les problèmes de conception du réseau et des lignes ferroviaires à opérer. Les problèmes de planification des ressources viennent au niveau tactique : il s'agit des grilles horaires, du matériel roulant et des personnels. Enfin, le niveau opérationnel se concentre sur les problèmes de replanification en présence d'aléas. Certains problèmes visent à intégrer plusieurs horizons décisionnels dans le but d'obtenir des solutions de meilleure qualité. Ces problèmes intégrés sont d'autant plus complexes, on pense par exemple à la planification conjointe des lignes ferroviaires, des horaires des trains et du matériel roulant. Dans l'état de l'art de cette thèse, nous intéressons particulièrement aux problèmes de planification d'horaires de sillons ferroviaires. Nous commencerons par présenter des problèmes du niveau tactique avec des durées de transport nominale. Nous détaillerons ensuite les indicateurs de performance du système ferroviaire. Enfin, nous présenterons les deux approches principales pour replanifier des plans de transport en temps-réel.

Le problème tactique de planification de sillons ferroviaires consiste à déterminer

l'horaire pour un ensemble de trains sur les points remarquables de leur parcours, notamment les départs et arrivées en gares, afin de répondre à une demande de mobilité tout en respectant les contraintes de circulation liées à l'utilisation de l'infrastructure. Certains auteurs se sont intéressés à la version de ce problème utilisant un horizon de temps linéaire, permettant par exemple de planifier les horaires d'une journée avec des demandes de sillons hétérogènes au fil des heures. De nombreux travaux traitent la variante périodique du problème, où tous les trains traversant la fin de la période coïncident avec ceux planifiés en début de période. Cette configuration du problème permet la planification des heures de pointe où la circulation ferroviaire est la plus dense, puis d'adapter la trame obtenue pour planifier le reste de la journée. Le *Periodic Event Scheduling Problem* est le modèle de prédilection pour ce problème. Une reformulation basée sur des variables de cycles a permis la conception d'inégalités valides pour améliorer l'efficacité de méthodes exactes de résolution. D'autres modèles intègrent des caractéristiques complémentaires comme la flexibilité des horaires dans les solutions proposées, l'aspect dynamique de la demande, ou encore la prise en compte de l'itinéraire emprunté par les passagers en fonction des horaires proposés. Tous ces modèles sont en grande majorité conçus pour une infrastructure à l'échelle macroscopique. Néanmoins, quelques auteurs ont initié une tendance dans l'utilisation d'infrastructures à l'échelle mésoscopique pour la modélisation du problème tactique. Concernant la résolution de ces problèmes complexes, des méthodes heuristiques ont été proposées pour traiter de problèmes de grande taille. Une technique populaire consiste à ajouter des sillons dans une grille horaire et de résoudre les éventuels conflits induits au fur et à mesure.

Pour mesurer la qualité des grilles horaires produites, une variété d'indicateurs de performance ont été défini dans la littérature. L'*occupation de l'infrastructure* permet de calculer si la densité de circulation ferroviaires sur un périmètre donné et durant une certaine période ne dépasse pas un certain seuil dépendant notamment de la topologie de l'infrastructure et des types de trains pour lesquels cette infrastructure est adaptée. Un indicateur mesurant la possibilité d'absorber des retards est la *stabilité* : il s'agit de déterminer si un certain nombre de minutes de retard sur un train peut être complètement résorbée en l'espace d'une heure sans créer d'effet boule de neige et sans effectuer de replanification des trains. La *robustesse* est un indicateur assez connexe, déterminant entre quelles circulations et à quels points du réseau il est le plus intéressant de garder des marges temporelles afin que les horaires soient le plus insensibles possibles aux légers retards les plus fréquents. Un système ferroviaire *résilient* permet, en cas d'aléas, de déclencher des actions comme par exemple la mise en service de rames de réserves afin de retrouver un service nominal non perturbé le plus rapidement possible. Un indicateur qui gagne en popularité est l'*efficacité énergétique* qui permet à un train de respecter des horaires de passage donnés tout en optimisant la traction à fournir tout au long du trajet

de manière à minimiser l'énergie consommée.

Quand le système ferroviaire fait face à des aléas générant d'importants retards, il est nécessaire de replanifier un ensemble de circulations en temps-réel de manière à limiter l'impact des perturbations. Des modèles en programmation par contraintes et en programmation linéaire en nombres entiers utilisant une infrastructure à l'échelle microscopique ont été proposés dans la littérature pour minimiser le retard total des trains. L'intégration de la gestion du retard subi par les passagers dans les décisions de replanification a aussi intéressé des auteurs, donnant lieu à des modèles utilisant une infrastructure macroscopique et prenant en compte la capacité de l'infrastructure, la possibilité de réorienter des passagers, et la capacité des trains. D'autres modèles visent à intégrer à la fois la minimisation du retard des circulations et des passagers.

Les deux problèmes traités dans cette thèse permettent de décider des horaires et une affectation de voies pour des sillons à un niveau tactique, sur un horizon de temps périodique, et en utilisant une infrastructure à l'échelle mésoscopique. Dans la première variante du problème, nous cherchons à obtenir des solutions réalisables, c'est à dire respectant trois familles de contraintes que nous allons détailler ci-après. Dans la deuxième variante du problème, nous modélisons une réoccupation du matériel roulant par les sillons en gares terminus et nous optimisons les durées de stationnement en gares de sorte à satisfaire au mieux les normes établies dans les référentiels de SNCF Réseau. Deux trains ne pouvant occuper les mêmes portions de voies simultanément, les horaires de sillons doivent respecter trois familles de contraintes appelées normes de tracé. Les contraintes d'espacement permettent de garantir que deux trains circulant sur les mêmes voies dans le même sens ne puissent le faire que s'ils sont espacés d'une certain nombre de minutes, dépendant de caractéristiques de l'infrastructure, de la vitesse maximale des trains sur cette infrastructure et de leur capacité de freinage. Les contraintes de réoccupation, qui s'appliquent notamment sur les voies en gares et sur les voies uniques à double sens de circulation, permettent de s'assurer qu'un premier train a libéré ces voies pendant une certaine durée avant que le train suivant puisse s'engager sur ces voies. Enfin, les contraintes de cisaillement de cisaillement permettent d'empêcher que des trains empruntant des itinéraires incompatibles sur l'infrastructure le fassent trop proche dans le temps. En effet, ces trains seraient susceptibles de générer des perturbations sur le trafic en cas de retard. Lors du déroulé des opérations, la signalisation ferroviaire est responsable du respect de l'ensemble de ces contraintes. Le rôle de l'optimisation des horaires au niveau tactique permet de créer un plan de transport sans conflit, où chaque mécanicien chargé de la conduite des trains ne rencontrerait que des signaux à *voie libre*, lui permettant de continuer son itinéraire sans provoquer de ralentissements. Nous modélisons ces problèmes et les résolvons sur des instances utilisant trois infrastructures mésoscopiques aux carac-

téristiques différentes. La première infrastructure utilisée permet la création d'instances jouets. Cette infrastructure est constituée de 4 gares satellites reliées à une gare centrale, où 8 sillons sont à planifier périodiquement toutes les 20 minutes. Notre deuxième infrastructure fictive est constituée de plus d'une vingtaine de gares avec plusieurs lignes entremêlées et une planification allant jusqu'à plusieurs dizaines de sillons par heure. Enfin, nous avons reconstitué à l'échelle mésoscopique une infrastructure fidèle au périmètre de l'Etoile de Savoie, configurée en étoile et centrée autour de la gare de Chambéry - Challes-les-Eaux. Cette troisième infrastructure est plus contrainte que les précédentes, notamment du fait de sa double voie desservant trois voies uniques à double sens de circulation.

Nous modélisons le problème de décision par un programme linéaire en nombres entiers inspiré de travaux de la littérature. En augmentant le nombre de sillons à planifier pour une période de une heure sur notre deuxième infrastructure fictive, nous observons que le temps de résolution par un solveur commercial devient de plus en plus long, jusqu'à ne plus pouvoir déterminer en temps raisonnable si une instance à 40 sillons est insoluble ou s'il aurait fallu davantage de temps de calcul pour trouver une solution réalisable. Afin de pouvoir résoudre nos instances basées sur des données réelles, nous adoptons la stratégie de décomposer le problème en commençant par décider les horaires des sillons avant de procéder à une affectation de voies compatibles. En effet, nous avons observé que pour des solutions réalisables, le solveur est efficace pour trouver une affectation de voies compatibles dès lors que les horaires des sillons permettent l'existence de cette affectation de voies. La partie difficile de notre problème se réduit alors à la recherche de tels horaires qui permettent ensuite de garantir la possibilité d'affectation des voies compatibles à partir de ces horaires. Pour ce faire, nous avons conçu un modèle permettant d'évaluer les conflits entre sillons à partir de leurs horaires et sans connaissance explicite de leur affectation de voies. Puis, nous utilisons ce modèle afin de choisir les horaires de sillons en minimisant les conflits ainsi évalués. Si des horaires sans conflits sont obtenus, alors ils sont injectés dans le programme linéaire en nombre entiers qui se charge de calculer efficacement une affectation de voies compatible avec ces horaires. Afin de résoudre des instances avec un nombre de sillons croissant, nous concevons une méthode de recherche tabou permettant, au lieu de chercher directement des horaires sans conflits pour la totalité des sillons, de modifier itérativement l'horaire du sillon le plus conflictuel pour espérer aboutir à une solution sans conflits. Cette méthode heuristique a été implémentée en deux versions, la première sous la forme de matheuristique utilisant le solveur pour rechercher localement le meilleur horaire du sillon le plus conflictuel avec les autres, la deuxième sous la forme d'une métaheuristique où les conflits entre sillons sont évalués sans utiliser le solveur. Cette dernière méthode, couplée au calcul de l'affectation de voies une fois les horaires

sans conflits déterminés, a permis de résoudre une instance sur l'infrastructure de l'Etoile de Savoie en moins de deux minutes, alors que le solveur ne permettait pas d'aboutir à une solution en plusieurs heures à partir du premier programme linéaire en nombres entiers modélisé. Nous proposons des perspectives afin d'améliorer cette méthode, notamment en concevant des opérateurs locaux complémentaire permettant de modifier simultanément l'horaire d'un sous-ensemble de sillons judicieusement sélectionnés.

Nous nous sommes intéressés à la possibilité de traiter le problème de décision à partir d'une modélisation en programmation par contraintes. Notre modélisation repose sur l'utilisation de variables d'intervalle permettant de représenter la durée de l'occupation de voies par un sillon. Le caractère *optionnel* de ces variables d'intervalle nous a permis de modéliser naturellement le choix des voies à affecter pour les sillons dans chacune des gares de leur parcours. Un défi intéressant a été la modélisation d'occupations de voies ayant lieu à la fin d'une période et se poursuivant au début de la période suivante. Pour ce faire, nous avons proposé une nouvelle structure de variable, l'*opération périodique*, qui est une juxtaposition de deux variables d'intervalle liées avec des contraintes spécifiques leur permettant de se comporter comme une seule variable d'intervalle ayant la possibilité de s'exécuter sur deux périodes. A l'aide d'une formulation complète du problème modélisant les mêmes contraintes que le programme linéaire en nombre entiers du début de paragraphe précédent, nous avons résolu la même instance sur l'infrastructure de l'Etoile de Savoie en quelques secondes seulement avec un solveur commercial de programmation par contraintes avec des paramètres par défaut. Ce résultat nous a motivé à investiguer une nouvelle variante de notre problème avec un degré de réalisme complémentaire. Par la suite, nous nous sommes donc intéressés à un problème d'optimisation de la durée de stationnement des rames en gares terminus. Cette variante du problème possède des contraintes horaires additionnelles dans les gares terminus, où la durée de stationnement est calculée comme la différence entre l'horaire de départ d'un sillon et l'horaire d'arrivée d'un sillon précédent partageant la même rame. Les durées de stationnement optimales sont définies dans les recueils d'exploitation des gares de SNCF Réseau comme respectant des intervalles de temps minimaux, robustes et maximaux qui dépendent des gares considérées. Nous avons donc modélisé des coûts de pénalité correspondant à des durées de stationnement ne respectant pas ces référentiels. Le problème est alors de produire une grille horaire tactique respectant d'une part les normes de tracé entre les sillons, et d'autre part minimisant la somme des coûts de pénalité liés au non respect des durées de stationnement en gares terminus renseignées dans les référentiels métier de l'entreprise. Nous avons amélioré notre modèle en renforçant la détection des contraintes de cisaillement et en concevant un ensemble d'algorithmes de filtrage permettant d'améliorer le maintien de l'arc-consistance des contraintes du modèle, et notamment des contraintes liées à la

périodicité temporelle et dont le solveur par défaut ne permettait pas de tirer profit d'une propagation exhaustive. Nous avons aussi conçu une méthode pour dériver des bornes inférieures à partir de la détermination du coût minimal lié à la durée de stationnement de chaque couple de sillons utilisant les mêmes rames. Enfin, nous avons conçu des algorithmes de branchement et une méthode de vérification de l'existence d'affectation de voies permettant d'obtenir de meilleures performances que le solveur de programmation par contraintes avec les paramètres par défaut. La conjonction de ces techniques nous a permis d'optimiser l'instance de l'infrastructure de l'Etoile de Savoie en l'espace d'une seconde. Ce résultat encourageant permettra d'envisager la résolution d'instances avec plus de sillons et sur des périmètres géographiques plus importants. Une perspective d'amélioration de notre approche est la conception d'un algorithme plus affiné pour la vérification de l'existence d'affectation de voies à partir d'une solution partielle d'horaires de sillons.

Les modèles et méthodes d'optimisation conçus et développés dans ces travaux ont été interfacés avec l'outil OptiSillons, une interface graphique développée dans le cadre de sujets de stages qui ont été menés par deux étudiants au cours de cette thèse. Cet outil permet de créer, d'importer, de visualiser et de modifier des données d'infrastructures ferroviaires à l'échelle mésoscopique ainsi que des demandes de sillons pour lesquels les horaires et l'affectation de voies est à déterminer. Via l'interface graphique, il est alors possible d'appeler les algorithmes d'optimisation de façon à obtenir des plans de transport réalisables et optimisant les durées de stationnement des rames en gares terminus. Les résultats obtenus sont alors visualisables dans l'interface graphique. OptiSillons donne aussi la possibilité d'effectuer des gestes métier à partir de ces suggestions de plans de transport. Un chargé d'études horaires peut facilement faire glisser des sillons pour en changer l'horaire de départ, ou modifier plus finement la durée du sillon sur certaines sections de son parcours. L'affectation de voies peut également être modifiée depuis l'interface de visualisation des graphiques d'occupation des voies. A tout moment, si la modification d'horaires ou d'affectation de voies génère des conflits entre les sillons, ceux-ci sont alors affichés à l'utilisateur. Un retour arrière permet d'annuler le dernier geste métier effectué s'il s'avère insatisfaisant. Enfin, une fois le travail au graphique achevé, la session peut-être enregistrée et les données exportées.

De multiples perspectives concernant la définition du problème pourraient être intégrées, comme la considération de plusieurs périodes avec un sous-ensemble de sillons non périodiques, ou encore une meilleure considération de la durée de trajet des voyageurs afin de suggérer des horaires et des affectations de voies assurant des correspondances attractives et robustes aux aléas.

Chapter 1

Introduction

The aim of this thesis is to provide decision support to the timetable planners at *SNCF Réseau*. This chapter provides motivations of our work. After giving elements of context on the importance of the timetabling task in railway transport companies, we set the goals we look to reach and describe the contributions that constitute the following chapters of this manuscript.

Contents

1.1 Industrial context

Railway is one of the most efficient transportation mode considering safety, decarbonisation and comfort. Everyday, 5 millions of passengers and 250 thousand tons of goods are transported in 15000 trains [155] on the French railway network, composed by 28000 km of tracks [154] in very large part electrified. Passengers and freight transportation demand is expected to increase in Europe, like in France where the volume of railway circulations is expected to double in the next one or two decades, see Farandou (2022) [72].

The railway planning process that allows a train to run on the infrastructure is complex. Traditionally, this process is decomposed in sequential phases easier to handle. First, decisions are made on the investments to renovate, modernise and develop the infrastructure. In 2023, 1600 projects are planned to contain the French network ageing [153]. Once the infrastructure state is known for the upcoming years, railway lines to be operated are decided 5 years before operations. This is achieved from socio-economic studies, mobility forecasting, and coordination with several stakeholders including transport organisation authorities, the Infrastructure Manager *SNCF Réseau* and Railway Undertakings like *SNCF Voyageurs* that operate trains [156]. About 3 to 2 years before operations, periodic timetables are built assessing the feasibility in peak hours of the line plans decided at the previous planning phase. Generating periodic timetables automatically to provide decision support to *SNCF Réseau* timetable planners through an interactive tool is the topic of the work of this thesis. The subsequent timetabling steps include the construction of daily timetables and the insertion of additional train demands in the residual capacity of the timetable until few days before operations. In order to operate a timetable, necessary resources are planned, namely the rolling stock composing the trains, and the crews on board, at stations and in the operational traffic management centers. The latter mentioned assets are crucial to monitor the real-time traffic and to adapt operations in case of potential delays or larger disruptions that could make the nominal timetable impossible to operate. To avoid disturbances caused by material failures as much as possible during daily operations, preventive maintenance of the rolling stock also needs to be scheduled.

At SNCF, the first purpose of the conception support tools for the timetable planners was to determine the running time of a train on the infrastructure depending on its characteristics like the mass of the train and the power of the locomotive. Lancien and Fontaine (1981) [111] describe how to compute automatically minimal energy-consumption running times to replace analog machines that were not convenient for industrial production. Two years later, Quinchon (1983) [151] details the duties of a timetable planner and observes that the only available automated tasks are the computation and the storage of train run-

ning times. The author mentions a new project that will allow conflict detection between trains. Bertereix and Beurrier (1990) [9] present a tool suitable to assist timetable planners of dense suburban traffic areas. In this software, conflict detection is implemented for several types of signalling systems, namely the automatic bloc signalling system, a track-to-train transmission based system, and a moving block system. The first two systems give speed instructions to train drivers depending on the number of free track sections in front of them, whereas the third system updates continuously the drivers about the latest moment they must brake to no overtake the preceding train. The latter one is still particularly relevant. Indeed, an automated Communication Based Train Control system will be installed in the upcoming years in the central portion of a dense railway line in Paris to increase its frequency [71].

More recently, operational research techniques have been widely used to tackle railway planning problems in academia. In particular, much effort has been dedicated to solve timetabling problems. These achievements have also given new opportunities for the realisation of timetable planners duties in practice: not only the tools verify if a timetable is conflict-free, but also they are able to provide automatised suggestions of such timetables. Nowadays at SNCF, the current projects making use of operational research cover subjects like platforming trains at stations from a given timetable, robustness simulations, scheduling maintenance operations for rolling stock and on the infrastructure, supply-chain planning for maintenance projects, freight operations planning, crew scheduling, passengers place affectation aboard trains, decision support for real-time rescheduling and optimal control in hyper dense lines. Research on line planning and periodic timetabling has also been conducted in the company. However, these tasks are still mainly performed by the expertise of timetable planners with modern conception-aid tools but without automated suggestions. The motivation of our work is therefore to initiate a movement towards an upgraded experience for the timetable planners in the company. In one hand, the determination of train running times and the robustness evaluations could still be performed using existing tools. On the other hand, we propose an interactive decision support process for the periodic timetabling task.

1.2 Research objectives and contributions

The aim of our work is to create a tool to provide decision support to timetable planners, focusing on feasibility of the resulting periodic timetables with respect to the constraints imposed by the railway infrastructure. The outcome of this thesis seeks to meet two objectives:

- **O1**: Propose and develop efficient methods to generate periodic timetables respond-

ing to a line plan and satisfying rules that regulate railway operations on the infrastructure at the mesoscopic scale.

- **O2**: Allow timetable planners to benefit from algorithms to generate periodic timetables through a user interface without needing programming skills.

These objectives are accomplished by the proposition of two main contributions (**C1** and **C2**) in relation with objectives **O1** and **O2** respectively. Contribution **C1** corresponds to the formal definition of the problem (Chapter 3) and to the conception of models and algorithms to solve them, using Integer Linear Programming (Chapter 4) and Constraint Programming (Chapter 5) techniques. An adaptive procedure for periodic timetabling and tracks assignment, detailed in Chapter 4, has been presented at the 11th Triennial Symposium on Transportation Analysis (TRISTAN XI, 2022). The constraint-based approach presented in Chapter 5 is currently submitted to an international Journal. This approach has also given rise to a communication to the 24th French Operational Research and Decision Support Society annual congress (ROADEF, 2023). Contribution **C2** corresponds to the conception and the development of a decision support tool (Chapter 6) that integrates the approaches presented in Chapter 4 and Chapter 5 to allow timetable planners to obtain automatic suggestions of periodic timetables. The front-end of the application corresponding to this contribution has been realised in collaboration with two intern students. The following section provides additional details on the contributions through a description of the chapters that compose this thesis.

1.3 Outline of the thesis

The thesis structure is presented in Figure 1.1 and described as follows.

In Chapter 2, we provide an overview of techniques used in the literature to model and solve railway planning problems. As it is the central topic of this thesis, we will mainly focus on reviewing train timetabling research, that we place in perspective among other problems that compose the whole railway planning process. Moreover, we will highlight a growing trend towards an integration of some of these problems as a way to make a better use of main resources.

In Chapter 3, we present background notions that will be used throughout this thesis. We describe two variants of the periodic Train Timetabling with mesoscopic tracks assignment problem. The advantage of the mesoscopic infrastructure scale is the consideration of more detailed tracks occupation constraints than at the macroscopic scale, while the model size remains smaller than for microscopic models. The first variant is a decision

problem, for which we assume that trains exit the railway system after their arrival to their terminus station. The second variant is an optimisation problem with improved realism by linking trains at their terminus stations and optimising the turnover time of the rolling stock. The two subsequent chapters can be read independently.

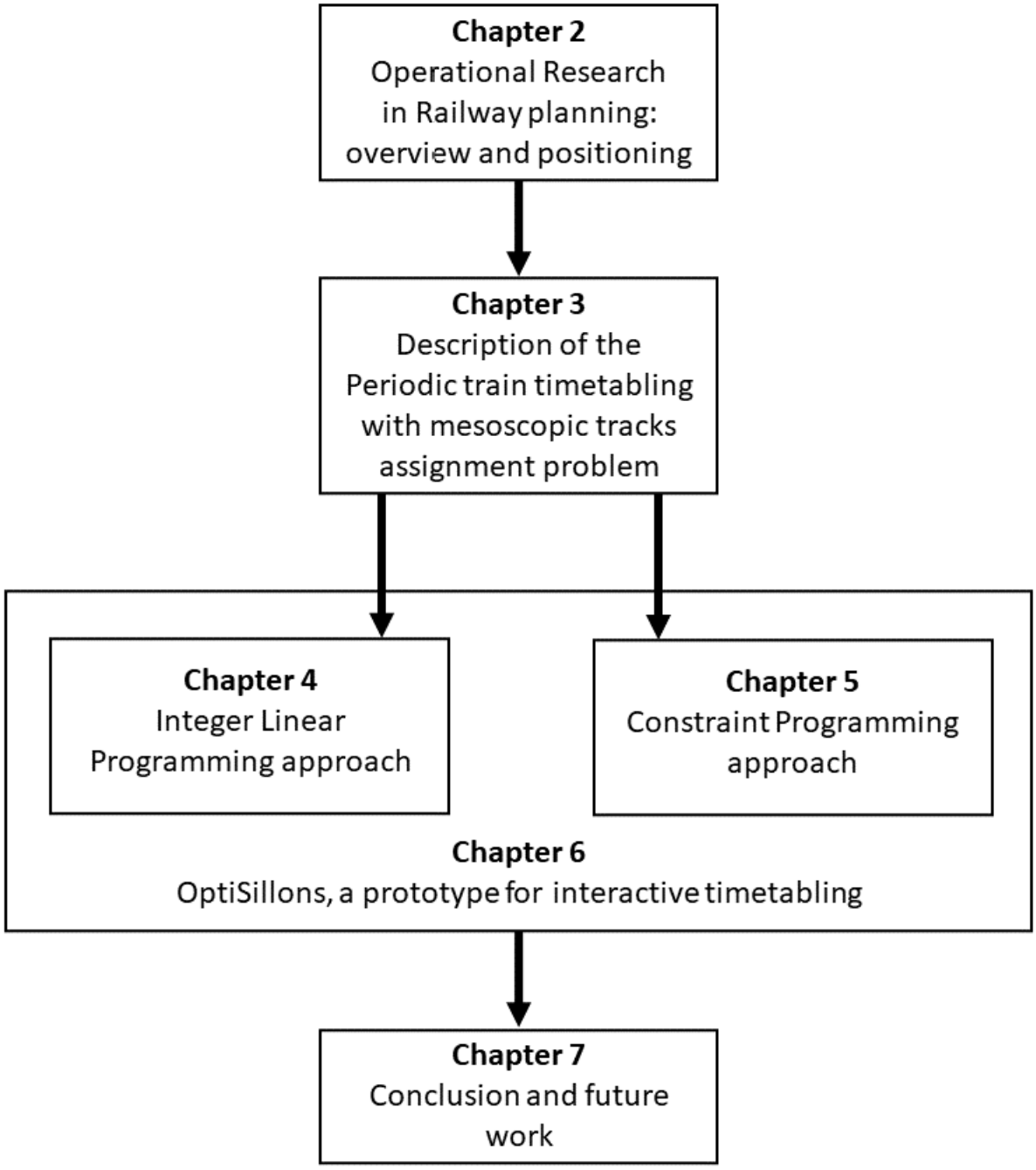


Figure 1.1: Thesis structure.

In Chapter 4, the first variant of the problem introduced in Chapter 3 is modelled by an Integer Linear Program (ILP) inspired from the literature. Instances of the problem are solved by Branch-and-Bound techniques using a commercial solver. As the problem

is $\mathcal{NP}$-complete, this exact generic approach becomes intractable when increasing the number of trains to be scheduled. Therefore, we decompose the model by first solving a new ILP formulation to find the time decisions that minimises a function evaluating conflicts in a timetable without requiring an explicit knowledge of the tracks assignment. Then, we inject this solution in the global model to find compatible tracks assignment. The time decision model being still challenging to solve using a solver, we propose three approaches to attempt getting solutions faster. The first method relies on a variables and constraints generation framework. At the beginning of the procedure, a heuristic provides starting times to the trains. Conflicts are evaluated and forbidden adding cuts. New possible starting times are sought and the ILP is solved again. The process is iterated until a conflict-free solution is found. The second method is a matheuristic. We also begin from a solution found heuristically. Iteratively, we select a train that is involved in the maximum number of conflicts and we solve a specific ILP to find which new starting time for this train allow the best conflict resolution improvements. The third method is similar to the second one, except that we use specific algorithms to solve each iteration of the conflict resolving faster than the solver. We indicate limitations of these methods and propose perspectives in order to improve them.

In Chapter 5, we work on the second variant of the problem described in Chapter 3. This variant increases the timetables realism by reusing the rolling stock and optimising their turnover times at terminus stations using real exploitation rules. We propose a Constraint Programming model that we solve with a solver propagation and branching techniques. We observed that solving the instances of the first variant of the problem with a solver is much faster than the best results obtained with the techniques described in Chapter 4. This result is the motivation of the introduction of the second variant of the problem, for which we find not only a conflict-free solution, but we desire the optimal one respecting to a criterion, in our case the turnover time of trains at terminus stations. In order to solve optimally our instances for this variant of the problem faster than the solver, we propose several enhancements to our model. First, we improve the detection of crossing constraints in particular settings, and we conceive families of filtering algorithms that improve the propagation process. Second, we derive lower bounds by relaxing the problem when considering the optimisation of turnover time of trains line by line at terminus stations. Third, we adapt a technique from the literature to make as many deductions as possible on inconsistent starting times during a preprocessing step to help the propagation process. We observed that the best use of these enhancements is made when integrated in a Branch-and-Check framework where we sequentially decide the starting time of a train and verify if a feasible tracks assignment still exist for the chosen times.

In Chapter 6, we describe a decision support tool for interactive timetabling, intended to be used by the timetable planners at *SNCF Réseau*. This software, resulting from a work with two intern students, allows to create or import, visualise and modify an infrastructure at a mesoscopic scale and a line plan. Using these data, the user can launch the automatic periodic timetable generation. If a conflict-free timetable along with compatible tracks assignment is returned, the user is still able to operate any modifications to the timetable and visualise potential conflicts that may appear. In particular, the user can modify the starting time of a train, extend the duration of any running or dwelling activity, modify the tracks assignment, and visualise the effect of these modifications.

Finally, in Chapter 7, we draw our conclusions about the contributions presented in this thesis and propose perspectives for future research.

1.4 List of publications

Article submitted to a Journal

G. Joubert, A. Jouglet, D. Nace, M. Postec and R. Bocquillon. A constraint-based approach to model and solve the periodic train timetabling with mesoscopic tracks assignment problem. Submitted.

International Conference communication

G. Joubert, A. Jouglet, D. Nace and M. Postec. An adaptive procedure for railway periodic timetabling and tracks assignment. *Proceedings of the 11th Triennial Symposium on Transportation Analysis*, Mauritius, 2022.

National Conference communication

G. Joubert, A. Jouglet, D. Nace, M. Postec and R. Bocquillon. Une approche basée sur la Programmation par Contraintes pour résoudre le problème d'ordonnancement de sillons périodiques à une échelle mésoscopique. *Proceedings of the 24th French Operational Research and Decision Support Society (ROADEF) annual congress*, Rennes, France, 2023.

Miscellaneous communications

- Metaheuristics Summer School 2020+1 workshop, online event, June 2021
- Ma Thèse en 180 secondes contest (SNCF), Saint-Denis, December 2021
- Heudiasyc SCOP team seminar (UTC), Compiègne, December 2021
- LIFAT ROOT team day, Tours, June 2022
- DOR/REST common days (GDR ROD), Saclay, October 2022
- GOTHA day (GDR ROD), Metz, November 2022
- Forum DGEX Solutions (SNCF), Saint-Denis, April 2023

Chapter 2

Operational Research in Railway planning: overview and positioning

In this chapter, we provide a literature review on research in railway planning. The traditional railway planning phases, as well as approaches aiming at integrating several planning phases, are described in Section 2.1. A focus on model and techniques to solve the Train Timetabling Problem is given in Section 2.2. We present complexity results and indicators assessing performance of railway schedules. When operations cannot be realised as planned because of delays, it may be necessary to reschedule duties and resources in real-time. Contributions in this direction are also presented. Finally, we position our work in tactical timetabling among the depicted railway planning landscape in Section 2.3.

Contents

2.1 Railway planning phases

Railway is a convenient transportation mode for passengers and freight. Organising railway circulations, though, is not a trivial task. Indeed, various resources are required to be available at the right place and at the right moment in order to make trips possible. These resources include the crews operating aboard trains, at stations and in operational traffic management centers, the rolling stock and the infrastructure. The railway planning process implies the collaboration of numerous actors on different temporalities, from several years until the day of operations. To handle this complexity, the scheduling of the whole system has traditionally been separated in subsequent problems easier to solve but still challenging, see Goossens et al. (2004) [88] for the planning stages of the Netherlands Railways, and Bussieck et al. (1997) [29], Huisman et al. (2005) [99], Caprara et al. (2007) [46], Lusby et al. (2011) [124] and Lusby et al. (2018) [123] for surveys on railway planning. Each of these problems belongs to an identified level depending on the remaining time before operations when the problem is solved. The strategic level corresponds to decisions made several years before operations and includes the Network design and the Line planning problems. The tactical level takes place some years to days before operations and includes several resource scheduling problems like Train timetabling, Rolling stock planning and Crew management. The operational level deals with the monitoring of operations in real-time and their rescheduling when necessary to limit the impact of potential delays or failures on the network. Borndörfer et al. (2017) [18] present successful implementations of decision support tools in the information systems of railway companies to solve problems of the three levels, i.e. a strategic freight train routing, a tactical rolling stock rotation planning and an operational real-time train dispatching. The aim of this section is to describe classic railway planning problems and to provide the interested reader with relevant literature for further details. We also present approaches looking at the integration of several planning problems to build solutions that make a better use of the resources.

2.1.1 Strategic level

At the strategic level of the railway planning process, decisions are made for a long term horizon and have a strong impact on the subsequent planning levels. Problems of the strategic level include Network Design, when the infrastructure is conceived or improved to satisfy future forecasted mobility flows, and Line Planning, when lines to be operated on the infrastructure are decided.

2.1.1.1 Network Design

The Network Design Problem aims at determining tracks and stations of a railway infrastructure to be built while satisfying a budget and maximising the utility of the investments. Contributions focused on railway Network Design exclusively are scarce. Indeed, most of railway networks have been built many decades ago, before the increasing popularity in the use of operational research techniques for railway planning. Magnanti and Wong (1984) [125] survey models, algorithms and computational results on the Network Design Problem for transportation planning. The authors describe a general model based on a multi-commodity flow where potential arcs representing new tracks should be added to the network to satisfy capacity constraints while minimising the construction costs. Other possible constraints are described like the selection of necessary arcs or the implication of selection of an arc depending on the selection of another arc. The model can be further specialised depending on the applications considered.

2.1.1.2 Line Planning

The goal of the Line Planning Problem is to determine an optimal set of railway lines to be operated on a given infrastructure assumed to be known. Each chosen line has a service frequency and a specific stop pattern i.e. a set dwelling operations at stations allowing passengers to board and alight. A solution to this problem is called a *Line Concept*, or a *Service Intention* when additional details like train connections to be ideally maintained and ideal turnover times at stations are indicated. A review on models and methods to solve this problem is proposed by Schöbel (2012) [169]. An operator-oriented objective can be the minimisation of the cost of the lines, whereas a passenger-oriented model would put a budget on the cost of the line concept and try to maximise the quality for the passengers that can be seen as an important number of direct travellers or fast connections between lines. A solution to the Line Planning Problem does not guarantee *a priori* that a feasible train schedule exists for this line plan. Instead of looking for a line plan from scratch, a classic method is to precompute a admissible *line pool* heuristically from which the lines of the solution are selected. In both cases, constraints bounding the total frequency of the selected lines on the infrastructure edges are also included in order to find more reasonable line plans. Bussieck et al. (1997) [28] state that the problem of finding a feasible line plan is $\mathcal{NP}$-hard by a polynomial reduction to EXACT COVER BY 3-SETS. The proof is given in Bussieck (1998) [27]. The authors present a MIP formulation for the Line optimisation problem maximising the number of direct travellers. In order to solve more instances, they simplify their model and adopt a cutting plane generation approach. Goossens et al. (2004) [88] propose a Branch-and-Cut approach using valid inequalities to minimise the cost of a hourly served line plan. Bull et al. (2019) [23] find

line plans minimising the estimated passenger travel time using an arc-flow formulation and a heuristic based on its relaxation.

2.1.2 Tactical level

Problems of the tactical level of the railway planning process focus on scheduling resources including crews, capacity on tracks and rolling stock, years to days before operations. The Train Timetabling, Train Platforming and Train Routing problems are closely connected. They aim at scheduling the departure and arrival time of trains at stations, at choosing the tracks and platforms where the trains should dwell at served stations, and at deciding the precise routes taken by trains between line tracks and platform tracks in stations. Those problems are traditionally solved assuming known infrastructures and line plans. However, the Train Timetabling Problem has recently also been addressed at the strategic level to evaluate or improve scenarios given from the Network Design and Line Planning problems, see Polinder et al. (2021) [148] and Sartor et al. (2023) [164]. Knowing the train timetables, a subsequent task is to organise the schedule of the vehicles that will compose the trains and to determine their movements from and to shunting yards. Other important problems of the tactical level are the Maintenance Planning of the infrastructure and of the rolling stock. Finally, crew schedules have to be set in order to perform all commercial and technical duties.

2.1.2.1 Train Timetabling

Given a railway infrastructure and a set of lines to be operated, the Train Timetabling is the problem of determining all the departure and the arrival times at the stations served by the trains of these lines. Generally, lower and upper bounds are known for any activity of running between a departure and an arrival, or any *activity* of dwelling at a station. Headway constraints prevent trains from occupying the same portion of track at the same moment. Cacchiani and Toth (2012) [38] survey the literature on Train Timetabling, where both the periodic and the aperiodic versions of the problem are presented. Models aiming at bringing robustness to the solutions are also discussed. In the periodic version of the problem, departures and arrivals of trains, often called *events*, occur at the same time of each period of T units of time, usually corresponding to one hour. Most of the existing periodic timetabling models are based on the *Periodic Event Scheduling Problem*, a suitable model presented in a seminal paper by Serafini and Ukovich (1989) [174]. A more recent survey on Train Timetabling is given in Caimi et al. (2017) [41], where several nominal and robust models are presented depending on the point of view of the infrastructure manager or of train operator companies that own the rolling stock and employ the crew on board. A more detailed focus on the Train Timetabling Problem is

provided in Section 2.2. We will recall basic complexity results, present existing models and solving methods and detail performance indicators sought in timetabling optimisation.

2.1.2.2 Train Routing

The Train Routing Problem occurs on the railway infrastructure around a station area. Given a train timetable, trains have to be routed through the station such that no two trains occupy the same platform track too close in time nor circulate on incompatible routes simultaneously. When all the considered trains have a stop at the station, the problem is also called Train Platforming Problem (see Cacchiani et al. (2015) [35] for a tutorial on the aperiodic version). In the decision problem, proved to be $\mathcal{NP}$-complete in Zwaneveld et al. (1996) [191], a conflict-free solution is thought including all considered trains. In case of infeasibility, dummy tracks can be used at stations, as well as small deviations of the timetable to gain flexibility. In the optimisation problem, the authors propose to maximise the number of routed trains using a Node Packing formulation. Zwaneveld et al. (2001) [192] model a Weighted Node Packing Problem to break ties with technical preferences while maximising the number of routing trains. The authors worked on a preprocessing step to speed-up the resolution. Caprara et al. (2011) [45] propose an Integer Linear Programming formulation for the Train Platforming Problem with a linearised quadratic objective function. The authors obtained good solutions using basic heuristics based on the strong linear relaxation of their model. Sels et al. (2014) [173] platform the maximum number of current trains on their preferred station tracks while considering a new set of trains to be platformed as well. Bai (2015) [3] addresses the Train Platforming Problem in the context of a dense railway traffic in complex stations. Bešinović and Goverde (2019) [13] propose a multi-objective model to minimise the average train delays and balance the traffic over the infrastructure to increase its lifetime before maintenance. Milliet de Faverges (2020) [129] solves a robust version of the Train Platforming Problem by analysing train delay distributions from historical traffic data. A survey on Train Routing problems is found in Lusby et al. (2011) [124].

2.1.2.3 Rolling Stock Planning

The Rolling Stock Planning or Vehicle Scheduling Problem is the task of deciding the specific train units to compose trains to be assigned to the scheduled trips of a timetable. According to the nominal problem described in Cacchiani et al. (2012) [31], important criteria to be minimised are the number of unseated passengers by choosing appropriate train compositions according to the expected demand, energy consumption and maintenance costs directly impacted by the number of vehicles used, and the number of composition changes at stations, e.g. operations of coupling or uncoupling of train units, that can have

a worsening effect on the robustness of a timetable in case of train delays. The authors tackle a robust version of the problem where large disruptions on the infrastructure between stations are considered. Borndörfer et al. (2017) [18] model the problem using a hypergraph to facilitate the expression of constraints representing in detail the possible changes of train compositions between two trips. Not only the number and the position of train units is indicated, but also their orientation conditioning the presence of the first and second classes at the front or at the back of the vehicles. The authors point out localisation constraints that rolling stock should also satisfy. First, at a given station area including shunting yards, enough train units have to be present at the end of a day to be able to operate all the scheduled trips of the beginning of the following day. Second, the train units, that run hundreds of kilometers, have to arrive on time at their specific maintenance site to realise their periodic maintenance operations.

2.1.2.4 Crew Management

The Crew Management is a famous problem in Operational Research, on wich Dantzig (1963) [58] worked to improve the planning of the military operations of the US Air Force and popularised the Linear Programming technique. The Crew Management Problem is to build a schedule for crews in order to cover a set of planned tasks. It have been widely used in airlines companies, as comment Caprara et al. (1997) [44]. The authors describe the problem in the railway context, and present various models and methods to solve it. Because of different temporalities regulating the time interval from the end of a task and the beginning of a new task performed by a same crew, the Crew Management Problem has naturally been decomposed in two subproblems. Let us consider a set of trips to be covered by crews every day. The Crew Scheduling Problem allows to create duties by grouping several trips which total span is no longer than a given maximum duration, e.g. two days in the case of the authors. The duties are then packed in rosters representing weeks in the Crew Rostering Problem, while details of each trip are not considered anymore at this step. The main objective of those problems is to require as less crew as possible to cover the set of trips, by a clever sequencing of the trips in duties and of the duties in rosters. Caprara et al. (1998) [48] solve the Crew Rostering Problem with a heuristic algorithm using lower bounds and find near optimal solutions for real-world instances.

2.1.3 Operational level

The operational level of planning stands for monitoring the railway traffic in real-time and making decisions to recover from degraded situations that may occur due to infrastructure or rolling stock failures, train delays, or any unexpected events that could disturb the

nominal planned operations. Cacchiani et al. (2014) [36] survey optimisation models and algorithms dedicated to re-schedule train timetables, rolling stock and crew plans individually as well as in an integrated manner. In order to avoid heavy adaptations of resource schedules in real-time as much as possible, performance indicators like timetable robustness can be optimised during the tactical phase, as we will see in Section 2.2.2. If these precautions are not sufficient with respect to the faced unexpected events, we will focus on proposed strategies to manage optimally the timetable re-scheduling in real-time in Section 2.2.3.

2.1.4 Integrated approaches

We saw that the railway planning process has traditionally been decomposed in subsequent problems easier to handle, but that each of these problems is already challenging. In the following, we present research projects looking at the integration of several problems previously mentioned. The results generally show that the solutions found in the integrated context make a better use of resources than solutions obtained when solving the problems subsequently.

2.1.4.1 Network Design and Train Timetabling

Nachtigall and Voget (1997) [134] search to build particular timetables for which passenger does not have to wait to take their connections. To attain this goal, the authors consider the possibility to upgrade track sections to increase their speed limit and thus reduce the running time on them. The objective is then a compromise between the waiting time of the passengers at stations and the cost of the network upgrading plan. Sub-optimal solutions are reached using a genetic algorithm. Qi et al. (2016) [150] work with a set of stations along a corridor consisting of a single or a double track railway line either in one-way or in two-way traversing mode. The authors show that for a given set of trains to be timetabled on the corridor, adding siding tracks at stations allow a reduction of the journey times thanks to the obtained extra dwelling capacity. The objective is to choose the relevant sidings tracks to be added at relevant stations in order to find a compromise between the cost of these extensions and benefits for the trip times.

2.1.4.2 Line Planning, Train Timetabling and Rolling Stock Planning

Schöbel (2017) [170] investigates the interaction between the Line Planning, the Train Timetabling and the Vehicle Scheduling problems. She proposes an *eigenmodel* to explore all the possible combinations to solve the three problems iteratively until convergence. A path in the eigenmodel corresponds to the order in which the problems are solved at each node on the path. The author points out that depending on the problems selected

at the beginning of the path, new Integer Linear Programming formulations could be found because such paths do not necessarily follow the traditional sequence for solving these problems. The author concludes that the eigenmodel could be extended to an integration of more planning stages, by considering the problems traditionally occurring before Line Planning and after Vehicle Scheduling. She also observes a possible use of such eigenmodels in other areas like supply chain and production scheduling. Pätzold et al. (2017) [137] explore three specific paths of the eigenmodel and derive methods to integrate aspects of vehicle scheduling during the Line Planning Problem to avoid getting bad quality solutions as it can be when the three problems are solved sequentially and without iterative loop. We will now focus on problems integrating Line Planning and Train Timetabling in the one hand, and on problems integrating Train Timetabling and Rolling Stock Planning on the other hand.

Line Planning and Train Timetabling. Jiang et al. (2017) [102] search to increase the number of train running on the high-speed highly congested double track corridor JingHu between Beijing and Shanghai respecting to an actual timetable in order to meet the future passenger demand. To do so, the authors allow an increase of dwelling times and a modification of the stopping patterns, i.e. adding or removing few stops. The latter option is usually a part of the Line Planning Problem. The problem is solved in short computation time by a heuristic based on the one of Cacchiani et al. (2010) [33]. Qi et al. (2021) [149] integrate time-dependant passenger demand to this problem and show the effect of this consideration on the resulting stop plans and timetables. Cacchiani et al. (2020) [37] addressed the problem in the case of an uncertain passenger demand distribution. Brethomé (2018) [22] presents a passenger-oriented multi-objective Line Planning model integrated with a Train Timetabling model taking into account the route choice of passengers, for the mass transit of the Île-de-France region. Burggraeve et al. (2017) [24] look for robust timetables obtained from the construction of a line plan from scratch. The line plan is questioned at each iteration of the process in order to improve the buffer times between the trains. The authors validate their heuristic on instances of the DSB S-tog network of Copenhagen. Yan and Goverde (2019) [185] propose an iterative model to cope with strongly heterogeneous railway lines on a high-speed corridor. In their Line Planning model, a wide flexibility is given for the frequency to be determined for each line. The regularity of the train timetable is then optimised in a subsequent model, that feeds back the Line Planning model with constraints on the frequency of the lines for further improvements. Fuchs et al. (2022) [78] solve iteratively the Line Planning and the Train Timetabling problems. A MIP and a SAT formulations are proposed for the latter problem. In case of infeasibility of the timetabling model, constraints representing *conflicts* between lines are added to the Line Planning model. The authors show that

banning a conflict between potentially numerous combinations of lines is more efficient than banning a specific combination of lines for the convergence of the method. Zhang et al. (2022) [188] integrate Line Planning and Train Timetabling using an ILP model to cover the passenger demand along with a multi-commodity network flow model. The authors develop a dual decomposition strategy based on the Alternating Direction Method of Multipliers to solve efficiently the coupling constraints of the two models. The approach is applied on a corridor with trains running in one direction.

Train timetabling and rolling stock planning. Kroon et al. (2014) [108] exploit rolling stock flexibility constraints based on the *Periodic Event Scheduling Problem* model to authorise vehicles to be scheduled in different lines and obtain a gain on the number of necessary vehicles to cover the timetable. Carosi et al. (2019) [50] use a matheuristic to solve a multicommodity-flow model for the integration of timetabling and vehicle scheduling in the bus transportation context. The resolution of real-world instances show how the possibility to use less vehicles and drivers. Grafe et al. (2022) [93] integrate vehicle circulation decisions in real-time rescheduling and manage to reduce the delays incurred by passengers in close-to real-world instances. The problem is solved by a heuristic using derived bound in reasonable computation time.

2.1.4.3 Train Timetabling and Routing

In the following, we take a look at contributions on timetabling with platforming and routing considerations. Petering et al. (2016) [144] present a Mixed-Integer Linear Programming based approach to minimise the cycle time and the journey time of a train timetable while considering the platforming of the trains on a single line traversed in one-way with multiple tracks at stations. Using computed bounds and a preprocessing step, life-size instances are solved within one hour with a commercial solver. Zhou et al. (2020) [190] minimise the total travel times of trains timetabled on a high-speed one-way network considering their platforming at stations. A constraint reduction strategy allows the Mixed-Integer Linear Programming formulation of the authors to be solved faster.

Integration with Maintenance Scheduling. Some approaches consider an integration of Train Timetabling with Routing and Maintenance decisions. Wust et al. (2019) [181] extend the *Periodic Event Scheduling Problem* to model track-choices at stations, as well as flexibility in the timetable. That is, a solution for train arrival or departure corresponds to an interval of times instead of a unique time. This flexibility allows to find a feasible timetable by considering several track maintenance scenarios, i.e. some double tracks are converted into two-way single tracks during the maintenance of the other track. Zhang et al. (2022) [187] integrate the planning of maintenance tasks on the

infrastructure to the timetabling and routing optimisation. The authors take into account different running times depending on the routes taken by the trains. The objective is to minimise the deviation from the initial schedule of the maintenance tasks while minimising the weighted train running times. A heuristic is used to reduce the search space. The approach presented by the authors outperforms the performance obtained using a commercial solver.

Robust approaches. Several contributions look for an improvement of the available buffer times between trains. Dewilde et al. (2014) [66] propose a tabu search framework to improve the spread of the trains in the timetable. Using a simulation tool, the authors observe a diminution of 25% in the delay propagation of their solutions. Dewilde et al. (2013) [65] pursue the same objective with an iterative method consisting at optimising the timetable, the routes taken by the trains, or their assigned platforms at stations. The method has been validated on instances from two busy stations in Belgium. Burggraeve and Vansteenwegen (2017) [26] present an approach in which the routing of the trains is decided in the first place in order to minimise the maximum use of specific tracks in the station area of Brussels. The timetable is built once the optimal routing has been sought. The method is improved in Burggraeve and Vansteenwegen (2017) [25] by evaluating the passenger robustness using a simulation tool in order to further adapt the buffer times between trains.

2.1.4.4 Rolling Stock Planning and Train Unit Shunting

The Train Unit Shunting Problem aims to manage train units that are coupled or uncoupled at stations due to Rolling Stock Planning decisions. As a consequence, the train units that stay in station areas have to be parked in shunting yards composed of parallel tracks that most of the time are accessible from one end only. The shunting movements from stations tracks to shunting yard tracks and vice versa, the track assignment in the shunting yard, and the train matching with an arrival and a departure train for a given train unit, are the key features of the Train Unit Shunting Problem. As this problem may easily be infeasible and thus provoking delays at stations, some approaches look at the integration of the Rolling Stock Planning with the Train Unit Shunting Problem to gain degrees of freedom. Haahr and Lusby (2017) [96] demonstrate that solving the integrated problem with Branch-and-Cut approaches, a solution can be found on instances of the DSB S-tog that were infeasible when considering the two problems subsequently. Kamenga (2020) [104] defines the Generalized Train Unit Shunting Problem by adding rolling stock maintenance operation scheduling in the integrated problem of Rolling Stock Planning with Train Unit Shunting. The author proposes a Mixed Integer Linear Programming formulation, and obtains good solutions in about one hour of computation

for instances of the Metz-Ville station. Additional algorithmic contributions allowed the author to speed-up the resolution.

2.1.4.5 Rolling Stock Planning and Crew Scheduling

We finalise our overview on integrated railway planning phases with approaches considering simultaneously the scheduling of rolling stock and crews. Freling et al. (2003) [77] model the integrated problem in the context of bus transportation. The objective is to minimise the number of used vehicles and the cost of scheduled crews. The problem is solved using a column generation approach. The authors use the solutions obtained from the sequential approach to evaluate the potential benefits of the integrated approach. It turns out that the integration is beneficial, even more when the cost of a scheduled crew is important with respect to the cost of a scheduled vehicle. Benhizia (2012) [8] proposes a Mixed Integer Linear Programming model to integrate the railway Rolling Stock Planning and the Crew Scheduling problems satisfying all the technical and legal requirements. The problems are either considered separately and jointly by applying coupling constraints. As the entire MILP struggles to be solved in reasonable amount of time for real-world instances using a commercial solver, algorithmic contributions based on the decomposition of the model are proposed to improve the performance of the approach.

2.2 Train Timetabling Problem

We dedicate this section to the Train Timetabling Problem, which is one of the keys of the railway planning process. We saw in Section 2.1 that the problem can be addressed as early as during the strategic level, when integrating Network Design or Line Planning decisions. In Section 2.2.1, we focus on various models and methods dealing with the tactical nominal problem. Several railway performance indicators that improve the quality of timetables like robustness are presented in Section 2.2.2. Finally, we present two streams of research aiming at solving the problem at the operational level in Section 2.2.3.

2.2.1 Tactical nominal problem

Solving the Train Timetabling Problem is the art of deciding departure and arrival times to a set of trains running on a railway infrastructure while minimising the passengers travel time and satisfying constraints modelling a conflict-free use of the tracks. This problem is generally solved at the tactical level, years to months before actual operations. Two variants can be identified, depending on the type of time horizon considered. Caprara et al. (2002) [43] address the *aperiodic* version of the problem, for which the scheduled arrival or departure times of trains are not enforced to repeat cyclically e.g. hourly within

the day. The notion of period is however mentioned in the contribution as the timetable can be applied every day. A tutorial on the aperiodic Train Timetabling Problem is given by Cacchiani et al. (2015) [35]. On the other hand, the *periodic* Train Timetabling Problem has also encountered a high interest from researchers thanks to the attractiveness of the obtained timetables for the passengers. Indeed, such timetables offer several mobility alternatives easily memorised for the same line. Serafini and Ukovich (1989) [174] introduce the *Periodic Event Scheduling Problem* (PESP), a seminal contribution that has been the basis of many more works on periodic Train Timetabling. The authors model the train arrival and departure times by nodes of an *event* graph, in which arcs between nodes stand for *activities* e.g. running from a departure event to an arrival event. The nodes are the variables of the problem, a time to be decided in the domain $[0, \mathrm{T} - 1]$, so that the events are repeated at the same time after at each period of duration T. The arcs are the constraints of the model, representing the activities duration, bounded between a minimal value and a maximal value. For instance, the travel time between stations A and B has to take between 5 and 7 minutes. Or, the dwelling time at station B has to be 3 minutes, representing the case where the lower bound of the activity equals its upper bound. The authors prove the $\mathcal{NP}$-completeness of the problem by polynomial reduction to HAMILTONIAN CIRCUIT, and propose a Branch-and-Bound procedure to solve the feasibility problem. Odijk (1998) [136] also proved that the problem is $\mathcal{NP}$-complete by reduction to GRAPH COLOURING. Recent parameterised complexity results given by Lindner and Reisch (2022) [122] show that deciding the feasibility of PESP is $\mathcal{NP}$-complete even for small instances with specific structural properties. Nachtigall (1996) [130] adds an objective function to periodic programs and minimises the waiting time of passengers during their connections at stations. The author proposes an improvement of a Branch-and-Bound method to solve more efficiently instances for which operated lines have different frequencies. Liebchen and Möhring (2002) [116] improve the timetables of the Berlin underground by solving a PESP formulation using preprocessing steps and a commercial solver. Liebchen (2004) [113] extends the PESP model by adding symmetry constraints. This requirement is applied in practice, e.g. in Switzerland, in France, and means that all arriving trains should be at a given station at a given symmetry minute and leave the station after the symmetry minute. This nice property allows the passengers to find comfortable connections, at the price of an increased rigidity in the timetable and difficulty to find feasible solutions. Liebchen and Möhring (2007) [117] further extends the PESP model to a vehicle scheduling aspect by showing how to minimise the number of required vehicles in a timetable. Liebchen (2008) [114] presents the 2005 Berlin underground timetable that is the first one to have been optimised in practice, saving travel and connection times for passengers and one train for BVG, the operator company.

In the following, we present modelling specificities and solving techniques arising in contributions on the Train Timetabling Problem.

Service intention. In periodic Train Timetabling, a *line concept* (see Schöbel (2012) [169]) is a set of lines operated periodically on the railway infrastructure, for which we know the *stop pattern* i.e. the list of stations served by the line, and the frequency of the line. The notion of *service intention* (see Wust et al. (2008, 2019) [183, 181]) enriches the line concept by also providing rotation times at terminus stations and desired connections between trains. Caimi et al. (2011) [42] introduce the *periodic service intention* as a manner to specify the periodicity of the lines and therefore to also deal with aperiodic lines.

Fixed activity durations. The following contributions as well as a group of instances in Fuchs et al. (2022) [78] have considered the case where the durations of the trains activities are fixed, i.e. the upper bound is moved to the lower bound of the activity. The authors observe that feasible solutions are harder to be found in this setup. Domschke (1989) [70] presents a quadratic assignment model for the Synchronisation Problem and heuristics to minimise the waiting time of passengers. Nachtigall and Voget (1996) [133] address the same problem with a Mixed-Integer linear Program solved by a Genetic Algorithm. The authors show that the problem is NP-Hard even with fixed activity durations. Großmann et al. (2012) [95] encodes the PESP in a SAT formulation and concludes that SAT solvers perform better than Constraint-based ones. Pätzold and Schöbel (2016) [138] use the term of *reduced PESP* to call the PESP for which the activity durations are determined. The objective is to minimise the passengers waiting time. The authors formulate the problem and propose an exact approach, a matching-based heuristic, and a hybrid algorithm combining the two methods.

Mesoscopic infrastructure scale. The PESP is suited to build timetables at the macroscopic scale of the infrastructure. At this scale, specific routing constraints are not considered, and stations are seen as black boxes. The maximal number of simultaneous authorised trains at a station can be given to increase the realism of the timetables. On the other hand, the microscopic scale of the infrastructure describes faithfully the train dynamics and the section blocking times on the tracks through a detailed representation of the signalling system. A majority of contributions using at the microscopic scale address the real-time Train Scheduling problem on a delimited sector, as we will see in Section 2.2.3. De Fabris et al. (2014) [62] introduce the *mesoscopic* infrastructure scale, an compromise that eliminates some drawbacks of the two other scales. First, the mesoscopic scale is more detailed than the macroscopic one as it explicitly allows to assign tracks to

trains at stations and to check for (mesoscopic) routes incompatibilities around stations. Second, it is still possible to generate timetables on relatively expanded infrastructure sectors at the mesoscopic scale, for which models are much less combinatorial than large models of the microscopic scale. As an example in a railway node, several microscopic routes between a given line track and a given station track would correspond to a unique mesoscopic route. Wust et al. (2019) [182] extend the PESP by considering the possibility to choose tracks at stations. Isobe et al. (2019) [100] generate periodic train timetables at the mesoscopic infrastructure scale using a solver based on the Satisfiability Modulo Theories. Masing et al. (2023) [127] address a periodic train timetabling problem using a mesoscopic scale infrastructure to take into account more precisely headway constraints in the context of construction sites that render some tracks unavailable.

Single track line. Heydar et al. (2013) [98] consider the problem of scheduling a set of two different types of trains on a single track line with siding tracks at stations to allow overtakings, and traversed in one-way direction. The objective is to minimise the cycle time in which the lines periodically repeat. The authors solve a Mixed-Integer Linear Program using a commercial solver. Petering et al. (2016) [144] extend this work by adding more siding tracks at stations. Daudet (2017) [61] presents algorithms to solve the train timetabling problem on single track lines of the tunnel under the Channel, between Calais and Folkestone. The ticket pricing problem for the trains taking the Eurotunnel has also be addressed by the author.

Flexible timetable. Caimi et al. (2011) [40] extend the PESP by providing *flexibility* to the events. That is, conflict-free time intervals are searched for each train arrival and departure instead of unique times. The authors conclude that this method only increase the computation time moderately, while allowing to find feasible timetables easier at the microscopic scale from the flexible macroscopic solutions.

Dynamic demand. Barrena et al. (2014) [5] address the Train Timetabling problem of the Madrid suburban area where the dynamic nature of the mobility demand over time is considered. Thus, the researched timetable is not periodic. The authors propose three linear formulations based on flow variables and minimise the passenger waiting time using a Branch-and-Cut technique.

Passenger routing. The Line Planning Problem for periodic timetables aims at selecting a relevant set of lines to be operated on the transport network in order to propose high quality connections for passengers. However, two distinct feasible periodic train timetables obtained from the same line concept can result in drastically different journey times for specific origin-destination trips. Therefore, a research stream has emerged to

take into account the passenger routing so that the travel times remain comfortable for most passengers. Siebert and Goerigk (2013) [175] present an iterative approach by re-timetabling trains and changing line frequencies to improve the passengers journey times. The authors use a quadratic formulation, choose the passengers paths by solving randomised shortest paths in the transport network, and iterate until no improvement is found during two iterations. Schmidt and Schöbel (2015) [166] point out the fact that the Line Planning and Train Timetabling problems are solved assuming a knowledge on the passenger routes while in practice these routes strongly depend on the timetable at hand. The authors propose approaches to solve the timetabling problem simultaneously with the passenger routing. Robenek (2016) [157] places the passengers at the centre of timetabling process. To do so, the author develops models that decrease the passenger waiting time and number of transfers, allow a partial periodicity in the timetable, and integrate revenue management. Schiewe and Schöbel (2020) [165] argue that the Train Timetabling Problem taking passenger routing into account is even harder than the classic version of the problem. They propose a preprocessing step to reduce the problem size and a heuristic that considers a subset of the passengers to derive bounds on the objective function. The method is evaluated on close-to-real-world instances. Grafe and Schöbel (2021) [94] propose to solve the PESP with a larger non-periodic constraint network. The authors show that this can have an interest for integrating PESP and Delay Management which is presented in Section 2.2.3.2, and suppose that it is a promising way to integrate PESP and passenger routing. Polinder et al. (2022) [146] minimise the passenger travel times in a train timetable while also considering the waiting times at the origin stations. The authors compute the ideal timetable from the passenger perspective without taking the infrastructure constraints into account. Then, they restore the feasibility of the timetable using a Lagrangian heuristic, and search for further improvements for the passengers while conserving the feasibility of the timetable. Gattermann et al. (2016) [81] use a SAT based approach to integrate passenger routing when solving the periodic Timetabling Problem. Kümmling et al. (2016) [110] improve the traffic assignment in train timetables using constraint propagation techniques. More recently, Yao et al. (2023) [186] developped a multi-commodity flow model solved with the Alternating Direction Method of Multipliers in order to integrate passenger routing and stop planning in their train timetabling problem. The authors tested the algorithm on the high-speed railway network of Shandong region in China.

Valid inequalities. Authors e.g. Peeters (2003) [139], Liebchen and Möhring [117] or Lusby et al. (2011) [124] present the alternative *Cycle Periodicity Formulation* of the PESP attributed to Nachtigall (1999) [131], a Habilitation Thesis that we did not access. In this formulation, constraints are not any more applied on *potentials* (event

times), but on *tension* variables defined as the periodic difference between the potentials at extremities of arcs of the PESP graph. Peeters (2003) [139] observes that the *Cycle Periodicity Formulation* is solved more efficiently than the classic PESP formulation. To further increase the solving performance of the PESP, various *valid inequalities* have been explored. A necessary condition on the feasibility of periodicity variables of the PESP depending on lower and upper bounds of activities forming cycles has been proposed as the Theorem 2 in Serafini and Ukovich (1989) [174]. The authors however remark that this condition is not sufficient, because of the existence a small infeasible PESP instance for which all the cycles though meet the condition. Therefore, there is more room to find properties able to better detect infeasible instances. Let S a set of linear inequalities determining a polytope P. Chvátal (1973) [52] defines the closure $S^{(1)}$ of S such that $S^{(1)}$ contains S and the smallest set of linear combinations of inequalities of S with the right-hand side rounded down and while they have a role in cutting off a new part of P. It is established that the polytope $P^{(1)}$ determined by $S^{(1)}$ contains all the integral points of P. It is theoretically possible to search for the Chvátal closure of $S^{(1)}$ and so on until determining a polytope equivalent to the convex hull of the integral points of P. The so-called *Chvátal rank* of S is the minimum number of Chvátal closures necessary to determine the convex hull of the integral points of P. Odijk (1996) [135] explores the first Chvátal closure of the PESP polytope by introducing the *cycle inequalities*. A cycle inequality is a linear combination of PESP constraint inequalities along an elementary cycle of activities in the constraint graph, for which the right-hand side is rounded down. As the number of these inequalities can be exponentially large in the number of arcs of the PESP constraint graph, Odijk (1996) [135] implements a *PESP Cut Generation* procedure that iteratively adds a cycle inequality to the constraints in the case where periodicity variables are found without allowing to build a feasible timetable, thus returning an infeasible cycle to be cut. Liebchen and Swarat (2008) [119] present the *change-cycle inequalities* introduced in Nachtigall (1999) [131], which definition is more complex than the cycle inequalities but use the same variables. The authors use the change-cycle inequalities to prove that the Chvátal rank of the PESP is at least $\frac{T}{2}$ where T is the periodic time horizon. They indeed showed that on a simple PESP instance, a specific change-cycle inequality could not be generated earlier than at the $\frac{T}{2}$-th Chvátal closure. The authors also explore the second Chvátal closure by introducing the *multi-circuit cuts*. These cuts are obtained by combining several cycle inequalities and by rounding down the obtained right-hand side. The relevance of these cuts has been proved on a class of infeasible PESP instances forming wheel graphs with an even number of nodes. The cycle inequalities and the change-cycle inequalities turned out to be valid for these infeasible instances while the multi-circuit cuts proved the infeasibility. The authors show that these new cuts are relevant to speed-up the solving computation time on some PESP instances, but that heuristics should be

applied to better determine in which context these cuts should be generated during the resolution. Borndörfer et al. (2020) [17] give an algorithm to separate cycle inequalities in pseudo-polynomial time and a proof of pseudo-polynomial time separability for the change-cycle inequalities. The authors also show that there is no algorithm with a better complexity for the separation of this inequalities. Lindner and Liebchen (2020) [120] show that flipping the orientation of some arcs of the PESP constraint graph, it is possible to obtain *flip inequalities* enclosing both cycle and change-cycle inequalities.

Model decompositions. Several contributions describe strategies to decompose the train timetabling problems to ease the resolution of large instances. Caimi et al. (2009) [39] propose to decompose railway networks with *condensation zones* where the railway network is the more complex and saturated like main railway stations, and *compensation zones* where time supplements are easier to introduce. The authors also reduce the model by adjusting the time granularity. They formulate the problem on condensation zones by an independent set problem in a conflict graph and solve it heuristically. Zhou et al. (2017) [189] formulate the Multi-periodic Train Timetabling problem on a double track railway corridor between Bei-jing and Shang-hai stations. In this problem, a distinct periodicity is to be optimised for different set of trains. The authors propose a Lagrangian relaxation of the problem that allows a decomposition into several subproblems responsible for the periodicity of one set of trains each. Herrigel et al. (2018) [97] present a sequential decomposition for the PESP using two control parameters. The trains are scheduled by priority groups which quantity depends on the first parameter. Once trains are scheduled, their times can still be shifted by a number of minutes depending on the second parameter, in order to allow flexibility for the last priority groups. If infeasibility occurs, a backtrack is performed and the shift parameter is increased. Leutwiler and Corman (2022) [112] address the problem of aperiodic train timetabling at the microscopic scale of the infrastructure, in which the network is composed by dense zones sparsely connected. The authors propose a logic-based Benders decomposition for which the logic Benders cuts are computed with a SAT-based algorithm. Other approaches based on Constraint Programming (see Baptiste et al. (2001) [4]) have also been proposed. Tormos et al. (2006) [179] propose three decomposition strategies to solve the Railway Scheduling problem more efficiently. The distributed Constraint-based models are divised by a partition of the constraint network, by trains or by contiguous stations. The second decomposition performed the best. Salido et al. (2007) [160] qualifies the general constraint network decomposition as *domain-independent* whereas the decompositions obtained from a deeper problem study e.g. by trains or by contiguous stations are said *domain-dependent* and provide better results.

Modulo network simplex. Nachtigall and Opitz (2008) [132] adapt the network simplex algorithm to the case of programs dealing with modulo equations. The authors thus introduce the *modulo network simplex method* that they use to minimise heuristically the waiting time of passengers in a multi-periodic train timetabling problem. Goerigk and Schöbel (2013) [86] improve the modulo network simplex algorithm by speeding-up the running times of the inner loop of the algorithm and improving the quality of the cuts found in the outer loop of the algorithm so that it is possible to escape from local optima. The authors conclude that this method outperforms the classic one as well as a state-of-the-art commercial solver. Goerigk and Liebchen (2017) [85] describe an heuristic aggregation method to reduce the size of PESP instances, along with an iterative procedure using the benefits of both the improved modulo network simplex algorithm and a commercial solver. Indeed, the authors remark that the solutions are not encoded in the same way in the two methods, and that switching form one method to the other thus allows to better escape from local optima and to improve the overall performance. This iterative approach allowed the authors to improve the objective value of a all PESPlib[1] instances (16 at this time).

Conflict resolving. Polinder et al. (2018) [147] address the problem of resolving initially infeasible PESP instances by making the minimal number of allowed modifications on the right-hand side of the PESP constraints to eliminate the conflicts. Reisch et al. (2021) [152] present a local search algorithm to build the annual timetable for all freight trains in Germany. A column generation approach allows to attribute time slots for a part of freight trains to be scheduled. The problem is represented as a Maximum Independent Set (MIS) problem where each conflicting slots are linked in a conflict graph. In order to schedule more trains and escape local optima, the solution is perturbed, creating conflicts that are then resolved. The approach of the authors performs better than a commercial solver and has a comparable performance to the Iterated Local Search, that is the best known MIS heuristic. The authors are able to schedule more than 5000 freight trains with their method. Kümmling et al. (2015) [109] present a state-of-the-art method to compute periodic train timetables of the German network including long distance trains, local trains and freight trains. The solver used is applied on a SAT encoding of the instances. In case of infeasibility, minimally unsatisfiable subformula are identified to perform slight modifications on the data and restore the solution feasibility. The authors argue that this method can be used to easily handle future modifications in the timetable requirements without demanding much computation.

[1] `https://num.math.uni-goettingen.de/~m.goerigk/pesplib/` (Accessed: 25 June 2023).

Heuristic approaches. Caprara et al. (2006) [47] address the Train Timetabling Problem with additional requirements like the maximal number of trains simultaneously present at stations, fixed timetables for a subset of trains, or maintenance on a set of tracks. The authors incorporate and solve these new constraints using a Lagrangian heuristic. Cacchiani et al. (2008) [32] propose both exact and heuristic approaches to solve the periodic versions of the Train Timetabling Problem. The originality of their methods rely on the variables expressing the full timetable of each train, allowing to obtain a stronger LP relaxation of the integer formulation. Borndörfer et al. (2010) [20] present a rapid branching approach to solve the maximal weighted railway track allocation problem. Garrisi and Cervelló-Pastor (2020) [80] present a Mixed-Integer Linear Programming formulation to model the Train Scheduling problem that aims to minimise the train delays. A heuristic is used to build an initial population of solutions that are then improved through a genetic algorithm.

Concurrent solver. Borndörfer et al. (2020) [19] introduce a PESP solver putting algorithms in concurrency. Contrary to Goerigk and Liebchen (2017) [85] that solved PESP by using iteratively the modulo network simplex method and a commercial solver, the authors launch the two methods in parallel along with a heuristic based on maximum cuts. This concurrent approach allowed to improve lower and upper bounds of all PESPlib instances (20 at this time). Lindner and Liebchen (2022) [121] present two strategies to merge a set of good quality periodic timetables. These strategies are implemented in the concurrent solver. As a result, the authors improve the objective value of the first and a last instances of the first set of PESPlib instances. Bortoletto et al. (2022) [21] propose a heuristic based on tropical geometry that uses the min-plus algebra. The authors implement their heuristic in the concurrent solver and improve the objective value of five PESPlib instances.

2.2.2 Performance indicators

The main objective of tactical train timetabling that we have seen above is to find arrival and departure times for a set of trains at traversed stations such that the resulting planned operations are conflict-free and minimise the passengers travel time. However, timetables implemented in practice should also prove an acceptable quality with respect to railway performance indicators. These indicators inform about the ability of a timetable to cope with delays occuring during railway operations. Among them, definitions of *infrastructure occupation*, *robustness*, *stability* and *resilience* are presented by Goverde and Hansen (2013) [92]. In the following, we focus on contributions aiming at measuring or optimising such railway performance indicators.

2.2.2.1 Infrastructure occupation

Given a train timetable in which all the train departures from their origin station span a time horizon H, a method described in the UIC 406 Leaflet [180] allows to compress the timetable, i.e. to minimise the span period of the same trains by removing all idle times between trains without changing their speed profile. If the maximum span between all train departures in the compressed timetable is h, then the infrastructure occupation of a timetable is defined by the ratio of h over H, usually given as a percentage called *Occupancy Time Rate*. The UIC 406 Leaflet [180] gives recommendations on the maximum Occupancy Time Rates tolerable depending on the topology of the infrastructure considered, the type of traffic or homogeneity of the train speed profiles, and the period during the day. Timetables for which the Occupancy Time Rate is superior to the prescribed one should thus be difficult to be operated if delays occur, because of the lack of residual capacity on the infrastructure to absorb the delays. Abril et al. (2008) [1] use timetable compression methods and timetable saturation to evaluate the capacity of Spanish railway infrastructures. The saturation of a timetable consists in adding as many trains as possible in a timetable in order to evaluate its residual capacity. Other authors worked on timetable saturation. Delorme et al. (2001) [64] use a Constraint Programming model and a unicost set packing formulation, respectively solved by a greedy heuristic and the GRASP metaheuristic (see Feo and Resende (1995) [73]), to saturate timetables on the Pierrefitte-Gonesse junction near Paris. Sels et al. (2014) [173] measure the *realistic capacity consumption* of a station by platforming a set of extra trains together with a set of current trains. In several contributions, extra trains represent an increase of the mobility demand to consider for the upcoming years. Cacchiani et al. (2016) [34] propose an Integer Linear Programming formulation to solve the Train Timetabling Problem in complex railway nodes linking several stations, from the Infrastructure Manager point of view. From the *ideal timetables* requested by the Train Operators, the authors schedule as many trains as possible while trying to satisfy the ideal times expressed. The authors also apply a saturation iterative heuristic to try to schedule more trains and thus to evaluate the saturation of railway nodes. This method allowed to conclude that the available capacity was not sufficient to schedule all the Train Operator demands, even when adding some new tracks in the railway nodes. Pellegrini et al. (2017) [142] use an algorithm based on a Mixed-Integer Linear Program to saturate the railway line between Rouen and Mantes-la-Jolie at the microscopic scale of the infrastructure, which represents a large network at this scale of detail. Petering et al. (2016) [144] perform the capacity analysis of a single line traversed in one-way with multiple tracks at stations by minimising the cycle time allowing to schedule a given set of periodic trains. Jensen et al. (2017) [101] use compressing methods to assess the infrastructure occupation of a sequence of trains. They also compute the capacity consumption of the sequence of trains by keeping

the critical headways during the compression. Results are obtained within little computation time and using a small amount of input data. The method allows to compare the number of trains circulating in a given period for several infrastructure scenarios.

2.2.2.2 Stability

A timetable is said to be stable if, when a delay of few minutes occur on a train, the consequences of this delay should be absorbed within a known *settle time* without needing to reorder or reschedule trains in real-time. Goverde and Hansen (2013) [92] use the example of the German norm stating that the consequences of a primary delay of ten minutes should be absorbed in two hours. It is therefore necessary to plan time allowances additionally to the minimal required technical times to perform railway operations. Indeed, normal small variations on the duration of some tasks are expected in practice and not taking these variations into account makes it difficult for train drivers to catch up a potential delay on their trip. Goverde (2007) [89] develop methods relying on the max-plus algebra to compute the delay propagation in a large-scale railway network in real-time and measure the duration of the perturbation. Goverde (2010) [90] improves the previous method by also considering train arrival and passing events instead of only departure events. The author develops a graph algorithm that allow computing the delay propagation in the system proportionally to the number of delayed events. Delorme et al. (2009) [63] present a multi-objective optimisation framework to obtain a timetable which minimises the number of conflicts between a set of trains, maximises the set of scheduled saturating trains, and maximises the preferences of the decision-maker like the trains to be selected, their routes and arrival dates. The authors also measure the stability of the solutions depending on several durations of initial delays impacting trains. Sparing and Goverde (2017) [177] propose a model optimising both the feasibility and the stability of the timetable in a single step, by minimising the cycle time of the periodic timetable within which operations every railway operation is repeated. Bešinović et al. (2019) [15] work on the case where infrastructure capacity is not sufficient to schedule an entire set of trains. To avoid generating unstable timetables, the authors solve heuristically a Mixed-Integer Linear Programming formulation minimising the cycle time of the timetable while relaxing the train line journey times as few as possible and cancelling as few trains as possible. The results show that higher transportation demand can be met using this method. The common point with the work of Sparing and Goverde (2017) [177] is that the minimisation of the cycle time allowed the authors of both contributions to inject cleverly allowance times between the critical train operations of the timetable.

2.2.2.3 Robustness

Robustness is a very popular indicator for solutions of optimisation problems applied to various industrial domains. Seeking for robust solutions to real-world problems is indeed attempting to maintain feasibility of the solutions when the uncertainty of input data at hand will be revealed. Several strategies has been set up to face uncertainty, we are now describing classic ones.

Robust optimisation. Consider constraints of a convex linear program using parameters which values are delimited in uncertainty sets. *Strict robustness* (Soyster (1973) [176]) aims at finding an optimal solution that remains feasible for all possible realisations of the parameters within the uncertainty sets. This technique is rarely applied in practice because of its overconservatism. The objective value of strictly robust programs can be heavily impacted, and even the existence of feasible strictly robust solutions can be questioned in some cases. Moreover, in many settings it can be unrealistic to want a protection against the worst case for each of the uncertain parameters. This first approach inspired many researchers to search for methods to achieve robust solutions with a fair objective value with respect to the nominal optimal solution. Ben-Tal and Nemirovski (2000) [7] remark that optimal solutions for a set of linear programs are not feasible any more when adding a slight level of uncertainty to the input data. The authors present a *Robust Counterpart* of the problem as a conic formulation to find optimal solutions having a parameterised probability to be feasible against a parameterised level of uncertainty. The authors observe that the loss of optimality respecting to the nominal solutions is small for a reasonable level of *immunity* against uncertain data. Bertsimas and Sim (2004) [10] present a linear tractable method in which at most Γ parameters of the model can be simultaneously subjected to uncertainty. A difficulty is to find the relevant value of Γ to keep an acceptable probability to find feasible solutions after the uncertainty is revealed, without decreasing too much the objective value of the robust solutions. This compromise to be found is the so-called *Price of Robustness*. In some other settings, the uncertain problems to be solved are such that a first set of decisions are to be taken under uncertainty, while a second set of decisions called *recourse decisions* can be taken after the uncertainty is revealed. The solutions obtained are therefore less conservative because of the adaptation possibility in a second phase. This kind of two-stage robust optimisation problems can be formulated as an *Adjustable Robust Counterpart* that integrates the two sets of decisions. To make this problem tractable, Ben-Tal et al. (2004) [6] restrict the second-stage variables to be an affine function of the uncertain data, thus formulating the *Affine Adjustable Robust Counterpart* of the problem.

Stochastic Programming. A common characteristic of the robust methods described above is that the values of the problem parameters are contained within given *uncertainty sets*. In other contexts, however, such sets are unknown and parameters are instead stochastic variables for which the distribution is either given or evaluated by sampling methods. This second family uncertainty settings can be dealt with using *Stochastic Programming*, a framework introduced by Birge and Louveaux (2011) [16]. The main idea is to optimise the expected value of the objective function of a problem subjected to a set of independent *scenarios* or *realisations* of the uncertain parameters. Considering many scenarios in the formulation allows to find a more robust solution but is also more difficult to solve because of the growth of the model.

Robustness is a goal that has been widely researched in the solutions of railway planning problems. In particular, the robust Train Timetabling Problem aims at determining a clever distribution of *buffer times* between train circulations at their potential conflict points in order to absorb as much as possible the propagation of potential delays. Cacchiani et al. (2012) [38], Caimi et al. (2017) [41] and Lusby et al. (2018) [123] surveyed the literature of train timetable robustness. In the following, we focus on the description of some contributions looking for robust timetables. While many approaches are based on the techniques mentioned above, we will see that new robust schemes like *light robustness* or *recoverable robustness*, applicable in various industrial domains, emerged from railway research.

Timetable robustness measures

According to Goverde and Hansen (2013) [92], a train timetable is considered robust if it minimises both the occurrence of initial delays by planning a sufficient amount of time allowance between each operation of a train, and the occurrence of secondary delays (induced by other delayed trains) by distributing buffer times to protect optimally the operations against the most potential delays. Many ways to capture the timetable robustness have been proposed, among which we mention two examples in the following.

Robustness in Critical Points. Andersson et al. (2013) [2] present a measure that quantifies *Robustness in Critical Points*. Such critical points correspond to dependencies between two trains that overtake each other or run on incompatible routes relatively close in time in a train timetable. Contrary to other previous approaches of the literature, this one not only points out the sensible parts of the infrastructure where the timetable is not robust, but it also gives indications on which train dependencies should be treated in order to increase the timetable robustness. The Robustness of a Critical Point between

two trains at a station is computed as the sum of (i) the buffer time that the first train disposes before entering the station, of (ii) the time difference between the departures of the two trains from the station, and of (iii) the buffer time that the second train disposes after leaving the station. That is, in case of delays, the more the value of this sum, the more adaptations can be made to avoid propagating the delay too much. The authors conclude that this measure could be included as an objective function in an optimisation model in order to build robust timetables instead of solely evaluate timetables robustness.

Expected passenger time. Sels et al. (2016) [172] define the robustness of a train timetable depending on the expected time that passengers should spend in the railway network in practice. To do so, the authors use distributions for the initial delays of the trains as well as for the secondary delays impacting two trains depending on the buffer time between their conflicting activities. Resulting expected delays are explicitly penalised in the (sum of picewise linear) objective function of a Mixed-Integer Linear Programming formulation, weighted by the known number of passengers that should be impacted by those delays. This method allowed to reduce the expected passenger time of the Belgian timetable of about 3.8% in peak hours in about two hours of computation time. Sels et al. (2015) [171] use the same objective function as a practical tool to evaluate and compare the robustness of several existing timetables and determine the one to apply in practice.

In the following, we describe three families of contributions aiming at finding robust train timetables. We classify them either among iterative, one-stage or two-stage approaches.

Iterative approaches using simulation

Kroon et al. (2008) [107] present an iterative approach to improve the robustness of given train timetables for which the order of the trains (or timetable structure) remains unchanged. The research question is how to re-allocate the buffer times in the timetable so that it become less sensitive to delays. The authors use an iterative framework with a stochastic program using delay realisations to re-allocate buffer times and a simulation tool to evaluate the modified timetable. Convergence is obtained when no more improvements are obtained. Bešinović et al. (2016) [14] solve a Linear Integer Program to obtain a timetable at the macroscopic scale of the infrastructure, with both attractive passenger travel times and safe buffer times between trains. A Monte Carlo simulation is used to assess the robustness of the timetable. The process is repeated until the timetable is robust enough. Afterwards, the timetable feasibility and stability of the timetable are evaluated at the microscopic scale of the infrastructure. Additional time modifications are performed

through the various modules of the framework until obtaining a robust timetable at the macroscopic scale that is also feasible and stable at the microscopic scale. Cornet (2020) [57] presents a *simheuristic* to allocate buffer times optimally between suburban trains in dense railway lines connected to Paris Saint-Lazare station. A timetable is given in input with fixed train orders. The robustness of this timetable is evaluated with a simulation tool. A simulated annealing metaheuristic is applied to optimise the re-allocation buffer times. The framework is iterated until reaching a convergence criterion.

One-stage approaches

Lagrangian heuristic. Cacchiani et al. (2012) [30] studied the optimal distribution of buffer time for a set of lines scheduled on a corridor using methods of Kroon et al. (2008) [107]. The authors use a Lagrangian heuristic to solve the Robust Train Timetabling Problem. Artificial parameters are added in the formulation to control the robustness of the solutions. During the solving procedure, the weight associated to this control parameters are progressively increased, in order to find a diversified panel of Pareto-optimal solutions going from more efficient (in term of the cost of the solutions) to more robust. The choice of the relevant solution to apply in practice is leaved to the final user.

Light Robustness. Fischetti and Monaci (2009) [74] introduce a framework called *Light Robustness* to deal with optimisation under uncertainty. The authors comment that the flexibility of the solutions obtained with Light Robustness is similar than when using two-stage Stochastic Programming thanks to slack variables representing recourse actions to be taken in case of important revealed uncertainty. However, the model is as simple to set as a classic robust linear program. Indeed, Light Robustness consists in fixing (i) a limit on the degradation of the objective value with respect to the one of the nominal optimal solution, and (ii) the ideal level of additional conservatism of the parameters in the constraints. Slack variables are integrated in the formulation in case of impossibility to find a fairly qualitative solution that respects the added conservatism. The objective of Light Robustness is therefore to minimise the sum of the slack variables. The authors compared this method with Stochastic Programming on instances of the Train Timetabling Problem. It turns out that for equivalent achieved level of robustness, the solutions are found is less computation time using Light Robustness. Fischetti et al. (2009) [76] present the design of an application aiming at computing nominal timetables, improving their robustness and validate the robustness level of the improved solutions. The contribution particularly concentrates on the step of robustness improving, given computed nominal timetables. For the validation step, a tool based on Linear Programming is used. Regarding the robustness improvement step, the Light Robustness model is competitive in

term of computation time, gained robustness and implementation ease when compared with three distinct Stochastic Programming settings. Liebchen et al. (2010) [118] improve the robustness of timetables by slightly extending the Light Robustness model in the sense that no slack variables are used but the objective function is adapted instead. The achieved timetables are qualified *delay resistant*, meaning that realistic delays do not propagate too much when testing the timetables in real-time conditions. More details about Delay Management are provided in Section 2.2.3.2. Goerigk et al. (2011) [84] address the problem of *robust Timetable Information* corresponding to the selection of particular trains to be taken by passengers from their origin to their destination and that remain good options even in presence of delays. The authors analyse the journey time difference with the nominal solutions when using strict robustness meaning that no train connection should be broken under delays, and Light Robustness. The first method turns out to be too conservative while the second provides relatively small journey time supplements for most passengers.

Two-stage approaches

Recoverable robustness. Liebchen et al. (2009) [115] introduce a two-stage method to find *delay resistant* timetables. *Recoverable robustness* of a solution is reached if for all considered delay scenarios, an algorithm exists to recover the feasibility of the solution with a limited effort. The recovery phase corresponds to the Delay Management problem.

Recovery-to-optimality. Goerigk and Schöbel (2014) [87] introduce the concept of *Recovery-to-optimality* as a new two-stage approach to deal with uncertainty. Depending on the application, finding a feasible solution can be not a requirement at the first stage. Given a definition of distance between two solutions representing a recovery cost, the objective is to find a solution for which the maximum distance to the optimal solution of each uncertainty scenario is minimised. The main differences with Recoverable robustness is that *Recovery-to-optimality* seeks to recover a not only feasible solution but an optimal one, and that it is not specified in advance how the recovery should be done but it should be possible thanks to the low recovery cost. The authors exemplify the concept by comparing the resolution of the aperiodic Train Timetabling Problem in the nominal situation and under uncertainty with strict robustness, Light Robustness, Recoverable robustness and *Recovery-to-optimality*. The results show that *Recovery-to-optimality* solutions offer good objective values without guaranteeing feasibility, and are hence useful if changes can be done after the uncertainty is revealed.

Adjustable robustness. Polinder et al. (2019) [145] present an adjustable Robust counterpart of the Periodic Timetabling Problem. That is, a timetable is found along with a set of recourse variables that represent a selection of decisions rules to be taken depending on a set of disturbances scenarios. Real-world instances are solved heuristically by limiting the recourse variables to be linearly dependent to the disturbances.

2.2.2.4 Resilience and vulnerability

Resilience is defined in Goverde and Hansen (2013) [92] by the flexibility given to a timetable to cope with delays by taking actions in real-time. In this sense, it is highly connected with the two-stage robust approaches present above, as they prepare a timetable to be easily operable to face disturbances. Mattson and Jenelius (2015) [128] survey two related concepts called *vulnerability* and *resilience* of transport systems. The first concept evaluates what are the risks for the system, their probability to happen and their consequences for the society. It is summarised as *knowing what to expect*. It can be assessed either by a topological study of the network, or either more in detail by analysing a larger amount of data that describes the expected flows on the network and the consequences of disruptions on them. The second concept represents the ability of a system to recover its nominal state after undergoing a disruption. It is summarised as *knowing what to do*. Bešinović (2020) [11] reviews the literature on *resilience* and related notions. The evolution of a resilient system performance is described as follows. Robustness aims at maintaining the system in its nominal states when facing small disturbances. If a disruption occurs, a phase of *survivability* begins where the performance of the system goes from the nominal to a degraded state. Afterwards, a phase of *response* is launched with actions to cope with the disruption while no drastic improvement of the system performance is yet visible. Finally, the situation is handled and the performance of the system is progressively restored to the nominal state during the *recovery* phase. *vulnerability* corresponds to the performance level difference between the nominal state and a potential degraded state of the system. Szymula and Bešinović (2020) [178] model the problem of vulnerability assessment in railway networks. A Mixed-Integer Linear Programming formulation aiming at finding the location where a given number track failures has the most impact for passengers is presented. The problem is solved heuristically by column-and-row generation. Results show that the worst failure locations are more related to the traffic demand than to the infrastructure topology in itself. Yan et al. (2019) [184] propose a multi-objective model to minimise the vulnerability of periodic timetables while minimising the journey times, the number of overtakings and maximising the trains regularity. Solutions approaches to build the four-dimensional Pareto frontier are tested on two instances and a lower capacity utilisation is used to break the tie between equivalent solutions in the Pareto frontier. Bešinović et al. (2022) [12] present an approach to ad-

dress the *response* phase of the resilience scheme presented in Bešinović (2020) [11] and assuming that multiple disruptions occurring in a railway network make the *survivability* phase duration negligible. The authors assess the resilience of the system when explicitly considering infrastructure restoration additionally to traffic management. The method helps decision makers to quantify adequately the impact of potential several disruptions occurring simultaneously in the network.

2.2.2.5 Energy efficiency

On the top of the railway performance indicators presented above, Goverde and Hansen [92] argue that the total energy required to operate a given timetable is also a valuable objective to optimise. Chevrier et al. (2013) [51] compute optimal train running time to be used as input for the Train Timetabling Problem. To do so, the authors use train dynamics and minimise both the running time and the energy consumption of train trips between stations by using an bi-objective evolutionary algorithm. Goverde et al. (2016) [91] present a multi-level approach to compute train timetables that optimise most of the performance indicators presented in Section 2.2.2, including energy efficiency. A microscopic module is used to verify feasibility and the stability of the timetable. At the macroscopic scale, journey times and robustness are optimised. Finally, robustness is further improved in a fine-tuning mesoscopic model, as well as energy consumption. The authors conclude that the integration of performance indicators during the timetable construction avoids a cumbersome validation step using simulation to detect flaws that are then not trivial to be resolved.

2.2.3 Real-time rescheduling

When important disturbances occur meaning that some operations take more time than planned, even robust and stable timetables may no longer be feasible without the action of train dispatchers in real-time. In this case, recoverable robust or resilient timetables may provide the train dispatchers with more options to cope with the degraded situation and eventually recover the nominal timetable as soon as possible. Cacchiani et al. (2014) [36] survey the literature on Train Timetabling and Rolling Stock rescheduling under disturbances and disruptions. The latter case designates larger perturbations caused by failures or unexpected events and that could require rolling stock and crew rescheduling and even train cancellations. In the following, we focus on contributions aiming to manage the railway traffic in real-time under disturbances. Two streams of research have been addressed in recent years. In the one hand, the Train Scheduling looks at the detection and the resolution of conflicts at the microscopic scale of the infrastructure. On the other hand, the Delay Management searches to answer the question if, given a known delay

occurring on a train, is it more beneficial for other trains to wait at stations so that passengers of the delayed train could maintain their connection, or to depart on time to avoid creating a snowball effect in the propagation of delays? We will present approaches for each stream of research as well as contributions integrating Train Scheduling and Delay Management.

2.2.3.1 Train Scheduling

Train Scheduling is a problem faced in real-time when railway operations are delayed at such a point that the nominal planned timetable is no more operable. To recover from this degraded situation, train dispatchers who monitor the railway traffic can reorder the arrival of trains at specific points of the infrastructure by presenting restrictive signal aspects to drivers or by giving them adapted speed indications. The dispatchers have also the possibility to reroute the trains so that they avoid the conflicting zone. Among various subjects, a review on real-time Train Scheduling is provided in Pellegrini and Rodriguez (2013) [143]. The authors compare air and rail transportation in Europe, by analysing the differences between the infrastructures, the strategic process of capacity allocation on the infrastructure, and the real-time traffic management. In the following, we present models and methods that have been used to solve Train Scheduling in real-time.

Integer Linear Programming formulations. D'Ariano et al. (2008) [59] model the Conflict Resolution Problem using an alternative graph formulation, where priority decisions are taken between trains in case of delays. The authors show the beneficial effect of flexible timetables for the real-time rescheduling. In flexible timetables, a time interval is planned for each arrival or departure event, letting more degrees of freedom to the train dispatchers. It turns out that timetable flexibility can ease the reduction of delays during real-time operations. D'Ariano and Pranzo (2009) [60] present a dispatching tool able to solve the short-term Train Scheduling and evaluate the propagation train delays for an upcoming time horizon of several hours, that is usually intractable. To do so, the authors decompose the time horizon into smaller time windows and solve the problem in cascade. The relevance of the approach has been validated on instances of a railway line between Utrecht and Den Bosch. Corman et al. (2010) [54] improve the dispatching tool by implementing new heuristics in a tabu search framework, indicating that better results are found in less computation time compared with a previous version of the tool. Corman et al. (2011) [56] extend the cases handled by their dispatching tool by addressing the problem in the complex station area of Utrecht. The dynamic rerouting and reordering possibilities implemented in their dispatching tool allow a diminution of train delays compared to the basic management rules applied in practice. Gély (2010) [82] proposes a continuous time model, a discrete time model and a hybridisation of both to solve the

real-time Train Scheduling on instances of lines between Paris and Lyon and between Bordeaux and Toulouse, and a Parisian line between Saint-Lazare and Gare du Nord stations. The author observed that few minutes of perturbations could generate thousands of seconds of delay in a network. Pellegrini et al. (2014) [141] introduce the real-time Railway Traffic Management Problem as Mixed-Integer Linear Program. Random traffic instances generated on the infrastructures of the triangle of Gagny near Paris and in the area of Lille-Flandres station are solved. The authors observe statistically that the use of a fine granularity of the infrastructure is crucial to be able to contain the delay propagation. Pellegrini et al. (2015) [140] extend the previous approach. In particular, they use valid inequalities and implement time-limited heuristics selected using an algorithm-configuration procedure. High quality results satisfying the real-time requirements are reported on instances based on three topologically different railway networks in France. Samà et al. (2016) [162] formalise the real-time Train Routing Selection Problem, that consists in selecting a relevant subset of routes to speed-up the resolution of the real-time Railway Traffic Management Problem. The routes selection is done through Ant Colony Optimisation. The authors observed an improvement of the results when using routes selection before solving the real-time Railway Traffic Management Problem. Samà et al. (2017) [163] wonder when is the most beneficial moment to compute an optimal subset of routes to be used during the real-time Train Scheduling. The authors observe that when computing routes at the tactical level right after constructing a Train Timetable, then these routes are the optimal ones if the faced disturbances in real-time are similar to historically known disturbances. A gain of computation time is therefore obtained in real-time thanks to the subset of routes already available. However, if disturbances are strong and different from usual ones, then the authors prefer computing dedicated routes in real-time as it becomes more efficient. Samà et al. (2017) [161] solve the Conflict Detection and Resolution problem using a Mixed Integer Linear Programming formulation and metaheuristics like Variable Neighbourhood Search or Tabu Search to reroute trains. The method is applied on various European busy railway networks. Fischetti and Monaci (2017) [75] show that real-time Train Scheduling can be solved efficiently using a Mixed Integer Linear Programming commercial solver on the top of which a heuristic preprocessing method has been applied. Results on instances based on a UK network are reported.

Constraint Programming approaches. Rodriguez and Kermad (1998) [159] address the problem using Constraint Programming. An assumption taken in their model is that all trains run at the maximum speed. Results on instances from the complex junction of Pierrefitte-Gonesse near Paris-Nord station validate the method. Rodriguez (2007) [158] improves the precedent approach by considering speed variation of the trains. Two

Constraint-based models are presented. The first one takes into account acceleration and deceleration times and allows a diminution of up to 95% wasted time in the system compared to actual decisions taken by the operators. The second model neglects acceleration times, slightly reducing the quality of the solutions while accelerating the resolution and allowing to tackle larger instances. Marlière et al. (2023) [126] present a new Constraint Programming approach using interval variables to solve the real-time Railway Traffic Management Problem. The authors compare the efficiency of their method with a MILP model and propose an hybridisation of both approaches to obtain even better solving performance.

2.2.3.2 Delay Management

Delay Management aims to minimise the overall time spent in the railway system by passengers under disturbances. Most of the models dealing with Delay Management are described using the macroscopic scale of the infrastructure, neglecting assuming the feasibility of the taken decisions. The question is, considering the average benefit of all the passengers, if connecting trains should wait for delayed trains, or if they should depart on time to contain the effect of delay propagation. König (2020) [105] provides a review on Delay Management contributions and classifies them following on a taxonomy describing if the objective function is passenger oriented, the importance of the disturbances, the infrastructure scale, the determinism and completeness of the input, and the exactness of the approaches. The concept of Delay Management has been introduced in Schöbel (2001) [167]. The author formulates an Mixed Integer Quadratic Program that is then linearised. Upper and lower bounds are proposed and a Branch-and-Bound procedure is indicated in perspective to solve the problem.

Tracks capacity constraints. Schöbel et al. (2009) [168] integrates macroscopic tracks capacity constraints in Delay Management, a feature that had been neglected so far. Solving the problem on instances with different headways and numbers of sources of delays, the author shows that the headway constraints only have a small impact on the sum of passengers delays and on the number of cancelled connections, while improving the feasibility of obtained *disposition timetables*. The two proposed heuristics are fast compared with the performance of a solver. The author remarks that an additional microscopic model could indicate only few more adaptations to the solutions to validate their microscopic feasibility. Indeed, the macroscopic headway constraints integrated to the Delay Management model are not as accurate than ones representing microscopic blocking times on specific locations of the infrastructure. However, the solutions obtained while integrating macroscopic tracks capacity constraints to Delay Management are more realistic than those obtained with the classic version of Delay Management.

Rerouting of passengers. Dollevoet et al. (2012) [69] present a model to incorporate rerouting of passengers to Delay Management. Without this consideration in the model, passengers who miss their connection have to wait for the next train of the same line, what may not be the case in practice. An integer programming model is presented, in which the decisions of waiting or departing for the trains are connected with a shortest path problem for the passengers. Preliminary results are provided on a small instance of the Dutch railway network, where the rerouting of passengers is observed in the solution.

Rerouting of passengers and tracks capacity constraints. Dollevoet et al. (2015) [68] propose a model both considering the rerouting of passengers and improving the tracks capacity constraints of Schöbel et al. (2009) [168] by taking into account the specificities of tracks occupation at stations. A first approach is to constrain the maximum number of trains that can be at a station at the same time, whereas a more powerful approach is to explicitly assign trains to station tracks. An iterative heuristic is applied to solve the problem. An optimal solution for the Delay Management problem is found with fixed tracks assignment at stations. From this *disposition timetable*, a second module improves the station tracks assignment in polynomial time.

Rerouting of passengers and train capacity constraints. König and Schön (2021) [106] formulate a Mixed Integer Nonlinear Programming model to allow the rerouting of passengers due to train delays and to the limited number of passengers that can be present in a train. Several approaches using different linearisations of the model are tested and compared to models of the literature that neglect the train capacity constrains. It turns out that rerouting of passengers due to train capacity limits is observed in the solutions of all considered scenarios. The results validate the impact of the train capacity constraints in Delay Management decisions.

2.2.3.3 Integration of Train Scheduling and Delay Management

We saw that Train Scheduling in real-time allows to provide train dispatchers with optimal decisions considering the microscopic scale of the infrastructure to recover from disturbances and eventually operating back the nominal planned timetable. However, these solutions may not be the best from the passengers point of view. Delay Management addresses this issue by minimising the sum of delays of the passengers. Decisions are taken on connecting trains that should wait, and if not, reroute passengers in other trains. This approach is generally applied at the macroscopic scale of the infrastructure, and several capacity constraints has been integrated to improve the feasibility of the solutions. Some contributions bridge the gap between these two streams of research, in order to resolve disturbances optimally for the passengers while assessing the feasibility of the decisions

at the microscopic scale of the infrastructure. Corman et al. (2012) [55] formulate a bi-objective model to find a compromise between the minimisation of the overall train delays and the minimisation of the sum of valued connections. Dollevoet et al. (2014) [67] present an iterative approach for solving the macroscopic Delay Management problem and validate the *disposition timetable* microscopically by solving the Train Scheduling and adapt the solution if necessary. The approach is tested on real-world instances around Utrecht station. Corman et al. (2017) [53] propose a Mixed Integer Linear Programming formulation in which Delay Management and Train Scheduling are integrated. This single model allows the authors to evaluate the optimality gap of the solutions found by fast heuristics that they use to solve the problem. For large instances, the authors propose a lower bound computation that outperforms the one found with a commercial solver. The heuristics are based on the decomposition of the model into a Train Scheduling model and a Passenger Routing problem. The results show the possibility to reduce passenger delays in practice.

2.3 Positioning

We saw in this chapter that the railway planning process is a complex task often decomposed into well studied smaller problems that are themselves very challenging. Based on these strong bases, however, many contributions have put their interest in integrating planning phases, decreasing the computation times and improving the quality of the solutions. The Train Timetabling Problem is an example of still very active topic of research in all the strategic, tactical and operational planning levels, as can attest the numerous recent works reviewed in this chapter. Several maps connecting railway planning problems have been proposed in the literature e.g. in Lusby et al (2011) [124] and in König et al. (2020) [105]. In Figure 2.1, we propose to emphasise the influence of Train Timetabling in all the railway planning levels and its interaction with several problems.

At SNCF Réseau, the French Infrastructure Manager, the annual train timetables are constructed through the expertise of timetable planners using conception tools, and are then validated by microscopic conflict detection, by evaluation of the infrastructure capacity consumption and by stability assessment under stochastic delay injections. In this thesis, we aim to conceive algorithmic approaches to automatically generate periodic train timetables, mainly focusing on *feasibility* of the solutions. We indeed believe that generating conflict-free timetables is necessary to allow the drivers not to meet restrictive signals and generate delays, as mention Goverde and Hansen (2013) [92]. Our work is however intended to be extended, as we do not take passenger connections and solutions robustness explicitly into account.

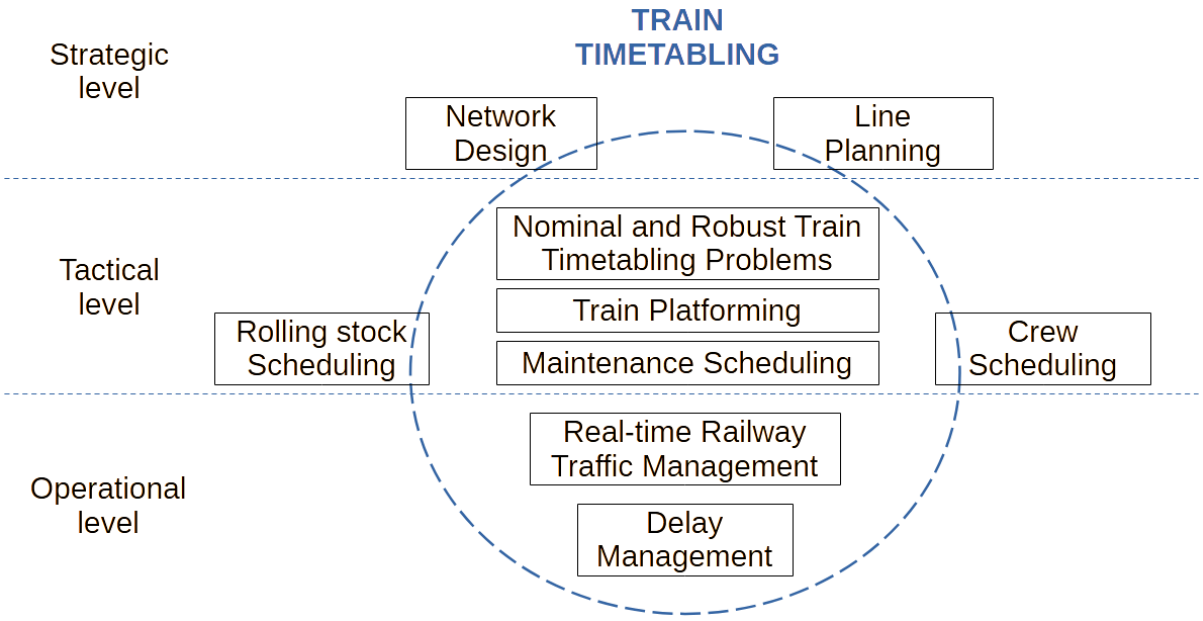


Figure 2.1: Influence of Train Timetabling on all railway planning levels.

We address the nominal tactical periodic Train Timetabling Problem integrated with the Train Platforming Problem at the mesoscopic scale of the infrastructure (see De Fabris et al. (2014) [62]). This infrastructure scale allows a fair consideration of track occupation constraints that we will present in detail, while keeping a model size closer to a macroscopic formulation than to a microscopic one. Like in the reduced PESP (see Pätzold and Schöbel (2016) [138]), we limit in this work our exploration of the timetables with nominal activity durations, i.e. the minimal technical durations plus already included classic percentage robustness buffer times. This feasibility problem thus minimises the passenger time for all direct travellers. It is still known to be $\mathcal{NP}$-complete even if the activity durations are fixed. Moreover, in our context the PESP constraint graph is not entirely known given that we also make mesoscopic routing decisions by choosing tracks for the trains in stations. These elements comfort us in the idea that the periodic Train Timetabling with mesoscopic tracks assignment problem is also $\mathcal{NP}$-complete. It is therefore vain to hope finding an algorithm to solve the problem in polynomial time (see Garey and Johnson [79]).

We start our investigations in Chapter 4 by proposing a model taking the assumption that trains leave the system few minutes after the arrival at their terminus station. We present a Mixed-Integer Linear Program for which we were inspired by works including non exhaustively Serafini and Ukovich (1989) [174] for the periodic model, Wüst et al. (2019) [182] for its mesoscopic track-choice extension, and Yan and Goverde (2019) [185] for the choice of boolean periodicity variables. As larger instances are difficult to solve in a reasonable amount of time, we decompose the model by deciding first the timetable and then the tracks assignment. Our main contribution is a conflict evaluation objective

function that does not need explicit tracks assignment knowledge. We use this function in a tabu search framework (see Glover (1986) [83]) to find timetables for which a mesoscopic tracks assignment is reachable in most of the cases. Additional investigations are necessary to detect more complex types of conflicts.

We improve the realism of the solutions in Chapter 5 by reusing the rolling stock at terminus stations. This feature allows us to optimise the turnover times of the rolling stock at terminus stations through the point of view of the Infrastructure Manager, using real data describing the exploitation rules at stations. This optimisation can be viewed as one ingredient towards the construction of robust solutions and should be extended to other constraints of the model, namely the track occupation headway constraints. We present a Constraint Programming (see Baptiste et al. (2001) [4]) formulation, lower bounds and strengthening techniques, dedicated filtering algorithms based on feasibility-led dominance rules (see Jouglet and Carlier (2011) [103]) and using edge finding techniques (see Carlier and Pinson (1994) [49]), and branching strategies embedded in a Branch-and-Check framework.

In the next chapter, we describe more into the details the problems and the instances that we will solve in the two subsequent chapters using the methods mentioned in the two previous paragraphs. Afterwards, we present the user application OptiSillons in Chapter 6, developed in collaboration with two intern students at DGEX Solutions, that is destined to be used by timetable planners to automatically generate periodic timetables at the mesoscopic infrastructure scale and optimal in the sense of the rolling stock turnover time at terminus stations. We finally draw conclusive thoughts and future work in Chapter 7.

Chapter 3

Description of the periodic Train Timetabling with mesoscopic tracks assignment problem

This chapter presents a description of the problems studied in this thesis. We first introduce background notions and notations in Section 3.1. Then, we succinctly describe the differences between the decision problem and the optimisation problem. Finally, we present instances in Section 3.3 that we will solve in Chapter 4 and in Chapter 5.

Contents

3.1 Background notions and notations

In this section, we introduce notations to describe the problems addressed by presenting the data in Section 3.1.1 and decision variables in Section 3.1.2.

3.1.1 Parameters

We present the mesoscopic infrastructure in Section 3.1.1.1, the Service Intention in Section 3.1.1.2 and the safety headways in Section 3.1.1.3.

3.1.1.1 Infrastructure data

Sections. The infrastructure considered in our application is composed of sections containing a set of parallel mesoscopic tracks K_s, with $|K_s| \geq 1$. Each track is composed by two rails on which a train can roll and an overhead catenary supplying traction power. Let S be the set of sections. Each section $s \in S$ is either a station or a line section. We note S^l the set of line sections and S^t the set of stations, together defining a partition of S.

Track directions. Depending on the mesoscopic track of the infrastructure, trains may be allowed to circulate in direction + that we arbitrary call the *downstream* direction from the left to the right in Figure 3.1. On the same track, trains may also be allowed to circulate in direction −, called the *upstream* direction, from the right to the left in Figure 3.1. The set of allowed direction(s) of track k is noted $D_k \subseteq \{+, -\}$.

Routes. Two sections $s_1, s_2 \in S$ are connected if and only if there is a track $k_1 \in s_1$ and a track $k_2 \in s_2$ such that there is a mesoscopic route from k_1 to k_2 or from k_2 to k_1. By existence of a mesoscopic route from a mesoscopic track to another, we mean there is at least a route from the former track to the latter at a microscopic scale. When s_1 and s_2 are connected, we say by extension that there are routes from s_1 to s_2 or vice versa, without clarifying the tracks involved in the routes. The set of routes of the infrastructure R is partitioned in two sets, namely the set of downstream routes R^+ and the set of upstream routes R^-. Hence, $R = R^+ \cup R^-$.

Zones. Let Z, a partition of S, be the set of zones. That is, each section of S is included in a single zone and a zone contains one or several sections. We call bizone a couple $< z_1, z_2 >$ with $z_1, z_2 \in Z$ such that there is a section $s_1 \in z_1$ and a section $s_2 \in z_2$ such that s_1 and s_2 are connected. That is, there are routes of R^+ from z_1 to z_2 and routes of R^- from z_2 to z_1.

Routes compatibility. Let us consider two mesoscopic routes $r_1, r_2 \in R$ of a bizone $< z_1, z_2 >$ with $z_1, z_2 \in Z$. We say that r_1 and r_2 are compatible if and only if there are routes r'_1 and r'_2 at the microscopic level corresponding to r_1 and r_2 and on which the signaling system allows two trains to circulate simultaneously.

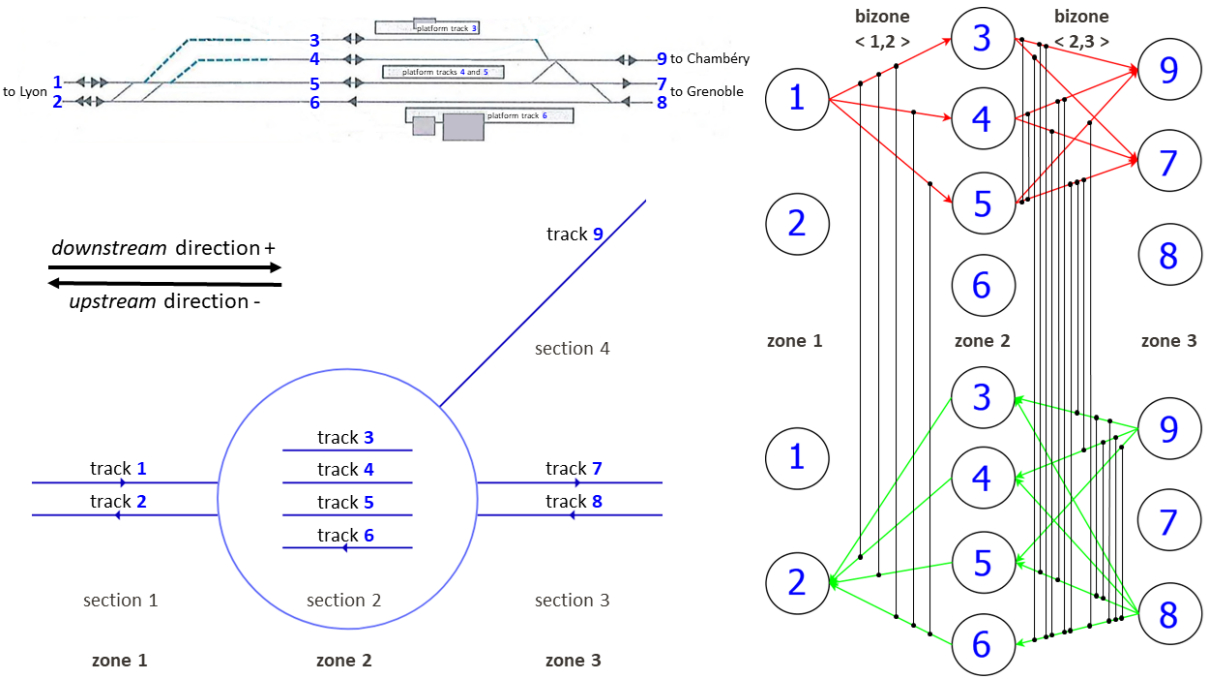


Figure 3.1: Mesoscopic railway routes and their compatibilities derived from the microscopic scale of the infrastructure. **Top left part**: A microscopic representation of Saint-André-le-Gaz station. Arrows on tracks 1 and 2 indicate that in absence of disruption or work on those tracks, the direction on track 1 is downstream while the direction on track 2 is upstream. Blue dashed lines represent planned microscopic tracks to be installed. **Bottom left part**: Mesoscopic infrastructure around Saint-André-le-Gaz station at section 2. Line sections 1, 3 and 4 go to Lyon, Grenoble and Chambéry respectively. Section 4 has a single track. Sections 3 and 4 are both elements of zone 3. **Right part**: Graph of routes compatibility. Vertices, representing the mesoscopic tracks, are duplicated. On the top graph, red arcs represent downstream mesoscopic routes between tracks of a bizone. On the bottom graph, green arcs represent upstream mesoscopic routes between tracks of a bizone. Black edges show compatibility between two routes of a bizone.

To illustrate those concepts, we derive a mesoscopic infrastructure from a microscopic one in Figure 3.1, with 3 zones, 4 sections and 9 tracks. Section 2 is a station. Tracks 1 and 7 are circulated downstream only, whereas tracks 2, 6 and 8 are circulated upstream only. Tracks 3, 4, 5 and 9 can be circulated in both directions. There is a downstream mesoscopic route from track 1 to track 4 and an upstream mesoscopic route from track 5 to track 2. Even though these routes seem not to cross each other in the mesoscopic representation, note that no compatibility edge is linking downstream route (1,4) with upstream route (5,2). Indeed, the two microscopic routes corresponding to (5,2) are using

common portions of the infrastructure with the microscopic route corresponding to (1,4). This example shows that pairwise compatibility between mesoscopic routes are taken into account using microscopic considerations instead of only being deduced from the graph of mesoscopic routes.

3.1.1.2 Service Intention

Let the Service Intention (SI) be a set of $\frac{N}{2}$ bidirectional lines to be timetabled within a cyclic period. Each line of the SI corresponds to a downstream slot and to an upstream slot that are scheduled on the same sections and that are potentially linked at terminus stations. Depending on the period and on the chosen time granularity, we obtain a cyclic time horizon of T time units. For example, a period of one hour with a granularity of half a minute gives $T = 120$ time units.

Each slot $i \in I = \{1, \ldots, N\}$ circulates in one known direction provided by $d(i) \in D$ and is composed of n_i successive operations. Each operation $j \in J_i = \{1, \ldots, n_i\}$ of a slot $i \in I$ is noted $o_{i,j}$ and represents either a passing through or a dwelling activity at a station, or a running activity on a line section. An operation $o_{i,j}$ has a known duration $p_{i,j}$. We note $J_{i-} = \{1, \ldots, n_i - 1\}$ the set of operations of slot i except the last one. In this work, we will assume that no additional slack time is added between operations, i.e. operation $o_{i,j+1}$ starts immediately at the end of operation $o_{i,j}$, for all $i \in I$ and $j \in J_{i-}$. This assumption has previously been made in some realisations, see Domschke (1989) [70], Nachtigall and Voget (1996) [133], Großmann et al. (2012) [95], Pätzold and Schöbel (2016) [138], and a group of instances in Fuchs et al. (2022) [78]. Any operation $o_{i,j}$ takes place on an identified section $s_{i,j} \in S$.

About the choice of *operation* term. In the problems studied through this thesis, we borrow the term *operation* from the scheduling community. In this sense, an analogy also exists between a *slot* and a *task*. We are aware that in the context of the *Periodic Event Scheduling Problem* and its extensions, the term *activity* has been proposed. As we do not explicitly consider passenger changeover activities between trains in this work, the term operation has seemed a natural alternative option for us to describe a train that runs on a line track, dwells at a station or passes through it.

3.1.1.3 Safety headways

On the railroad, a line section is microscopically decomposed into several block sections separated by insulated joints and protected by signals. The role of a signalling system is to give appropriate indications to the drivers depending on the location of the trains on the infrastructure in order to guarantee the safety of the operations. Various signalling

systems exist. The simplest one, called Automatic block signalling system at 3 aspects, is presented in Figure 3.2. A signal aspect is a combination of positions, colours and continuity of the lights presented in a signal. A protection signal (red colour) indicates that the protected block section is currently occupied by a train. A warning signal (yellow colour) informs the driver that the train must be stopped before the following signal, that presents a protection signal. A running signal (green colour) means that the train can run at it scheduled speed. It can be observed in Figure 3.2 that when the first train releases the insulated joint, then the former warning signal becomes a running signal. Therefore, using the Automatic block signalling system at 3 aspects, a signal presents a green aspects whenever its protected block section and the following one are not occupied by a train. A train running at scheduled speed may run several hundreds of meters before being able to stop. As soon as a driver perceives a warning signal, the train must begin breaking instantaneously until reaching a limited speed at which it will be able to stop by perceiving a protection signal. To avoid such a situation where a following train looses precious minutes by running at a limited speed during an entire section while the first train was almost releasing the insulated joint two signals away, it is crucial to schedule enough free track time before a train reaches a signal, by taking into account the visibility range of the signal and an additional time margin. Other signalling systems may present more than 3 aspects, like Automatic block signalling systems that include blinking signals. Another system, called the track-to-train transmission, is used for the high-speed lines and gives precise speed indications directly in the cabin depending on the number of released block sections ahead.

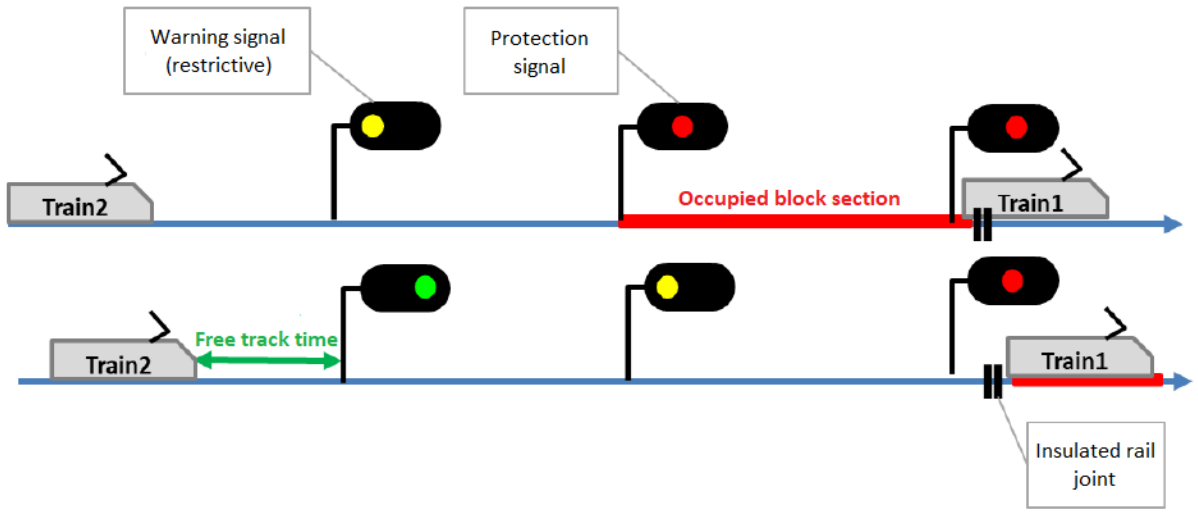


Figure 3.2: Automatic block signalling system, translated from the published AR02384 SNCF Réseau document.

At the mesoscopic scale of the infrastructure, the dynamic of the signalling system

is not explicitly taken into account. Instead, sufficient safety headways are respected between slots using shared infrastructure resource in order to satisfy all the blocking times at the microscopic scale. More specifically in this work, we consider three types of safety headways:

- **Spacing headways**: considering slots running in the same direction, a slot can be scheduled on a line track Δ_{spa} time units after a previous slot have been scheduled on the same track.
- **Reuse headways**: (i) a slot can be scheduled on a station track Δ_{reu} time units after a previous slot has liberated this track; (ii) considering two slots running in the opposite direction, the second slot can be scheduled on a line track Δ_{reu} time units after the first slot has liberated this track. This rule prevents two trains from running face to face.
- **Crossing headways**: considering slots running on incompatible routes r_1 and r_2, a slot can run on r_1 (resp. on r_2) Δ_{cro} time units after a previous slot ran on r_2 (resp. on r_1).

3.1.2 Decision Variables

Starting times. Let $t_{i,j} \in \{0, \ldots, T-1\}$ be the starting time of operation $o_{i,j}$, $i \in I$, $j \in J_i$. Since all durations $p_{i,j}$ are known, then fixing any $t_{i,j}$ enforces the starting time of all other operations of slot i.

Tracks assignment. Let $x_{i,j,k}$ be equal to 1 if operation $o_{i,j}$ is processed on track $k \in K_{i,j} = \{1, \ldots, |K_{s_{i,j}}|\}$ of section $s_{i,j}$, $i \in I$, $j \in J_i$ and be equal to 0 in the other cases. Contrary to the classical PESP model tackling the periodic TTP at a macroscopic level, the constraint network of the periodic TTP at a mesoscopic level is not fully determined but dynamically refined depending on the track choices.

3.2 Problem statement

In this section, we qualify a solution of the decision problem in Section 3.2.1 and we define the criterion to be optimised in Section 3.2.2.

3.2.1 Decision problem

We search to attribute a value for each $t_{i,j}$ and $x_{i,j,k}$, $i \in I$, $j \in J_i$, $k \in K_{i,j}$ with $s_{i,j} \in S$ such that:

- Operations are assigned on tracks with a compatible running direction.
- Given a slot, precedence constraints between its operations hold.
- Tracks assignment for any two consecutive operations of a slot verifies the existence of a corresponding mesoscopic route.
- All headways detailed in Section 3.1.1.3 are satisfied.

As commented by Pätzold and Schöbel (2016), even finding a feasible schedule at a macroscopic level with all the running and dwelling times fixed is already a challenging task.

Assumptions. It should be kept in mind that we state our problem under the following assumptions:

- The infrastructures that we study in this work are not spatially cyclic. This does not mean that, considering the stations as vertices and line sections as edges of a graph, we only limit our study to trees. The medusa network presented in Section 3.3.2 is an example of infrastructure which topology is different from a tree. What we mean, instead, is that our models do not fit with infrastructures that allow trains to flip from downstream to upstream direction (or vice versa) while conserving a strictly positive speed e.g. using a triangle structure or a loop. In other words, a train that does not stop cannot traverse a same mesoscopic track twice in the same run on our studied infrastructure topologies. The only way to run on a same track for a given train is to turnover i.e. to stop and to be driven from the opposite cabin. This assumption is a limitation when compared with the possible real-world infrastructures. Nevertheless, our models can still apply on a wide variety of infrastructure topologies including corridors and even more complex networks, as we will see in the following.
- In our decision problem, trains exit the system after accomplishing their last dwelling operation i.e. few minutes after arriving at their terminus station. This is the case in particular for the models presented in Chapter 4. The realism of the timetables is improved in Chapter 5 by reusing the linking slots at terminus stations and reusing the rolling stock.
- We limit our study to direct lines such that all operations of a given slot are performed in the same direction i.e. we can classify slots to run either downstream or upstream but not in both directions. In the French real-world instances however, there may exist lines requiring a turnover at a station that is not the origin nor the

destination station. Such special slots can however be handled by artificially linking two slots running on the opposite direction at a station.

3.2.2 Optimisation problem

A feasible schedule to our problem ensures minimising the travel time for direct travellers. We now assume in the optimisation problem addressed in Chapter 5 that a schedule is feasible if slots operating the same line in opposite directions are linked at their terminus stations, use the same tracks on these stations and share the same rolling stock. In addition, we aim at optimising the turnover times at terminus stations. It means that we want the schedule to respect as much as possible specified turnover time constraints. Moreover, when so, we want turnover times to be slightly greater than the minimal prescribed value in order to be robust to small potential delays. Typically, a terminus station $s \in S^t$ is associated with a minimal turnover time μ^s_{min}. Some of them are also characterised by a robust turnover time μ^s_{rob}, with $\mu^s_{\text{rob}} > \mu^s_{\text{min}}$ and by a maximal turnover time μ^s_{max}. We note $(i, i', s) \in L_s$ if a slots i and i' are linked at station s such that the arriving vehicle of slot i is reused at the departure of slot i' from station s. From these data, we design a personalised turnover cost function for each terminus station. More specifically, for any two slots i and i' linked at a terminus station $s \in S^t$, we penalise the turnover time $q^s_{i,i'}$ by cost function $c^s_{i,i'}(q^s_{i,i'})$ whose value depends on the following cases:

- $(\mu^s_{\text{rob}} - q^s_{i,i'}) * c_{\text{rob}}$ when $q^s_{i,i'} \geq \mu^s_{min}$ but $q^s_{i,i'} < \mu^s_{\text{rob}}$, only if μ^s_{rob} is specified for station s.
- $(\mu^s_{\text{min}} - q^s_{i,i'}) * c_{\text{bound}}$ when $q^s_{i,i'} < \mu^s_{\text{min}}$ and $q^s_{i,i'} \geq \mu_{\text{feas}}$.
- $(q^s_{i,i'} - \mu^s_{\text{max}}) * c_{\text{bound}}$ when $q^s_{i,i'} > \mu^s_{\text{max}}$, only if μ^s_{max} is specified for station s.
- c_M when $q^s_{i,i'} < \mu^s_{\text{feas}}$.

To prioritise turnover time bounds compliance while increasing robustness of the schedule when possible, we tune c_{rob} and c_{bound} such that turnover costs are minimised in a lexicographic-like fashion. To achieve this, we set $c_{\text{rob}} = 1$ and $c_{\text{bound}} > \sum_{s \in S^t}(\mu^s_{\text{rob}} - \mu^s_{\text{min}}) * |L_s|$. To ease results interpretation, we set c_{bound} to the power of 10 immediately superior to $\sum_{s \in S^t}(\mu^s_{\text{rob}} - \mu^s_{\text{min}}) * |L_s|$. It may be desired that turnover times are not too short, so that passengers can alight and board, and crews get ready for the next trip. To this end, we activate hard constraints using the minimal allowed turnover time μ_{feas} in all terminus stations. The value of μ_{feas} is set to the maximum value inferior or equal to all $\mu^s_{\text{min}}, s \in S^t$, such that the instance of the problem is feasible. Then any feasible schedule with two slots i and i' linked at a terminus station s always verifies $q^s_{i,i'} \geq \mu_{\text{feas}}$. Lastly, c_M is a huge cost that never participates to the objective function of feasible schedules. Figure 3.3 illustrates turnover cost functions in the general case.

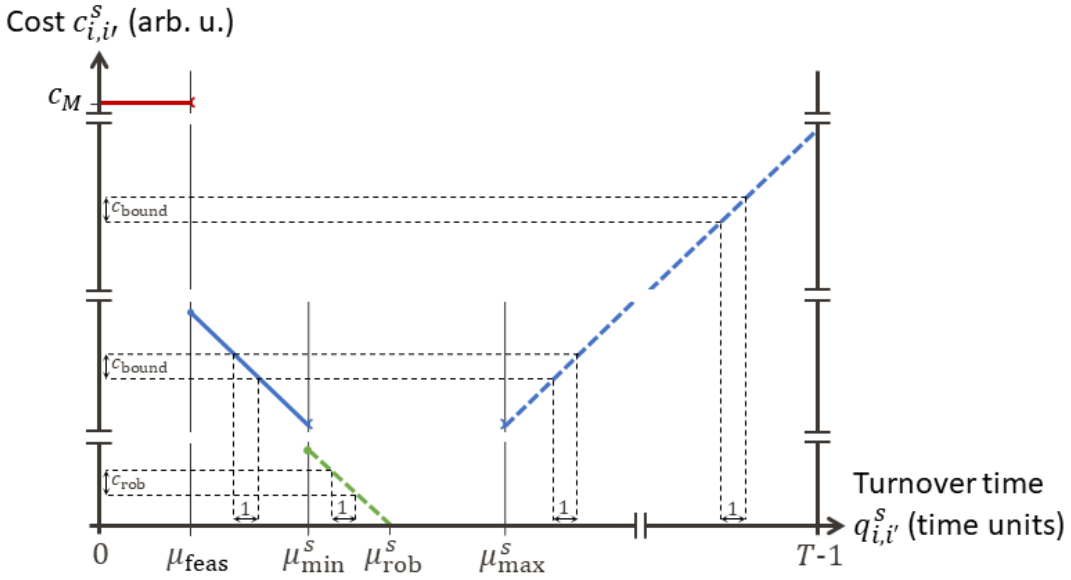


Figure 3.3: Turnover cost function of a terminus station $s \in S^t$ in the general case. In our case, $c_{\text{bound}} > c_{\text{rob}}$. Dashed lines indicate that the corresponding costs are applied only if a robust turnover time μ^s_{rob} and maximal turnover time μ^s_{max} are specified for station s. The discontinuity on the turnover time axis, that implies the second discontinuity on the cost axis, aims to reduce the graphic size since generally, $\mu^s_{\text{max}} \leq \frac{T}{2}$.

3.3 Instances

In this section, we present the instances that we consider in this work, all at the mesoscopic scale of the infrastructure. In the first place, we describe a toy example that allows to detail a Service intention and a possible representation for a solution. Then, we present a larger fictive infrastructure on which we will try to schedule an increasing number of periodic slots using CPLEX 12.10 on our Mixed-Integer Linear Program in Section 4.1.2. Third, we present instances inspired from the Savoie area around Chambéry, which are not solved in a reasonable amount of time using a MIP solver. Instead, we solve these instances heuristically in Section 4.2.3, after presenting a didactic execution of the heuristic on the toy example in Section 4.2.2.3. Finally, we also solve the Savoie instances using our Constraint Programming approach in Chapter 5, that turned out to be more efficient.

Some characteristics of our three families of instances are summarised in three tables. Table 3.1 compares the number of slots in the instances, the infrastructure shape and the quality of the available routes as introduced in sections 3.3.1 to 3.3.3. We will see that our fictive instances offer more compatible routes around stations that in the real-world Savoie area infrastructure. Table 3.2 indicates the number of time units considered per instances family. Finally, Table 3.3 provides descriptive statistics on the largest instance (with the largest number of slots, solved and unsolved where applicable) for each instance family. The number of operations per slot is indicated for these instances, as well as the number

of slots scheduled in the three most busy stations within the periodic time horizon. For example, in the largest solved instance on the medusa network infrastructure, the number of slots scheduled in the three most busy stations during one hour were 24, 16 and 14 slots respectively. The longest slots get more operations in the lager infrastructures. However, there is an heterogeneity in the Savoie area instance with roughly a half of small slots around stations at extremities of the network and another half of long slots. Table 3.4 summarises our experiments. A check mark indicates that most of the instances of the family are solved and even optimised depending on the focus. A check mark with a plus symbol determines the method that worked best for the decision problem and the optimisation problem respectively.

Table 3.1: Instance size and infrastructure configuration.

Instance	Number of slots	Topology	Routes and compatibilities
Toy example	8	star	ideal
Medusa network	10 to 40	network	ideal
Savoie area	28 and 32	star	more constrained

Table 3.2: Time horizon settings.

Instance	Period	Granularity	Number of time units
Toy example	20 min	minute	20
Medusa network	1 hour	minute	60
Savoie area	1 hour	half-minute	120

Table 3.3: Slot lengths and densities in busy stations on the largest instance for each infrastructure.

	Slot lengths (number of operations per slot)				Maximal number of slots
Instance	Minimal	Maximal	Average	Median	in the three most busy stations
Toy example	3	5	4.5	5	8, 4 and 4
Medusa network	7	17	(feasible) 9.4 (intractable) 9.7	9	(feasible) 24, 16 and 14 (intractable) 26, 16 and 16
Savoie area	2	29	10.2	6.5	18, 14 and 14

Table 3.4: Experiments conducted through this thesis. The Constraint-based solver performs better than our ILP-based approaches for the *Decision* problem, and our Branch-and-Check procedure outperforms the solver for the *Optimisation* problem.

Approach	**ILP-based** (Chapter 4)		**Constraint-based** (Chapter 5)		
Focus	*Decision*			*Optimisation*	
Instance groups	Solver	Heuristic	Solver	Solver	Branch-and-Check
Toy example		demo			
Medusa network	✓				
Savoie area	intractable	✓	✓+	✓	✓+

3.3.1 Toy example

Sections and tracks. The toy example infrastructure illustrated in Figure 3.4 is composed by 9 sections. Among these sections, 5 of them represent stations, alphabetically ordered from A to E. These stations contain 2 to 3 station tracks ordered form the top to the bottom of each station. The stations are connected by line sections that contain line tracks. In particular, the line section $A - B$ connecting station A and station B contains a single line track. Almost all the tracks of this infrastructure can be circulated in both directions, except the first station track of section B and the first line track of section $B - C$ that go downstream on the one hand, and the third station track of section B and the second line track of section $B - C$ that go upstream on the other hand.

Routes. On this infrastructure, all the possible mesoscopic routes between tracks exist. This means that, given any two tracks of distinct adjacent sections, both a downstream route and an upstream route exist between these two tracks by default, except if specified circulation directions on the tracks prevent the existence of these routes. The sets of existing routes between the tracks of sections $B - C$ and C are given in Table 3.5 as an illustrative example. In particular, only one route exists between any two tracks of these sections because of the specified direction on the tracks of section $B - C$.

Table 3.5: Routes between the tracks of sections $B - C$ and C. A right arrow ($\rightarrow$) indicates a downstream route whereas a left arrow ($\leftarrow$) stands for an upstream route.

Tracks of section $B - C$	Routes of section C	C.1	C.2	C.3
$B - C.1$		$\rightarrow$	$\rightarrow$	$\rightarrow$
$B - C.2$		$\leftarrow$	$\leftarrow$	$\leftarrow$

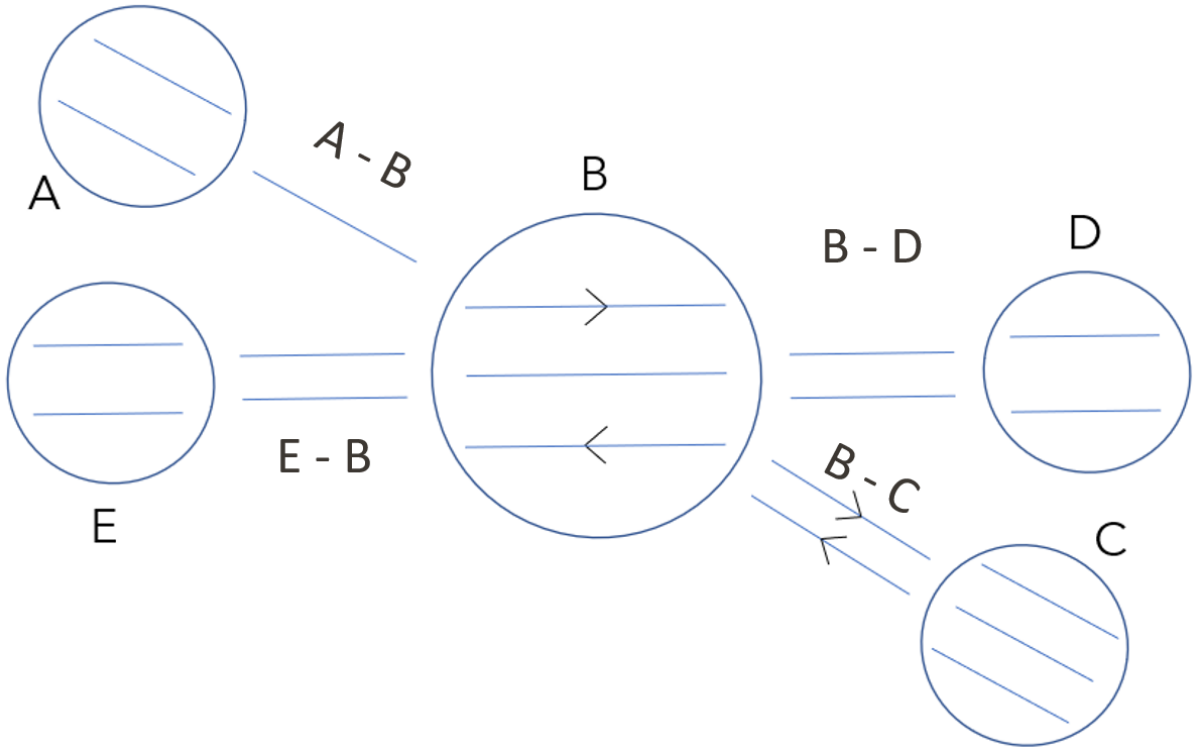


Figure 3.4: Toy example infrastructure.

Routes compatibility. Two routes are said compatibles if two trains are allowed to circulate on these routes simultaneously. Verifying the compatibility between routes is required at any point of the infrastructure where several sections connect, in order to activate crossing constraints in case two trains take incompatible routes. A route is not compatible with itself. Two distinct routes are incompatible if they connect the same extremity of the same track. Moreover, in the toy example infrastructure, there is no bridge nor tunnel, the altitude remains the same anywhere. We can thus observe that two routes are also incompatible if they cross each other when trying to draw them in Figure 3.4. In any other case for this infrastructure, we will consider two routes as compatible. Note that real-world French railway infrastructures may be more restrictive. Indeed, we saw in Figure 3.1 that the routes (1,4) and (5,2) do not cross each other, but they are incompatible because they use a common portion of microscopic track. The toy example infrastructure thus represents an idealistic case where the network is built such that non crossing routes in Figure 3.4 are compatible. Table 3.6 reports the compatibility between routes of opposite directions at the connecting point between sections $B - C$ and C. Note that we also take into account compatibilities between routes of the same direction, giving two additional compatibility matrices that are symmetric.

Service intention. We consider a period of 20 minutes. Each of the 8 slots presented in Figure 3.5 has to be scheduled within this period. In particular, a start time is needed

Table 3.6: Routes compatibility at the connecting point between sections B – C and C. A green cell indicates compatible routes whereas a red cell stands for incompatible routes.

Routes	upstream	B – C.2 ← C.1	B – C.2 ← C.2	B – C.2 ← C.3
downstream	Compatibility			
B – C.1 → C.1				
B – C.1 → C.2				
B – C.1 → C.3				

for their first operation, and a track must be assigned for each of their operations on the sections of the network. The slots are divided in two types. On the one hand, slots s4 and s5 ensure a fast service between stations D and E, without stopping at station B. Therefore, we take the rough hypothesis that the trains using these slots will spend approximately 0 minutes to pass through station B. Slots s4 and s5 provide a running time of 3 minutes on the line sections E – B and B – D. On the other hand, all the other slots make a one minute stop at station B, and provide a running time of 5 minutes on any line sections they traverse. Table 3.7 summarises these data. Finally, all slots spend 1 minute at their origin station before departing and 1 minute at their terminus station after arriving.

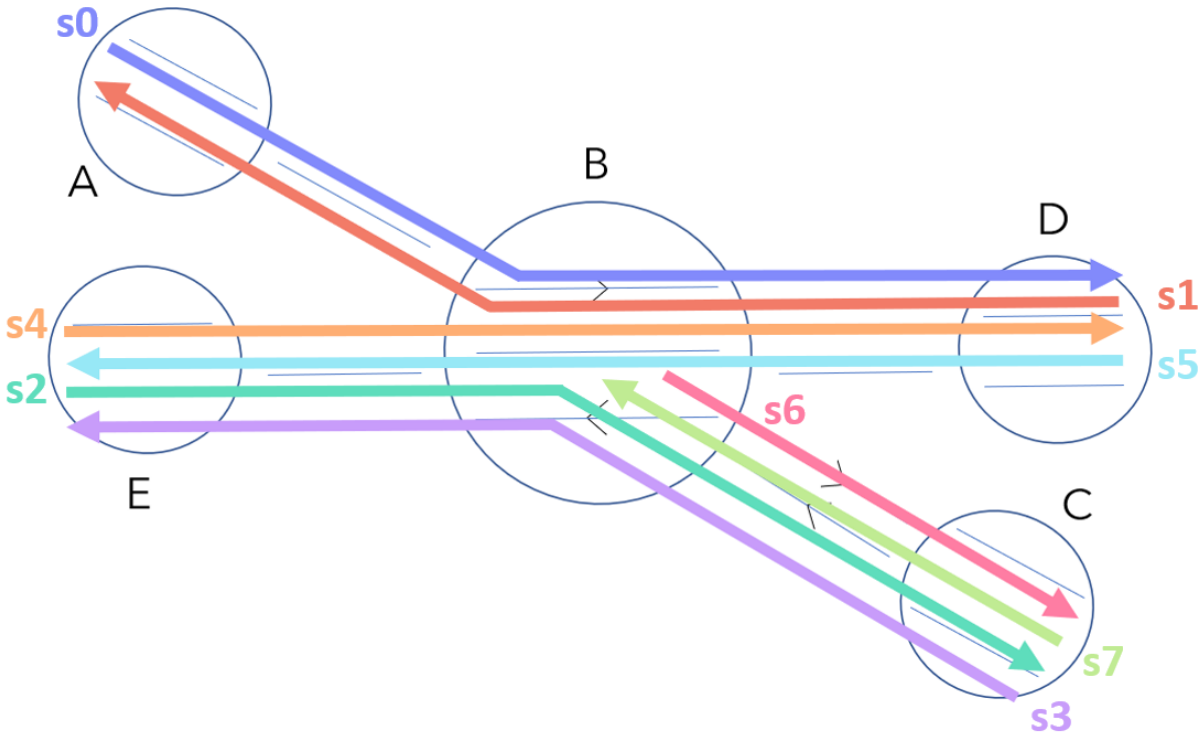


Figure 3.5: Slots to be scheduled every 20 minutes on the toy example infrastructure.

Table 3.7: Parameters of the toy example instance.

Slots	Dwell one minute at station B	Running time on line sections
$\{s4, s5\}$	no	3 min
$\{s0, s1, s2, s3, s6, s7\}$	yes	5 min

A possible solution representation. We propose to represent a feasible solution for the toy example instance in Figure 3.6 similarly to a netgraph[1] style enriched with mesoscopic tracks assignment. Note that in the original netgraph[1], stations are displayed macroscopically. Moreover, arrival and departure times are displayed near corresponding stations borders, either inside for passing through events or outside otherwise. In Figure 3.6, however, we adopt a slightly different convention. The coloured tracks correspond to the assigned track of any train running, dwelling or passing through operation. Then, the minute displayed at the beginning of the track (depending on the slot direction) represents the starting time of the operation. When the starting time of any operation of a slot is decided, then the starting time of its other operations is deducted from the hypothesis taken in the previous paragraph and in Table 3.7. For example, observe how slots s4 and s5 spend no time at station B or how slot s6 runs 5 minutes on line section $B - C$. It could be misinterpreted that slot operations do not respect precedence constraints, e.g. the second operation of slot s0 starts at minute 15 while its third operation starts at minute 0. This phenomenon is due to cyclicity of the schedule time horizon. Recall that in this instance, each operation occurs every 20 minutes. Therefore the displayed minute is always the principal one i.e. between 0 and 19 included. We set a headway time of 4 minutes for all the infrastructure constraints, namely the spacing, the track reuse and the crossing constraints. It can be observed in Figure 3.6 that all these constraints are satisfied. For example, the single track in the line section $A - B$ is alternatively released by slots s0 and s1 at least 4 minutes before the other one runs occupies the track again. The solution is illustrated in 4 duplicates of the infrastructure, sorted by slots sharing the same terminal stations. This representation simplifies the visualisation of the timetable, but the infrastructure constraints apply on every pair of two slots, in particular at station B. Finally, slots sharing two terminus stations are not necessarily scheduled on the same track at their terminus stations in the context of our decision problem, where the trains leave the system after their last arrival. A more realistic case where the rolling stock is reused and the slots are linked at terminus stations is presented in Chapter 5.

[1] `https://sma-partner.com/en/software/timetable-planning-with-viriato/functionality` (Accessed: 30 June 2023).

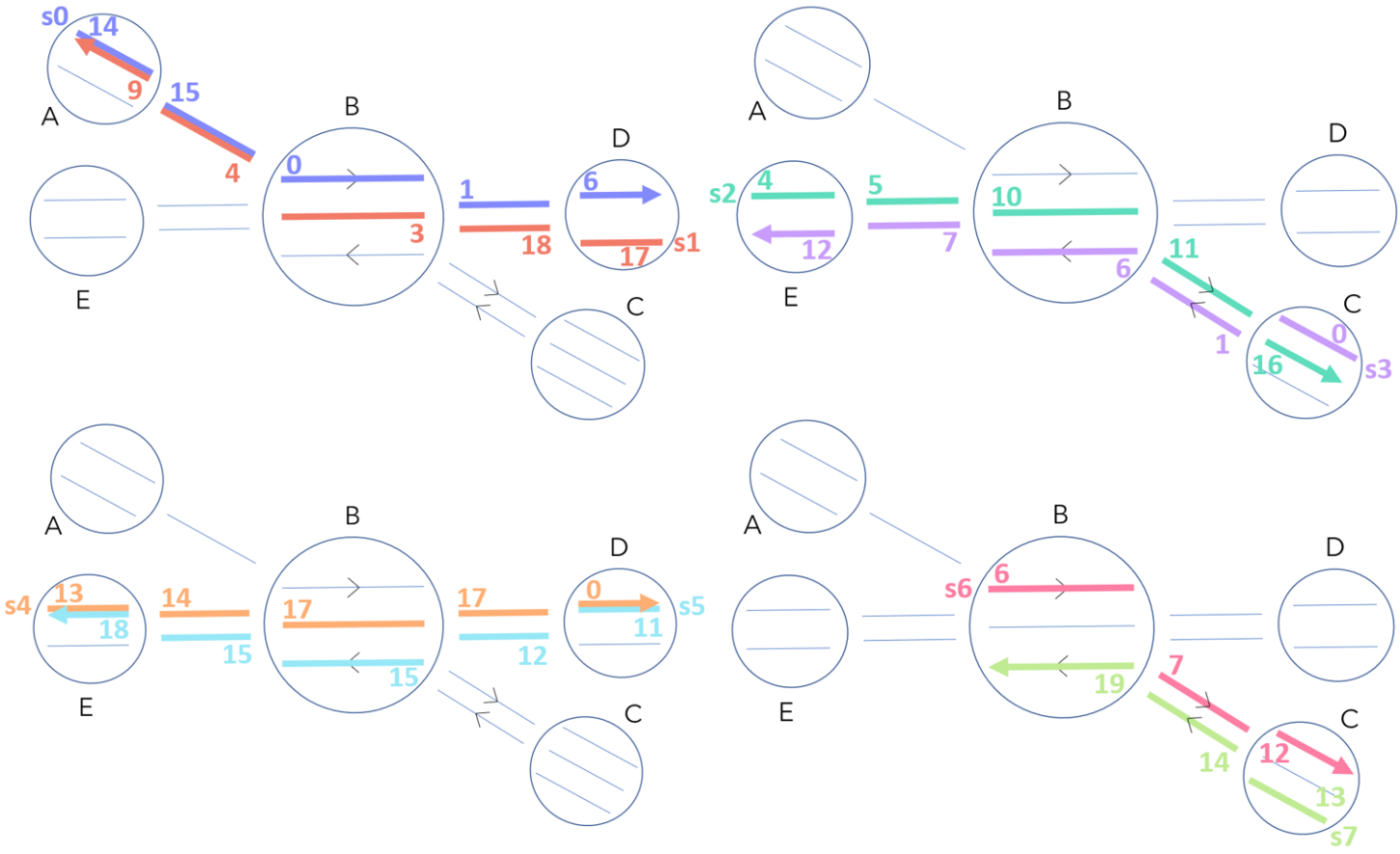


Figure 3.6: A netgraph like representation of a feasible solution for the toy example instance.

3.3.2 Medusa network

Representation. The medusa network illustrated in Figure 3.7 is larger than the toy example infrastructure and follows generally the same rules regarding the existence of mesoscopic routes and their compatibilities. Moreover, we keep the hypotheses that all the dwelling times span one minute, that trains spend approximately 0 minutes in stations where they do not stop, and that fast services run 3 minutes on line sections whereas omnibus trains run 5 minutes on line sections. Yet, four slight differences with the conventions of the previous instance are now introduced. First, small green dots on the infrastructure indicate limitations on the routes existence. More precisely, the routes of tracks connected by a small green dot uniquely pass through this dot. For example, a downstream route and an upstream route exist between the first track of station n and the first track of line section n − O, and the same apply for the second track of station n and the second track of line section n − O. However, there is no route between the first track of station n and the second track of line section n − O, nor between the second track of station n and the first track of line section n − O. Second, stations are classified among two levels of importance, indicated by the lower or upper case of their label. Anecdotally, further indications allow to distinguish between stations I and l in Figure 3.7. The two last

differences relate to the representation of the slots to be scheduled in Figure 3.8. Third, on coloured line now represent 2 distinct slots that share the same stops and terminus stations. Fourth, contrary to the display of the toy example instance, the slot types are now explicitly indicated. All the slots have several stops. The fast slots have large dots at the stations where they stop, and also stop at their terminus station. The remaining slots are omnibus and thus stop at every station they traverse.

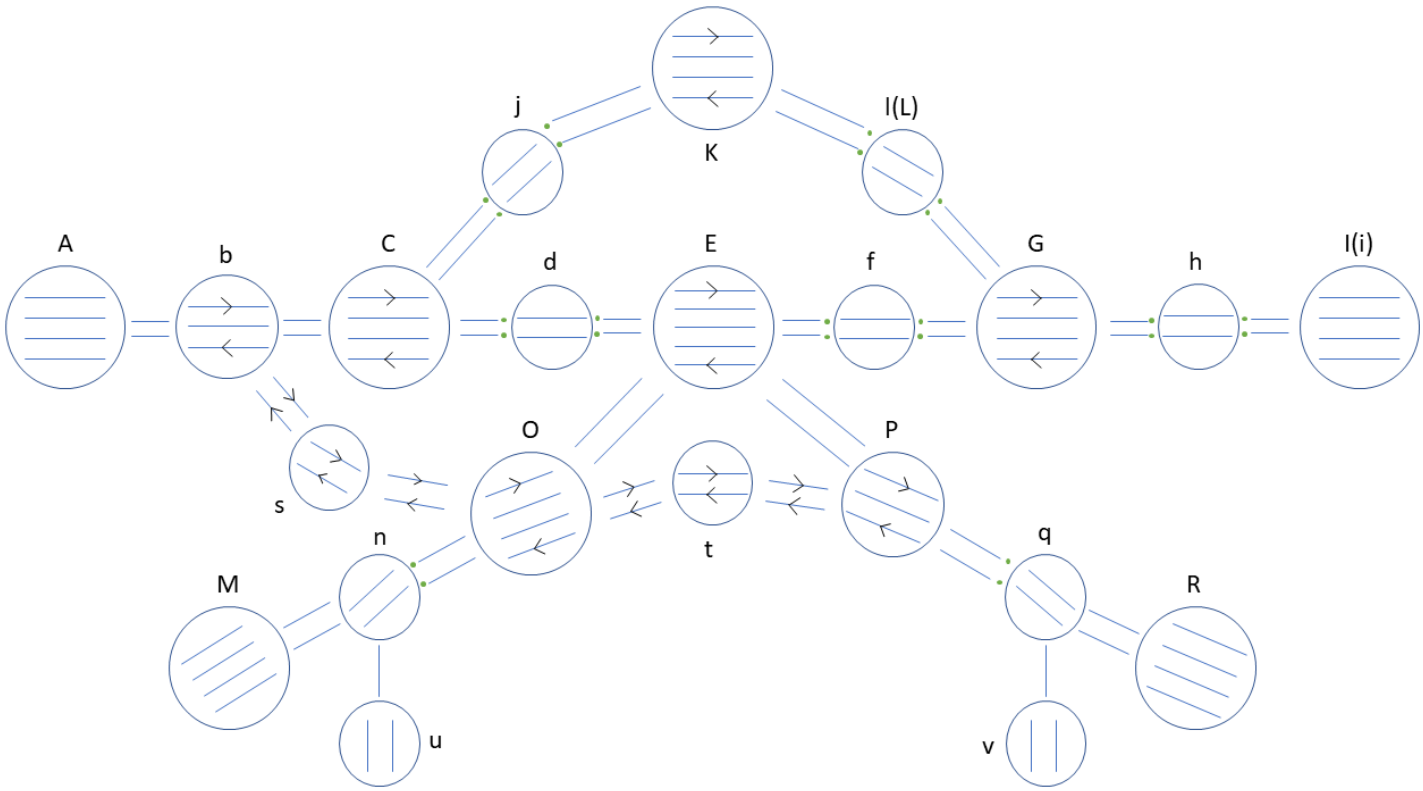


Figure 3.7: Medusa network topology.

Service intentions. We set 11 instances of the decision problem on the medusa network, in which the number of periodic slots to be scheduled within a period of 60 minutes increases gradually. The first three instances are illustrated in Figure 3.8. A particularity of the first instance is that slots running in the same direction only meet around stations and not on line sections. Therefore, spacing constraints that prevent two trains running in the same direction from caching up each other on line tracks are not activated. On the other hand, track reuse constraints at stations and in line tracks when the trains run in opposite direction, as well as crossing constraints between sections, apply and guarantee that a found solution is conflict-free. In the second instance, the omnibus slots between stations C and G passing by K are replaced by omnibus slots between stations A and K and between K and I respectively. Moreover, 8 additional omnibus slots converge at station E: between stations A and E, u and E, E and I and between stations E and v,

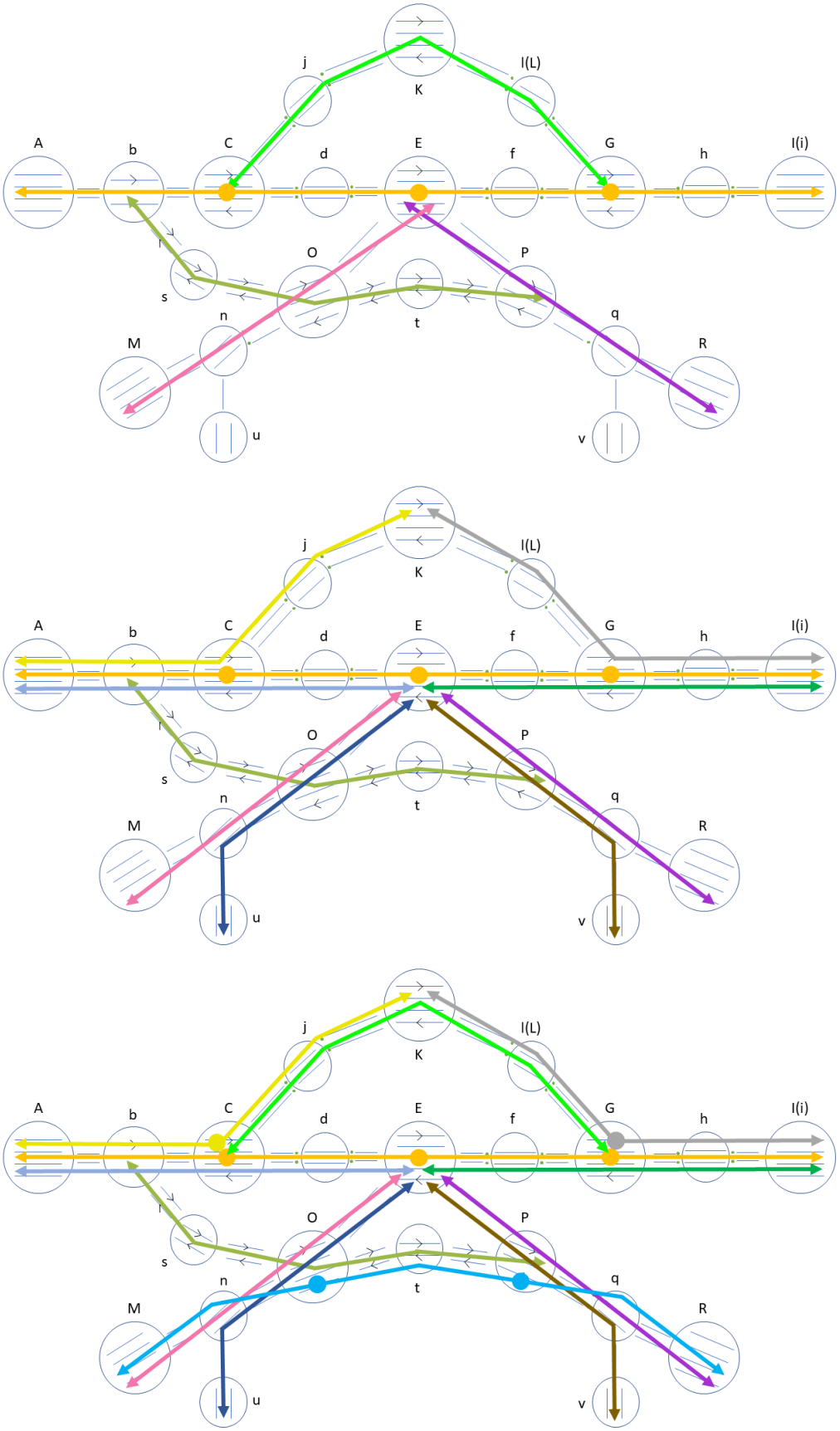


Figure 3.8: First three instances on the medusa network, with 10, 20 and 24 hourly periodic fast and omnibus slots.

respectively. In the third instance, the slots between stations C and G passing by K are restored and fast slots between stations M and R dwelling at stations O and P are also included. Instances 4 to 11 are incremented in the same spirit and are specified in Table 3.8. For example, instances 4 and 5 are both built using the slots of the third instance as a basis. Two slots, one in the downstream direction and the other in the upstream direction, with terminus stations C and G and passing by E, are included. For instance 4, these slots are omnibus whereas for instance 5, they correspond to fast services with a stop at station E.

Table 3.8: Characteristics of instances 4 to 11 on the medusa network.

			Additional slots		
Instance	Number of slots	Base instance	Terminus stations	Omnibus	Stops if fast services
4	26	3	C and G by E	✓	
5	26	3	C and G by E		E
6	28	3	C and G by E	✓	
			C and G by E		E
7	30	6	C and G by K		K
8	32	7	A and R by t		O and P
9	36	8	M and E		O
			E and R		P
10	38	9	C and R		E and P
11	40	10	M and I		O, E and G

3.3.3 Savoie area around Chambéry station

Instance description. In our third family of instances, we consider the Savoie infrastructure area in star configuration around Chambéry station (section 15) illustrated in Figure 3.9. This infrastructure includes double track line sections between Aix-les-Bains (section 45) and Modane (section 34) stations (D), and single track line sections between Saint-André-le-Gaz and Chambéry stations (S1), between Saint-Pierre-d'Albigny (section 19) and Albertville (section 41) stations (S2), and between Annecy (section 55) and Aix-les-Bains stations (S3). Table 3.9 describes the Service intentions of two instances of the problem. Some of the bidirectional hourly lines are planned on a large proportions of one or several groups of tracks of the infrastructure. The remaining slots are scheduled at extremities of the infrastructure and arrive from and go to other destinations. The latter

slots, even with few operations, are important because their turnover time participate in the objective function, because they occupy station tracks, and because they activate crossing constraints with the former slots. Instance 1 contains 28 slots per hour all having at least one terminus station in the Savoie infrastructure. The line plan of Instance 2 has 4 additional slots per hour, i.e. one Intercity and one Freight lines served in both directions for which origins and destinations are outside of the infrastructure of interest. Other approaches in the literature solve instances with larger networks and more slots per hour, see De Fabris et al. (2014) [62] at a mesoscopic level and Kümmling et al. (2015) [109] at a macroscopic level. Still, our instances are challenging in finding feasible solutions because of slots partially scheduled in single tracks, and because no additional time is allowed between operations in our presented approaches. The performance obtained when solving Instances 1 and 2 in Chapter 5 are encouraging to challenge our method with additional instances. Table 3.10 provides minimal, robust and maximal prescribed turnover times at terminus stations of the infrastructure. These reference times are used to compute the objective function depending on the train timetable, and more precisely on the turnover times. Table 3.11 specifies values used for costs and safety headways.

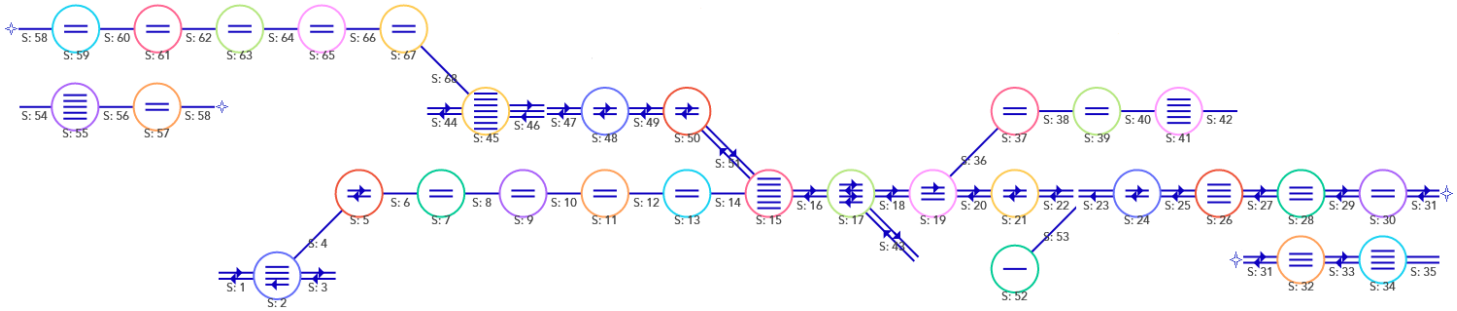


Figure 3.9: Savoie infrastructure area at a mesoscopic scale in star configuration around Chambéry station (section 15). Circled sections are stations and halts while the remainder are line sections. Line sections 31 and 58 are represented twice.

Further considerations. Contrary to the case explained in Section 3.3.1 valid for the toy example infrastructure and for the medusa network, the existence of mesoscopic routes is not as systematic in the real-world Savoie area infrastructure. As an example, the last track of Chambéry station (section 15) is not accessible to or from the southern end of the station, corresponding to the right and side of station 15 in Figure 3.9. The same holds regarding the compatibility between mesoscopic routes: they are much more permissive in the toy example infrastructure and in the medusa network than in the Savoie area infrastructure. This is because the 2 fictive infrastructure represent ideal topologies that are not realistic since they should take too much space and being too costly if implemented in reality. In the end, the multiple use of the same portion of track around stations make routes be easily incompatible in our real-world Savoie area infrastructure. The detail

Table 3.9: Bidirectional lines to be hourly timetabled on the Savoie area infrastructure. Slot types are Regional (R), Intercity (IC) and Freight (F). The infrastructure is displayed by increasing order of sections id whenever at least one line is planned on it. Sections of group D contain stations and double track line sections between Aix-les-Bains and Modane. Sections of group S1 contain stations and single track line sections between Saint-André-le-Gaz and Chambéry. Those of group S2 contain stations and single track line sections between Saint-Pierre-d'Albigny and Albertville. Finally, sections of group S3 contain stations and single track line sections between Annecy and Aix-les-Bains. Zones are displayed at the junction of several groups of sections. Squares represent terminus stations. Dots represent other dwelling operations at stations. Where applicable, arrow dots specify the direction in which a stop is planned. Vertical bold lines are running operations in line sections or passing through operations at stations.

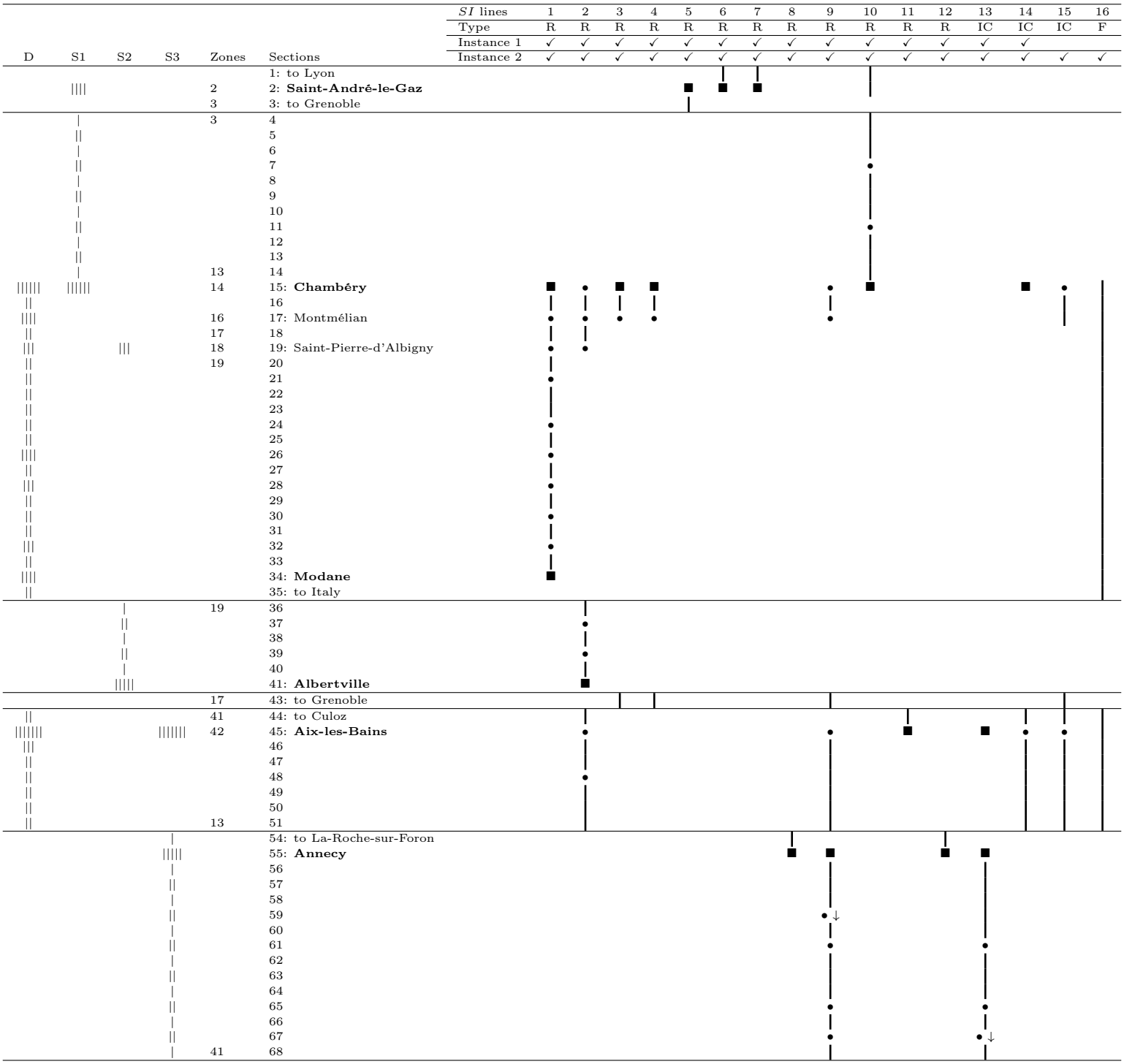


Table 3.10: Prescribed turnover times at terminus stations.

s	$\mu^s_{\min}$ (time units)	μ^s_{rob} (time units)	$\mu^s_{\max}$ (time units)
Chambéry (section 15)	16	20	24
Albertville (section 41) and Annecy (section 55)	16	20	40
Aix-les-Bains (section 45)	24	30	
Saint-André-le-Gaz (section 2) and Modane (section 34)	16		

Table 3.11: Values of cost and safety headway parameters.

c_{rob}	1	arb. u.
c_{bound}	100	arb. u.
c_M	10^5	arb. u.
$\Delta_{\text{spa}} = \Delta_{\text{reu}} = \Delta_{\text{cro}} = \mu_{\text{feas}}$	8	time units

of the existing routes and their compatibilities is not provided in this thesis, however it is accessible to the timetable planners who navigate in the OptiSillons tool presented in Chapter 6. About the Service intentions, we can qualify them to be close-to-real-world. In fact, the 2 instances presented in our real-world infrastructure uniquely include hourly served lines including both downstream and upstream slots. However, in real-world settings, there are some lines with a two-hour frequency and for which the downstream slot runs on a given single track line during odd hours whereas the upstream slot runs on the same single track line during even hours. Our models only apply on a unique parameterised period and thus we do not consider two-hourly railway services on single track lines in our instances. Of course, we could set our period to 2 hours and double the model size by including two slots when they are hourly served. However, there is certainly a clever way to extend our models by instead considering two distinct but almost identical hourly served timetables where the first one contains one two-hourly served slot and the second timetable contains the remaining two-hourly served slot that runs in the opposite direction.

3.A List of Symbols

The Table 3.12 summarises symbols and notations used in this chapter.

Table 3.12: Signification of symbols and notations used in Chapter 3.

Symbol	Meaning
T	Time period
SI	Service Intention, set of bidirectional lines to be timetabled within period T
$S = S^l \cup S^t$	Set of sections, partitioned in the set of line sections and the set of stations
K_s	Set of mesoscopic tracks of section s
$D_k \subseteq \{+, -\}, \mathrm{d}()$	Set of possible running directions on track k and running direction of a slot operand, respectively
$R = R^+ \cup R^-$	Set of mesoscopic routes, partitioned by direction
Z	Set of zones, a partition of S
$< z_1, z_2 >$	A bizone containing mesoscopic routes in any direction between z_1 and z_2
I, J_i and J_{i-}	Set of slots, set of operations of slot i and set of operations of slot i except the last one, respectively
$o_{i,j}, p_{i,j}$, $s_{i,j}$ and $t_{i,j}$	Operation j of slot i, its known duration, the section on which it is processed and its starting time, respectively
$\lvert\ \rvert$	Cardinal of a set operand inside the symbol
$K_{i,j} = \{1, \ldots, \lvert K_{s_{i,j}} \rvert\}$	Set of alternative tracks for operation $o_{i,j}$
$x_{i,j,k}$	Assignment of track k for operation $o_{i,j}$
$\Delta_{\mathrm{spa}}, \Delta_{\mathrm{reu}}$ and Δ_{cro}	Spacing, reuse and crossing safety headways, respectively
$\mu_{\mathrm{min}^s}, \mu_{\mathrm{rob}^s}$ and μ_{max^s}	Minimal, robust and maximal prescribed turnover times at terminus station s, respectively
μ_{feas}	Unforced minimal turnover time at terminus stations
L_s	Set of linked slots sharing the same track and using the same vehicle at station s
$c_{\mathrm{rob}}, c_{\mathrm{bound}}$ and c_M	Costs penalising turnover times at terminus stations that does not respect the prescriptions
$q^s_{i,i'}$	Turnover time of slots i and i' at terminus station s
$c^s_{i,i'}()$	Cost function depending on the turnover time $q^s_{i,i'}$ of linked slots i and i' at station s

Chapter 4

Integer Linear Programming approaches

In this chapter, we present approaches to solve the decision problem stated in Section 3.2.1 based on Integer Linear Programming. We formulate a model in Section 4.1 and we solve it on the medusa network instances with a commercial solver. To improve the tractability of the method, we propose a model decomposition in Section 4.2 such that the timetable is decided before the tracks assignment. We conceived an objective function to evaluate conflicts in a timetable without explicit tracks assignment knowledge, and we developed an adaptive heuristic based on tabu search to address the problem.

Contents

4.1 Modelling the problem as an Integer Linear Program

In this section, we model our decision problem as an Integer Linear Program. Among the works that inspired our formulation, we can mention the *Periodic Event Scheduling Problem* of Serafini and Ukovich (1989) [174], its track-choice extension by Wüst et al. (2019) [182] and the use of boolean periodicity variables like in Yan and Goverde (2019) [185]. We considered the possibility to investigate a flow formulation, however it seemed more appropriate to adopt a model closer to job-shop problems because of the specificities of the safety headway constraints arising from the infrastructure usage. As we saw in Chapter 2, flow formulations are more suited for Network Design, Line Planning and Passenger Routing problems. Our model is given is Section 4.1.1 and solved using CPLEX 12.10 on the medusa network instances in Section 4.1.2.

4.1.1 Model formulation

We address a decision problem, for which we want to determine if a feasible timetable exists or not from a set of slots to be scheduled. Therefore, the formulation presented in this section does not include an objective function. If we consider passenger origin-destination data, our model can easily be extended with an objective function to minimise the waiting time of passengers during changeovers, as it is proposed in Nachtigall and Voget (1996) [133]. Note that as we enforce the travelling time to be minimal, solving the decision problem already allows to minimise the travel time for all direct travellers.

4.1.1.1 Parameters

The input data of the decision problem, corresponding to some of the notations summarised in Sub-appendix 3.A, are the following. The Service intention SI contains n slots to be scheduled on a railway infrastructure described at the mesoscopic scale. Each slot i is composed by n_i operations that consist either in running on a line section, dwelling or passing through a station. Each operation $o_{i,j}$ is scheduled in the section $s_{i,j}$ of the infrastructure and has a known duration $p_{i,j}$. The remainder of the parameters includes the safety headways constants Δ_{spa}, Δ_{reu} and Δ_{cro}. The first is used in spacing constraints to regulate the access to a section line track between two following trains. The second is used in track reuse constraints at stations and to prevent the simultaneous access of two trains in opposite direction on the same section line track. Finally, the third parameter is used in crossing constraints to prevent two trains to run through incompatible routes too close in time.

4.1.1.2 Variables

Every operation $o_{i,j}$ is associated with an integer variable $t_{i,j}$ representing its start time in the range $[0, T-1]$. Let $t_{i,j}$ and $t_{i',j'}$ be the start time of $o_{i,j}$ and $o_{i',j'}$, we introduce the boolean variable $z_{i,i',j,j'}$ that equals 0 if $t_{i,j} \leq t_{i',j'}$ and 1 otherwise. To obtain this behaviour, we make the two following inequalities always hold for any two operations of distinct slots (this is a disjunctive constraint):

$$t_{i,j} \leq t_{i',j'} + z_{i,i',j,j'} * T$$

$$t_{i',j'} + 1 \leq t_{i,j} + (1 - z_{i,i',j,j'}) * T$$

Each operation $o_{i,j}$ occurs in a determined section $s_{i,j}$ containing $K_{i,j}$ tracks. There is one boolean variable $x_{i,j,k}$ per track k in the section $s_{i,j}$ on which $o_{i,j}$ will potentially be assigned. We have $x_{i,j,k} = 1$ if $o_{i,j}$ is assigned on track k and $x_{i,j,k} = 0$ otherwise. We introduce the auxiliary boolean variables $y_{s,i,i',k}$ to indicate the simultaneous assignment of operations $o_{i,j}$ and $o_{i',j'}$ occurring in section $s = s_{i,j} = s_{i',j'}$ on a same track $k \in K_s$. To do so, the following equations hold for each track k of a shared section s among two slots i and i':

$$y_{s,i,i',k} \leq x_{i,j,k}$$

$$y_{s,i,i',k} \leq x_{i',j',k}$$

$$y_{s,i,i',k} \geq x_{i,j,k} + x_{i',j',k} - 1$$

The first two above equations enforce $y_{s,i,i',k}$ to be equal to 0 whenever $x_{i,j,k}$ or $x_{i',j',k}$ equal 0. The third above equation enforces $y_{s,i,i',k}$ to be equal to 1 if $x_{i,j,k}$ and $x_{i',j',k}$ equal 1. These three equations thus implement a logical AND between $x_{i,j,k}$ and $x_{i',j',k}$ which result corresponds to $y_{s,i,i',k}$. Now, consider any two incompatible routes (k_1,k_2) and (k_3,k_4). These routes are either in the same direction or in the opposite direction. We introduce the auxiliary boolean variables $y^{cro}_{i,i',j,j',k_{1..4}}$ to detect the routes incompatibility, i.e. if $o_{i,j}$ is assigned to track k_1, $o_{i,j+1}$ to track k_2, $o_{i',j'}$ to track k_3 and $o_{i',j'+1}$ to track k_4. In this case, $y^{cro}_{i,i',j,j',k_{1..4}}$ equals 1, and equals 0 otherwise. The following equation enforces $y^{cro}_{i,i',j,j',k_{1..4}}$ to be equal to 1 if x_{i,j,k_1}, $x_{i,j+1,k_2}$, x_{i',j',k_3} and $x_{i',j'+1,k_4}$ are all equal to 1:

$$y^{cro}_{i,i',j,j',k_{1..4}} \geq x_{i,j,k_1} + x_{i,j+1,k_2} + x_{i',j',k_3} + x_{i',j'+1,k_4} - 3$$

Note that if at least two tracks among $\{k_1, k_2, k_3, k_4\}$ are the same one, thus the routes incompatibility can even be detected if only the two responsible track assignment variables equal 1. So, if slots i and i' run on the same direction, the following equations independently hold if $k_1 = k_3$ or $k_2 = k_4$, respectively:

$$y^{\text{cro}}_{i,i',j,j',k_{1..4}} \geq x_{i,j,k_1} + x_{i',j',k_3} - 1$$

$$y^{\text{cro}}_{i,i',j,j',k_{1..4}} \geq x_{i,j+1,k_2} + x_{i',j'+1,k_4} - 1$$

Symmetrically, if slots i and i' run on the opposite direction, the following equations independently hold if $k_1 = k_4$ or $k_2 = k_3$, respectively:

$$y^{\text{cro}}_{i,i',j,j',k_{1..4}} \geq x_{i,j,k_1} + x_{i',j'+1,k_4} - 1$$

$$y^{\text{cro}}_{i,i',j,j',k_{1..4}} \geq x_{i,j+1,k_2} + x_{i',j',k_3} - 1$$

Contrary to auxiliary variables $y_{s,i,i',k}$ for the detection of 2 assignment variables on the same track, auxiliary variables $y^{\text{cro}}_{i,i',j,j',k_{1..4}}$ do not exactly represent a logical AND between 4 assignment variables. Therefore, we do not impose $y^{\text{cro}}_{i,i',j,j',k_{1..4}}$ to be equal to 0 whenever one of the 4 corresponding assignment variables is equal to 0 because it can be contradictory with one of the four last above equations and thus wrongly leads to infeasibility. In the following, M refers to an arbitrarily large value (Big-M).

4.1.1.3 Constraints

Now that the variables of the model have been presented, the decision problem formulation reads:

$$t_{i,j+1} = t_{i,j} + p_{i,j} - z_{i,i,j,j+1} * T, \forall i \in I, \forall j \in J_{i-} \tag{4.1a}$$

$$\begin{cases} t_{i',j'} - t_{i,j} + z_{i,i',j,j'} * T + (1 - y_{s,i,i',k}) * M \geq \Delta_{\text{spa}}, & (4.1b) \\ t_{i',j'} - t_{i,j} + z_{i,i',j,j'} * T \leq T - \Delta_{\text{spa}} + (1 - y_{s,i,i',k}) * M, & (4.1c) \\ \forall i \neq i' \in I, \text{d}(i) = \text{d}(i'), \forall j \in J_i, \forall j' \in J_{i'}, s_{i,j} = s_{i',j'}, s_{i,j} \in S^l, \forall k \in K_{i,j} \end{cases}$$

$$\begin{cases} t_{i',j'} - t_{i,j} + z_{i,i',j,j'} * T + (1 - y_{s,i,i',k}) * M \geq p_{i,j} + \Delta_{\text{reu}}, & (4.1d) \\ t_{i',j'} - t_{i,j} + z_{i,i',j,j'} * T \leq T - (p_{i',j'} + \Delta_{\text{reu}}) + (1 - y_{s,i,i',k}) * M, & (4.1e) \\ \forall i \neq i' \in I, \forall j \in J_i, \forall j' \in J_{i'}, s_{i,j} = s_{i',j'}, s_{i,j} \in S^t \;||\; \text{d}(i) \neq \text{d}(i'), \forall k \in K_{i,j} \end{cases}$$

$$\begin{cases} t_{i',j'} - t_{i,j} + z_{i,i',j,j'} * T + (1 - y^{\text{cro}}_{i,i',j,j',k_{1..4}}) * M \geq \Delta_{\text{cro}}, & (4.1f) \\ t_{i',j'} - t_{i,j} + z_{i,i',j,j'} * T \leq T - \Delta_{\text{cro}} + (1 - y^{\text{cro}}_{i,i',j,j',k_{1..4}}) * M, & (4.1g) \\ \forall i \neq i' \in I, \forall j \in J_{i-}, \forall j' \in J_{i'-1}, \forall k_1 \in K_{i,j}, \forall k_2 \in K_{i,j+1}, \forall k_3 \in K_{i',j'}, \forall k_4 \in K_{i',j'+1}, \\ r_1 = (k_1, k_2), r_2 = (k_3, k_4) \in R, r_1 \text{ and } r_2 \text{ not compatible} \end{cases}$$

$$x_{i,j,k} = 0, \forall i \in I, \forall j \in J_i, \forall k \in K_{i,j}, \mathrm{d}(i) \notin D_k \tag{4.1h}$$

$$\sum_{k \in K_{i,j}} x_{i,j,k} = 1, \forall i \in I, \forall j \in J_i \tag{4.1i}$$

$$x_{i,j,k} \leq \sum_{\substack{k' \in K_{i,j+1} \\ (k,k') \in R}} x_{i,j+1,k'}, \forall i \in I, \forall j \in J_{i-}, \forall k \in K_{i,j} \tag{4.1j}$$

Equations (4.1a) are the precedence constraints for the operations of each slot. Variables $z_{i,i,j,j+1}$ equal 1 whenever $t_{i,j} + p_{i,j} \geq T$.

Constraints (4.1b) to (4.1g) ensure the safety headways between the start times of slot operations. More precisely, these constraints are to be considered pairwise, i.e. constraints (4.1b) and (4.1c) work for the spacing headways, constraints (4.1d) and (4.1e) work for the track reuse headways, and constraints (4.1f) and (4.1g) work for the crossing headways. All these constraints apply on the periodic time difference between $t_{i',j'}$ and $t_{i,j}$, that has to respect a lower bound in the case of constraints (4.1b), (4.1d) and (4.1f), and an upper bound in the case of constraints (4.1c), (4.1e) and (4.1g). Indeed, as the time horizon is cyclic, then not satisfying a lower bound or an upper bounds of these constraints both means that operations start too close in time and create a conflict. The spacing headway constraints (4.1b) and (4.1c) allow a following train to access a line track at least Δ_{spa} time units after the previous train entered the track. The track reuse constraints (4.1d) and (4.1e) allow a new train to reuse a track at least Δ_{reu} time units after a first train has liberated the track. These constraints apply in stations and also for slots going in opposite direction on line section tracks. Remark how in the spacing and in the track reuse headway constraints (4.1b) to (4.1e), the Big-M like form using $y_{s,i,i',k}$ variables allow to inhibit the constraints while the operations are not assigned to the same track. Finally, the crossing headway constraints allow the train to take a route that is not compatible with a route taken by a previous train after at least Δ_{cro} time units. If incompatible routes are assigned to slot operations, then the variables $y^{\mathrm{cro}}_{i,i',j,j',k_{1..4}}$ activate these constraints. However, we saw above that in case of compatible routes, we did not explicitly model an enforcement for variables $y^{\mathrm{cro}}_{i,i',j,j',k_{1..4}}$ to be equal to 0. The model could thus be refined in order to avoid branching errors on these variables that could imply backtracks and more computation time in case the constraints are not appropriately activated for two slots taking compatible routes simultaneously.

Constraints (4.1h) prevent operations to be assigned to tracks with an incompatible running direction. Constraints (4.1i) require exactly one track assigned by operation. Finally, constraints (4.1j) guarantee the coherence of the tracks assignment i.e. the assigned tracks have to correspond to existing mesoscopic routes of the infrastructure. The operations successors are used for this purpose. Symmetric constraints could also use the operations predecessors for additional enforcement.

Furthermore, some valid inequalities are added to the model:

$$\begin{cases} t_{i,j} + p_{i,j} \leq t_{i,j+1} + z_{i,i,j,j+1} * M, & \text{(4.1k)} \\ t_{i,j+1} \leq t_{i,j} + (1 - z_{i,i,j,j+1}) * M, & \text{(4.1l)} \\ \forall i \in I, \forall j \in J_{i-} \end{cases}$$

$$\sum_{j \in J_{i-}} z_{i,i,j,j+1} \leq \left\lceil \sum_{j \in J_i} p_{i,j}/T \right\rceil, \forall i \in I \quad \text{(4.1m)}$$

Constraints (4.1k) to (4.1m) strengthen equations (4.1a). Indeed, depending on the value of variables $z_{i,i,j,j+1}$, variables $t_{i,j+1}$ are enforced to be either greater than $t_{i,j} + p_{i,j}$ or less than $t_{i,j}$ in constraints (4.1k) and (4.1l). On the other hand, constraints (4.1m) control the number of variables $z_{i,i,j,j+1}$ that can equal 1 for a given slot depending on the total duration of the slots, as presented in Table 4.1. Lower bounding the left-hand side in constraints (4.1m) is also possible but less immediate because it depends on the distribution of the durations of the slot operations and not only on the total of operations durations.

Table 4.1: Right-hand side of constraints (4.1m) depending on the slot duration.

$\sum_{j \in J_i} p_{i,j}$	$\left\lceil \sum_{j \in J_i} p_{i,j}/T \right\rceil$
0	0
$[1, T]$	1
$[T+1, 2T]$	2
$\vdots$	$\vdots$

Assuming that T is strictly greater than any operation duration plus any headway duration, we tried T as a candidate for the M value in constraints (4.1b) to (4.1g), (4.1k) and (4.1l). It turned out that this value was not sufficient in practice, so we instead

applied a sufficient arbitrary large value for M.

4.1.2 Experimental results

We implement Formulation (4.1) in C++ using the IBM Concert Technology. The model is solved using CPLEX 12.10 with a computer running on Windows 10, 64 bits with a i5-8265U CPU at 1.60-1.80 GHz and 24 Go of RAM. We populated our set of instances on the medusa network infrastructure gradually to evaluate improvements of model variants that we will clarify in the following. The results summarised in Table 4.2 give the order of magnitude in seconds to find a solution on medusa network instances depending on the following model variants.

Table 4.2: Computation time in seconds to find a solution for the decision problem on the 11 medusa network instances using CPLEX 12.10.

		Model variants			
Instance	Number of slots	Reference	Relaxation	Linearisation	Partial tracks assignment
1	10	$1-10$	$1-10$	$0.1-1$	$0.01-0.1$
2	20	$100-1$k	$100-1$k	≈ 10	$1-10$
3	24	≈ 10k	$100-1$k	≈ 10	$1-10$
4	26			$10-100$	$1-10$
5	26			$10-100$	$1-10$
6	28			1k-10k	$1-10$
7	30			1k-10k	≈ 100
8	32			1k-10k	$100-1$k
9	36				1k-10k
10	38				$100-1$k
11	40				intractable

The *Reference* model variant uses all the constrained variables described above Formulation (4.1), as well as constraints (4.1a) to (4.1j) such that headway constraints (4.1b) to (4.1g) were not linearised yet i.e. without Big-M forms. For example, constraints (4.1b) were instead written as these logical constraints: $y_{s,i,i',k} \Rightarrow (t_{i',j'} - t_{i,j} + z_{i,i',j,j'} * T \geq \Delta_{\text{spa}})$. The *Relaxation* model variant differs from the *Reference* variant only by the continuous relaxation of all the start time variables $t_{i,j}$. These two variants are tested on the medusa network instances 1 to 3. We observe that the computation time has improved for the third

instance when the start time variables integrality is relaxed. The motivation for testing the time relaxation was to find feasible solutions easier while still being valid. Moreover, the time granularity is finer for Île-de-France region timetables, i.e. the decasecond, versus the half-minute in other regions, and the minute for the fictive medusa network instances. However, a solution rounding is then still necessary even for the decasecond granularity, and this can be a cause of infeasibility. On the other hand, we temporary tried to add slack variables to the operations durations of the model to be minimised, and to maximise a unique robustness variable in supplement in all the headway constraints. In this special experiment, we observed that the robustness variable value increases when the start time variables integrality is relaxed. We did not explore more about time relaxation, slack variables and robustness maximisation using formulation (4.1). Instead, we were interested to challenge the search of feasible solutions on larger instances. Therefore, in the two next model variants, we go back with our decision problem with integral start time variables without additional slack.

In the *Linearisation* model variant, constraints (4.1a) to (4.1j) are linearised as displayed in Formulation (4.1), and valid inequalities (4.1k) to (4.1m) are included. The model is solved on the 3 previous instances and on 5 additional ones where slots are included as indicated in Table 3.8. These modifications significantly improved the computation time on the first three instances.

Finally, we experimented to enforce some symmetry breaking by arbitrary fixing a partial tracks assignment for slots in some line sections. More precisely, in Figure 3.7, we remark for example that to run between stations C and E in any direction, a train can either take the first track in all sections C − d, d and d − E, or either the second track in all these same sections. Indeed, the little green dots indicate the impossibility to switch to another track, as discussed in Section 3.3.2. What we tried is to assign the operations of the downstream slots to the first track of these sections and the operations of the upstream slots to the second track of these sections a preprocessing step. Note that the same setting is performed for slot operations on any other sections of the medusa network infrastructure around little green dots. We solve the *Partial tracks assignment* model variant and improve the computation time for the first eight instances. We also solved two additional instances. However, the eleventh instance remained undecided after hours of computation time.

Until here, we proposed an Integer Linear Program to model our decision problem and solve it using an off-the-shelf solver while varying the settings of the solve by introducing model variants. This approach is useful to begin the apprehension of the problem and to

solve small to medium instances, observing that it becomes harder to decide the problem in reasonable time when the instance size grows. However, this model in itself is not enough satisfactory to address real-world cases like the Savoie area which infrastructure is even more constrained than the medusa network infrastructure. The remainder of this thesis is therefore devoted to find faster techniques to be able to solve the problem on our real-world infrastructure.

4.2 Decomposing the model

During the experiments undertaken to solve Formulation (4.1), we observed that once a feasible solution is found, and if we try to solve the model again while fixing the timetable from this solution and looking uniquely for tracks assignment, then this specific problem is easily handled by an off-the-shelf solver on our instances. We take precautions about this phenomenon by expressing the following comments. Finding the slot start times allowing a conflict-free tracks assignment is a challenging task. When such a timetable is obtained, the subsequent tracks assignment problem can be seen as a set of Train Platforming Problems (TPP) on sections of the mesoscopic scale infrastructure. We saw in Chapter 2 that the TPP is a difficult problem but in our case, we are not searching to assign a maximal number of trains as we already know by assumption that a feasible solution exists for all the trains. Moreover, the problem sizes stay reasonable because we do not consider microscopic signalling, nor the aperiodic problem during a day time span, and we do not address (yet) the main stations of the French railway infrastructure e.g. the Parisian stations, Lyon Part Dieu, Lille Flandres or Marseille Saint-Charles to give only few future interesting cases to consider. To summarise our main idea, we now look at our decision problem from another point of view: the goal is to find a timetable for which we can assume that a tracks assignment exists, and we then use an off-the-shelf solver as an oracle in a second step to find the tracks assignment for this timetable. Note that this approach can be extended if more challenging stations are included in our instances, namely by conceiving dedicated tracks assignment algorithms if this task becomes intractable using the solver. In Section 4.2.1, we present a method to evaluate the conflicts in a timetable without explicit tracks assignment knowledge. This function is used in the remainder of this section to find conflict-free timetables. Solution approaches are presented in Section 4.2.2. More precisely, in Section 4.2.2.1, we present a variable-and-constraint generation framework that progressively theoretically converges towards a conflict-free solution if there is one. This method should however be refined to be tractable and we provide some hints for that. In Section 4.2.2.2, we describe a tabu search heuristic able to minimise timetable conflicts. A didactic execution of the heuristic is given in Section 4.2.2.3. Finally, experimental results on our largest Savoie area instance

are presented in Section 4.2.3. Note that in the context of Section 4.2 and Section 4.1.2, all the safety headway times are equal, e.g. parameterised to 4 minutes in our case. The approach that we present can be extended in order to consider variable headway times depending on the trains following each other and on infrastructure specificities.

4.2.1 Conflict evaluation function using the timetable only

This section aims at describing a method to evaluate the conflicts in a timetable containing any scheduled slots but without requiring knowledge about the tracks assignment. The benefit of the resulting conflict evaluation function is that a timetable evaluated to have 0 conflicts should allow the existence of tracks assignment satisfying all the constraints of Formulation (4.1). To illustrate the mechanism of this function, we will consider the toy example infrastructure on which we schedule the start time of a fast slot between stations E and D and the start time of an omnibus slot between stations E and C, without assigning tracks to the operations. This timetable is displayed in Figure 4.1, where the fast slot is blue and the omnibus slot is red. In the following, we explain how we compute the spacing conflicts, the track reuse conflicts and the crossing conflicts from this timetable. To this end, we first introduce the notion of infrastructure occupation by the slots at the mesoscopic scale in section 4.2.1.1. Then, we describe how we consider the capacity of the infrastructure in Section 4.2.1.2. Finally, the conflicts evaluation is presented in Section 4.2.1.3.

4.2.1.1 Occupation and use variables

Occupation variables. As we search to evaluate conflicts between slots by just knowing the timetable, we compute an additional needed ingredient that is a kind of footprint left by the slots when traversing the infrastructure. We call these footprints the *occupation* variables. The occupation of a given slot is measured depending on operations durations, on the traversed infrastructure elements and on the corresponding safety headways. By infrastructure element, we mean any infrastructure section and potential crossing areas where mesoscopic routes connect several sections. As detailed in Chapter 3, the spacing headway constraints apply on line sections, so we compute *spacing* occupations on line sections. We saw that track reuse headway constraints apply at stations, but also in line sections to avoid trains running in the opposite direction. Therefore, we compute *reuse* occupations for each section traversed by slots. Finally, the crossing headway constraints prevent any two trains to run on incompatible routes to close in time. Hence, we compute *crossing* occupations for each traversed potential crossing area where mesoscopic routes connect several sections. Table 4.3 presents the occupation of the slots scheduled in Figure 4.1 on elements of the toy example infrastructure. Black squares indicate the

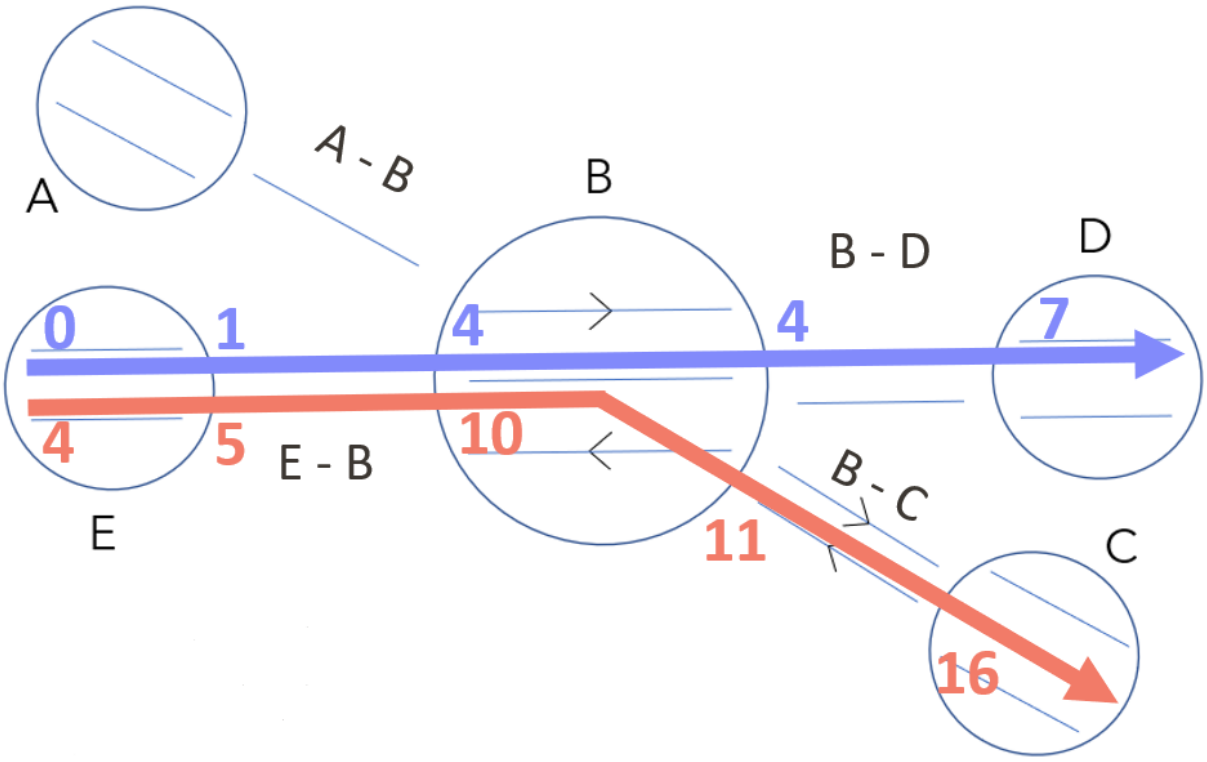


Figure 4.1: Two scheduled slots without assigned tracks for which conflicts are to be evaluated.

section where the rolling stock of the corresponding slot is located at the beginning of the corresponding minute. Squares coloured in red, gold and turquoise correspond to spacing, reuse and crossing headways, respectively. Several observations can be made from the displayed occupations. First, the spacing occupations are only composed by the spacing headways, that begin as soon as the rolling stock enter the line section. We indeed recall that two following trains can be located on the same mesoscopic line track if the spacing headway is respected at the track entrance. Second, the reuse occupations, however, are composed by the time interval when the rolling stock is in the section, followed by the track reuse headway. Indeed, two trains cannot occupy the same track at stations or in line sections in the opposite direction. Finally, the crossing occupations are composed by the crossing headways and start as soon as the rolling stock enters the corresponding potential crossing area i.e. when the operation on the next section starts. In compliance with our hypothesis made in Section 3.3.1 stating that trains passing through a station without dwelling spend a negligible amount of time in this station, remark in Table 4.3 how the fast slot triggers occupations in many close infrastructure elements around station B at minute 4.

Use variables. Now that we know the occupation left by individual slots on elements of the infrastructure, we make the link with the global use of each infrastructure element.

The *use* variables can be seen as a numerical aggregation of the slot occupations variables. Like for the occupation variables, there are 3 types of use variables depending on the safety headways considered. Moreover, the aggregations are sorted on the one hand by running direction in all the sections of the infrastructure, and on the other hand by connecting sections in all potential crossing areas. To illustrate this concept from the timetable of Figure 4.1, we compute the use variables corresponding to the reuse occupations at station E in Table 4.4 and the use variables corresponding to the crossing occupations in the potential crossing area connecting sections B, B − D and B − C in Table 4.5. We observe in Table 4.4 that the use of station E in the downstream direction is equal to 1 track from times 0 to 8, except at time 4 where this use is equal to 2 tracks. The same kind of computations can be performed on line sections for the use variables corresponding to the spacing and to the reuse occupations respectively, and are sorted by running direction. In Table 4.5, the use variables corresponding to the crossing occupations are sorted depending on the sections traversed by the slots. We observe that the use between sections B and B − D is equal to 1 downstream route from times 4 to 7 and that the use between sections B and B − C is equal to 1 downstream route from times 11 to 14. In the following, we estimate the capacity infrastructure on each of its elements i.e. on sections and on potential crossing areas connecting sections. Then, we will see how conflicts are computed with the use variables and the infrastructure capacity estimations.

4.2.1.2 Infrastructure capacity

In this section, we propose a capacity measure of elements of the infrastructure. More precisely, we search to determine how many tracks can be used depending on the direction of the traffic on sections at any time, and how many routes can be run on potential crossing areas at any time.

Capacity of sections. On a given section s of the infrastructure, we define the capacity Cap_s of this section as an integer value equal to the number of tracks $|K_s|$ of this section. We also specify additional parameters tracking the number of tracks by allowed running directions. The parameter $\overleftrightarrow{Cap_s}$ indicates the number of tracks of section s that can be run in both directions. Moreover, parameters $\overrightarrow{Cap_s}$ and $\overleftarrow{Cap_s}$ indicate the number of tracks of section s that can be run uniquely in the downstream direction and uniquely in the upstream direction, respectively. In the end, we have $Cap_s = \overleftrightarrow{Cap_s} + \overrightarrow{Cap_s} + \overleftarrow{Cap_s}$ for any section s of the infrastructure.

Capacity of potential crossing areas. A less straightforward task than defining the capacity of infrastructure sections is to determine the capacity of potential crossing areas where mesoscopic routes connect several sections. What we look for is the maximal sets

Table 4.3: Occupation variables of a fast and an omnibus slots on elements of the toy example infrastructure, determined from their timetable in Figure 4.1.

Infra. element	Time / Slot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Δ_{reu} E	fast																				
	om.																				
Δ_{cro} E|E − B	fast																				
	om.																				
Δ_{spa} Δ_{reu} E − B	fast																				
	om.																				
Δ_{cro} A − B/ E − B|B	fast																				
	om.																				
Δ_{reu} B	fast																				
	om.																				
Δ_{cro} B|B − D /B − C	fast																				
	om.																				
Δ_{spa} Δ_{reu} B − D	fast																				
Δ_{cro} B-D|D	fast																				
Δ_{reu} D	fast																				
Δ_{spa} Δ_{reu} B − C	om.																				
Δ_{cro} B-C|C	om.																				
Δ_{reu} C	om.																				

Table 4.4: Use variables corresponding to the reuse occupations at station E.

Infra. element	Time / Slot	0	1	2	3	4	5	6	7	8
Δ_{reu} E	fast									
	om.									
	$\overrightarrow{u}_{\text{reu}[E]}$	1	1	1	1	2	1	1	1	1

Table 4.5: Use variables corresponding to the crossing occupations in the potential crossing area connecting sections B, B − D and B − C.

Infra. element	Time / Slot	4	5	6	7	8	9	10	11	12	13	14
Δ_{cro} B|B − D /B − C	fast	■	■	■	■							
	omnibus								■	■	■	■
	$u_{\text{cro[B to B−D]}}$	1	1	1	1	0	0	0	0	0	0	0
	$u_{\text{cro[B to B−C]}}$	0	0	0	0	0	0	0	1	1	1	1

of routes that can be used by trains simultaneously without being incompatible. To do so, we perform a preprocessing step that consists in dressing the list of maximal cliques in the compatibility graph of routes in each potential conflict area. This preprocessing step is represented in Figure 4.2 for the case of the potential crossing area between sections B, B − D and B − C. The downstream routes in red and the upstream routes in green are the nodes of the compatibility graph on the centre of Figure 4.2. The nodes are clustered in four blue polygons depending on the initial and the terminal sections of the routes. For example, the first, the second, the fourth and the fifth downstream routes in red belong to the upper left blue polygon and go from section B to section B − D, as indicated in the top-right part of Figure 4.2. The other nodes are clustered in the same logic depending on the ordered sections that the corresponding routes traverse. Black links represent a compatibility between two routes, indicating that two trains can run these routes simultaneously. For example, the first red route between the first track of station B and the first track of line section B − D is compatible with the third green route between the second track of line section B − D and the second track of station B. Remark that black links in the middle of the graph represent compatibilities between routes in the opposite direction whereas black links in the sides of the graph represent compatibilities between routes in the same direction. We search to find maximal sets of nodes such that each pair of nodes in the sets are connected by a black link. Representative elements of the list of maximal cliques is provided on the right side of Figure 4.2. We observe that the largest cardinal of cliques in this list is 3, so up to 3 trains can run simultaneously between sections B, B − D and B − C. For example, the first downstream route in red can be run simultaneously with the third and sixth upstream routes in green. Next to each displayed clique, we report a blue square with four positions, where we sum the routes of the clique depending on the blue polygons in which they are included. For example, in the first displayed clique, there is one route from section B to section B − D, another route from section B − D to section B, and a third route from section B − C to section B. Thus, the value of 1 is assigned to all positions of the blue square except the bottom-left

position where the value 0 is assigned because there is no route from section B to section $B - C$ in this clique. If several routes are compatible and going from and to the same sections in the same clique, a greater integer value can be assigned in the corresponding blue square, as it is the case for the second displayed clique in Figure 4.2. We can also observe that distinct cliques can correspond to a same blue square because distinct routes of these cliques can share the same initial and terminal sections. This is the case of the third and the fourth displayed cliques.

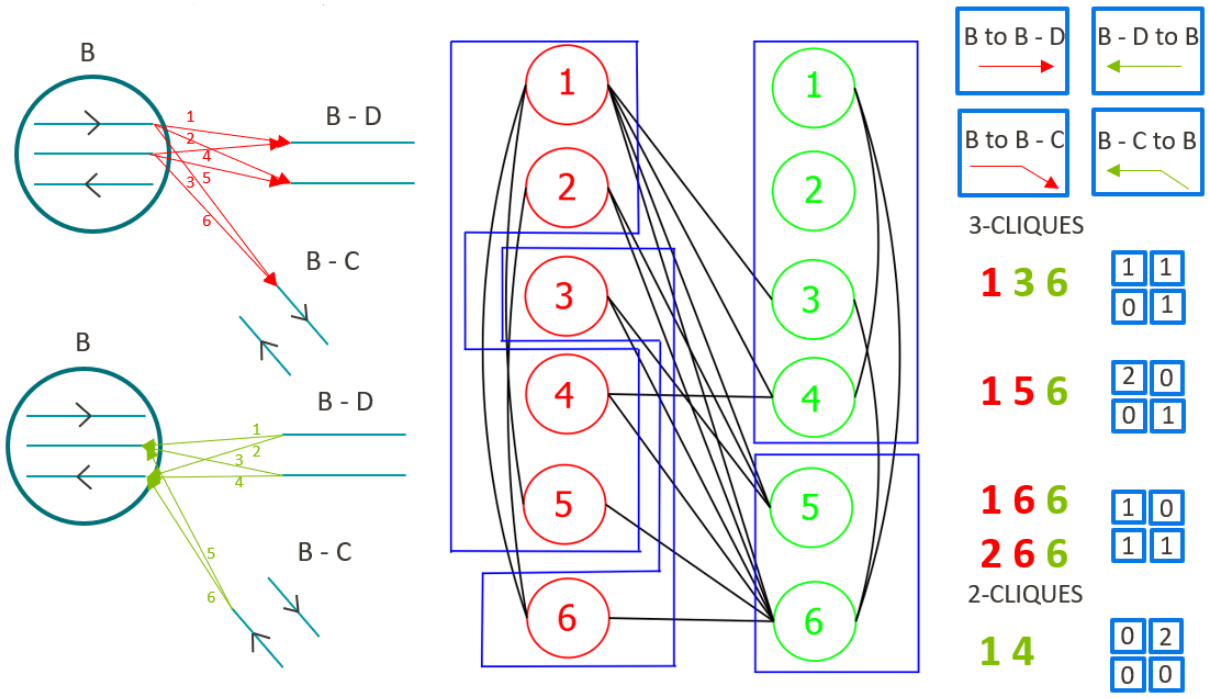


Figure 4.2: Determining capacity in potential crossing areas.

There are some maximal cliques of cardinal 2, but only one of them corresponds to a blue square which information is not already contained in the blue squares of cliques of cardinal 3 is fifth displayed clique, composed by the first and the fourth upstream routes in green. Indeed, another maximal clique of cardinal 2 is composed by the first and the sixth upstream routes in green, but the corresponding blue square is composed of values 1 in the two right positions. The information provided by this square is already included in the square of the first displayed clique. Therefore, the maximal clique of cardinal 2 composed by the first and the sixth upstream routes in green is not representative in this context. We do not report maximal cliques of cardinal 1 because they do not represent compatibility between several routes. We can assume that in the absence of cliques of carnality greater than 1 in a potential crossing area, then a blue square is initialised with a value 1 in a unique position corresponding to one existing route between sections. When the search of representative maximal cliques is over, the resulting blue squares determine the capacity of the potential crossing area. More precisely, each blue square indicates a

maximal number of trains that can run on routes from specific sections to specific other sections simultaneously.

4.2.1.3 Conflicts evaluation

At this point, we defined the occupation and use variables from scheduled slots in a timetable, and we also introduced a method to estimate the capacity of elements of the infrastructure. We have all the ingredients to evaluate conflicts between scheduled slots in a timetable and without knowing their tracks assignment explicitly. In the following, we describe how we evaluate conflicts between slots in multi track sections, in line sections containing a single track, and in potential crossing areas. Note that all use variables and conflict evaluations are time indexed whereas infrastructure capacities are constant at each time unit. In order to simplify the notations, the four conflict formulas that we introduce in this section correspond to a given time unit i.e. the time indices are not explicitly represented here.

Conflicts in multi track sections. In stations and line sections containing more than one track, conflicts between slots are evaluated by comparing the value of use variables at each unit of time with the capacity of the sections.

Given a multi track station s, the conflicts at this station at each time unit are computed using the following formula:

$$\begin{aligned} v_s = \max(0, &\overrightarrow{u}_{\text{reu}[s]} - (\overrightarrow{Cap_s} + \overleftrightarrow{Cap_s}), \\ &\overleftarrow{u}_{\text{reu}[s]} - (\overleftrightarrow{Cap_s} + \overrightarrow{Cap_s}), \\ &\overrightarrow{u}_{\text{reu}[s]} + \overleftarrow{u}_{\text{reu}[s]} - Cap_s) \end{aligned}$$

The conflict v_s expresses a *violation* of the infrastructure capacity of station s during a unit of time. We compute this conflict as the maximum of four terms. The first term is 0 and indicates that there is no conflict in the case of negativity of the 3 subsequent terms. The second term is the difference between the value of the use variable of station s corresponding to the reuse occupations in the downstream direction and the number of tracks allowing trains running downstream. The third term is symmetric with respect to the second term and applies on the upstream direction. Finally, the fourth term is the difference between the value of the use variable of station s corresponding to the reuse occupations in both directions and the total capacity of station s. To illustrate this calculation, we can look again to Table 4.5 where only two downstream slots occupy the station E. By applying the above conflict evaluation formula to station E, we can observe

that v_{E} equals 0 for all times from 0 to 8 included. This means that a track assignment is possible in station E for the fast slot and the omnibus slot. In particular, the slots will be assigned to 2 different tracks at station E because the value of the use variable at time 4 equals the total capacity of station E, so track reuse constraints of Formulation (4.1) would not allow these slots to be assigned to the same track.

Given a multi track line section s', the conflict $v_{s'}$ is computed similarly to the conflict v_s at station s but using the use variables corresponding to the spacing occupations instead. The formula reads:

$$\begin{aligned} v_{s'} = \max(0, &\overrightarrow{u}_{\mathrm{spa}[s']} - (\overrightarrow{Cap_{s'}} + \overleftrightarrow{Cap_{s'}}), \\ &\overleftarrow{u}_{\mathrm{spa}[s']} - (\overleftarrow{Cap_{s'}} + \overleftrightarrow{Cap_{s'}}), \\ &\overrightarrow{u}_{\mathrm{spa}[s']} + \overleftarrow{u}_{\mathrm{spa}[s']} - Cap_{s'}) \end{aligned}$$

We saw in Section 4.2.1.1 and in Table 4.3 that the spacing occupations are generally shorter than the reuse occupations because the latter also includes the time interval when the rolling stock is present in the section whereas the former does not. We indeed already commented that two trains can follow each other on the same mesoscopic line track if they respect spacing headways at the track entrance. We noticed that in some particular cases on our fictive infrastructures where all the tracks of a line section can be run in both directions like in line section E − B of the toy example infrastructure, a timetable evaluated with no conflicts could result in an impossibility to assign tracks in the line section using Formulation (4.1). This is due to the fact that trains running in the same direction following each other to close in time at the entrance of line section E − B should therefore be assigned to different tracks, thus making impossible for trains running in opposite direction to enter the line section because of the track reuse constraints, even if the spacing occupations terminated. However, our evaluation of conflicts in line sections is accurate for real-world infrastructures since a preferred running direction is generally provided for almost all tracks of multi track line sections i.e. $\overleftrightarrow{Cap_{s'}}$ generally equals 0 in a real-world multi track line section s'. In the next paragraph, we address more precisely the relationship between use variables related to both spacing and reuse occupations in line sections tracks containing a single track. In this case, indeed, it is more natural to see trains running in different directions on a single track. We will see that the resulting formula is more complex than the two previous ones, and that it can be further extended in order to take into account the issue discussed above in this paragraph for the case of multi track line sections with line tracks allowing trains in both directions.

Conflicts in single track line sections. Mesoscopic line sections containing a single track should allow several trains to run on the track if they run in the same direction and they respect spacing headway constraints, while also forbidding two trains from running on the track in the opposite direction if they do not respect the track reuse constraints. Here, a first alternative could be to use the same conflict formula as when we compute conflicts in stations. However, this alternative is not satisfactory because it is too conservative as it would not allow two trains to follow each other directly after the end of the spacing headways. Instead, the following train would have to wait the first train to liberate the whole track plus a track reuse headway. Therefore, we will describe a method to evaluate accurately the conflicts on single track line sections while allowing to exploit the full capacity i.e. by not being too conservative for trains following each other in the same direction. We introduce auxiliary integer variables sim_s that indicate simultaneous reuse occupations of slots running on the opposite direction in the single line track section s at a given time. More precisely, we have $sim_s = \overrightarrow{u}_{\text{reu}[s]} * \overleftarrow{u}_{\text{reu}[s]}$. That is, a strictly positive value of sim_s at a given time is unacceptable as it would translate a simultaneous reuse occupation of trains running in opposite direction on the single track. However, when sim_s equals 0, then use variables corresponding to spacing occupations are sufficient to evaluate conflicts on the single track. Hence, we use the following formula to evaluate conflicts in single track line sections:

$$\begin{aligned} v_s = \max(&\overrightarrow{u}_{\text{spa}[s]} - Cap_s, \\ &\overleftarrow{u}_{\text{spa}[s]} - Cap_s, \\ &sim_s) \end{aligned}$$

The first two terms trigger a conflict detection if two trains follow each other to close in time while the third term triggers a conflict detection when trains running in the opposite direction activate reuse occupations at the same time. Note that as the third term is never strictly negative, then the maximum with 0 is omitted in the computation of v_s in the case of single track line sections. Extending this method should be investigated in future work in order to compute conflicts more accurately in the same fashion in multi track line sections containing tracks allowing trains to run in both directions.

Conflicts in potential crossing areas. We finally describe how the conflicts between slots are evaluated in potential crossing areas without knowing explicitly the routes on which they run. Once again, we will compare use variables at each time unit with the available capacity of the infrastructure element. We recall that in the case of potential crossing areas, the capacity corresponds to a maximal set of slots that can run simultaneously on routes between specific initial and terminal sections, determined in a preprocessing step

detailed in Section 4.2.1.2. We note Cl_a the set of capacity blue squares corresponding to representative maximal cliques of the crossing area a, illustrated in Figure 4.2 in the case of the crossing area between sections B, B − D and B − C of the toy example infrastructure. Each blue square associated to a representative maximal clique contains a set of positions $p \in Pos_a$ that correspond to initial and terminal sections of routes in the potential crossing area a. The conflicts in the potential crossing area a are computed using the following formula:

$$v_a = \min(\sum_{p \in Pos_a} \max(0, u_{\text{cro}[p]} - Cap_p^{cl}) \mid \forall cl \in Cl_a)$$

We now provide a simple image of how this conflict computation can be understood, and we use the potential crossing area between sections B, B − D and B − C of the toy example infrastructure to illustrate these computations. We start from the timetable of Figure 4.1 for which the use variables corresponding the our potential crossing area are computed at each unit of time in Table 4.5. We observe in particular that as there are no upstream slots in the timetable, then use variables $u_{\text{cro[B−DtoB]}}$ and $u_{\text{cro[B−CtoB]}}$ equal 0 at all time units. Now, looking at the blue squares of Figure 4.2, we observe that the two relevant positions of this study, corresponding to the downstream slots of the considered timetable, are located at the top-left and bottom-left part of the blue squares. Let us focus on a given unit of time and on a given blue square clique. We compute the difference between the use variable and the capacity at each position and we replace these values by 0 if they are strictly negative. Then, we sum these values. The result corresponds to the conflict of the use variables with respect to the considered blue square clique. However, other blue square cliques could be more permissive for our use variables. Therefore, we do the same computations considering the other blue square cliques, still for the same unit of time. Finally, the minimal conflict value among the computations done for all blue square cliques is kept, because each blue square clique corresponds to a set of compatible routes that trains can run simultaneously. We repeat the whole process for all units of times. In the case of use variables of Table 4.5, it can be established that there are no crossing conflicts for all time units.

4.2.2 Solution approaches

In Section 4.2.1, we described a function able to compute conflicts between scheduled slots in a timetable without using tracks assignment knowledge explicitly. About the computation of the conflicts from a given timetable, we dispose of two methods. First, we can solve an Integer Linear Program minimising the sum of all conflicts v described in Section 4.2.1.3. The model is therefore linearised as follows: we replace the "max"

operations by "greater than" inequalities for each argument, and we replace the "min" operations by Big-M forms to be able to select only one blue square clique i.e. the most permissive one with respect to the use variables corresponding to crossing occupations. Second, the computations can be done algorithmically by using the provided formulas in Section 4.2.1.3.

In this section, we present two solution approaches using the conflict evaluation function of Section 4.2.1 to look for conflict-free timetables where we then use Formulation (4.1) as a fast oracle to solve tracks assignment. In Section 4.2.2.1, we briefly describe a variables and constraints generation approach that we believe to be a promising approach but that however needs more investigations on its components in order to become tractable. We improve the problem resolution by proposing a second method that consists in an adaptive heuristic embedded in a tabu search framework described in Section 4.2.2.2. A didactic execution of this heuristic on the toy example instance is proposed in Section 4.2.2.3. Finally, experimental results on our largest Savoie area instance are provided in Section 4.2.3.

4.2.2.1 Using variables and constraints generation

We present a first solution approach to obtain conflict-free timetables without using tracks assignment decisions. This approach consists to begin with a small number of variables and constraints and to include new columns and rows to the model as necessary. A flowchart illustrating the method is given in Figure 4.3. We start by describing the Timetabling Problem at the centre of the figure.

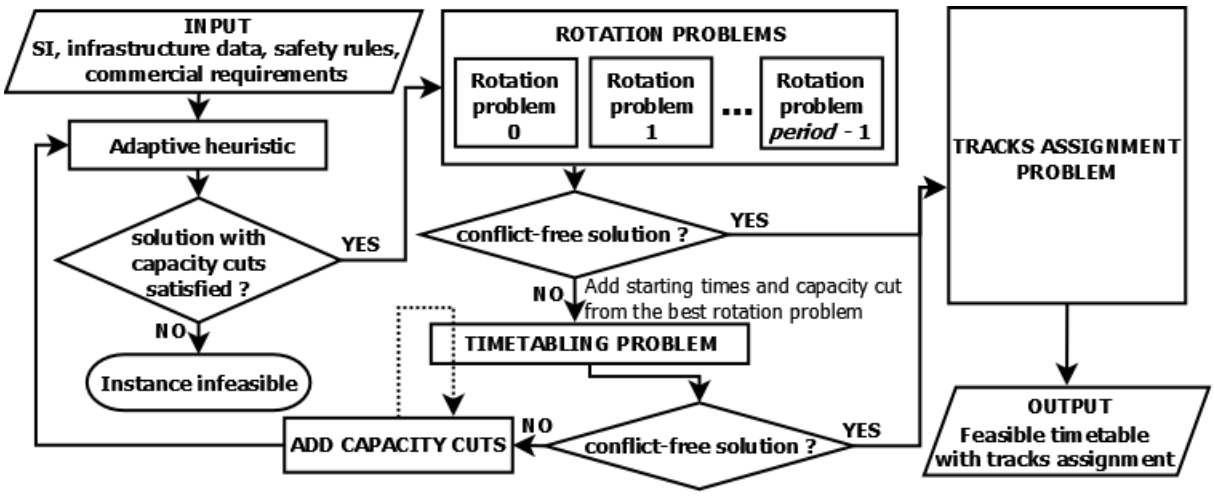


Figure 4.3: Variables and constraints generation procedure.

Timetabling problem. In the Timetabling problem, we solve a model similar to Formulation (4.1) but without the tracks assignment variables and without the safety headway constraints. Instead, we include the conflict evaluation function of Section 4.2.1 as the objective function to be minimised. Another particularity is that the slot starting times variables cannot take any value at the beginning of the procedure. Instead, each slot has its own *candidate starting times* set that are progressively filled by the Adaptive heuristic and by the Rotation problems that we describe later on. The Timetabling problem provides a starting time to each slots among their candidate starting time set such that the objective function is minimised. If the objective value is strictly positive, we saw in Section 4.2.1 that it means the presence of conflicts and then the impossibility to launch the Tracks assignment problem. In this case, *capacity cuts* are added to the model in the form of hard constraints to prevent a set of starting times to take values leading to the same resulting conflicts in the objective function. The next step is to add new values to some candidate starting times sets in order to attempt improving the solutions in the next iteration of the Timetabling problem.

Adaptive heuristic and Rotation problems. The goal of the adaptive heuristic is to schedule greedily as much slots as possible with the full range of possible starting times and while enforcing the objective value to remain 0 and activating the current capacity cuts. The heuristic is adaptive in the sense that at each iteration, the priority list determining slots to be scheduled change depending on the emptiness of their candidate starting times set and on the decreasing number of capacity cuts they are involved in. When the largest scheduled subset of slots satisfying the capacity cuts is found, then the objective value is not enforced anymore to equal 0 and the partial timetable obtained is fixed. The rotation problem 0 aims at providing a starting time to the potential remaining slots among their candidate starting times set while minimising the objective function. The resulting objective value is recorded and the subsequent rotation problems will attempt to improve it. In the rotation problem 1, the partial fixed timetable is shifted by one minute later, and a starting time for the potential remaining slots is again checked among their candidate starting times set. The procedure is repeated with the following rotation problems for which the initial partial timetable from the adaptive heuristic is always more shifted. The rotation problem with the best objective value allows to fill the candidate starting times sets with the values of the corresponding shifted partial timetable. If the objective value of the Timetabling problem is still strictly positive when using the augmented candidate starting times sets, then new capacity cuts are added and the heuristic is restarted with the updated priority list of slots to be scheduled first.

Drawbacks and possible improvements. This approach was able to solve the toy example instance. However, when testing it on our real-world instances, it turned out that the Timetabling problems have become intractable after already few iterations. We suppose that adding the capacity cuts as hard constraints do not make this approach simpler to solve than the original Formulation (4.1), and that the values added to the candidate starting times sets are not impacting enough to obtain conflict-free solutions fast. We take these observations into consideration and propose an improved method in Section 4.2.2.2 that makes a smarter use of the conflict evaluation function of Section 4.2.1.

4.2.2.2 A tabu search heuristic to minimise the conflicts

We present a second solution approach to obtain conflict-free timetables without using tracks assignment decisions. Contrary to the previous approaches, additional hard constraints are not considered any more and the full range of starting times is always considered in this new approach. The procedure is illustrated in Figure 4.4.

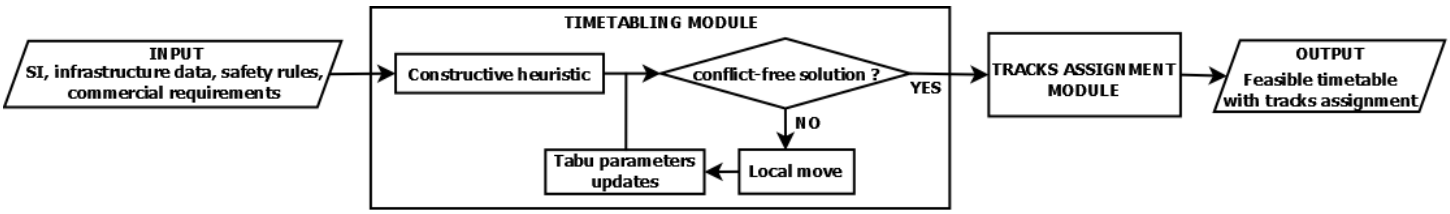


Figure 4.4: Tabu search conflict minimisation heuristic.

Constructive heuristic. The procedure begins with a constructive heuristic. At each step, a slot is selected and a starting time is chosen for this selected slot such that the value of conflict evaluation function increases as less as possible. The heuristic is greedy, so the remaining slots are scheduled step by step without modifying the schedule of the previous slots. When the heuristic scheduled all the slots, the conflict evaluation value can equal 0. In this case, the tracks assignment is performed by solving Formulation (4.1) with the timetable provided by the heuristic. If however the conflict evaluation value is strictly positive, then we enter in the tabu search framework in order to solve the existing conflicts.

Local move. In order to solve the conflict and thus decrease the conflict evaluation value, we select the slot that is involved in the maximal number of conflicts. We then search to minimise the conflict evaluation function by finding the best new starting time for the selected slot. We call this action a *local move* as it allows to navigate from one solution to another by modifying the starting time of a slot. When a local move is

performed on a slot, the combination of this local move with the given slot becomes tabu i.e. it is not immediately possible to modify the timetable using the same slot.

Tabu parameters update. A tabu countdown associated to each combination of local moves and slots is decreased at each iteration until being equal to 0. When the tabu countdown of a combination is equal to 0, it means that the associated local move can be performed again on the associated slot. The tabu duration at which the tabu countdown is initialised when a local move is performed on a slot may increase each time that the same combination of a local move on a given slot is performed. These tabu durations updates allow to escape potential local optima where the same local moves could be selected again and again. Moreover, we also increase a multiplicative coefficient associated to the remaining conflicts in which the modified slot is still involved, in order to concentrate in hard conflicting areas on the infrastructure. If all conflicts are resolved, the tracks assignment oracle can be launched. In the following, we show how this tabu search framework can be applied on the toy example instance.

4.2.2.3 Heuristic execution on the toy example instance

We described a conflict evaluation function and solution approaches to obtain conflict-free timetables without using tracks assignment variables explicitly. In this section, we present a didactic execution of our tabu search framework on the toy example instance of Section 3.3.1. This resolution is provided for 3 purposes. First, conflicts between slot will naturally occur and they will be visualised on the infrastructure by displaying the corresponding conflicts variables v. Second, this resolution gives a detailed view of the heuristic mechanism introduced in Section 4.2.2.2. Third, we will see that the conflict-free timetable obtained allows to find a tracks assignment satisfying all the constraints of Formulation (4.1).

Initialisation. Our solution approach normally begins by a constructive heuristic aiming at attributing starting times to slots in a greedy way while minimising the conflict evaluation value for each slot added to the timetable. However, for this instance, the constructive heuristic provided a timetable for which the conflict evaluation value already equalled 0. In order to show how the tabu framework is applied, we therefore propose to start from an initial solution obtained with random starting times. This initial solution and the resulting heuristic state are presented in Table 4.6 and counts 17 conflicts in total. These conflicts will be addressed by our tabu framework that works as follows. The *tabu countdown* of each slot is equal to 0 i.e. the local moves associated to each slots can be used. The *tabu duration* of each slot is initialised to value 1, thus the tabu countdown of a slot will take value 1 as soon as the local move associated to this slot is used. At

each iteration of the procedure, a tabu duration may be increased in order to increase the tabu countdown if the same local move is selected in a future iteration. We observe that among the 17 conflicts aforementioned of this timetable, each slot can be involved in some of these conflicts. The sum of the values in last line of Table 4.6 is not necessarily equal to the total number of conflicts in the timetable, because several slots can be involved in a same conflict. We select the slot on which we will perform a local move in the first iteration. The selected slot is the one involved in the largest weighted sum of conflicts. By weighted, we mean that we take into account the multiplicative coefficient associated to any conflict variable, as specified in last the paragraph of Section 4.2.2.2. We observe that slots s0 and s1 are involved in the same amount of conflicts. The tabu countdown of these to slots is equal to 0, so we can apply a local move either on slot s0 or on slot s1. Here, we arbitrarily select s0, but it would have been possible to select one of these two slots randomly or using other tie breaking rules. The full timetable of slot s0, determined by its starting time, is given in Figure 4.5. We also provide a visualisation of the conflicts in which slot s0 is involved. More precisely, from Table 4.7 we observe that sim_{A-B}, computed by $\overrightarrow{u}_{\text{reu}[A-B]} * \overleftarrow{u}_{\text{reu}[A-B]}$, is equal to 1 from time 4 to time 10, thus explaining 8 conflicts in which slot s0 is involved. Looking at Table 4.8, we can see that $\overrightarrow{u}_{\text{reu}[B]}$ is strictly greater than $\overleftarrow{Cap_B} + \overrightarrow{Cap_B}$ at time 10, thus clarifying the ninth conflict in which slot s0 is involved.

Table 4.6: Heuristic state after the initialisation, slot s0 is involved in the maximal number of conflicts.

Slots	s0	s1	s2	s3	s4	s5	s6	s7
Starting time	2	17	4	15	13	11	6	13
Tabu countdown	0	0	0	0	0	0	0	0
Tabu duration	1	1	1	1	1	1	1	1
Conflicts involved in	**9**	9	4	8	0	0	2	8

Table 4.7: Slot s0 involved in 8 conflicts in the single track line section A − B.

Infra. element	Time / Slot	3	4	5	6	7	8	9	10	11	12
Δ_{reu} A − B	s0										
	s1										
	$\overrightarrow{u}_{\text{reu[A−B]}}$	1	1	1	1	1	1	1	1	1	0
	$\overleftarrow{u}_{\text{reu[A−B]}}$	0	1	1	1	1	1	1	1	1	1
	$v_{\text{A−B}}$	0	1	1	1	1	1	1	1	1	0

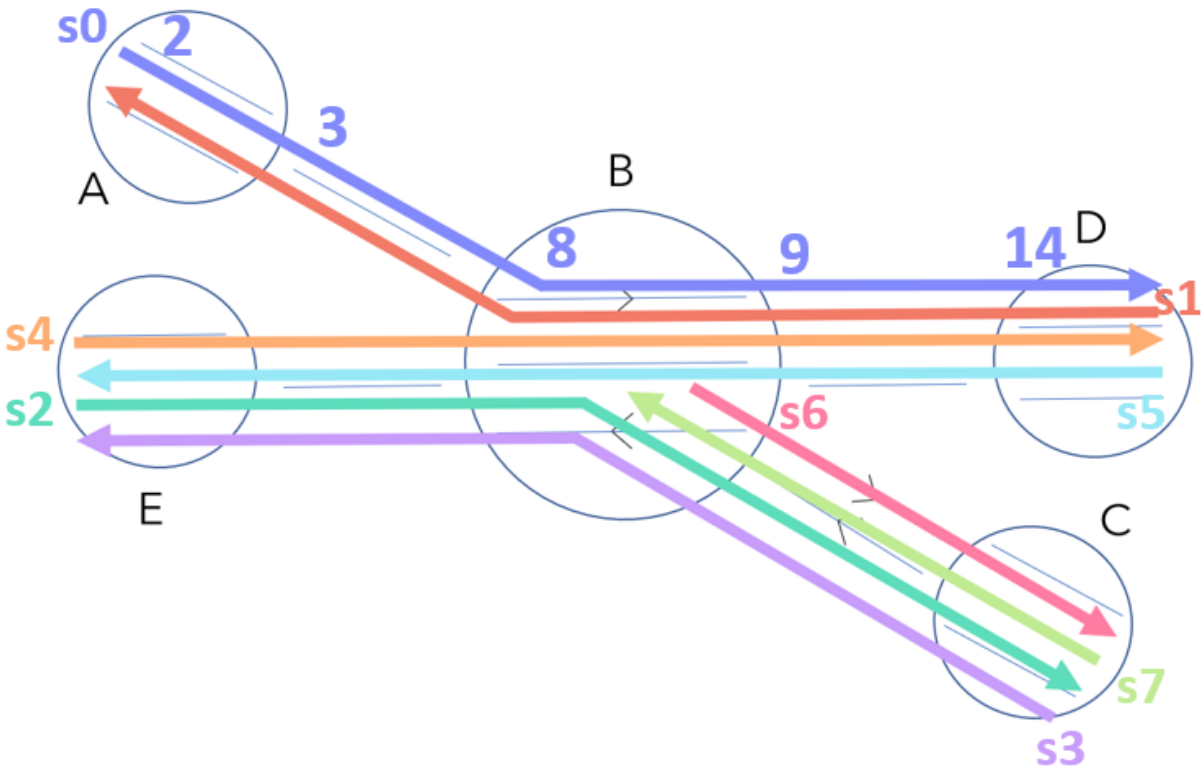


Figure 4.5: Initial timetable of slot s0 involved in 9 conflicts out of 17 in total in the complete timetable.

Iteration 1. We selected slot s0 to perform a local move, which consists in finding a better starting time for this slot such that the overall timetable conflicts are minimised. The new starting time of s0 after this local move is 14 instead of 2, and the amount of conflicts passed from 17 to 8. However, even if the new starting time of slot s0 minimises the overall conflicts in the timetable, we can observe in Figure 4.6 and in Table 4.9 that slot s0 is still involved in 2 residual conflicts in the potential crossing area between sections B, B – D and B – C. Indeed, if we look at top-left and at the bottom-right positions of the blue square cliques in Figure 4.2, we observe that a train running from section B to

Table 4.8: Slot s0 involved in 1 conflict at station B.

Infra. element	Time / Slot	6	7	8	9	10	11	12	13	14
Δ_{reu} B	s0									
	s2									
	s6									
	$\overrightarrow{u}_{\text{reu[B]}}$	1	1	2	2	**3**	2	2	1	1
	v_{B}	0	0	0	0	**1**	0	0	0	0

section B – D and two other trains running from section B – C to B are never allowed to take such routes too close in time. Therefore, when applying the last formula given in Section 4.2.1.3 on the potential crossing area between sections B, B – D and B – C, we obtain 1 crossing conflict from time 1 to time 2.

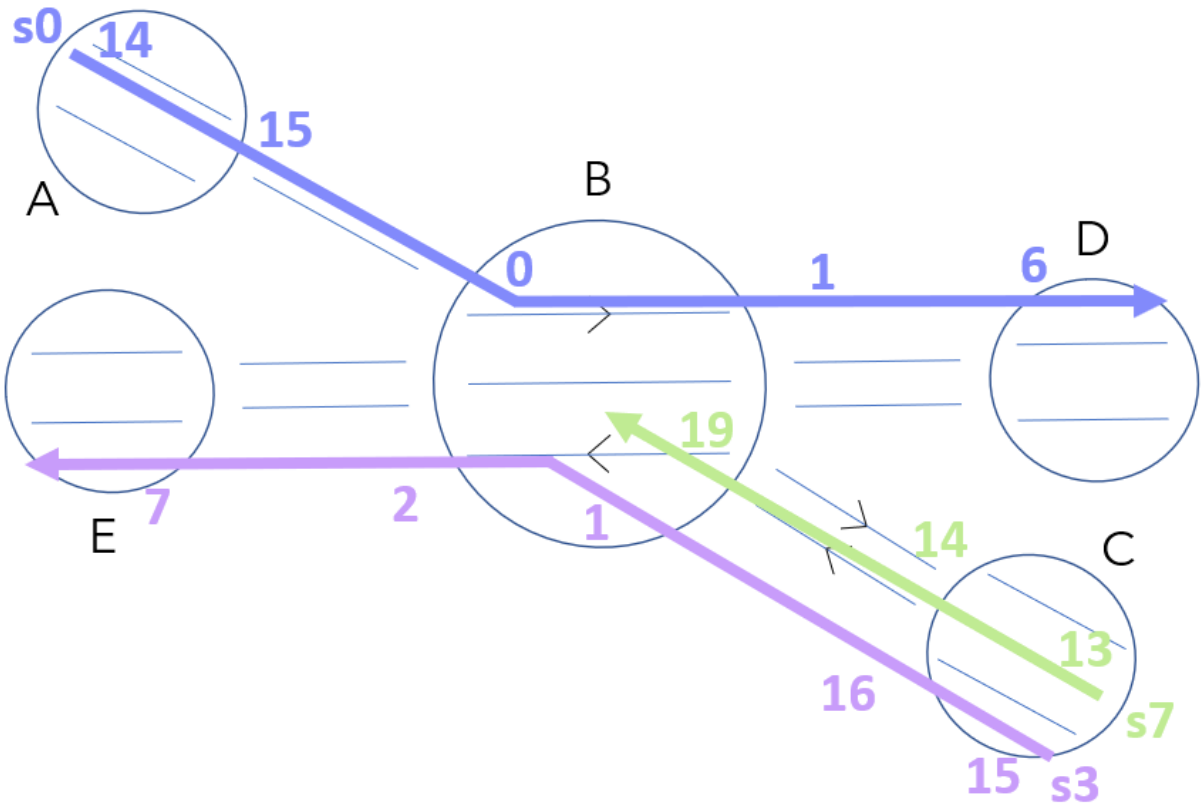


Figure 4.6: After the local move of the first iteration, slot s0 is still involved in crossing conflicts between sections B, B – D and B – C.

Because slot s0 is still involved in conflicts after the application of a local move to this slot, we increase the multiplicative coefficient associated to these conflicts. The consequences of this action is a degradation of the conflict evaluation value of the timetable, and in particular for slot s0 but also for slots s3 and s7 that are involved in the discussed crossing conflicts as well. The goal of this multiplicative coefficient increase is to give more focus on the identified conflicts in order to select future local moves that should be more likely to solve them in the next iterations. Table 4.10 presents the updated heuristic state at the end of the first iteration of the procedure. We observe that the tabu countdown of slot s0 is updated to value 1. Therefore, it would have been impossible to perform a local move on slot s0 if it has been involved in the maximum amount of conflicts. At each new iteration of the procedure, the strictly positive tabu countdowns will be decremented. We also increased the tabu duration of slot s0, thus the tabu countdown of this slot will be updated to value 2 the next time a local move will be applied on it. Among the slots for which the tabu countdown equals 0, those involved in the maximal amount of conflicts

Table 4.9: Slot s0 still involved in 2 crossing conflicts between sections B, B − D and B − C.

Infra. element	Time / Slot	0	1	2	3	4		19
Δ_{cro} B|B − D /B − C	s0							
	s3							
	s7							
	$u_{\text{cro[B to B−D]}}$	0	**1**	**1**	1	1		0
	$u_{\text{cro[B−C to B]}}$	1	**2**	**2**	1	1		1
	$v_{\text{B\|B−D/B−C}}$	0	**1**	**1**	0	0		0

are slots s3 and s7. Again, we select a slot arbitrarily between these two, in this case we select s3. The starting time of slot s3 will then be modified at the next iteration in order to attempt improving the conflict evaluation value of the timetable.

Table 4.10: Heuristic state after the first iteration, slot s3 is involved in the maximal number of conflicts.

Slots	s0	s1	s2	s3	s4	s5	s6	s7
Starting time	14	17	4	15	13	11	6	13
Tabu countdown	1	0	0	0	0	0	0	0
Tabu duration	2	1	1	1	1	1	1	1
Conflicts involved in	6	2	3	**9**	0	0	1	9

Iteration 2 and tracks assignment. We repeat instructions executed in the last iteration from the updated heuristic state presented in Table 4.10. The updated starting time of slot s3 is 0 instead of 15. As a result, there is no more conflict in the timetable and Table 4.11 presents the last heuristic state, obtained after the second iteration. At this point of the procedure, a conflict-free timetable has been found without using tracks assignment variables explicitly. Hence, we are now able to launch the tracks assignment module that consists to solve Formulation (4.1) while fixing the starting time variables provided by Table 4.11. A feasible solution returned by solving this problem by an off-the-shelf solver correspond tho the timetable with mesoscopic tracks assignment presented in Figure 3.6.

Table 4.11: Heuristic state after the second iteration, no conflicts remain.

Slots	s0	s1	s2	s3	s4	s5	s6	s7
Starting time	14	17	4	0	13	11	6	13
Tabu countdown	0	0	0	1	0	0	0	0
Tabu duration	2	1	1	2	1	1	1	1
Conflicts involved in	0	0	0	0	0	0	0	0

4.2.3 Experimental results

We present the performance of our tabu search heuristic in solving the Savoie area instance with the largest number of slots presented in Section 3.3.3. The heuristic is executed in two distinct implementations in C++. The first implementation uses the IBM Concert Technology and CPLEX 12.10 to execute the local moves, as briefly commented at the beginning of Section 4.2.2. The second implementation uses dedicated algorithms to compute each local move and tabu parameter updates. In this second implementation, CPLEX 12.10 is used for the tracks assignment when a conflict-free timetable with respect to the conflict evaluation function has been found. The experiments are performed on a computer running under Windows 10, 64 bits with a i5-8265U CPU at 1.60-1.80 GHz and 24 Go of RAM. Table 4.12 summarises the performance of the two implementations. We resolved conflicts of 20 randomly initialised timetables.

Table 4.12: Performance of the heuristic to find a conflict-free solution for the decision problem on the second Savoie area instance with 20 random initial solutions and using the constructive heuristic.

Tabu search		Using CPLEX 12.10		Using dedicated algorithms	
	Initial conflicts	#iterations	time/it (**min**)	#iterations	time/it (**sec**)
Average on 20 random initial solutions	≈ 468	≈ 40	> 5	24	< 4
Using the constructive heuristic	28			14	

It turns out that several minutes to compute each local move while our dedicated algorithms are able to compute a local move in few seconds only. Indeed, the linearised max and min operators of formulas described in Section 4.2.1.3 might be challenging to be solved to optimality when asking the solver to modify the starting time of a slot. Also, we remark that using our algorithms, less iterations of the procedure are needed to find conflict-free solutions. This can be explained because the Savoie area instances have a

period of one hour and a time granularity of half a minute, resulting in 120 time steps. Therefore, several locally optimal solutions could exist for a given local move, and it may appear that the two implementations do not find the same optimal solution, resulting in significantly different executions in the next iterations. We tried to start the tabu framework from a solution obtained by our constructive heuristic, having less conflicts in average than the randomly initialised solutions. We observe that using the constructive heuristic initial solution, less iterations of the tabu framework are needed in average to find a conflict-free solution than when starting from a randomised solution.

Note that we slightly modified the conflict evaluation function by creating additional artificial trigerrable conflicts before executing the experiments presented above. We indeed had to face some situations where timetables were evaluated with no conflicts, but for which the tracks assignment was not possible. We analysed these issues and they resulted in more extended crossing conflicts that are not yet captured in our model and appearing notably around Saint-André-le-Gaz, Aix-les-Bains and Montmélian stations. We discuss and treat these extended constraints in Section 5.1.3.1.

4.3 Conclusion

In this chapter, we presented an Integer Linear Programming formulation inspired from the literature to solve the periodic Train timetabling with mesoscopic tracks assignment problem. In order to solve larger instances, we decomposed the problem by deciding first the slot starting times and then the tracks assignment. To do so, we introduced a conflict evaluation function able to detect most of the conflicts between slots without using tracks assignment knowledge explicitly. The conflict resolution method of Reisch et al. (2021) [152] inspired the solution approaches that we proposed to address the problem. Our tabu search framework using dedicated algorithms is able to solve our largest Savoie area instance in about two minutes.

The present work is open to several future improvements that we summarise. First, it could be investigated how we can extend valid inequalities of the literature presented in Section 2.2.1 to the case of periodic timetabling problem with mesoscopic tracks assignment in order to strengthen Formulation (4.1). Second, our variables and constraints generation method could be improved, for example by banning conflicts as proposed in Fuchs et al. (2022) [78] and by conceiving procedures that generate relevant new starting time values for slots. Third, our methods should be extended to take into account headways that can variate depending on train orderings and on traversed infrastructure elements. Fourth, we would like to refine our conflicts evaluation in order to propose more

accurate conflict-free timetables for which a tracks assignment is effectively obtained. We would like to solve more challenging instances on complex topologies for which a tracks assignment heuristic could be necessary. Finally, we want to investigate a wider range of local moves in our tabu framework in order to obtain conflict-free schedule in less iterations and computation time. In particular, we consider a future local move consisting in shifting the starting times of several slots at once. We hope that our contributions give new perspectives to railway researchers and that we will extend the present work as proposed in the future.

4.A List of Symbols

The Table 4.13 summarises symbols and notations used in this chapter.

Table 4.13: Signification of symbols and notations used in Chapter 4.

Symbol	Meaning
T	Time period
SI	Service Intention, set of bidirectional lines to be timetabled within period T
$S = S^l \cup S^t$	Set of sections, partitioned in the set of line sections and the set of stations
K_s	Set of mesoscopic tracks of section s
$D_k \subseteq \{+,-\}, \mathrm{d}()$	Set of possible running directions on track k and running direction of a slot operand, respectively
$R = R^+ \cup R^-$	Set of mesoscopic routes, partitioned by direction
Z	Set of zones, a partition of S
$< z_1, z_2 >$	A bizone containing mesoscopic routes in any direction between z_1 and z_2
I, J_i and J_{i-}	Set of slots, set of operations of slot i and set of operations of slot i except the last one, respectively
$o_{i,j}, p_{i,j}$, $s_{i,j}$ and $t_{i,j}$	Operation j of slot i, its known duration, the section on which it is processed and its starting time, respectively
$\lvert\ \rvert$	Cardinal of a set operand inside the symbol
$K_{i,j} = \{1, \ldots, \lvert K_{s_{i,j}} \rvert\}$	Set of alternative tracks for operation $o_{i,j}$
$x_{i,j,k}$	Assignment of track k for operation $o_{i,j}$
$\Delta_{\mathrm{spa}}, \Delta_{\mathrm{reu}}$ and Δ_{cro}	Spacing, reuse and crossing safety headways, respectively
$z_{i,i',j,j'}$	Boolean variable equalling 0 if $t_{i,j} \leq t_{i',j'}$ and 1 otherwise
$y_{s,i,i',k}$	Boolean variable equalling 1 if ($x_{i,j,k}$ AND $x_{i',j',k}$) equals 1, and 0 otherwise
$y^{cro}_{i,i',j,j',k_1,k_2,k_3,k_4}$	Boolean variable equalling 1 if routes (k_1, k_2) and (k_3, k_4) between assigned tracks x_{i,j,k_1}, $x_{i,j+1,k_2}$, x_{i',j',k_3} and $x_{i',j'+1,k_4}$ are incompatible
M	Arbitrarily large value to linearise disjunctive constraints using so-called Big-M forms
$\overrightarrow{u}_{\mathrm{spa}[s]}$	Use variables corresponding to spacing occupations in the downstream direction on section s
$\overleftarrow{u}_{\mathrm{spa}[s]}$	Use variables corresponding to spacing occupations in the upstream direction on section s
$\overrightarrow{u}_{\mathrm{reu}[s]}$	Use variables corresponding to reuse occupations in the downstream direction on section s
$\overleftarrow{u}_{\mathrm{reu}[s]}$	Use variables corresponding to reuse occupations in the upstream direction on section s
$u_{\mathrm{cro}[s \text{ to } s']}$	Use variables corresponding to crossing occupations from section s to section s'
Cap_s	Global tracks capacity at section s, equalling $\lvert K_s \rvert$
$\overleftrightarrow{Cap_s}$	Number of tracks of section s that can be run in both downstream and upstream directions
$\overrightarrow{Cap_s}$	Number of tracks of section s that can be run in downstream direction only
$\overleftarrow{Cap_s}$	Number of tracks of section s that can be run in upstream direction only
v_s	Conflicts detected at section s
sim_s	Expression defined by $\overrightarrow{u}_{\mathrm{reu}[s]} * \overleftarrow{u}_{\mathrm{reu}[s]}$
v_a	Conflicts detected in the potential crossing area a where mesoscopic routes connect sections of a bizone

Chapter 5

Constraint Programming approach

In this chapter, we present approaches based on Constraint Programming to solve the rolling stock turnover times optimisation problem stated in Section 3.2.2. In Section 5.1, we present a constraint-based model formulation and enhancements such as extended crossing constraints, filtering algorithms, lower bounds, and a constraints strengthening procedure. A Branch-and-Check framework to solve the problem is described in Section 5.2. Finally, Section 5.3 reports experimental results on Savoie area instances centred around Chambéry station presented in Section 3.3.3. The decision problem of Section 3.2.1 is solved in few seconds with our Constraint Programming approach, a result which has motivated the extension to the optimisation problem presented in this chapter.

Contents

5.1 Modelling the problem as a Constraint Optimisation Problem

We model the problem described in Section 3.2.2 as a Constraint Optimisation Problem using interval variables to represent operations of slots. Interval variables are defined by start and end times, and deal with a presence status that can either be *present*, *absent* or *unfixed*. In Section 5.1.1, we present how periodicity is tackled at the level of one operation of a slot. Then, the model is described in Section 5.1.2, showing how the presence status of interval variables is a natural way to represent tracks assignment. Section 5.1.3 details model enhancements including extended crossing constraints, dedicated filtering algorithms, lower bounds and a strengthening constraints procedure.

5.1.1 Periodic operations

Each slot of the SI is composed by operations to be assigned on mesoscopic tracks, given that operation j of slot i has a known duration $p_{i,j}, i \in I, j \in J_i$. Since the timetable to be produced is periodic, it is frequent that operations of a same slot are scheduled within more than one period. Even considering only one operation, it may occur that $t_{i,j} + p_{i,j} > T$ so the operation itself straddles to consecutive periods. In such cases and assuming $p_{i,j} < T$, then the end time of $o_{i,j}$ is lower than its start time due to periodicity. However, the start time of an interval variable is conceptually lower or equal than its end time. In order to model operations generically while capturing periodicity when needed, we always model an operation $o_{i,j}$ relating two interval variables $\alpha_{i,j}$ and $\beta_{i,j}$ to create a seamless higher level object: the periodic operation. This structure is illustrated in Figure 5.1, where $p_{i,j}$ remains constant in all cases and $t_{i,j}$ varies.

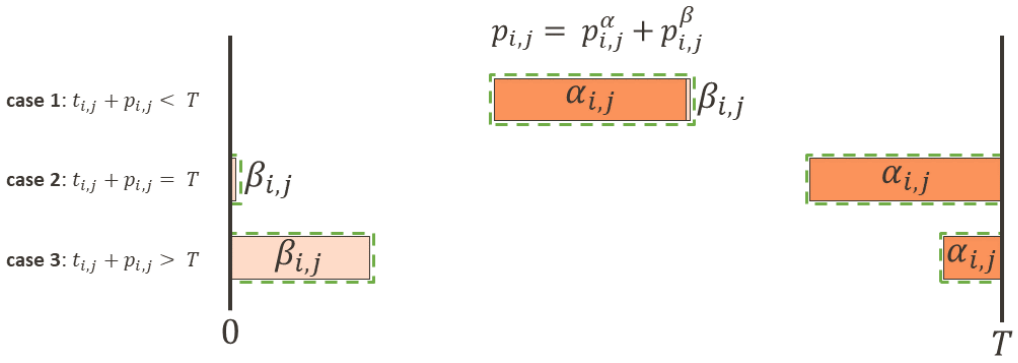


Figure 5.1: Periodic operation $o_{i,j}$ of constant duration $p_{i,j}$, highlighted as a green dashed rectangle in the 3 cases depending on the value of $t_{i,j}$.

Variables $p^{\alpha}_{i,j}$ and $p^{\beta}_{i,j}$ represent the length of interval variables α_{ij} and $\beta_{i,j}$, respectively. Equality $p_{i,j} = p^{\alpha}_{i,j} + p^{\beta}_{i,j}$ always holds. The idea is to maximise $p^{\alpha}_{i,j}$ such that the end time

$\alpha_{i,j}$ never exceeds T. As a result, $p^{\beta}_{i,j} = 0$ in the two firsts cases. As $\alpha_{i,j}$ is executed until T in cases 2 and 3, $\beta_{i,j}$ is then executed from time 0. The latter particularity facilitates the expression of precedence constraints between periodic operations. Constraints allowing this behaviour are derived in Section 5.1.2. From Figure 5.1, we remark that properties $\text{end}(\alpha_{i,j}) \geq p_{i,j}$ and $\text{end}(\beta_{i,j}) \neq T$ always hold.

5.1.2 Model formulation

We use *optional* interval variables $\alpha_{i,j,k}$ and $\beta_{i,j,k}$ to represent the assignment of operation $o_{i,j}$ on one of the tracks $k \in K_{i,j}$ for all $i \in I$ and $j \in J$. The start and end times of the chosen *optional* interval variables are then synchronised with the start and end times of $\alpha_{i,j}$ and $\beta_{i,j}$. The constraint-based formulation of the periodic train timetabling with mesoscopic tracks assignment problem optimising the turnover times at terminus stations, reads:

$$\text{Minimise} \sum_{(i,i',s)\in L_s, s\in S^t} c^s_{i,i'}(q^s_{i,i'}) \tag{5.1a}$$

$$\text{s.t.: } q^s_{i,i'} = (t_{i',2} - t_{i,n_i} + T) \ \% \ T, \forall (i,i',s) \in L_s, s \in S^t \tag{5.1b}$$

$$\text{pr}(\alpha_{i,n_i,k}) = \text{pr}(\alpha_{i',1,k}), \forall (i,i',s) \in L_s, s \in S^t, \forall k \in K_{i,n_i} \tag{5.1c}$$

$$\text{end}(\alpha_{i,j}) = \min(T, t_{i,j} + p_{i,j}), \forall i \in I, \forall j \in J_i \tag{5.1d}$$

$$\text{start}(\beta_{i,j}) = \text{end}(\alpha_{i,j}) \ \% \ T, \forall i \in I, \forall j \in J_i \tag{5.1e}$$

$$\text{end}(\beta_{i,j}) = (t_{i,j} + p_{i,j}) \ \% \ T, \forall i \in I, \forall j \in J_i \tag{5.1f}$$

$$\text{alternative}(\alpha_{i,j}, \{\alpha_{i,j,1}, \ldots, \alpha_{i,j,|K_{s_{i,j}}|}\}), \forall i \in I, \forall j \in J_i \tag{5.1g}$$

$$\text{synchronise}(\beta_{i,j}, \{\beta_{i,j,1}, \ldots, \beta_{i,j,|K_{s_{i,j}}|}\}), \forall i \in I, \forall j \in J_i \tag{5.1h}$$

$$\text{pr}(\alpha_{i,j,k}) = \text{pr}(\beta_{i,j,k}), \forall i \in I, \forall j \in J_i, \forall k \in K_{i,j} \tag{5.1i}$$

$$\text{startAtEnd}(\alpha_{i,j+1}, \beta_{i,j}), \forall i \in I, \forall j \in J_{i-} \tag{5.1j}$$

$$\text{d}(i) \notin D_k \Rightarrow \text{setAbsent}(\alpha_{i,j,k}), \forall i \in I, \forall j \in J_i, \forall k \in K_{i,j} \tag{5.1k}$$

$$\mathrm{pr}(\alpha_{i,j,k}) \Rightarrow !\mathrm{pr}(\alpha_{i,j+1,k'}), \forall i \in I, \forall j \in J_{i-}, \forall k \in K_{i,j}, \forall k' \in K_{i,j+1}, (k,k') \notin R \tag{5.1l}$$

$$!\mathrm{pr}(\alpha_{i,j,k}) \;||\; !\mathrm{pr}(\alpha_{i',j',k}) \;||\; ((t_{i,j} - t_{i',j'} + T) \;\%\; T \geq \Delta_{\mathrm{spa}} \;\&\&\; (t_{i',j'} - t_{i,j} + T) \;\%\; T \geq \Delta_{\mathrm{spa}}),$$
$$\forall i \neq i' \in I, \mathrm{d}(i) = \mathrm{d}(i'), \forall j \in J_i, \forall j' \in J_{i'}, s_{i,j} = s_{i',j'}, s_{i,j} \in S^l, \forall k \in K_{i,j} \tag{5.1m}$$

$$!\mathrm{pr}(\alpha_{i,j,k}) \;||\; !\mathrm{pr}(\alpha_{i',j',k}) \;||\; ((t_{i,j} - t_{i',j'} + T) \;\%\; T \geq p_{i',j'} + \Delta_{\mathrm{reu}} \;\&\&\; (t_{i',j'} - t_{i,j} + T) \;\%\; T \geq p_{i,j} + \Delta_{\mathrm{reu}}),$$
$$\forall i \neq i' \in I, \forall j \in J_i, \forall j' \in J_{i'}, s_{i,j} = s_{i',j'}, s_{i,j} \in S^t \;||\; \mathrm{d}(i) \neq \mathrm{d}(i'), (i,i',s_{i,j}) \notin L_s, \forall k \in K_{i,j} \tag{5.1n}$$

$$!\mathrm{pr}(\alpha_{i,j,k}) \;||\; !\mathrm{pr}(\alpha_{i,j+1,k'}) \;||\; !\mathrm{pr}(\alpha_{i',j',k''}) \;||\; !\mathrm{pr}(\alpha_{i',j'+1,k'''}) \;||\; \begin{matrix}(t_{i,j+1} - t_{i',j'+1} + T) \;\%\; T \geq \Delta_{\mathrm{cro}} \\ (t_{i',j'+1} - t_{i,j+1} + T) \;\%\; T \geq \Delta_{\mathrm{cro}}\end{matrix},$$
$$\forall i \neq i' \in I, \forall j \in J_{i-}, \forall j' \in J_{i'-1}, \forall k \in K_{i,j}, \forall k' \in K_{i,j+1}, \forall k'' \in K_{i',j'}, \forall k''' \in K_{i',j'+1}, \tag{5.1o}$$
$$r_1 = (k,k'), r_2 = (k'',k''') \in R, r_1 \text{ and } r_2 \text{ not compatible}$$

var.: $\alpha_{i,j}$ and $\beta_{i,j}$ are *present* interval variables, $\forall i \in I, \forall j \in J_i$ (5.1p)

$$t_{i,j} = \mathrm{start}(\alpha_{i,j}) \in \{0,\ldots,T-1\}, \mathrm{end}(\alpha_{i,j}) \in \{p_{i,j},\ldots,T\}, \forall i \in I, \forall j \in J_i \tag{5.1q}$$

$$\mathrm{start}(\beta_{i,j}) \in \{0,\ldots,T-1\}, \mathrm{end}(\beta_{i,j}) \in \{0,\ldots,T-1\}, \forall i \in I, \forall j \in J_i \tag{5.1r}$$

$\alpha_{i,j,k}$ and $\beta_{i,j,k}$ are *optional* interval variables, $\forall i \in I, \forall j \in J_i, \forall k \in K_{i,j}$ (5.1s)

The objective function (5.1a) minimises the total cost induced by the turnover times at terminus stations. Symbol % stands for the modulo operation between two positive expressions on its sides. Constraints (5.1b) define expressions $q^s_{i,i'}$ of turnover times at terminus station s between the start of the last operation of slot i (i.e. arriving at station s) and the start of the second operation of slot i' (i.e. departing after dwelling at station s) for each (i,i',s) in L_s, s in S^t. Constraints (5.1c) make the last operation of an arrival slot at its terminus station s having the same presence status on track k as that of the first operation of the linked departing slot, for all tracks of station s. Indeed, symbol pr() is a constraint that equals true if the interval variable in argument is set *present* and equals false if the interval variable in argument is set *absent*. Note that we make use of dedicated filtering algorithm to apply two additional families of constraints on turnover times at terminus stations. First, we constrain $q^s_{i,i'}$ to be greater or equal than μ_{feas}. Second, we prevent operations of other slots from occupying the same track at station s as that of slots i and i' while their rolling stock dwells on it.

Constraints (5.1d) to (5.1f) link the interval variables of each operation $o_{i,j}$ so that

they implement the periodic behaviour described in Section 5.1.1.

Constraints (5.1g) to (5.1i) enforce the assignment of a unique track for all operations $o_{i,j}$ among the tracks of $s_{i,j}$. More precisely, constraints (5.1g) ensure that only one interval variable in $\{\alpha_{i,j,1}, \ldots, \alpha_{i,j,|K_{s_{i,j}}|}\}$ is present while the other are absent, and that the chosen interval variable has the same start and end times as that of $\alpha_{i,j}$. Constraints (5.1h) make all present interval variables in $\{\beta_{i,j,1}, \ldots, \beta_{i,j,|K_{s_{i,j}}|}\}$ having the same start and end times as that of $\beta_{i,j}$. Finally, constraints (5.1i) make that the only present interval variable in $\{\beta_{i,j,1}, \ldots, \beta_{i,j,|K_{s_{i,j}}|}\}$ corresponds to the interval variable in $\{\alpha_{i,j,1}, \ldots, \alpha_{i,j,|K_{s_{i,j}}|}\}$ selected in constraint (5.1g).

Constraints (5.1j) are the precedence constraints between the periodic operations of the slots. Constraints (5.1k) and (5.1l) prevent slots from running on tracks in a not compatible direction and from running on successive tracks between which no route exists, respectively.

Constraints (5.1m) to (5.1o) are operational constraints that implement capacity consumption on the infrastructure using safety headway introduced in Section 3.1.1.3, with symbol ! standing for the negation symbol || standing for the disjunction. More precisely, constraints (5.1m) are the spacing constraints ensuring that any two slots can access a same line track on the same direction with a time difference greater or equal to Δ_{spa}. Constraints (5.1n) are the reuse constraints applying on station tracks for any two slots of any direction and on line tracks for any two slots running in the opposite direction. In those conditions, reuse constraints allow a slot to occupy a track at least Δ_{reu} units of time after a previous slot has cleared the track. Note that reuse constraints do not apply on two slots linked at a terminus station and sharing the same rolling stock. Finally, constraints (5.1o) are the crossing constraints ensuring that any two slots are able to run on incompatible routes if and only if they access those routes with a time difference greater than or equal to Δ_{cro}. The presence of interval variables on the four tracks forming the incompatible routes activates this constraints. Note that we can also detect incompatibility between two routes of a bizone if they share a common track, thus requiring the presence of only two interval variables to activate the constraint.

5.1.3 Model enhancements

In this section, we describe techniques to improve the model formulated in Section 5.1.2. Among those enhancements, extended crossing constraints that are detected at an upper level than constraints (5.1o) are presented in Section 5.1.3.1. Dedicated filtering algorithms implementing feasibility-led dominance rules to deal with time periodicity are

detailed in Section 5.1.3.2. Lower bounds that speed-up the optimality proof are explained in Section 5.1.3.3. Finally, a constraint strengthening procedure inspired from Kümmling et al. (2016) is presented in Section 5.1.3.4.

5.1.3.1 Extended crossing constraints

We saw in the model and in Figure 3.1 that crossing constraints are activated whenever two slots take incompatible routes of a same bizone. However, in some particular infrastructure configurations it is possible to detect induced routes incompatibility at an upper level because of sections in which slots circulate.

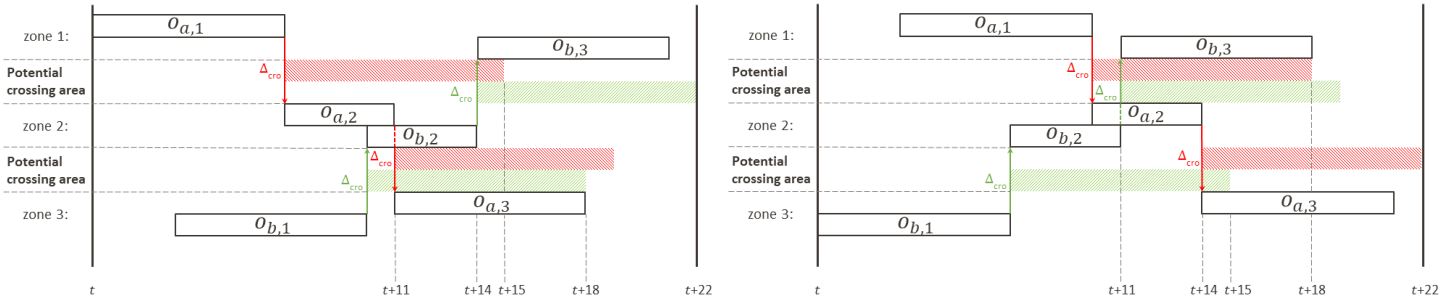


Figure 5.2: Detection of infeasible schedules without tracks assignment by using extended crossing constraints. We know that at least one route incompatibility will occur on bizones around Saint-André-le-Gaz station (zone 2) between slot a to Grenoble and slot b from Chambéry, for any tracks assignment at Saint-André-le-Gaz station. Horizontal axis represents relative time from a time t. Periodicity still holds, thus the end of period T could either occur at any time in $\{t, \ldots, t+22\}$ or outside of this scope. **Left part:** $o_{a,1}$ starts at time t and $o_{b,1}$ starts relatively as late as possible such that crossing headway times of the two slots still overlap over at least one unit of time both on bizones $< 1, 2 >$ and $< 2, 3 >$. As there are not simultaneous compatible routes on both bizones for those slots, then this case leads to an infeasible schedule or an opportunity to reduce starting times domains without using tracks assignment knowledge. **Right part**: The same logic applies when scheduling slot b first and relatively delaying slot a as much as possible such that the conflict is still present. Note that 5 other intermediary cases leading to infeasible schedules between the two illustrated extreme cases are also detected.

Let us consider in Figure 5.2 a downstream slot a from Lyon to Grenoble and an upstream slot b from Chambéry to Lyon, both dwelling to during 4 time units at Saint-André-le-Gaz station and running 7 time units in line sections. Moreover, we will take 8 time units as a value for Δ_{cro}. Since line sections 1 and 3 are one-way double tracks and line section 4 has a single track, we deduce that slot a runs on track 1 at in line section 1 and on track 7 in line section 3, and that slot b runs on track 9 in section 4 and on track 2 in section 1. At station 2, a choice remains to be made among tracks 3 to 5 for slot a and among 3 to 6 for slot b. Without making this choice and considering the crossing constraints of the model described in Section 5.1.2, no deduction can be made on starting times of operations of slot b even if starting times of operations of slot a are fixed.

We remark there are compatible routes in bizone $< 1, 2 >$ (e.g. routes (1,3) and (5,2)) and in bizone $< 2, 3 >$ (e.g. routes (5,7) and (9,4)). However, those routes cannot be run by slots a and b because downstream routes (1,3) and (5,7) don't share the same track in station 2, and because upstream routes (9,4) and (5,2) neither. In fact, it can be seen in Figure 3.1 that if any allowed tracks are chosen at station 2 for slots a and b, then this would always imply routes incompatibility at least on one of bizones $< 1, 2 >$ and $< 2, 3 >$, or even on both bizones. We can take advantage of this knowledge to perform domain reductions on the starting time variables of operations of slots a and b. Figure 5.2 illustrates this phenomenon. Exploiting the fact that no tracks assignment for slots a and b can ensure routes compatibility on both sides of zone 2, then fixing the start time of $o_{a,1}$ e.g. at time 0 allows to forbid $o_{b,1}$ from starting at times in $\{T-3, T-2, T-1, 0, 1, 2, 3\}$ without using information on tracks assignment in zone 2.

5.1.3.2 Improving propagation using dedicated filtering algorithms

With the model of Section 5.1.2 at hand, we can use a CSP solver to solve instances of the problem described in Section 3.2.2. We can also verify propagation quality of specific constraints of the model, and to improve the propagation process conceiving dedicated filtering algorithms when we observe that more deductions should be performed. In the following, we will describe a filtering algorithm dedicated to a family of constraints to manage propagation issues due to time periodicity. Then, we will mention other families of constraints that we either improve or create using additional filtering algorithms.

Periodic operations constraints We now focus on the propagation behaviour of start and end times of periodic operations of a slot when they are scheduled near the end of period T. More precisely, we will resolve a situation where the domain of the start time of $\alpha_{i,j}$ (resp. the domain of the end time of $\beta_{i,j}$) is not fully propagated to the domain of the end time of $\beta_{i,j}$ (resp. to the domain of the start time of $\alpha_{i,j}$) so the propagation is not performed on the successive (resp. preceding) operations of the slot. We consider an operation $o_{i,j}$ of known duration $p_{i,j} = 3$ and for which the start time is constrained such that $t_{i,j} \in \{T-7, \ldots, T-2\}$ so we are not able to decide in which of the 3 cases of Figure 5.1 it corresponds. Then we know that $\text{start}(\alpha_{i,j}) \in \{T-7, \ldots, T-2\}$, and we look at the effect of the propagation of constraints (5.1d) to (5.1f) on the domains of $\text{end}(\alpha_{i,j})$, $\text{start}(\beta_{i,j})$ and $\text{end}(\beta_{i,j})$. We expect that the size of the domain of variable $\text{end}(\beta_{i,j})$, domain which is shared with variable $\text{start}(\alpha_{i,j+1})$ after application of precedence constraints (5.1j), should be equal to the size of the domain of variable $\text{start}(\alpha_{i,j})$. Table 5.1 reports the state of the domains that we expect to obtain for those variables and the state of the domains that we obtain in practice using the propagation of a solver for

which the inference level has been parameterised to *Extended* value. We observe that in practice, no values have been removed from the domains of variables start($\beta_{i,j}$) and end($\beta_{i,j}$). Since values 0 and $T-1$ are elements of the expected propagated domains of those variables, we can deduce that the solver achieves *bound-consistency*, i.e. if two values are consistent with respect to the constraints, then all values in between are also conserved. This propagation mode is performed with an economic computational effort, however in this case there is a loss of knowledge that implies no propagation on time variables of next operation $o_{i,j+1}$. Algorithm 1 achieves *arc-consistency* for the periodic operations constraints, i.e. expected propagated domains of Table 5.1 are obtained if this algorithm is used instead or in complement of constraints (5.1d) to (5.1f). Note that the presented phenomenon is also observed when we know the domain of the end time of a periodic operation that we try to propagate to its start time. In lines 4 to 17, Algorithm 1 performs forward propagation as follows. For each value in the domain of the start time of $\alpha_{i,j}$, we compute the corresponding values of the end time of $\alpha_{i,j}$ (nDomEα), of the start time of $\beta_{i,j}$ (nDomSβ) and of the end time of $\beta_{i,j}$ (nDomEβ). Then, we update the actual domains of the start time of $\alpha_{i,j}$ and of the start and end times of $\beta_{i,j}$ by applying the intersection with sets nDomEα, nDomSβ and nDomEβ respectively. Backward propagation is performed in the same fashion in lines 20 to 33 by exploring the domain of the end time of $\beta_{i,j}$.

Table 5.1: Expected and practical propagation on domains of variables end($\alpha_{i,j}$), start($\beta_{i,j}$) and end($\beta_{i,j}$) using constraints (5.1d) to (5.1f) given $p_{i,j}=3$ and start($\alpha_{i,j}$) $\in \{T-7,\ldots,T-2\}$.

Variable		end($\alpha_{i,j}$)	start($\beta_{i,j}$)	end($\beta_{i,j}$)
Propagated domain	Expected	$\{T-4,\ldots,T\}$	$\{0\}\cup\{T-4,\ldots,T-1\}$	$\{0,1\}\cup\{T-4,\ldots,T-1\}$
	In practice	$\{T-4,\ldots,T\}$	$\{0,\ldots,T-1\}$	$\{0,\ldots,T-1\}$

Operational headway constraints Operational headway constraints (5.1m) to (5.1o) ensure that no two slots are using the same infrastructure resource too close in time. They apply safety headways Δ_{spa}, Δ_{reu} and Δ_{cro} described in Section 3.1.1.3 between train circulations. Satisfying these constraints prevent any train from meeting a restrictive signal during its journey. When propagating these constraints, we observed again, as detailed in the last paragraph, that the propagation performs *bound-consistency*. That is, if we consider 2 downstream slots assigned on the same line track, and if we timetable the start time of the operation of one of those slots on this track, then values will be removed by propagation of a spacing constraint (5.1m) from the domain of the start time of the operation of the other slot only if the inconsistent values are located at the extremities of the domain. We observe the same phenomenon for the propagation of reuse constraints (5.1n) and crossing constraints (5.1o). We then specify how we can systematically remove

Algorithm 1: Maintain *arc-consistency* on the periodic operations constraints (5.1d) to (5.1f)

1 **Input:** $\text{Dom}(\text{start}(\alpha_{i,j}))$, $\text{Dom}(\text{end}(\alpha_{i,j}))$, $\text{Dom}(\text{start}(\beta_{i,j}))$ and $\text{Dom}(\text{end}(\beta_{i,j}))$
2 **Output:** updated $\text{Dom}(\text{start}(\alpha_{i,j}))$, $\text{Dom}(\text{end}(\alpha_{i,j}))$, $\text{Dom}(\text{start}(\beta_{i,j}))$ and $\text{Dom}(\text{end}(\beta_{i,j}))$
3 **Initialisation:**
$\text{nDomS}\alpha = \varnothing$; $\text{nDomE}\alpha = \varnothing$; $\text{nDomS}\beta = \varnothing$; $\text{nDomE}\beta = \varnothing$; $\text{geqT} = false$; $\text{lt0} = false$;
4 **for** $v \in \text{Dom}(\text{start}(\alpha_{i,j}))$ **do**
5 **if** $v + p_{i,j} < T$ **then**
6 $\text{nDomE}\alpha := \text{nDomE}\alpha \cup \{v + p_{i,j}\}$;
7 $\text{nDomS}\beta := \text{nDomS}\beta \cup \{v + p_{i,j}\}$;
8 $\text{nDomE}\beta := \text{nDomE}\beta \cup \{v + p_{i,j}\}$;
9 **else**
10 **if** !geqT **then**
11 $\text{nDomE}\alpha := \text{nDomE}\alpha \cup \{T\}$;
12 $\text{nDomS}\beta := \text{nDomS}\beta \cup \{0\}$;
13 $\text{geqT} = true$;
14 $\text{nDomE}\beta := \text{nDomE}\beta \cup \{(v + p_{i,j}) \ \% \ T\}$;
15 $\text{Dom}(\text{end}(\alpha_{i,j})) := \text{Dom}(\text{end}(\alpha_{i,j})) \cap \text{nDomE}\alpha$;
16 $\text{Dom}(\text{start}(\beta_{i,j})) := \text{Dom}(\text{start}(\beta_{i,j})) \cap \text{nDomS}\beta$;
17 $\text{Dom}(\text{end}(\beta_{i,j})) := \text{Dom}(\text{end}(\beta_{i,j})) \cap \text{nDomE}\beta$;
18 $\text{nDomE}\alpha = \varnothing$;
19 $\text{nDomS}\beta = \varnothing$;
20 **for** $v \in \text{Dom}(\text{end}(\beta_{i,j}))$ **do**
21 **if** $v - p_{i,j} \geq 0$ **then**
22 $\text{nDomS}\alpha := \text{nDomS}\alpha \cup \{v - p_{i,j}\}$;
23 $\text{nDomE}\alpha := \text{nDomE}\alpha \cup \{v\}$;
24 $\text{nDomS}\beta := \text{nDomS}\beta \cup \{v\}$;
25 **else**
26 **if** !lt0 **then**
27 $\text{nDomE}\alpha := \text{nDomE}\alpha \cup \{T\}$;
28 $\text{nDomS}\beta := \text{nDomS}\beta \cup \{0\}$;
29 $\text{lt0} = true$;
30 $\text{nDomS}\alpha := \text{nDomS}\alpha \cup \{(v - p_{i,j} + T) \ \% \ T\}$;
31 $\text{Dom}(\text{start}(\alpha_{i,j})) := \text{Dom}(\text{start}(\alpha_{i,j})) \cap \text{nDomS}\alpha$;
32 $\text{Dom}(\text{end}(\alpha_{i,j})) := \text{Dom}(\text{end}(\alpha_{i,j})) \cap \text{nDomE}\alpha$;
33 $\text{Dom}(\text{start}(\beta_{i,j})) := \text{Dom}(\text{start}(\beta_{i,j})) \cap \text{nDomS}\beta$;

inconsistent values even in the middle of variable domains, maintaining an *arc-consistency* for operational constraints (5.1m) to (5.1o). In Figure 5.3, we describe a generic condition to detect inconsistent values in the domain of constrained start time variables depending on the specificities of constraints (5.1m) to (5.1o). We then implement an *arc-consistency* propagation in the filtering Algorithm 2 using the inconsistent values detection for constraints (5.1m) and (5.1o). Note that in lines 6, 14, 23 and 34 of Algorithm 2, we add another condition to also take into account the special case of the right part of Figure 5.3 where the start time of a constrained variable is fixed. Lines 4 to 20 are devoted to remove inconsistent values in the domain of the 2 constrained variables, since the presence of the interval variables activate the constraint. However, in lines 21 to 31 as well as in lines

32 to 42, one of the constrained variables is present but the other constrained still have an undetermined presence status. We then check if the latter variable has a support i.e. a value in its domain that is not inconsistent. Otherwise, we deduce that the presence of this latter variable would violated the constraint, thus we can set its status to *absent*. Note also that Algorithm 2 is slightly modified while keeping the same idea in the case of crossing constraints (5.1o): the presence on tracks of up to four interval variables is managed.

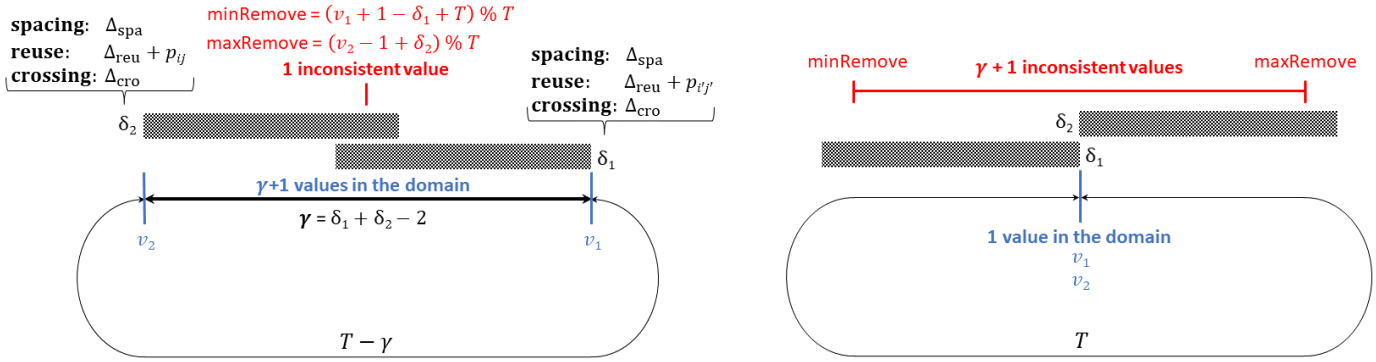


Figure 5.3: Improving the detection of inconsistent values in the domain of variables related to operational constraints (5.1m) to (5.1o). The spacing, reuse and crossing constraints have slightly different application contexts, but they all constrain the start time of two periodic operation variables. If the domain of one constrained variable have $\gamma + 1$ values or less and that two consecutive values v_1 and v_2 of the domain have at least a distance of $T - \gamma$, with $\gamma = \delta_1 + \delta_2 - 2$ and δ_1 and δ_2 depending on the family of constraints, then inconsistent values are detected and can be removed from the domain of the other constrained variable. **Left part:** This is an extreme case where the domain of a constrained variable has exactly $\gamma + 1$ values that are all cyclically contiguous so the extreme values of the domain v_1 and v_2 have a cyclic distance of $T - \gamma$. Then 1 inconsistent value is detected in the domain of the other constrained variable. **Right part:** When the start time of a constrained variable is fixed, then $\gamma + 1$ inconsistent value are detected in the domain of the other constrained variable. Many other cases exist between the 2 illustrated extreme cases. In particular, domains does not need to be contiguous. In the general case, inconsistent values are detectable whenever 2 cyclically consecutive (but not necessary contiguous) values v_1 and v_2 of the domain of a constrained variable have a distance at least equal to $T - \gamma$, what is possible only if the domain contains at most $\gamma + 1$ values.

Additional filtering algorithms The filtering algorithms described in the 2 previous paragraphs allow *arc-consistency* on constraints (5.1d) to (5.1f) and (5.1m) to (5.1o) to be maintained. Additionally, we implement filtering algorithms that have the following purposes not covered by the initial constraints of the model:

- Activate the extended crossing constraints presented in Section 5.1.3.1.
- Prevent any slot to be scheduled on a terminus station track while a rolling-stock of linked slots is dwelling on this track.

Algorithm 2: Sketch of maintain *arc-consistency* on the operational headway constraints (5.1m) to (5.1o)

Input: initial state of variables $\alpha_{i,j}, \alpha_{i',j'}, \alpha_{i,j,k}$ and $\alpha_{i',j',k'}$
Output: updated state of variables $\alpha_{i,j}, \alpha_{i',j'}, \alpha_{i,j,k}$ and $\alpha_{i',j',k'}$
Initialisation: minRemove $= 0$; maxRemove $= 0$;
if pr($\alpha_{i,j,k}$) && pr($\alpha_{i',j',k'}$) **then**
  if $|\text{Dom}(\text{start}(\alpha_{i,j}))| \leq \gamma + 1$ **then**
    if $\exists\ v_1, v_2 \in \text{Dom}(\text{start}(\alpha_{i,j})), (v_2 - v_1 + T)\ \%\ T \geq T - \gamma, v_1$ and v_2 cyclically consecutive **then**
      minRemove $= (v_1 + 1 - \delta_1 + T)\ \%\ T$;
      maxRemove $= (v_2 - 1 + \delta_2)\ \%\ T$;
      if minRemove $\leq$ maxRemove **then**
        $\text{Dom}(\text{start}(\alpha_{i',j'})) := \text{Dom}(\text{start}(\alpha_{i',j'})) - \{\text{minRemove}, \ldots, \text{maxRemove}\}$;
      else
        $\text{Dom}(\text{start}(\alpha_{i',j'})) := \text{Dom}(\text{start}(\alpha_{i',j'})) - (\{0, \ldots, \text{maxRemove}\} \cup \{\text{minRemove}, \ldots, T-1\})$;
  if $|\text{Dom}(\text{start}(\alpha_{i',j'}))| \leq \gamma + 1$ **then**
    if $\exists\ v'_1, v'_2 \in \text{Dom}(\text{start}(\alpha_{i',j'})), (v'_2 - v'_1 + T)\ \%\ T \geq T - \gamma, v'_1$ and v'_2 cyclically consecutive **then**
      minRemove $= (v'_1 + 1 - \delta_1 + T)\ \%\ T$;
      maxRemove $= (v'_2 - 1 + \delta_2)\ \%\ T$;
      if minRemove $\leq$ maxRemove **then**
        $\text{Dom}(\text{start}(\alpha_{i,j})) := \text{Dom}(\text{start}(\alpha_{i,j})) - \{\text{minRemove}, \ldots, \text{maxRemove}\}$;
      else
        $\text{Dom}(\text{start}(\alpha_{i,j})) := \text{Dom}(\text{start}(\alpha_{i,j})) - (\{0, \ldots, \text{maxRemove}\} \cup \{\text{minRemove}, \ldots, T-1\})$;
else if pr($\alpha_{i,j,k}$) && !abs($\alpha_{i',j',k'}$) **then**
  if $|\text{Dom}(\text{start}(\alpha_{i,j}))| \leq \gamma + 1$ **then**
    if $\exists\ v_1, v_2 \in \text{Dom}(\text{start}(\alpha_{i,j})), (v_2 - v_1 + T)\ \%\ T \geq T - \gamma, v_1$ and v_2 cyclically consecutive **then**
      minRemove $= (v_1 + 1 - \delta_1 + T)\ \%\ T$;
      maxRemove $= (v_2 - 1 + \delta_2)\ \%\ T$;
      if minRemove $\leq$ maxRemove **then**
        if $\nexists\ v' \in \text{Dom}(\text{start}(\alpha_{i',j'})), v' < \text{minRemove}\ ||\ v' > \text{maxRemove}$ **then**
          setAbsent($\alpha_{i',j',k'}$);
      else
        if $\nexists\ v' \in \text{Dom}(\text{start}(\alpha_{i',j'})), v' > \text{maxRemove}$ && $v' < \text{minRemove}$ **then**
          setAbsent($\alpha_{i',j',k'}$);
else if !abs($\alpha_{i,j,k}$) && pr($\alpha_{i',j',k'}$) **then**
  if $|\text{Dom}(\text{start}(\alpha_{i',j'}))| \leq \gamma + 1$ **then**
    if $\exists\ v'_1, v'_2 \in \text{Dom}(\text{start}(\alpha_{i',j'})), (v'_2 - v'_1 + T)\ \%\ T \geq T - \gamma, v'_1$ and v'_2 cyclically consecutive **then**
      minRemove $= (v'_1 + 1 - \delta_1 + T)\ \%\ T$;
      maxRemove $= (v'_2 - 1 + \delta_2)\ \%\ T$;
      if minRemove $\leq$ maxRemove **then**
        if $\nexists\ v \in \text{Dom}(\text{start}(\alpha_{i,j})), v < \text{minRemove}\ ||\ v > \text{maxRemove}$ **then**
          setAbsent($\alpha_{i,j,k}$);
      else
        if $\nexists\ v \in \text{Dom}(\text{start}(\alpha_{i,j})), v > \text{maxRemove}$ && $v < \text{minRemove}$ **then**
          setAbsent($\alpha_{i,j,k}$);

- Make the turnover time of any two slots linked at a terminus station at least equal to μ_{feas}.
- If a non proved optimal feasible timetable is found, its objective value is used as an upper bound to cut off partial schedules with a greater evaluated lower bound during the search of a potential better timetable.

5.1.3.3 Lower bounds

Proving optimality of a solution to an optimisation problem can be a tough task. Indeed, when solving such a combinatorial minimisation problem, it could be difficult to prove that no other variable affectations lead to a solution with better objective value. To cope with this issue, one can try to obtain high quality lower bounds, i.e. the closest as possible to the optimal objective value. If the lower bound computed for the global problem (at

route node, before branching decisions) is ever reached by the objective value of a feasible timetable, then this timetable is proved to be optimal.

In our context, the objective function to be minimised is a cost penalising turnover times at terminus stations if they are not robust, or worst if they do not respect minimal or maximal turnover time prescriptions of the Infrastructure Manager. Among different lower bounds that can be used, we experimented to solve the model of Section 5.1.2 while optimising the terms that compose the objective function (5.1a) one by one. Indeed, reaching optimality when optimising only one term is faster than optimising the whole objective function (5.1a), because the compensation effects of terms of the sum in the participation to the objective value then not occurs. The lower bound used in our approach is the sum of the optimal values of each turnover time cost function optimised separately.

We focus on a particular turnover time cost term $c^s_{i,i'}(q^s_{i,i'})$ of two given linked slots i and i' at a given terminus station s optimised separately. If the optimal value of this cost is strictly greater than 0, it means that there are no variable affectations satisfying all the constraints of the model while allowing a cost equal to 0 for this term of the objective function (5.1a). Moreover, the optimal cost of $c^s_{i,i'}(q^s_{i,i'})$ obtained when optimised separately can only increase when optimising the whole objective function (5.1a). That is, this value optimised separately participates as an additive term to the lower bound of the whole objective function.

5.1.3.4 Constraints strengthening procedure

Before solving our problem, we can attempt to make as much deductions as possible using propagation of the constraints of the model in order to reduce the search space. To this end, we will describe a constraints strengthening procedure inspired from approaches presented in Kümmling et al. (2016) [110].

In Kümmling et al. (2016) [110], authors tackle the periodic train timetabling with traffic assignment problem at a macroscopic scale of the infrastructure for long distance passenger trains of all Germany with constraint propagation techniques. In their problem, the duration of slot operations and transfer times are to be decided between minimal and maximal values. A preprocessing step consists in fixing the time of an event (equivalent to the start or the end time of an operation in our context) at 0 in a constraint graph composed of transport and transfer arcs between events. Then, they use constraint propagation in order to tighten minimal and maximal bounds of slot operation durations and of transfer times by detecting inconsistencies. The process is iterated resetting the domain of the events, fixing another event at time 0 and propagating the constraints while using

the previously tightened minimal and maximal bounds, until no new deduction can be made.

Algorithm 3: Constraints strengthening procedure

Input: model of Section 5.1.2, possible activation of model enhancements of Section 5.1.3.1, Section 5.1.3.2 and Section 5.1.3.3
Output: improved forbidden distances between variables $\text{start}(\alpha_{i,1}), \forall i \in I$
Initialisation: oldNbForbiddenDistances := 0; newNbForbiddenDistances := 0; forbiddenDistances = $\varnothing$; round = 1;
repeat
  oldNbForbiddenDistances := newNbForbiddenDistances;
  newNbForbiddenDistances := 0;
  reset(forbiddenDistances);
  for $i \in I$ **do**
    $\text{Dom}(\text{start}(\alpha_{i,1})) := \{0\}$;
    propagate();
    for $i' \in \{i+1, \ldots, N\}$ **do**
      if $|\text{Dom}(\text{start}(\alpha_{i',1}))| < T$ **then**
        $\text{forbiddenDistances}_{i,i'} := \{0, \ldots, T-1\} - \text{Dom}(\text{start}(\alpha_{i',1}))$;
        newNbForbiddenDistances := newNbForbiddenDistances + $|\text{forbiddenDistances}_{i,i'}|$;
    $\text{Dom}(\text{start}(\alpha_{i,1})) := \{0, \ldots, T-1\}$;
  if round > 1 **then**
    Remove constraints posted before the end of last round;
  for $i \in I$ **do**
    for $i' \in \{i+1, \ldots, N\}$ **do**
      if $\text{forbiddenDistances}_{i,i'} \neq \varnothing$ **then**
        Post a new constraint: $((\alpha_{i',1} - \alpha_{i,1} + T) \ \% \ T) \notin \text{forbiddenDistances}_{i,i'}$;
  round := round + 1;
until oldNbForbiddenDistances = newNbForbiddenDistances;

In our problem, the duration of slot operations is already known. As a consequence, when fixing the start time of any operation of a slot, the propagations of the periodic operation constraints (5.1d) to (5.1f) and of the precedence constraints (5.1j) fix the start time of all other operations of this slot. In our problem, the constraint graph is not fully determined at the beginning of the resolution since tracks assignment have to be performed for numerous slot operations that have an impact on the activation of safety headway constraints between slot operations. However, the constraint graph is partially available specifically because of slots running in single track sections or in double tracks sections with restricted circulated directions. That is, when fixing the start time of one slot at time 0, it is possible to deduce inconsistent values in the domain of the start time of some other slots using propagation since those specific slots are known to share common resource of the infrastructure with the slot for which the start time has been fixed. Thus, we use the technique inspired from Kümmling et al. (2016) [110] not for tightening minimal and maximal values of operations duration that in our case are already

fixed to the minimum, but for feeding new constraint that learn forbidden time distances between the start time of slots. Algorithm 3 implements this procedure. In lines 9 to 14, the start time of the first operation of slot i is fixed to time 0 and the constraints of the model are temporarily propagated in order to store forbidden start times for the first operation of other slots that represent forbidden time distances between the start time of slot i and the start time of other slots. The forbidden time distances are then applied and the whole process is repeated in further rounds until no new forbidden time distances between start times of slots are found. As a result, we observe that using the described constraints strengthening procedure and fixing the start time of slot i at time 0 allow inconsistent values to be detected in the domain of the start time of a slot i' that shares no constraints with slot i. This phenomenon is induced by transitivity because some other slots share constraints with both slots i and i'. However, the impact on slot i' is not observed anymore when the constraints strengthening procedure is not used.

5.2 Branching strategies

When solving an instance of the problem described in Section 3.2.2, one can use an off-the-shelf solver to solve the model presented in section 5.1.2, potentially making use of enhancements detailed in Section 5.1.3. However, at the solving stage it is still possible to take the control of the search and apply strategies to attempt converging faster to an high quality solution. In this section, we describe a Branch-and-Check framework that iteratively decide slot start times and check whether there is a feasible tracks assignment for the current timetabled slots. Note that constraint propagation is performed after each decision of start time for a new slot. As a result, a backtrack is done on previously made decisions if the domain of the start time of a remaining slot to be timetabled becomes empty. Algorithm 4 describes the recursive Branch-and-Check timetabling framework. The overall procedure works as follows. The next slot to be timetabled is selected by SelectSlot() procedure. Then, a start time is selected among the values still in the domain of the selected slot branchSlot, using SelectTime(branchSlot) procedure. We then apply the selected start time to the selected slot by recursively calling Timetable() procedure while posting a new constraint. If a backtrack occurs, we then prevent the start time of the selected slot from taking the wrong selected value. After propagation has been performed, we then proceed to checking whether allowed tracks assignment exist for the current timetabled slots. In the following, we describe SelectSlot(), SelectTime(branchSlot) and CheckTracksAssignment(slotsToBeTested) procedures in more details.

Algorithm 4: Branch-and-Check solving framework `TIMETABLE()`

```
1  Input: Infrastructure and headway data, SI, model of section 5.1.2 with possible
     enhancements of Section 5.1.3
2  Output: A timetable with mesoscopic tracks assignment which optimises the turnover times at
     terminus stations
3  TIMETABLE(){
4  oldFixedSlots := GETSTOREDFIXEDSLOTS();
5  newFixedSlots := ∅;
6  slotsToBeTested := ∅;
7  for i ∈ I do
8      if |Dom(start(α_{i,1}))| = 1 then
9          if i ∉ oldFixedSlots then
10             slotsToBeTested := slotsToBeTested ∪ {i};
11         newFixedSlots := newFixedSlots ∪ {i}
12 if !CHECKTRACKSASSIGNMENT(slotsToBeTested) then
13     FAIL();
14 STOREFIXEDSLOTS();
15 branchSlot := SELECTSLOT();
16 branchTime := SELECTTIME(branchSlot);
17 TIMETABLE(start(α_{branchSlot,1}) = branchTime) ||
     TIMETABLE(start(α_{branchSlot,1}) ≠ branchTime);}
```

SELECTSLOT() This procedure aims at selecting the next slot to be timetabled among the non yet timetabled ones. The following steps are tried one by one until a slot is eventually selected.

First, we verify if there is a terminus station where an arriving (resp. departing) slot is timetabled while the linked departing (resp. arriving) slot is not. This linked slot is selected in this case.

Second, we use a priority list to select the non yet timetabled slot for which starting time have the smallest domain among the group of non yet timetabled slots of highest priority. The priority criterion of the list can be as diversified as:

- The decreasing worst cost in which a slot can participate at a terminus station.
- The number of important sections on which slots are scheduled, like sections where tracks from different origins converge and stations, with terminus stations counting for 2.
- The decreasing number of operation of slots.
- The decreasing number of slots sharing the same sections.
- Combinations of those criteria.

Note that one priority list is used for the whole timetabling process. Additional lists can be experimented based on other data properties. Not all slots have to be included in the list. Even the empty list can be tried in order to evaluate the usefulness of priority lists.

Third, if not all slots are elements of the activated priority list, then the slot with the smallest domain is selected. A tie breaking could still be added at this stage.

SelectTime(branchSlot) Once the next slot branchSlot to be timetabled is selected, this procedure aims at selecting a start time among the values of its domain. We focus on 3 cases that are treated distinctly. In the first case, the selected slot branchSlot is linked with a slot that is not timetabled yet. The start time of slot branchSlot is then selected while taking into account the possible start time values of the linked slot in order to attempt minimising the turnover cost associated to those slots at their terminus stations. In the second case, the selected slot branchSlot is linked with a slot that is already timetabled. We select the start time that minimises the associated turnover cost of those slots. Note that during the solution search, when dealing with a partial timetable construction, we can store the best achieved cost obtained for two linked slots at a terminus station to explore more promising parts of the search space first. Finally, in the third case, the selected slot branchSlot passes through the delimited infrastructure without having a terminus station. Then, any start time value can be selected in the domain, e.g. the minimum one. An additional strategy could be applied for this case.

CheckTracksAssignment(slotsToBeTested) When a slot has been timetabled and the propagation process is terminated, we proceed to a tracks assignment check to verify whether infeasibility is detected at this state of the search. Indeed, the propagation process used the partially known tracks assignment, however there are timetabled slot operations for which tracks assignment is still unknown. So, the goal of this procedure is only to assess that there is at least a tracks assignment compatible with the previously decided start times. We need this check procedure to be exhaustive, flexible and parsimonious. By exhaustive, we mean that tracks assignment is checked for all slots newly timetabled, including slots for which the starting time has been induced by the propagation process. More precisely, tracks assignment for those slots is checked on every line section containing more than two tracks and on every stations where they are scheduled. By flexible, we mean that if the check is a success, then the tracks assigned during the check process are unassigned. This allow full flexibility for the time choice of the remainder slots to be timetabled. Indeed, keeping fixed tracks assignment among all possible ones could over-constrain the search and close opportunities to find solutions. Finally, by

parsimonious we mean that we launch the check only when necessary. This may seem obvious, but we technically have to deal with the case when a backtrack occurs so the check that would have been performed at the beginning of the TIMETABLE() procedure is unnecessary. To make this possible, we memorise the slots that have been timetabled only after a success of the check procedure. That is, it is possible to identify the sufficient set of slots to be tested at the beginning of the TIMETABLE() procedure, including when TIMETABLE() is called after a backtrack.

Putting all pieces together, the whole approach presented in Section 4 is summarised in Figure 5.4.

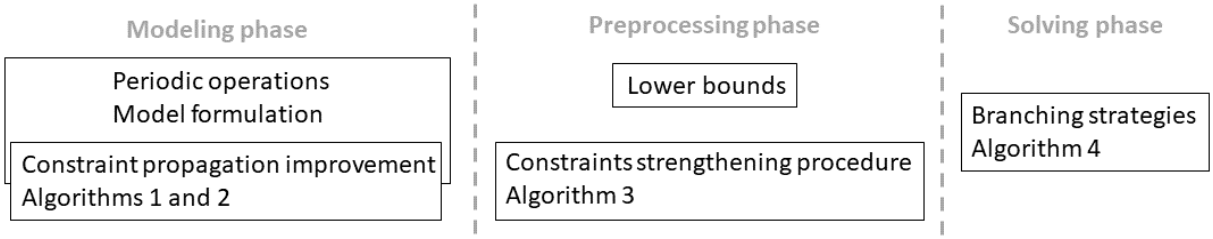


Figure 5.4: Components of the constraint-based approach at different phases of the problem handling.

5.3 Experimental results

We address the optimisation turnover times problem on instances based on the Savoie area infrastructure, presented in Section 3.3.3. Our experiments are set up in our C++ application using CP Optimizer 12.10 to solve the model of Section 5.1.2. The dedicated filtering algorithms of Section 5.1.3 and the branching strategies of Section 5.2 are implemented using the extensions library of CP Optimizer 12.10. The computer used runs on Windows 10, 64 bits with a i5-8265U CPU at 1.60-1.80 GHz and 24 Go of RAM. We use 1 thread when controlling the search using branching strategies, otherwise the 8 available threads work. Table 5.2 reports the results without using branching strategies. The model is solved in 8 configurations: when activated, improved constraints of Section 5.1.3.1 and 5.1.3.2 are indicated by **C**, lower bounds of Section 5.1.3.3 are indicated by **LB**, and constraints strengthening procedure by cascade propagation of Setion 5.1.3.4 is indicated by **P**. We set a time limit of 300 s to allow interactive planning with practitioners. The columns of Table 5.2 are detailed as follows. The objective value of the best found solution is indicated by **obj**. The number of branching decisions made when the best solution is found or when the time limit is attained is indicated by **br**. The amount of time is seconds to find the best solution is indicated by **t**: the value **lim** is printed when the optimal solution is not found within the time limit. A check mark in the **feas** column

means that a feasible solution has been found before the time limit. When an optimal solution is found before the time limit and when it is proved optimal by the solver, a check mark is placed in columns **opt** and **prov**, respectively.

Table 5.2: Performance of the solver on instances 1 and 2 based on both number of branches (br.) and computational time in seconds (t.) with a time limit of 300 s (lim.) and depending on activated enhancements : improved constraints of Section 5.1.3.1 and Section 5.1.3.2 (**C** if activated, $\overline{\mathbf{C}}$ if not), lower bounds of Section 5.1.3.3 (**LB** if activated, $\overline{\mathbf{LB}}$ if not) and constraints strengthening procedure by cascade propagation of Section 5.1.3.4 (**P** if activated, $\overline{\mathbf{P}}$ if not).

			LB						$\overline{\text{LB}}$					
			obj	br	t	feas	opt	prov	obj	br	t	feas	opt	prov
	P	C	4901	733k	lim	✓			4200	**138k**	40	✓	✓	
Instance 1		$\overline{\text{C}}$	4200	1.2M	188	✓	✓	✓	4200	1.9M	177	✓	✓	
	$\overline{\text{P}}$	C	5700	2.7M	lim	✓			6102	259k	lim	✓		
		$\overline{\text{C}}$	4200	646k	**19**	✓	✓	✓	4200	2.2M	65	✓	✓	
	P	C	10305	394k	lim	✓			10000	773k	lim	✓		
Instance 2		$\overline{\text{C}}$	5403	2.5M	lim	✓			13200	4.1M	lim	✓		
	$\overline{\text{P}}$	C	-	1.2M	lim				8400	2.8M	lim	✓		
		$\overline{\text{C}}$	4200	**3.3M**	**220**	✓	✓	✓	-	12.8M	lim			

As expected, we observe in Table 5.2 that for every solving configuration, it is harder to solve Instance 2 than Instance 1. In particular, Instance 2 is proved optimal in the configuration $(\text{LB}, \overline{\text{P}}, \overline{\text{C}})$, and so is Instance 1 but with less branches and a faster computation time. In other configurations that allow a feasible solution to be found for Instance 2, a feasible solution is also found for Instance 1 with a better objective function including the optimal one. Finally in configurations $(\text{LB}, \overline{\text{P}}, \text{C})$ and $(\overline{\text{LB}}, \overline{\text{P}}, \overline{\text{C}})$, no feasible solution is found for Instance 2 whereas a feasible and an optimal solution are found for Instance 1, respectively.

In the configurations for which an optimal solution is found, we observe that the use of the lower bounds LB is crucial to prove optimality within the time limit. Indeed, optimality of solutions with optimal objective value obtained for Instance 1 without lower bounds would not have been proved even in one hour. Surprisingly, proved optimal solutions are obtained faster for both instances in configuration $(\text{LB}, \overline{\text{P}}, \overline{\text{C}})$, that is without improved constraints. It means that the solver, without removing all inconsistent values from the variable domains during propagation, is still able to take reasonable branching decisions. Nevertheless, we observe that configuration $(\overline{\text{LB}}, \text{P}, \text{C})$ allows to find an optimal solution with the less number of branches for Instance 1, even if optimality is not proved at this stage of the search. It means that the improved constraints C and the constraint

strengthening procedure P reduced the search space enough to help the solver take less bad branching decisions. When looking at the first row of Table 5.2 i.e. for Instance 1 and configurations $(\mathrm{LB}, \mathrm{P}, \mathrm{C})$ and $(\overline{\mathrm{LB}}, \mathrm{P}, \mathrm{C})$, an optimal solution is found in the latter case but is not proved optimal by the solver within the time limit because the lower bounds are not used. Surprisingly, in the former case, an optimal solution is not even found within the time limit when the lower bounds are activated, indicating that the branching decisions should be better controlled.

As Instance 2 is more difficult to solve optimally within the time limit than Instance 1, we now present in Table 5.3 the results for Instance 2 when we apply the branching strategies described in Section 5.2. We observe that controlling the search, Instance 2 can be solved to optimality and proved optimal in less than 10^3 branches and in about 1 second. This result represents a significant improvement compared with the solve without branching strategies.

Table 5.3: Performance of the solver with branching strategies of Section 5.2 on Instance 2 based on both number of branches (br.) and computational time in seconds (t.) with a time limit of 300 s (lim.). All enhancements **C**, **LB** and **P** are activated.

Priority list criteria	obj.	br.	t.
Decreasing worst possible cost	4200	60681	6
Decreasing weighted sum of planned sections (*)	4200	**558**	**1**
(*) + tie break: slots with terminus stations first	4200	**478**	**1**
(*) + tie break: slots with terminus stations last	4200	**579**	**1**
Decreasing number of operations (**)	4200	3067	**1**
Decreasing number of slots on common sections (***)	-	3.4M	lim
Decreasing sum of (**) and (***)	4200	3083	2
(**) + tie break: (***)	4200	3067	2
(***) + tie break: (**)	4200	3036	2
Empty	4200	12862	21

We observe that the priority list used for SelectSlot() procedure have an impact on the performance of the resolution. We recall that when the priority list is empty, the next slot selected to be timetabled is either a linked slot at a terminus station to an already timetabled slot, either the slot whose the start time has the smallest domain. Instead of the latter case, using a priority list creates slot groups to be timetabled first, not only relying on the domain sizes. Almost all priority criteria described in Section 5.2 improve either the number of branches decisions or the computation time compared with the use of the empty list. However, no solution has been found within the time limit when slots

are prioritised by decreasing number of slots on common sections. Note that other criteria cloud be used to build new priority lists depending on the properties of the instances to be solved.

All enhancements **C**, **LB** and **P** are activated when solving the problem with branching strategies. Indeed, without the filtering Algorithm 1 described in Section 5.1.1 that helps *arc-consistency* to be maintained for constraints (5.1d) to (5.1f), the domain of start time of operations at the middle of a slot fails to be propagated backward to the domain of start time of the first operation of this slot. As a consequence, a value selected during the SelectTime(branchSlot) procedure can be directly unvalidated because of this lack of propagation completeness. This observation shows the importance of Algorithm 1 when controlling the search. Once again, activating the lower bounds is crucial to prove the optimality of the found solutions. Finally, we tried to solve the problem without activating the constraint strengthening procedure and we did not observe a significant loss of performance. This can be explained because the branching strategies associated with filtering algorithms are able to allow procedures SelectSlot() and SelectTime(branchSlot) to take good decisions enough since the beginning of the controlled search. Figure 5.5 illustrates the space time chart of an optimal solution on double track sections between Culoz and Modane.

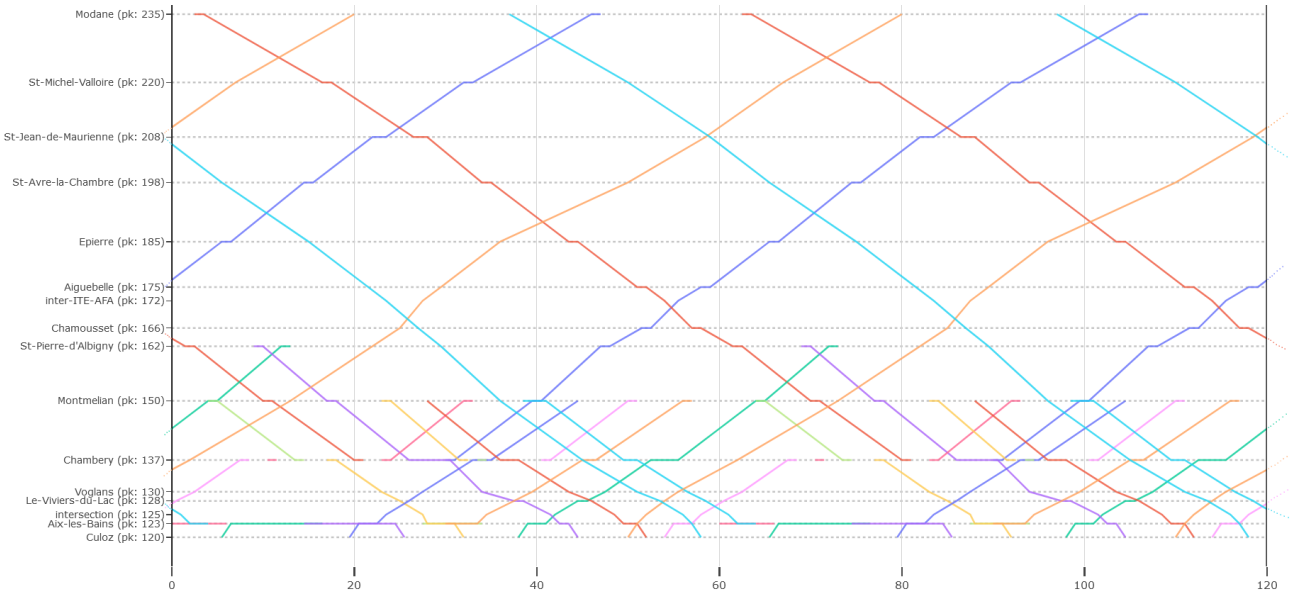


Figure 5.5: Space-time chart of an optimal timetable for Instance 2 on double track sections between Culoz and Modane. On this chart, time is represented in minutes with a granularity of half a minute. Two periods of 60 minutes are printed, allowing to observe the timetable periodicity.

5.4 Conclusion

In this chapter, we presented a constraint-based approach to model and solve the periodic train timetabling with mesoscopic tracks assignment problem while optimising the turnover times at terminus stations. The described model relate interval variables used for timetabling decisions with *optional* interval variables that represent assignment of slot periodic operations to tracks. Dedicated filtering algorithms have been described to improve the detection of inconsistencies when dealing with time periodicity constraints. An optimality-oriented Branch-and-Check procedure has been presented to allow controlling the search and improving the resolution performance.

Future work includes modifications of the model to improve the realism of the solutions. In one hand, spatial cyclicity could be handled as in other realisations like in Fuchs et al. (2022) [78]. This would allow to integrate slots in the SI whose some operations are downstream while other are upstream, e.g. a slot between Saint-André-le-Gaz (section 2) and Annecy (section 59). On the other hand, it should be of a high interest to authorise slack time between slot operations like in the classical PESP in order to find feasible timetables for more instances while controlling the added slack time in the objective function. Moreover, the objective function could be improved to integrate transfer times between lines.

Additionally to the turnover times robustness that we proposed to optimise, we could extend the robustness optimisation to the safety headway constraints (5.1m) to (5.1o) by adding complementary buffer times. Safety headway values should be personalised depending on the slots and the infrastructure resource where constraints occur. Algorithm 2 is already designed for such a customisation of parameter values. While overtakings constraints are not explicitly considered in our model, we check that none occur. In this concern, an additional filtering algorithm could be added in the same way than those presented in this chapter and could even improve the propagation process.

A promising perspective is the generation of minimal conflicts explanations before backtracks occur. The deductions learned while solving the problem can then be exploited as new constraints to improve the reduction of the search space. This kind of technique as been successfully applied in Kümmling et al. (2015) [109] and Fuchs et al. (2022) [78].

5.A List of Symbols

The Table 5.4 summarises symbols and notations used in this chapter.

Table 5.4: Signification of symbols and notations used in Chapter 5.

Symbol	Meaning
T	Time period
SI	Service Intention, set of bidirectional lines to be timetabled within period T
$S = S^l \cup S^t$	Set of sections, partitioned in the set of line sections and the set of stations
K_s	Set of mesoscopic tracks of section s
$D_k \subseteq \{+, -\}, \mathrm{d}()$	Set of possible running directions on track k and running direction of a slot operand, respectively
$R = R^+ \cup R^-$	Set of mesoscopic routes, partitioned by direction
Z	Set of zones, a partition of S
$< z_1, z_2 >$	A bizone containing mesoscopic routes in any direction between z_1 and z_2
I, J_i and J_{i-}	Set of slots, set of operations of slot i and set of operations of slot i except the last one, respectively
$o_{i,j}, p_{i,j}$, $s_{i,j}$ and $t_{i,j}$	Operation j of slot i, its known duration, the section on wich it is processed and its starting time, respectively
$\lvert\ \rvert$	Cardinal of a set operand inside the symbol
$K_{i,j} = \{1, \ldots, \lvert K_{s_{i,j}} \rvert\}$	Set of alternative tracks for operation $o_{i,j}$
$x_{i,j,k}$	Assignment of track k for operation $o_{i,j}$
$\Delta_{\mathrm{spa}}, \Delta_{\mathrm{reu}}$ and Δ_{cro}	Spacing, reuse and crossing safety headways, respectively
$\mu_{\mathrm{min}^s}, \mu_{\mathrm{rob}^s}$ and μ_{max^s}	Minimal, robust and maximal prescribed turnover times at terminus station s, respectively
μ_{feas}	Unforced minimal turnover time at terminus stations
L_s	Set of linked slots sharing the same track and using the same vehicle at station s
$c_{\mathrm{rob}}, c_{\mathrm{bound}}$ and c_M	Costs penalising turnover times at terminus stations that does not respect the prescriptions
$q^s_{i,i'}$	Turnover time of slots i and i' at terminus station s
$c^s_{i,i'}()$	Cost function depending on the turnover time $q^s_{i,i'}$ of linked slots i and i' at station s
$\alpha_{i,j}, \beta_{i,j}$	Interval variables that compose operation $o_{i,j}$
$\alpha_{i,j,k}, \beta_{i,j,k}$	Implementation of $x_{i,j,k}$: *optional* interval variables that represent the assignment of operation $o_{i,j}$ on track k
$p^{\alpha}_{i,j}, p^{\beta}_{i,j}$	Duration variables of $\alpha_{i,j}$ and $\beta_{i,j}$ which sum equals $p_{i,j}$
start(), end()	start and end time of an operand interval variable
%	Modulo constraint between 2 positive expressions on both sides of the symbol
$\lvert\lvert, \&\&, !, \Rightarrow$	Logical OR, AND, negation and implication symbols, respectively
pr()	Constraint that succeeds if the operand interval variable is *present*
alternative()	Constraint that makes exactly one interval variable to be *present* among a set of *optional* interval variables and synchronises its start and end times with start and end times of a principal interval variable
synchronise()	Constraint that synchronises start and end times of all *present* interval variables among a set of *optional* interval variables with start and end times of a principal interval variable
startAtEnd()	Constraint that make a first interval variable operand starting at the end of a second interval variable operand
setAbsent()	Procedure that sets an interval variable to be *absent*
Dom()	Set of values in the domain of a variable operand
γ	Maximal cyclic distance between 2 surrounding values of the domain of a constrained variable of constraints (5.1m) to (5.1o) to detect inconsistent values in the domain of the other constrained variable
δ_1, δ_2	Generic headways depending on the constraint (5.1m) to (5.1o) and on durations $p_{i,j}$ and $p_{i',j'}$
C	Improved constraints of Sections 5.1.3.1 and 5.1.3.2
LB	Lower bounds of Section 5.1.3.3
P	Constraints strengthening procedure of section 5.1.3.4
D, S1, S2 and S3	Several groups of line sections of the Savoie infrastructure
R, IC and F	Regional, Intercity and Freight line types, respectively
obj, br and t	Objective value, number of branches and computation time in seconds, respectively
feas, opt and prov	There is a feasible solution, the solution is optimal and the solution is proved optimal, respectively

Chapter 6

OptiSillons, a prototype for interactive timetabling

This chapter aims to present an application allowing to compute automatically periodic timetables without requiring algorithmic or programming skills. This work has been realised with two intern students at SNCF Réseau | DGEX Solutions. In particular, the implementation of the graphic user interface in JavaScript is integrally due to their work, undertaken during the third and the fifth semesters of this PhD. In Section 6.1, we present how the data is managed before calling the algorithms of Chapter 4 and Chapter 5. In Section 6.2, we show how a timetable planner at SNCF Réseau is still able to work on the suggested solution, thus providing an interactive experience.

Contents

6.1 Managing the input data

All the data needed to set a periodic timetable problem can be created or imported in OptiSillons and are saved in a json file. The parameters include the mesoscopic infrastructure containing sections, tracks, routes and their compatibilities in the one hand. In the other hand, slots of the Service intention are indicated with the sections they traverse and the duration of their operations. Spending a strictly positive time interval at a station means dwelling at this station in our model. When an infrastructure is provided, it looks like in Figure 3.9, but without the stars uniquely added to allow the infrastructure to better fit in the document. If the user clicks on a section e.g. on Chambéry station at section 15 of the Savoie area infrastructure, the page is scrolled and several view appear like in Figure 6.1. On the right part of the figure, the user has also clicked on track 27, making appearing blue downstream routes, red upstream routes, and green downstream and upstream routes connected to this track. On the left part of the figure, compatibilities between opposite direction routes on the left end of Chambéry station (the northern end) are given. The tracks on the grey cells correspond to initial tracks of routes while the tracks of the white cells correspond to terminal tracks of the routes. Note that compatibilities between several upstream routes and also between several downstream routes are also available.

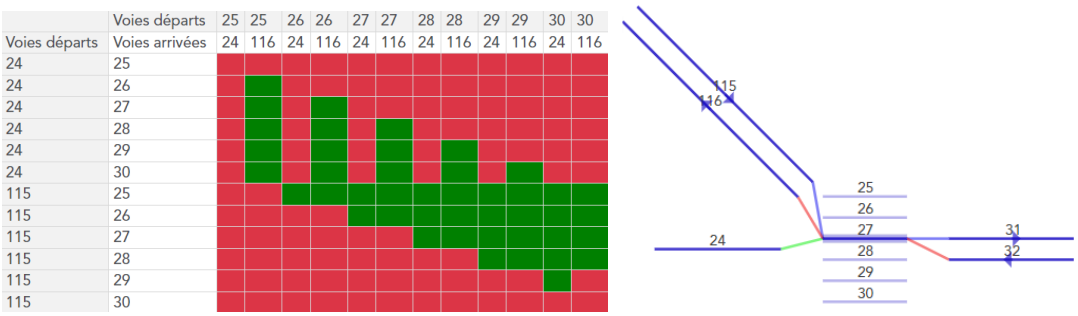
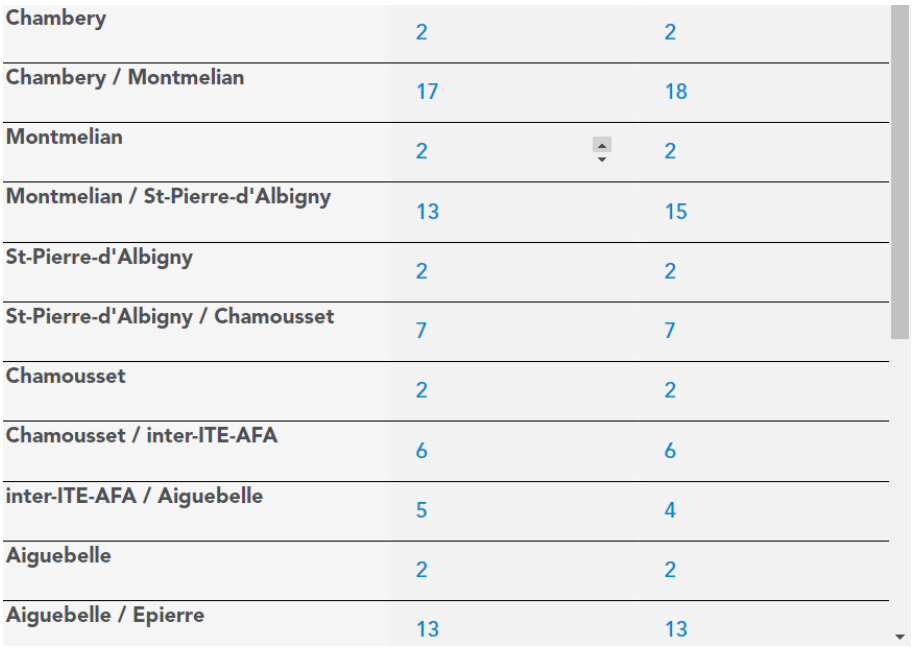


Figure 6.1: Routes and compatibilities around Chambéry station.

Modifications can be made to the parameters. Sections can be added in the infrastructure, as well as tracks and routes. The compatibilities between routes can also be easily indicated or modified by the user. The same applies for the operation durations of the slots included in the Service intention. In Figure 6.2, the given window show a subset of operations of omnibus slots of a line to be operated between Chambéry and Modane stations. The durations, displayed in half minutes, can be edited by the user. The first column corresponds to the operation durations of the downstream slot of the line and the second column corresponds to the operation durations of the upstream slot of the line.



Chambery	2	2
Chambery / Montmelian	17	18
Montmelian	2	2
Montmelian / St-Pierre-d'Albigny	13	15
St-Pierre-d'Albigny	2	2
St-Pierre-d'Albigny / Chamousset	7	7
Chamousset	2	2
Chamousset / inter-ITE-AFA	6	6
inter-ITE-AFA / Aiguebelle	5	4
Aiguebelle	2	2
Aiguebelle / Epierre	13	13

Figure 6.2: Modification of the duration (in half minutes) of operations of omnibus slots between Chambéry and Modane stations.

6.2 Obtaining a solution and working on it

6.2.1 Timetable representations.

When the infrastructure and the Service intention parameters are configured, solution methods of Chapter 4 or Chapter 5 can be called and a periodic timetable is returned if the instance is solved. The solution values are also saved in the aforementioned json file that can be exported. In OptiSillons, the timetables are visualised in the two following representations. First, a space-time chart like in Figure 5.5 gives the location of the slots on the sections of the infrastructure depending on time. Second, a tracks occupation chart informs about the mesoscopic tracks assignment for each slot operation. Traditionally, tracks occupation charts are independently used for different stations. We chose instead to represent all the tracks assignments in the same window, and also including the tracks assignment in line sections. Figure 6.3 shows a sample of the tracks assignment on the Savoie area infrastructure, namely on the single line track between Chambéry and Saint-André-le-Gaz stations. The timetable partially shown in this figure has been optimised to optimality by the Constraint Programming approach presented in Chapter 5. It means that the turnover times at terminus stations respect as much as possible the durations prescribed by the company presented in Table 3.10. The preferred turnover time at Chambéry station is between 10 and 12 minutes. We can observe in Figure 6.3 that indeed, most of the two-tone linked slots on Chambéry tracks are scheduled for a duration that is not penalised. The only longer turnover time is due to slots visualised on the

single track line between Chambéry and Saint-André-le-Gaz stations, and for which the turnover time in Chambéry could not be further optimised because of the restrictions imposed by the single track line sections. We can also obesrve that is Saint-André-le-Gaz station, turnover times might be significantly longer than is Chambéry station. Indeed, a maximal turnover time is not prescribed in this station, thus the turnover times have not be specifically optimised in this station because they are not penalised.

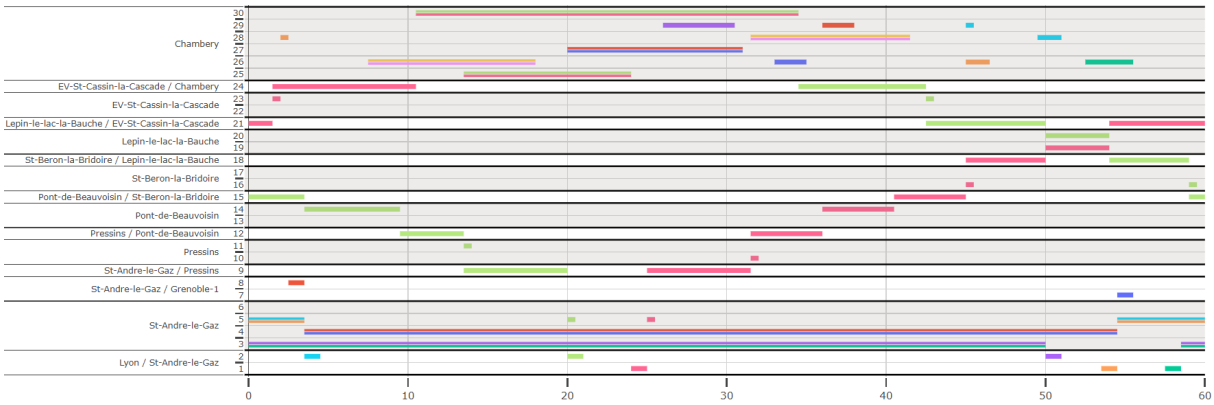


Figure 6.3: Tracks occupation chart between Chambéry and Saint-André-le-Gaz stations.

It is common for timetable planners to work on several screens on their desktop. The infrastructure window can be used on a laptop while the space-time chart and the tracks occupation chart are displayed on two additional screens.

6.2.2 Modifying a timetable.

At any moment of the navigation in OptiSillons, the control is left to the timetable planners. Therefore, it is still an option to work on timetables returned by the tool. Many possibilities are given to modify a timetable. The user can drag a slot and slide its position horizontally on one of the two charts. This action modifies the start time of this slot. The user can also modify the tracks assignment directly on the tracks occupation chart. Moreover, it is possible to add run time and dwell time supplements to the duration of slot operations. In any case, all modifications performed in the space-time chart are replicated to the tracks occupation chart in real-time, and vice versa. Moreover, these modifications are likely to temporarily create conflicts in the timetable. These conflicts are visualised in both charts. From the solution of Figure 5.5, we select on the red slot the line section from Saint-Pierre d'Albigny to Montmélian stations and we show in Figure 6.4 how to add run time supplements to the corresponding operation. These run time supplements will create conflicts that we will detail in the following. When the user clicks once in the plus button, 1 half minute is added to the duration of the operation, and the resulting situation is depicted in Figure 6.5.

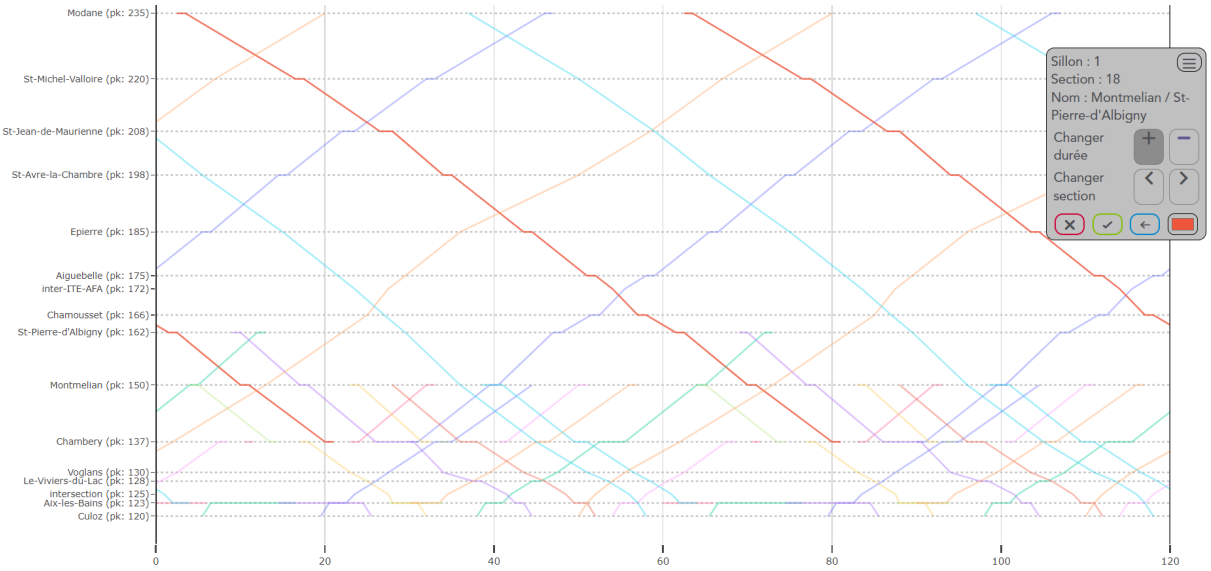


Figure 6.4: Adding run time supplements on the running operation of the red slot on the line section from Saint-Pierre d'Albigny to Montmélian stations.

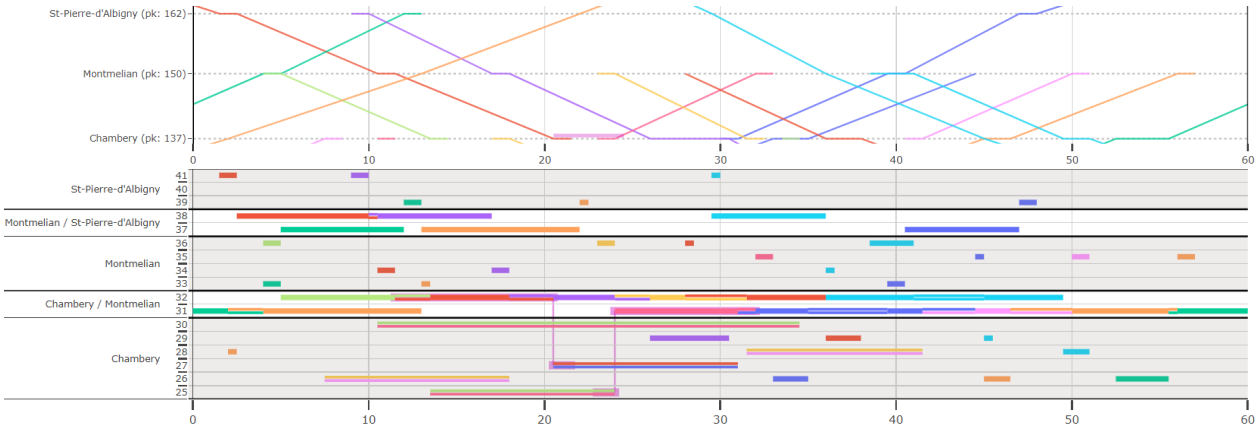


Figure 6.5: Visualisation of conflicts after adding 1 half minute to the red slot from Saint-Pierre d'Albigny to Montmélian.

On the space-time chart, we observe that a crossing conflict between routes taken by the red slot and by the pink slot appeared on the right end of Chambéry station (southern end) to Montmélian. This conflict is also visualised on the tracks occupation chart, where the conflicting routes are highlighted. When adding 4 additional half minutes to the same operation, a new crossing conflict appears between routes on the same end of Chambéry station, a track reuse conflict is created at Montmélian station, and a spacing conflict is detected at the exit of the line track from Saint-Pierre d'Albigny to Montmélian stations. These new conflicts are visualised in Figure 6.6, where the track reuse conflict is highlighted in green and the spacing conflict is highlighted in red. These conflicts are visible on both charts, however the spacing conflicts are more visible on the space-time chart. The supplement of 2 additional minutes on the running operation from Figure 6.5 to Figure 6.6 can be observed on both charts. On the line charts, the slope of the red line

corresponding to this operation has changed, the slot is slower. On the tracks occupation chart, this results in a longer occupation of track 38 for this operation.

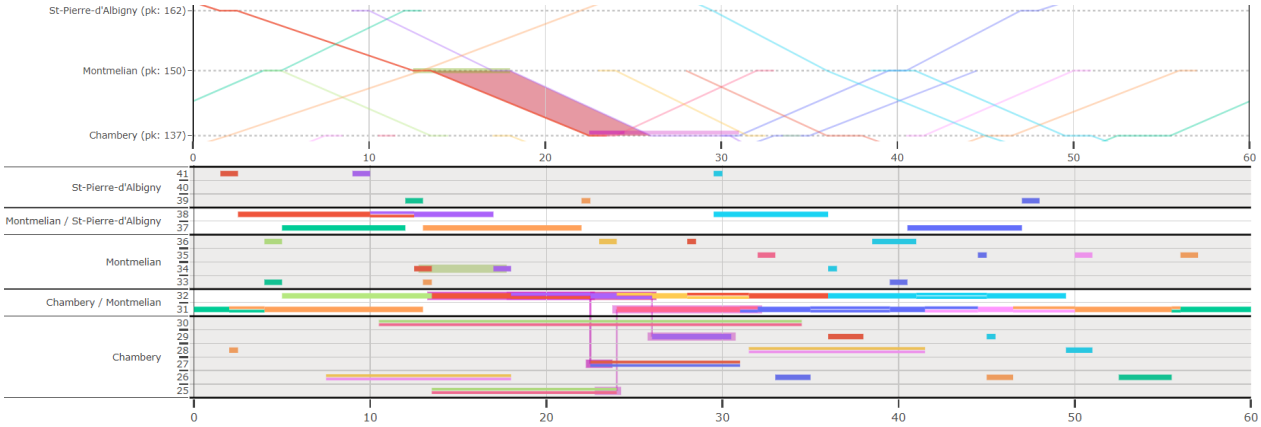


Figure 6.6: Visualisation of conflicts after adding 5 half minutes to the red slot from Saint-Pierre d'Albigny to Montmélian.

Chapter 7

Conclusion and future work

Through this thesis, we addressed the Periodic Train Timetabling with mesoscopic tracks assignment Problem considering known running and dwelling durations, modelled and solved by techniques based on Integer Linear Programming and Constraint Programming. This chapter aims to provide a conclusive picture of the proposed approaches. We summarise our contributions and discuss the completion of the research objectives in Section 7.1. Afterwards, we draw future research directions that could be undertaken from the present work in Section 7.2.

Contents

7.1 Synthesis

Our contributions can be summarised as follows.

After having presented an overview of the research landscape on railway planning in Chapter 2, we formally described the Train Timetabling with mesoscopic tracks assignment Problem considering known running and dwelling durations in Chapter 3, for which we proposed two variants. This task belongs to the tactical planning level which corresponds in our case to few years before operations.

The first variant is a decision problem for which we search to determine if there are a start time and tracks assignment for each slot of a Service intention such that all the constraints arising from the infrastructure use are respected, given that travel times are enforced to be minimised for all direct passengers. This problem variant is addressed in Chapter 4. We presented an Integer Linear Programming formulation to model this problem and we solved instances of increasing size on a fictive infrastructure using an off-the-shelf solver. To improve the tractability of the method, we decomposed the problem such that the time decisions are made in a first stage and the tracks assignment decisions are made in a second stage. We conceived a conflict evaluation function to assess conflicts between slots by knowing their timetable without knowing their tracks assignment explicitly. We presented two solution approaches for this problem, along with a didactic execution of our second approach that performed best to solve a close-to-real-world Service intention on the real-world Savoie area infrastructure.

The second variant is an optimisation problem for which we search to optimise the turnover time of rolling stock at trains terminus stations using real time prescriptions from the company and while still satisfying the infrastructure use constraints and ensuring minimal travel times for direct passengers. This problem variant is treated in Chapter 5. By linking the opposite directions slots of each line at their terminus stations, the realism of the timetables is improved compared to the decision problem in which this slots linking were not taken into account. We started our investigations by creating a Constraint Programming model to the decision problem with an off-the-shelf solver, and it turned out that solutions could be obtained in few seconds for the same instance Savoie area instance. This result encouraged us to upgrade the formulation and to consider the turnover time optimisation problem. As the solver struggled to prove the optimality of the solutions, we proposed model enhancements including extended crossing constraints, dedicated filtering algorithms, lower bounds based on the optimal cost contribution of linked slots and a constraint strengthening procedure inspired from the literature. Fur-

thermore, we set up a Branch-and-Check procedure using branching strategies based on priority lists between slot groups and starting times they should take, and checking routines verifying if tracks assignment are still possible whenever a new slot is scheduled. Our method performed better than the default solver on our Savoie area instance, allowing to find optimal solutions faster and with a small amount of branches.

We hope that Contribution entitled **C1** including the problem formulation, the models and the solution approaches are a strong base to meet Objective **O1** that consists in proposing methods to generate feasible periodic timetables in a reasonable amount of time.

We embedded our solution approaches in a user-friendly application allowing timetable planners to enter data corresponding to an infrastructure at the mesoscopic scale and a Service intention, automatically obtaining timetabling suggestions with tracks assignment, and to still being able to work on the solutions if further improvements are desired. The graphic user interface OptiSillons, integrally developed by two intern students, is presented in Chapter 6.

We believe that Contribution entitled **C2** corresponding to the prototype OptiSillons is a first step towards the possibility to generate periodic timetables in practice in the company by timetable planners without requiring algorithmic or programming skills, thus meeting in part Objective **O2**.

There are plenty of opportunities to improve Contributions **C1** and **C2**, leading to the following perspectives.

7.2 Perspectives

The future research directions from our work are numerous.

Our solutions approaches will need additional investigation to solve larger instances. The Integer Linear Programming methods should be upgraded to model the decision problem with linked slots at terminus stations, as well as the turnover time optimisation problem. This upgrade would imply slight modifications in Formulation (4.1) that can be inspired from Formulation (5.1). As a result, the conflict evaluation function would also be enriched by turnover conflicts. A bi-objective function may also be considered to minimise the conflicts while looking for good turnover times. Another important aspect that the conflict evaluation should integrate is the possibility to deal with safety headway parameters that could have different values. To solve the problem on larger

infrastructures, additional local moves may be integrated to our tabu search framework, like start time modifications involving several slots at once. A local move type selection heuristic could therefore be necessary. If more complex railway stations are present in the infrastructure, a heuristic should also be investigated for the tracks assignment problem that for now works well when a conflict-free timetable is known with our stations sizes.

Specificities of each formulation may additionally benefit to each other as follows. On the one hand, extended crossing conflicts presented in Section 5.1.3.1 could be translated to refine the crossing conflict evaluation formula of Section 4.2.1.3. On the other hand, the blue square cliques of Figure 4.2 determining the maximal capacity on potential crossing areas should definitely be used to derive new dedicated filtering algorithms in the Constraint Programming approach additionally to the existing ones described in Section 5.1.3.2.

The problem description in itself could be further adapted to improve the realism of the resulting solutions. Namely, a multi-periodic should be considered to take into account a small proportion of slots which period is twice the period of the other slots of the Service intention. Passenger data should be integrated to improve the quality of the timetables from their point of view. Additional variables and constraints modelling train composition modifications at stations and technical slots movements to shunting areas and maintenance facilities would be challenging but should be valuable extensions.

Robustness of the timetable is in part implicitly addressed because the nominal operation durations are larger than the minimal technically possible ones, and because the safety headway parameters also include buffer times in real-world settings. However, this is not sufficient to obtain an optimally robust timetable. Indeed, the time interval between slots heading to their conflict points should be optimised as well. We hope having begun a work towards this direction by penalising non robust turnover times at terminus stations. A natural extension of this work is therefore to translate this penalisation to the headway constraints to obtain additional buffer times and cope better with uncertainty.

Our graphic user interface timetabling prototype demonstrated its relevance to the timetable planners of SNCF Réseau. Additional work is however required to industrialise this tool i.e. to connect it with the data flows and the information systems of the company. Our team currently considers to include the present work as a plug-in module of a microscopic simulation tool already well established in the tools ecosystem of the company.

Bibliography

[1] Montserrat Abril, Federico Barber, Laura Ingolotti, Miguel A. Salido, Pilar Tormos, and Antonio Lova. An assessment of railway capacity. *Transportation Research Part E: Logistics and Transportation Review*, 44(5), 2008.

[2] Emma V. Andersson, Anders Peterson, and Johanna Törnquist Krasemann. Quantifying railway timetable robustness in critical points. *Journal of Rail Transport Planning and Management*, 3(3), 2013.

[3] Lijie Bai. *Train platforming problem in busy and complex railway stations.* PhD thesis, Ecole Centrale de Lille, 2015.

[4] Philippe Baptiste, Claude Le Pape, and Wim Nuijten. *Constraint-based scheduling : applying constraint programming to scheduling problems.* 2001.

[5] Eva Barrena, David Canca, Leandro C. Coelho, and Gilbert Laporte. Exact formulations and algorithm for the train timetabling problem with dynamic demand. *Computers and Operations Research*, 44, 2014.

[6] Aharon Ben-Tal, Alexander Goryashko, Elena Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2), 2004.

[7] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of Linear Programming problems contaminated with uncertain data. *Mathematical Programming, Series B*, 88(3), 2000.

[8] Faten Benhizia. *Optimisation du plan de transport par planification intégrée des ressources.* PhD thesis, Ecole Nationale Supérieure des Mines Saint-Etienne, 2012.

[9] Gérard Bertereix and Aline Beurrier. Conception des horaires assistée par ordinateur (CHAO). *Revue Générale des Chemins de Fer*, 1990.

[10] Dimitris Bertsimas and Melvyn Sim. The Price of Robustness. *Operations Research*, 52, 2004.

[11] Nikola Bešinović. Resilience in railway transport systems: a literature review and research agenda. *Transport Reviews*, 40(4), 2020.

[12] Nikola Bešinović, Raphael Ferrari Nassar, and Christopher Szymula. Resilience assessment of railway networks: Combining infrastructure restoration and transport management. *Reliability Engineering and System Safety*, 224, 2022.

[13] Nikola Bešinović and Rob M.P. Goverde. Stable and robust train routing in station areas with balanced infrastructure capacity occupation. *Public Transport*, 11(2), 2019.

[14] Nikola Bešinović, Rob M.P. Goverde, Egidio Quaglietta, and Roberto Roberti. An integrated micro-macro approach to robust railway timetabling. *Transportation Research Part B: Methodological*, 87, 2016.

[15] Nikola Bešinović, Egidio Quaglietta, and Rob M.P. Goverde. Resolving instability in railway timetabling problems. *EURO Journal on Transportation and Logistics*, 8(5), 2019.

[16] John R. Birge and François Louveaux. Introduction to Stochastic Programming. In *Springer Series in Operations Research and Financial Engineering*. 2011.

[17] Ralf Borndörfer, Heide Hoppmann, Marika Karbstein, and Niels Lindner. Separation of cycle inequalities in periodic timetabling. *Discrete Optimization*, 35, feb 2020.

[18] Ralf Borndörfer, Torsten Klug, Leonardo Lamorgese, Carlo Mannino, Markus Reuther, and Thomas Schlechte. Recent success stories on integrated optimization of railway systems. *Transportation Research Part C: Emerging Technologies*, 74, 2017.

[19] Ralf Borndörfer, Niels Lindner, and Sarah Roth. A concurrent approach to the periodic event scheduling problem. *Journal of Rail Transport Planning and Management*, 15, 2020.

[20] Ralf Borndörfer, Thomas Schlechte, and Steffen Weider. Railway track allocation by rapid branching. In *OpenAccess Series in Informatics*, volume 14, pages 13–23, 2010.

[21] Enrico Bortoletto, Niels Lindner, and Berenike Masing. Tropical Neighbourhood Search: A New Heuristic for Periodic Timetabling. In *OpenAccess Series in Informatics*, volume 106, 2022.

[22] Lucile Brethomé. *Passenger-oriented modelling and optimization of the railway transportation plan in a mass transit system.* PhD thesis, Ecole Centrale de Lille, 2018.

[23] Simon H. Bull, Jesper Larsen, Richard M. Lusby, and Natalia J. Rezanova. Optimising the travel time of a line plan. *4OR*, 17(3):225–259, sep 2019.

[24] Sofie Burggraeve, Simon H. Bull, Pieter Vansteenwegen, and Richard M. Lusby. Integrating robust timetabling in line plan optimization for railway systems. *Transportation Research Part C: Emerging Technologies*, 77, 2017.

[25] Sofie Burggraeve and Pieter Vansteenwegen. Optimization of supplements and buffer times in passenger robust timetabling. *Journal of Rail Transport Planning and Management*, 7(3), 2017.

[26] Sofie Burggraeve and Pieter Vansteenwegen. Robust routing and timetabling in complex railway stations. *Transportation Research Part B: Methodological*, 101, 2017.

[27] Michael R. Bussieck. *Optimal Lines in Public Rail Transport.* PhD thesis, 1998.

[28] Michael R. Bussieck, Peter Kreuzer, and Uwe T. Zimmermann. Optimal lines for railway systems. *European Journal of Operational Research*, 96(1), 1997.

[29] Michael R. Bussieck, Thomas Winter, and Uwe T. Zimmermann. Discrete optimization in public rail transport. *Mathematical Programming, Series B*, 79(1-3), 1997.

[30] Valentina Cacchiani, Alberto Caprara, and Matteo Fischetti. A lagrangian heuristic for robustness, with an application to train timetabling. *Transportation Science*, 46(1), 2012.

[31] Valentina Cacchiani, Alberto Caprara, Laura Galli, Leo G. Kroon, Gábor Maróti, and Paolo Toth. Railway Rolling Stock Planning: Robustness Against Large Disruptions. *Transportation Science*, 46(2):217 – 232, 2012.

[32] Valentina Cacchiani, Alberto Caprara, and Paolo Toth. A column generation approach to train timetabling on a corridor. *4OR*, 6(2), 2008.

[33] Valentina Cacchiani, Alberto Caprara, and Paolo Toth. Scheduling extra freight trains on railway networks. *Transportation Research Part B: Methodological*, 44(2), 2010.

[34] Valentina Cacchiani, Fabio Furini, and Martin P. Kidd. Approaches to a real-world Train Timetabling Problem in a railway node. *Omega (United Kingdom)*, 58, 2016.

[35] Valentina Cacchiani, Laura Galli, and Paolo Toth. A tutorial on non-periodic train timetabling and platforming problems. *EURO Journal on Transportation and Logistics*, 4(3), 2015.

[36] Valentina Cacchiani, Dennis Huisman, Martin P. Kidd, Leo G. Kroon, Paolo Toth, Lucas Veelenturf, and Joris C. Wagenaar. An overview of recovery models and algorithms for real-time railway rescheduling, 2014.

[37] Valentina Cacchiani, Jianguo Qi, and Lixing Yang. Robust optimization models for integrated train stop planning and timetabling with passenger demand uncertainty. *Transportation Research Part B: Methodological*, 136, 2020.

[38] Valentina Cacchiani and Paolo Toth. Nominal and robust train timetabling problems. *European Journal of Operational Research*, 219(3), 2012.

[39] Gabrio Caimi, Dan Burkolter, Thomas Herrmann, Fabian Chudak, and Marco Laumanns. Design of a railway scheduling model for dense services. In *Networks and Spatial Economics*, volume 9, 2009.

[40] Gabrio Caimi, Martin Fuchsberger, Marco Laumanns, and Kaspar Schüpbach. Periodic railway timetabling with event flexibility. In *Networks*, volume 57, 2011.

[41] Gabrio Caimi, Leo G. Kroon, and Christian Liebchen. Models for railway timetable optimization: Applicability and applications in practice. *Journal of Rail Transport Planning and Management*, 6(4), 2017.

[42] Gabrio Caimi, Marco Laumanns, Kaspar Schüpbach, Stefan Wörner, and Martin Fuchsberger. The periodic service intention as a conceptual framework for generating timetables with partial periodicity. *Transportation Planning and Technology*, 34(4), 2011.

[43] Alberto Caprara, Matteo Fischetti, and Paolo Toth. Modeling and solving the train timetabling problem. *Operations Research*, 50(5), 2002.

[44] Alberto Caprara, Matteo Fischetti, Paolo Toth, Daniele Vigo, and Pier Luigi Guida. Algorithms for railway crew management. *Mathematical Programming, Series B*, 79(1-3), 1997.

[45] Alberto Caprara, Laura Galli, and Paolo Toth. Solution of the train platforming problem. *Transportation Science*, 45(2):246–257, 2011.

[46] Alberto Caprara, Leo G. Kroon, Michele Monaci, Marc Peeters, and Paolo Toth. Chapter 3 Passenger Railway Optimization, 2007.

[47] Alberto Caprara, Michele Monaci, Paolo Toth, and Pier Luigi Guida. A Lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Applied Mathematics*, 154(5 SPEC. ISS.), 2006.

[48] Alberto Caprara, Paolo Toth, Daniele Vigo, and Matteo Fischetti. Modeling and solving the crew rostering problem. *Operations Research*, 46(6), 1998.

[49] Jacques Carlier and Eric Pinson. Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research*, 78(2), 1994.

[50] Samuela Carosi, Antonio Frangioni, Laura Galli, Leopoldo Girardi, and Giuliano Vallese. A matheuristic for integrated timetabling and vehicle scheduling. *Transportation Research Part B: Methodological*, 127, 2019.

[51] Rémy Chevrier, Paola Pellegrini, and Joaquín Rodriguez. Energy saving in railway timetabling: A bi-objective evolutionary approach for computing alternative running times. *Transportation Research Part C: Emerging Technologies*, 37, 2013.

[52] Václav Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4), 1973.

[53] Francesco Corman, Andrea D'Ariano, Alessio D. Marra, Dario Pacciarelli, and Marcella Samà. Integrating train scheduling and delay management in real-time railway traffic control. *Transportation Research Part E: Logistics and Transportation Review*, 105, 2017.

[54] Francesco Corman, Andrea D'Ariano, Dario Pacciarelli, and Marco Pranzo. A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B: Methodological*, 44(1), 2010.

[55] Francesco Corman, Andrea D'Ariano, Dario Pacciarelli, and Marco Pranzo. Bi-objective conflict detection and resolution in railway traffic management. *Transportation Research Part C: Emerging Technologies*, 20(1), 2012.

[56] Francesco Corman, Andrea D'Ariano, Marco Pranzo, and Ingo A. Hansen. Effectiveness of dynamic reordering and rerouting of trains in a complicated and densely occupied station area. *Transportation Planning and Technology*, 34(4), 2011.

[57] Sélim Cornet. *Formalisation et résolution du problème de construction de grilles horaires robustes pour les réseaux ferrés denses.* PhD thesis, Université Gustave Eiffel, 2020.

[58] George B. Dantzig. Linear Programming and Extensions. Technical report, 1963.

[59] Andrea D'Ariano, Dario Pacciarelli, and Marco Pranzo. Assessment of flexible timetables in real-time traffic management of a railway bottleneck. *Transportation Research Part C: Emerging Technologies*, 16(2), 2008.

[60] Andrea D'Ariano and Marco Pranzo. An advanced real-time train dispatching system for minimizing the propagation of delays in a dispatching area under severe disturbances. In *Networks and Spatial Economics*, volume 9, 2009.

[61] Laurent Daudet. *Algorithms for Train scheduling on a single line*. PhD thesis, Université Paris-Est, 2017.

[62] Stefano de Fabris, Giovanni Longo, Giorgio Medeossi, and Raffaele Pesenti. Automatic generation of railway timetables based on a mesoscopic infrastructure model. *Journal of Rail Transport Planning and Management*, 4(1-2), 2014.

[63] Xavier Delorme, Xavier Gandibleux, and Joaquín Rodriguez. Stability evaluation of a railway timetable at station level. *European Journal of Operational Research*, 195(3), 2009.

[64] Xavier Delorme, Joaquín Rodriguez, and Xavier Gandibleux. Heuristics for railway infrastructure saturation. In *Electronic Notes in Theoretical Computer Science*, volume 50, 2001.

[65] Thijs Dewilde, Peter Sels, Dirk Cattrysse, and Pieter Vansteenwegen. Robust railway station planning: An interaction between routing, timetabling and platforming. *Journal of Rail Transport Planning and Management*, 3(3):68–77, 2013.

[66] Thijs Dewilde, Peter Sels, Dirk Cattrysse, and Pieter Vansteenwegen. Improving the robustness in railway station areas. *European Journal of Operational Research*, 235(1):276–286, may 2014.

[67] Twan Dollevoet, Francesco Corman, Andrea D'Ariano, and Dennis Huisman. An iterative optimization framework for delay management and train scheduling. *Flexible Services and Manufacturing Journal*, 26(4), 2014.

[68] Twan Dollevoet, Dennis Huisman, Leo G. Kroon, Marie Schmidt, and Anita Schöbel. Delay management including capacities of stations. *Transportation Science*, 49(2), 2015.

[69] Twan Dollevoet, Dennis Huisman, Marie Schmidt, and Anita Schöbel. Delay management with rerouting of passengers. *Transportation Science*, 46(1), 2012.

[70] Wolfgang Domschke. Schedule synchronization for public transit networks. *OR Spektrum*, 11(1), 1989.

[71] RER Eole. Pourquoi NExTEO ? https://www.rer-eole.fr/actualite/pourquoi-nexteo/ (Accessed 12 May 2023).

[72] Jean-Pierre Farandou. Le fer contre le carbone - Doubler la place du train pour une vraie transition climatique, 2022.

[73] Thomas A. Feo and Mauricio G.C. Resende. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, 6(2), 1995.

[74] Matteo Fischetti and Michele Monaci. Light robustness. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5868 LNCS, 2009.

[75] Matteo Fischetti and Michele Monaci. Using a general-purpose Mixed-Integer Linear Programming solver for the practical solution of real-time train rescheduling. *European Journal of Operational Research*, 263(1), 2017.

[76] Matteo Fischetti, Domenico Salvagnin, and Arrigo Zanette. Fast approaches to improve the robustness of a railway timetable. *Transportation Science*, 43(3), 2009.

[77] Richard Freling, Dennis Huisman, and Albert P.M. Wagelmans. Models and algorithms for integration of vehicle and crew scheduling. In *Journal of Scheduling*, volume 6, 2003.

[78] Florian Fuchs, Alessio Trivella, and Francesco Corman. Enhancing the interaction of railway timetabling and line planning with infrastructure awareness. *Transportation Research Part C: Emerging Technologies*, 142, 2022.

[79] M R Garey and D S Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences). *Computers and Intractability*, 1979.

[80] Gianmarco Garrisi and Cristina Cervelló-Pastor. Train-scheduling optimization model for railway networks with multiplatform stations. *Sustainability (Switzerland)*, 12(1), 2020.

[81] Philine Gattermann, Peter Großmann, Karl Nachtigall, and Anita Schöbel. Integrating passengers' routes in periodic timetabling: A SAT approach. In *OpenAccess Series in Informatics*, volume 54, 2016.

[82] Laurent Gély. *Modélisation et Optimisation de la Gestion Opérationnelle du Trafic Ferroviaire en cas d'Aléas*. PhD thesis, Université de Bordeaux, 2010.

[83] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5), 1986.

[84] Marc Goerigk, Martin Knoth, Matthias Müller-Hannemann, Marie Schmidt, and Anita Schöbel. The price of robustness in timetable information. In *OpenAccess Series in Informatics*, volume 20, 2011.

[85] Marc Goerigk and Christian Liebchen. An improved algorithm for the periodic timetabling problem. In *OpenAccess Series in Informatics*, volume 59, 2017.

[86] Marc Goerigk and Anita Schöbel. Improving the modulo simplex algorithm for large-scale periodic timetabling. *Computers and Operations Research*, 40(5), 2013.

[87] Marc Goerigk and Anita Schöbel. Recovery-to-optimality: A new two-stage approach to robustness with an application to aperiodic timetabling. *Computers and Operations Research*, 52(PART A), 2014.

[88] Jan Willem Goossens, Stan P.M. Van Hoesel, and Leo G. Kroon. A branch-and-cut approach for solving railway line-planning problems. *Transportation Science*, 38(3), 2004.

[89] Rob M.P. Goverde. Railway timetable stability analysis using max-plus system theory. *Transportation Research Part B: Methodological*, 41(2), 2007.

[90] Rob M.P. Goverde. A delay propagation algorithm for large-scale railway traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(3), 2010.

[91] Rob M.P. Goverde, Nikola Bešinović, Anne Binder, Valentina Cacchiani, Egidio Quaglietta, Roberto Roberti, and Paolo Toth. A three-level framework for performance-based railway timetabling. *Transportation Research Part C: Emerging Technologies*, 67:62–83, jun 2016.

[92] Rob M.P. Goverde and Ingo A. Hansen. Performance indicators for railway timetables. In *IEEE ICIRT 2013 - Proceedings: IEEE International Conference on Intelligent Rail Transportation*, 2013.

[93] Vera Grafe, Alexander Schiewe, and Anita Schöbel. Delay Management with Integrated Decisions on the Vehicle Circulations. In *OpenAccess Series in Informatics*, volume 106, 2022.

[94] Vera Grafe and Anita Schöbel. Solving the periodic scheduling problem: An assignment approach in non-periodic networks. In *OpenAccess Series in Informatics*, volume 96, 2021.

[95] Peter Großmann, Steffen Hölldobler, Norbert Manthey, Karl Nachtigall, Jens Opitz, and Peter Steinke. Solving periodic event scheduling problems with SAT. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7345 LNAI, 2012.

[96] Jørgen Haahr and Richard M. Lusby. Integrating rolling stock scheduling with train unit shunting. *European Journal of Operational Research*, 259(2), 2017.

[97] Sabrina Herrigel, Marco Laumanns, Jacint Szabo, and Ulrich Weidmann. Periodic railway timetabling with sequential decomposition in the PESP model. *Journal of Rail Transport Planning and Management*, 8(3-4), 2018.

[98] Mojtaba Heydar, Matthew E.H. Petering, and Dietrich R. Bergmann. Mixed integer programming for minimizing the period of a cyclic railway timetable for a single track with two train types. *Computers and Industrial Engineering*, 66(1), 2013.

[99] Dennis Huisman, Leo G. Kroon, Ramon M. Lentink, Michiel J.C.M. Vromans, Dennis Huisman, Leo G. Kroon, Ramon M. Lentink, and Michiel J.C.M. Vromans. Operations research in passenger railway transportation. *Statistica Neerlandica*, 59(4):467–497, nov 2005.

[100] Yoshinao Isobe, Hisabumi Hatsugai, Akira Tanaka, Yutaka Oiwa, Takanori Ambe, Akimasa Okada, Satoru Kitamura, Yamato Fukuta, and Takashi Kunifuji. Automatic generation of train timetables from mesoscopic railway models by SMT-solver. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 2019.

[101] Lars Wittrup Jensen, Alex Landex, Otto Anker Nielsen, Leo G. Kroon, and Marie Schmidt. Strategic assessment of capacity consumption in railway networks: Framework and model. *Transportation Research Part C: Emerging Technologies*, 74, 2017.

[102] Feng Jiang, Valentina Cacchiani, and Paolo Toth. Train timetabling by skip-stop planning in highly congested lines. *Transportation Research Part B: Methodological*, 104, 2017.

[103] Antoine Jouglet and Jacques Carlier. Dominance rules in combinatorial optimization problems. *European Journal of Operational Research*, 212(3), 2011.

[104] Franck Kamenga. *Combinatorial optimization for integrating rolling stock management and railway traffic scheduling in passenger stations*. PhD thesis, Université Gustave Eiffel, 2020.

[105] Eva König. A review on railway delay management. *Public Transport*, 12(2), 2020.

[106] Eva König and Cornelia Schön. Railway delay management with passenger rerouting considering train capacity constraints. *European Journal of Operational Research*, 288(2), 2021.

[107] Leo G. Kroon, Gábor Maróti, Mathijn Retel Helmrich, Michiel J.C.M. Vromans, and Rommert Dekker. Stochastic improvement of cyclic railway timetables. *Transportation Research Part B: Methodological*, 42(6), 2008.

[108] Leo G. Kroon, Leon W.P. Peeters, Joris C. Wagenaar, and Rob A. Zuidwijk. Flexible connections in PESP models for cyclic passenger railway timetabling. *Transportation Science*, 48(1), 2014.

[109] Michael Kümmling, Peter Großmann, Karl Nachtigall, Jens Opitz, and Reyk Weiß. A state-of-the-art realization of cyclic railway timetable computation. *Public Transport*, 7(3), 2015.

[110] Michael Kümmling, Jens Opitz, and Peter Großmann. Improving traffic assignment and periodic timetable optimization by constraint propagation. In *Proceedings of OR 2016 Conference*, 2016.

[111] Daniel Lancien and Michèle Fontaine. Calcul de marche de trains minimisant l'énergie de traction - Le programme MARECO. *Revue Générale des Chemins de Fer*, 1981.

[112] Florin Leutwiler and Francesco Corman. A logic-based Benders decomposition for microscopic railway timetable planning. *European Journal of Operational Research*, 303(2), 2022.

[113] Christian Liebchen. Symmetry for periodic railway timetables. In *Electronic Notes in Theoretical Computer Science*, volume 92, 2004.

[114] Christian Liebchen. The first optimized railway timetable in practice. *Transportation Science*, 42(4), 2008.

[115] Christian Liebchen, Marco Lübbecke, Rolf H. Möhring, and Sebastian Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In *Lecture Notes in Computer Science (including subseries Lecture Notes*

in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 5868 LNCS, 2009.

[116] Christian Liebchen and Rolf H. Möhring. A case study in periodic timetabling. In *Electronic Notes in Theoretical Computer Science*, volume 66, 2002.

[117] Christian Liebchen and Rolf H. Möhring. The modeling power of the periodic event scheduling problem: Railway timetables - And beyond. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4359 LNCS, 2007.

[118] Christian Liebchen, Michael Schachtebeck, Anita Schöbel, Sebastian Stiller, and André Prigge. Computing delay resistant railway timetables. *Computers and Operations Research*, 37(5), 2010.

[119] Christian Liebchen and Elmar Swarat. The second Chvátal closure can yield better railway timetables. In *OpenAccess Series in Informatics*, volume 9, 2008.

[120] Niels Lindner and Christian Liebchen. Determining all integer vertices of the PESP polytope by flipping arcs. In *OpenAccess Series in Informatics*, volume 85, 2020.

[121] Niels Lindner and Christian Liebchen. Timetable merging for the Periodic Event Scheduling Problem. *EURO Journal on Transportation and Logistics*, 11, 2022.

[122] Niels Lindner and Julian Reisch. An analysis of the parameterized complexity of periodic timetabling. *Journal of Scheduling*, 25(2), 2022.

[123] Richard M. Lusby, Jesper Larsen, and Simon H. Bull. A survey on robustness in railway planning, 2018.

[124] Richard M. Lusby, Jesper Larsen, Matthias Ehrgott, and David Ryan. Railway track allocation: Models and methods, 2011.

[125] Thomas L. Magnanti and Richard T. Wong. Network Design and Transportation Planning: Models and Algorithms. *Transportation Science*, 18(1), 1984.

[126] Grégory Marlière, Sonia Sobieraj Richard, Paola Pellegrini, and Joaquin Rodriguez. A conditional time-intervals formulation of the real-time railway traffic management problem. *Control Engineering Practice*, 133:105430, 2023.

[127] Berenike Masing, Niels Lindner, and Christian Liebchen. Periodic timetabling with integrated track choice for railway construction sites. *Journal of Rail Transport Planning & Management*, 28:100416, 2023.

[128] Lars-Göran Mattsson and Erik Jenelius. Vulnerability and resilience of transport systems - A discussion of recent research. *Transportation Research Part A: Policy and Practice*, 81, 2015.

[129] Marie Milliet de Faverges. *Développement et implémentation de modèles apprenants pour l'exploitation des grandes gares.* PhD thesis, Conservatoire National des Arts et Métiers, 2020.

[130] Karl Nachtigall. Periodic network optimization with different arc frequencies. *Discrete Applied Mathematics*, 69(1-2), 1996.

[131] Karl Nachtigall. Periodic network optimization and fixed interval timetables. Habilitation Thesis. 1998.

[132] Karl Nachtigall and Jens Opitz. Solving periodic timetable optimisation problems by modulo simplex calculations. In *OpenAccess Series in Informatics*, volume 9, 2008.

[133] Karl Nachtigall and Stefan Voget. A genetic algorithm approach to periodic railway synchronization. *Computers and Operations Research*, 23(5), 1996.

[134] Karl Nachtigall and Stefan Voget. Minimizing waiting times in integrated fixed interval timetables by upgrading railway tracks. *European Journal of Operational Research*, 103(3), 1997.

[135] Michiel A. Odijk. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B: Methodological*, 30(6), 1996.

[136] Michiel A. Odijk. *Railway Timetable Generation.* PhD thesis, Technische Universiteit Delft, 1998.

[137] Julius Pätzold, Alexander Schiewe, Philine Schiewe, and Anita Schöbel. Look-ahead approaches for integrated planning in public transportation. In *OpenAccess Series in Informatics*, volume 59, 2017.

[138] Julius Pätzold and Anita Schöbel. A matching approach for periodic timetabling. In *OpenAccess Series in Informatics*, volume 54, 2016.

[139] Leon W.P. Peeters. *Cyclic Railway Timetable Optimization.* PhD thesis, Erasmus Universiteit Rotterdam, 2003.

[140] Paola Pellegrini, Grégory Marlière, Raffaele Pesenti, and Joaquín Rodriguez. RECIFE-MILP: An Effective MILP-Based Heuristic for the Real-Time Railway Traffic Management Problem. *IEEE Transactions on Intelligent Transportation Systems*, 16(5), 2015.

[141] Paola Pellegrini, Grégory Marlière, and Joaquín Rodriguez. Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B: Methodological*, 59, 2014.

[142] Paola Pellegrini, Grégory Marlière, and Joaquín Rodriguez. RECIFE-SAT: A MILP-based algorithm for the railway saturation problem. *Journal of Rail Transport Planning and Management*, 7(1-2), 2017.

[143] Paola Pellegrini and Joaquín Rodriguez. Single European Sky and Single European Railway Area: A system level analysis of air and rail transportation. *Transportation Research Part A: Policy and Practice*, 57, 2013.

[144] Matthew E.H. Petering, Mojtaba Heydar, and Dietrich R. Bergmann. Mixed-integer programming for railway capacity analysis and cyclic, combined train timetabling and platforming. *Transportation Science*, 50(3), 2016.

[145] Gert-Jaap Polinder, Thomas Breugem, Twan Dollevoet, and Gábor Maróti. An adjustable robust optimization approach for periodic timetabling. *Transportation Research Part B: Methodological*, 128, 2019.

[146] Gert-Jaap Polinder, Valentina Cacchiani, Marie Schmidt, and Dennis Huisman. An iterative heuristic for passenger-centric train timetabling with integrated adaption times. *Computers and Operations Research*, 142, 2022.

[147] Gert-Jaap Polinder, Leo G. Kroon, Karen Aardal, Marie Schmidt, and Marco Molinaro. Resolving Infeasibilities in Railway Timetabling Instances. *SSRN Electronic Journal*, 2018.

[148] Gert-Jaap Polinder, Marie Schmidt, and Dennis Huisman. Timetabling for strategic passenger railway planning. *Transportation Research Part B: Methodological*, 146, 2021.

[149] Jianguo Qi, Valentina Cacchiani, Lixing Yang, Chuntian Zhang, and Zhen Di. An Integer Linear Programming model for integrated train stop planning and timetabling with time-dependent passenger demand. *Computers and Operations Research*, 136, 2021.

[150] Jianguo Qi, Lixing Yang, Yuan Gao, Shukai Li, and Ziyou Gao. Integrated multi-track station layout design and train scheduling models on railway corridors. *Transportation Research Part C: Emerging Technologies*, 69, 2016.

[151] Claude Quinchon. Elaboration et tracé des horaires. *Revue Générale des Chemins de Fer*, 1983.

[152] Julian Reisch, Peter Großmann, Daniel Pöhle, and Natalia Kliewer. Conflict resolving – A local search algorithm for solving large scale conflict graphs in freight railway timetabling. *European Journal of Operational Research*, 293(3), 2021.

[153] SNCF Réseau. La modernisation du réseau. https://www.sncf.com/fr/reseau-expertises/reseau-ferroviaire/sncf-reseau/modernisation-reseau (Accessed 03 May 2023).

[154] SNCF Réseau. Réseau. https://www.sncf-reseau.com/fr/reseau (Accessed 13 June 2023).

[155] SNCF Réseau. Tout savoir sur SNCF Réseau. https://www.sncf.com/fr/reseau-expertises/reseau-ferroviaire/sncf-reseau (Accessed 03 May 2023).

[156] SNCF Réseau. Document de Référence du Réseau ferré national - Horaire de service 2024. Technical report, 2023.

[157] Thomáš Robenek. *Behaviorally Driven Train Timetable Design*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2016.

[158] Joaquín Rodriguez. A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B: Methodological*, 41(2), 2007.

[159] Joaquín Rodriguez and Lyes Kermad. Constraint programming for real-time train circulation management problems in railway nodes. In *Proceedings of the International Conference on Computer Aided Design, Manufacture and Operation in The Railway and Other Advanced Mass Transit Systems*, 1998.

[160] Miguel A. Salido, Montserrat Abril, Federico Barber, Laura Ingolotti, Pilar Tormos, and Antonio Lova. Domain-dependent distributed models for railway scheduling. *Knowledge-Based Systems*, 20(2), 2007.

[161] Marcella Samà, Andrea D'Ariano, Francesco Corman, and Dario Pacciarelli. A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. *Computers and Operations Research*, 78, 2017.

[162] Marcella Samà, Paola Pellegrini, Andrea D'Ariano, Joaquín Rodriguez, and Dario Pacciarelli. Ant colony optimization for the real-time train routing selection problem. *Transportation Research Part B: Methodological*, 85, 2016.

[163] Marcella Samà, Paola Pellegrini, Andrea D'Ariano, Joaquín Rodriguez, and Dario Pacciarelli. On the tactical and operational train routing selection problem. *Transportation Research Part C: Emerging Technologies*, 76, 2017.

[164] Giorgio Sartor, Carlo Mannino, Thomas Nygreen, and Lukas Bach. A MILP model for quasi-periodic strategic train timetabling. *Omega (United Kingdom)*, 116, 2023.

[165] Philine Schiewe and Anita Schöbel. Periodic timetabling with integrated routing: Toward applicable approaches. *Transportation Science*, 54(6), 2020.

[166] Marie Schmidt and Anita Schöbel. Timetabling with passenger routing. *OR Spectrum*, 37(1), 2015.

[167] Anita Schöbel. A model for the delay management problem based on mixed-integer-programming. In *Electronic Notes in Theoretical Computer Science*, volume 50, 2001.

[168] Anita Schöbel. Capacity constraints in delay management. *Public Transport*, 1(2), 2009.

[169] Anita Schöbel. Line planning in public transportation: Models and methods, 2012.

[170] Anita Schöbel. An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation. *Transportation Research Part C: Emerging Technologies*, 74, 2017.

[171] Peter Sels, Dirk Cattrysse, and Pieter Vansteenwegen. Practical macroscopic evaluation and comparison of railway timetables. In *Transportation Research Procedia*, volume 10, 2015.

[172] Peter Sels, Thijs Dewilde, Dirk Cattrysse, and Pieter Vansteenwegen. Reducing the passenger travel time in practice by the automated construction of a robust railway timetable. *Transportation Research Part B: Methodological*, 84:124 – 156, 2016.

[173] Peter Sels, Pieter Vansteenwegen, Thijs Dewilde, Dirk Cattrysse, B. Waquet, and A. Joubert. The train platforming problem: The infrastructure management company perspective. *Transportation Research Part B: Methodological*, 61, 2014.

[174] Paolo Serafini and Walter Ukovich. A Mathematical Model for Periodic Scheduling Problems. *SIAM Journal on Discrete Mathematics*, 2(4), 1989.

[175] Michael Siebert and Marc Goerigk. An experimental comparison of periodic timetabling models. *Computers and Operations Research*, 40(10), 2013.

[176] Allen. L. Soyster. Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming. *Operations Research*, 21(5), 1973.

[177] Daniel Sparing and Rob M.P. Goverde. A cycle time optimization model for generating stable periodic railway timetables. *Transportation Research Part B: Methodological*, 98, 2017.

[178] Christopher Szymula and Nikola Bešinović. Passenger-centered vulnerability assessment of railway networks. *Transportation Research Part B: Methodological*, 136, 2020.

[179] Pilar Tormos, Montserrat Abril, Miguel A. Salido, Federico Barber, Laura Ingolotti, and Antonio Lova. Distributed constraint satisfaction problems to model railway scheduling problems. In *WIT Transactions on the Built Environment*, volume 88, 2006.

[180] UIC. UIC Code 406: Capacity, 2nd ed. Technical report, Paris, 2013.

[181] Raimond Wüst, Stephan Bütikofer, Severin Ess, Claudio Gomez, Albert Steiner, Marco Laumanns, and Jacint Szabo. Maintenance timetable planning based on mesoscopic infrastructure and the transport service intention. *Journal of Rail Transport Planning and Management*, 11, 2019.

[182] Raimond Wüst, Stephan Bütikofer, Severin Ess, Claudio Gomez, Albert Steiner, Marco Laumanns, and Jacint Szabo. Periodic Timetabling with 'Track Choice'-PESP Based on Given Line Concepts and Mesoscopic Infrastructure. In *Operations Research Proceedings 2018*, pages 571–578. Springer International Publishing, 2019.

[183] Raimond Wüst, Felix Laube, Samuel Roos, and Gabrio Caimi. Sustainable global service intention as objective for controlling railway network operations in real time. In *Proceedings of the 8th World Congress of Railway Research (WCRR), Seoul, Korea*, 2008.

[184] Fei Yan, Nikola Bešinović, and Rob M.P. Goverde. Multi-objective periodic railway timetabling on dense heterogeneous railway corridors. *Transportation Research Part B: Methodological*, 125, 2019.

[185] Fei Yan and Rob M.P. Goverde. Combined line planning and train timetabling for strongly heterogeneous railway lines with direct connections. *Transportation Research Part B: Methodological*, 127, 2019.

[186] Zhiyuan Yao, Lei Nie, Yixiang Yue, Zhenhuan He, Yu Ke, Yuxin Mo, and Hongda Wang. Network periodic train timetabling with integrated stop planning and passenger routing: A periodic time–space network construct and admm algorithm. *Transportation Research Part C: Emerging Technologies*, 153:104201, 2023.

[187] Qin Zhang, Richard M. Lusby, Pan Shang, and Xiaoning Zhu. A heuristic approach to integrate train timetabling, platforming, and railway network maintenance scheduling decisions. *Transportation Research Part B: Methodological*, 158, 2022.

[188] Yongxiang Zhang, Qiyuan Peng, Gongyuan Lu, Qingwei Zhong, Xu Yan, and Xuesong Zhou. Integrated line planning and train timetabling through price-based cross-resolution feedback mechanism. *Transportation Research Part B: Methodological*, 155:240–277, 2022.

[189] Wenliang Zhou, Junli Tian, Lijuan Xue, Min Jiang, Lianbo Deng, and Jin Qin. Multi-periodic train timetabling using a period-type-based Lagrangian relaxation decomposition. *Transportation Research Part B: Methodological*, 105, 2017.

[190] Wenliang Zhou, Xiaorong You, and Wenzhuang Fan. A mixed integer linear programming method for simultaneous multi-periodic train timetabling and routing on a high-speed rail network. *Sustainability (Switzerland)*, 12(3), 2020.

[191] Peter J. Zwaneveld, Leo G. Kroon, H. Edwin Romeijn, Marc Salomon, Stéphane Dauzère-Pérès, Stan P.M. Van Hoesel, and Harrie W. Ambergen. Routing trains through railway stations: Model formulation and algorithms. *Transportation Science*, 30(3), 1996.

[192] Peter J. Zwaneveld, Leo G. Kroon, and Stan P.M. Van Hoesel. Routing trains through a railway station based on a node packing model. *European Journal of Operational Research*, 128(1), 2001.