Problématiques d'ordonnancement ferroviaire

Olivier Liess, Serigne Gueye

Problématiques d'ordonnancement ferroviaire

Olivier Liess, Serigne Gueye

18 juin 2008

Chapitre 1

Introduction aux problèmes d'optimisation ferroviaire

1.1 La gestion ferroviaire

Le transport ferroviaire, qu'il s'agisse de voyageurs ou de fret, est un mode de transport nécessitant à toutes les étapes de son développement une réflexion importante. De nombreux problèmes de décision et d'optimisation se posent de la phase prospective initiale jusqu'à l'exploitation proprement dite. Les réponses à ces problèmes conditionnent des investissements lourds ainsi que l'image de l'exploitant auprès des utilisateurs des services proposés.

Si le développement du réseau ferroviaire, entraînant une augmentation de la complexité des problématiques à traiter, a globalement toujours été d'actualité, l'importance de ce développement a augmenté ces dernières années. Les raisons sont nombreuses, citons par exemple :

- une demande de mobilité accrue, aussi bien concernant les déplacements nationaux qu'internationaux (extension de la taille du réseau, planification des dessertes et mise en correspondance, mise en cohérence avec l'international, cadencement) ;
- la privatisation du secteur, qui se traduit en France par une séparation entre la gestion matérielle du réseau, assurée par R.F.F. (Réseau Ferré de France, www.rff.fr), et la prestation de services, encore assurée principalement par l'opérateur historique S.N.C.F. (Société Nationale des Chemins de Fer, www.entreprise-sncf.com). Celle-ci devrait s'ouvrir aux opérateurs privés ;
- les motivations écologiques, qui se traduisent entre autres par le souhait de limiter le trafic routier en assurant le transport de marchandises ou de camions par le rail.

Les critères usuels de praticabilité, de fiabilité et de sécurité restent de plus présents, et doivent être pris en compte dans l'ensemble de la chaîne décisionnelle.

1.2 Structure d'un réseau

Les enjeux économiques importants, l'obligation d'assurer la sécurité des passagers, et la nécessité d'offrir des prestations fiables, aussi bien pour le trafic passager que pour le fret, offrent de nombreuses voies de recherche et domaines d'application pour l'optimisation et l'aide à la décision. Pour cela, il est essentiel de disposer d'une connaissance et d'une représentation adéquate du réseau ferroviaire et des éléments qui le constituent.

Il est de plus souhaitable, voire nécessaire, d'adapter cette représentation en tenant compte de la problématique traitée. Cette adaptation se traduit dans la pratique par une granularité plus ou moins importante. Nous présentons ci-dessous la structure des réseaux ferroviaires selon quatre niveaux de représentation, en partant d'une granularité importante (vision macroscopique, forte agrégation) pour aller vers une granularité fine (vision microscopique, faible agrégation).

1.2.1 1er niveau de représentation : réseau ferroviaire

Le premier niveau considère l'interconnexion des axes de circulation formant le réseau ferroviaire. Cette représentation peut être obtenue à l'aide d'un graphe, dans lequel les sommets représentent les *nœuds* du réseau, c'est à dire les gares et bifurcations, et les arcs (ou arêtes), appelés *tronçons*, représentent les liens directs entre ces nœuds. Des capacités et des temps de parcours peuvent être estimés pour ces différents *points* (nœuds et tronçons), éventuellement des temps d'arrêt (desserte) pour les nœuds de type gare. Une telle représentation, en raison de sa faible précision, est principalement utilisée pour le traitement de problématiques à long terme (niveau stratégique) [1], et permet d'avoir une vision globale, bien qu'approximative, du trafic pouvant transiter par le réseau établi. La figure 1.1 illustre cette représentation.

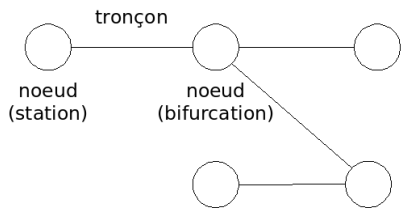

FIG. 1.1: Représentation du réseau ferroviaire : nœuds et tronçons.

1.2.2 2ème niveau de représentation : affectation des voies

Le principal inconvénient de la représentation précédente est la faible prise en compte de la capacité réelle du réseau, qui dépend directement de la capacité des points qui le composent. Cette capacité repose essentiellement sur deux paramètres :

– d'une part, le nombre de voies composant le point, ainsi que leur orientation (que l'on qualifiera dans ce document de *paire*, *impaire*, ou *banalisée*) ;

[1] voir Bussieck et al. [5]

– d'autre part, la nature (grande vitesse, régional, de fret, ...) et l'ordre des circulations empruntant le point.

Si le nombre et le type des voies influent de manière évidente sur la capacité, en permettant le passage d'un nombre plus ou moins important de circulations, il nous paraît nécessaire de clarifier le deuxième point. La construction d'un graphique (voir définition section 1.3.1) doit tenir compte de l'opposition entre ordonnancement alterné des circulations lentes et rapides, plus intéressant sur le plan commercial (voir figure 1.2, schéma de gauche) et ordonnancement en batteries homogènes, de meilleur rendement (voir figure 1.2, schéma de droite).

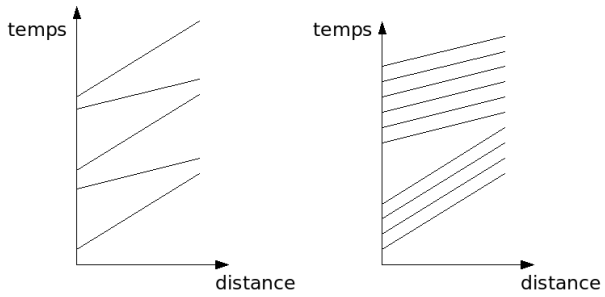

FIG. 1.2: Variation de la capacité en fonction de l'ordre des circulations.

Les règles de sécurité imposent de plus aux circulations empruntant une même voie de respecter un espacement minimal. Cet espacement permet de s'assurer qu'une circulation dispose d'une distance de freinage suffisante (transcrite ici par un temps) dans le cas où celle-ci serait amenée à rattraper la circulation qui la précède. La valeur de cet espacement dépend de la nature des circulations considérées, une circulation rapide nécessitant une distance (et donc un temps) de freinage plus importante. Dans la pratique, si le nombre de voies aux points du réseau et la nature des circulations constituent des données du problème, l'ordre des circulations est le plus souvent obtenu par résolution d'un problème d'ordonnancement.

La modélisation retenue pour tenir compte des voies est illustrée dans la figure 1.3. Elle consiste simplement à détailler les voies existantes en chaque noeud et tronçon du réseau. C'est l'une des plus couramment utilisées. Elle permet un bon compromis entre taille du problème à résoudre et qualité de la représentation [31, 33].

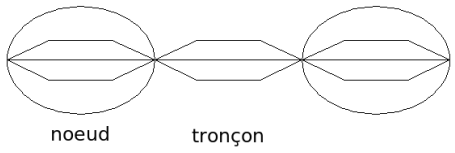

FIG. 1.3: Représentation du réseau ferroviaire : matérialisation des voies.

1.2.3 3ème niveau de représentation : cantons

Là encore, la différence avec le niveau de granularité précédent repose sur la prise en compte des contraintes de capacité. Plus exactement, il s'agit ici de considérer non plus un temps d'espacement estimé en fonction de la nature des circulations, mais la gestion des espacements telle qu'elle est faite en réalité.

Pour cela, il nous faut revenir à la gestion « réelle » des espacements. Chaque voie d'un point du réseau est en réalité divisée en *cantons* (aussi appelés *blocks*), délimités à leurs extrémités par une signalisation (pour certaines circulations, la signalisation étant embarquée). Il est important de souligner que le cantonnement n'est pas nécessairement identique selon le sens de circulation considéré.

La prise en compte de ces cantons dans la modélisation, si elle permet de mieux considérer les contraintes d'espacement, a toutefois l'inconvénient d'augmenter de façon importante la taille du modèle. Cette représentation est ainsi utilisée plutôt pour l'étude de problèmes sur des sous-réseaux de taille réduite, par exemple pour la gestion des circulations en des nœuds importants, pour lesquels une évaluation précise du trafic et de sa capacité est souhaitée. Une illustration de cette représentation est présentée figure 1.4.

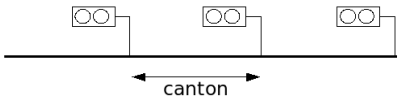

FIG. 1.4: Représentation du réseau ferroviaire : cantonnement d'une voie.

1.2.4 4ème niveau de représentation : circuits de voie

Le 4ème niveau est la granularité la plus fine. On tient compte ici des mécanismes de détection utilisés par les systèmes de signalisation. Chaque canton est pour cela divisé en *circuits de voie*, sur lesquels s'effectue réellement la détection. La modélisation tient alors compte de la présence de ces circuits afin de déterminer la position exacte des trains sur le sous-réseau ferroviaire étudié. Là encore, ceci conduit à accroître la taille du modèle, et ne peut être utilisé que sur des sous-réseaux de petite taille. À notre connaissance, une telle modélisation a été peu utilisée (voir [12, 16, 30]), principalement dans des problématiques d'ordonnancement dans des nœuds complexes.

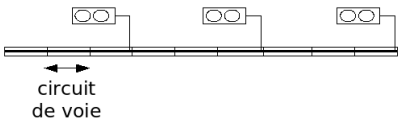

FIG. 1.5: Représentation du réseau ferroviaire : circuits de voie.

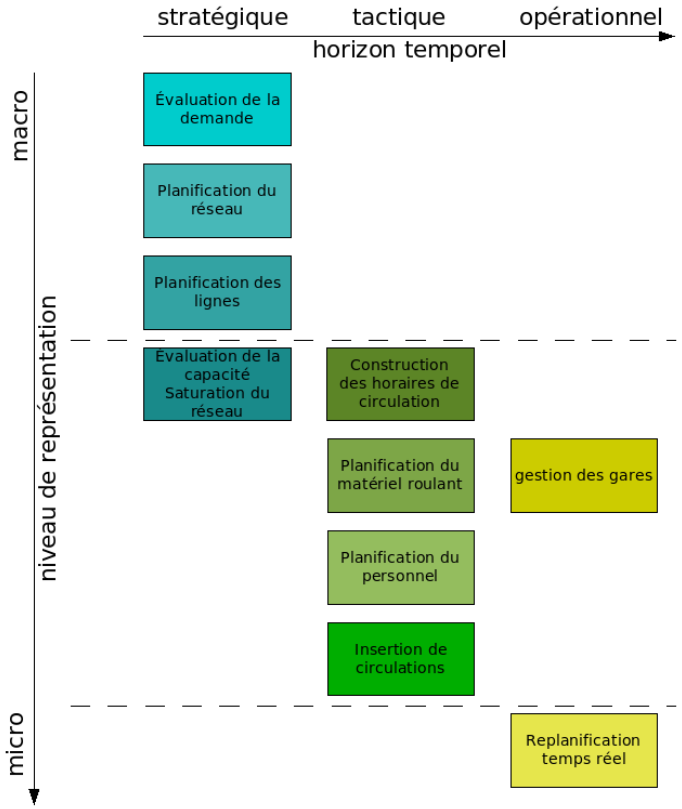

FIG. 1.6: Classification de problèmes d'optimisation ferroviaire

1.3 Problématiques ferroviaires

De nombreuses problématiques d'optimisation dans le contexte ferroviaire ont été définies et étudiées, à tous les niveaux de décision. Du point de vue théorique, elles ont conduit à la publication de nombreux travaux, tandis que du point de vue pratique, des outils logiciels ont été développés et sont utilisés pour la résolution de ces problématiques (voir chapitre 2).

Comme cela est mis en évidence sur la figure 1.6, ces différentes problématiques peuvent être classées selon deux critères principaux. Un premier critère correspond à la portée temporelle du problème. On distingue pour cela les portées stratégiques, d'horizon lointain (une à plusieurs années), tactiques, à moyen terme (de l'ordre de la semaine à l'année) et opérationnels, à très court terme (de quelques heures, minutes au temps réel). Un deuxième axe de classification peut être défini par le niveau de représentation nécessaire pour la modélisation du problème, allant d'une vision macroscopique (cf. section 1.2.1) à un niveau « microscopique » (cf. section 1.2.4). On pourra noter que ces deux classifications sont partiellement liées. En effet, plus l'horizon temporel est réduit plus la nécessité de représenter la structure finement est grande.

Dans notre étude, nous nous sommes particulièrement intéressés à trois problèmatiques que nous détaillons ci-dessous. Une présentation plus détaillée des

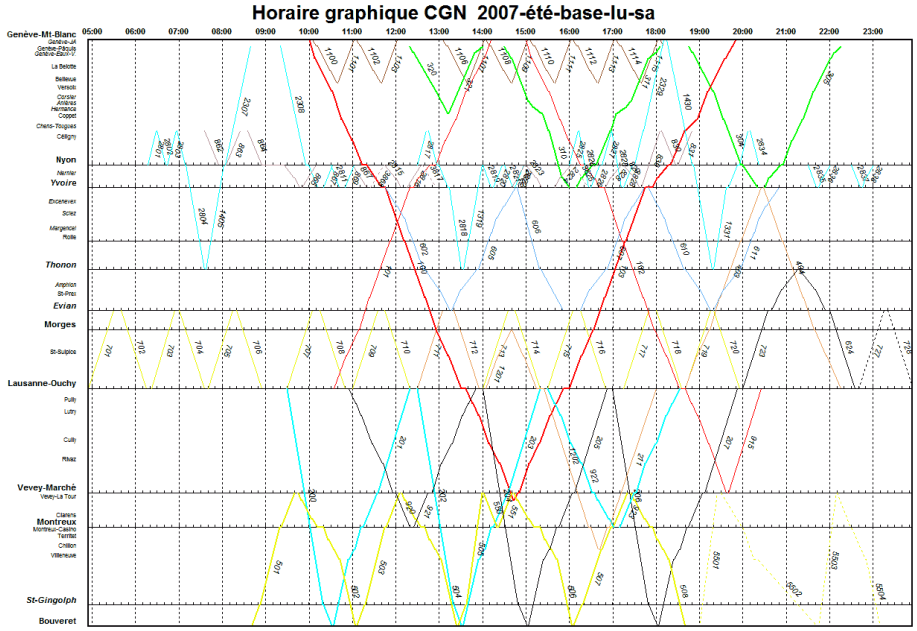

FIG. 1.7: Exemple de graphique de circulation.

éléments de la figure 1.6 peut être trouvée dans [5, 13].

1.3.1 Construction de graphiques de circulation

La construction de graphique de circulation fait parti des problémes tactiques portant sur des décisions le plus souvent à moyen ou long terme. Les graphiques de circulation (voir figure 1.7) représentent sur un diagramme espace-temps les circulations empruntant un itinéraire donné. La construction de ces graphiques est un problème fortement combinatoire. Elle nécessite la prise en compte d'un nombre important de contraintes qui peuvent être :

- le respect des régles de sécurité : espacement des trains sur les noeuds et tronçons
- des contraintes commerciales portant par exemple sur les temps d'arrêts maximums

Parmi les graphiques respectant les contraintes, on s'intéresse en général à déterminer ceux optimisant un ou plusieurs objectifs quantitatifs ou qualitatifs. Le nombre de circulations planifiées est par exemple un objectif quantitatif que l'on tentera de maximiser. la qualité d'un graphique se juge également à sa flexibilité et à sa robustesse. La capacité du graphique à autoriser des modifications (flexibilité) ou à « absorber » de petits aléas (robustesse) sont quant à eux des critères plutôt qualitatifs.

Une présentation générale de ce problème est faite dans [3]. L'auteur y décrit les contraintes qui conditionnent la construction d'un graphique de circulation, en prenant comme exemple concret la ligne TVG Méditerranée. La littérature présente de nombreux travaux de recherche visant à automatiser la construction

de ces graphiques, nous renvoyons le lecteur à la section 2.1 pour une présentation succinte de ces travaux.

1.3.2 Insertion de circulations

Cette problématique découle dans un sens de la précédente. Elle acquiert, avec la privatisation des services, une importance particulière. Il s'agit de positionner, dans un graphique de circulation existant, une nouvelle circulation qui doit respecter les contraintes liées à la capacité et à la sécurité du réseau. Cette insertion, idéalement, doit se faire sans modification des circulations préexistantes. Compte tenu du taux de saturation actuel du réseau, cela peut s'avérer impossible. Dans ce cas, des objectifs tels que la minimisation des retards, ou du nombre de modifications, doivent être considérés. Il y a ainsi une forte similitude avec les problèmes de gestion des aléas que nous présentons ci-après.

1.3.3 Gestion des aléas

La gestion des aléas fait parti des problémes opérationnels portant sur des décisions à court terme. Etant donné un graphique de circulations construit préalablement, il s'agit de suivre l'évolution de ce graphique et d'y apporter les correctifs induits par les aléas inévitables apparaissant dans l'éxecution temps-réel du plan de trafic prévu. Nous entendons par aléas tout incident non prévu dans la circulation. Nous ne considérons que les conséquences quantitatives de ces aléas se traduisant par le retard d'une ou plusieurs circulations. Une fois le retard détecté et quantifié, la grille de circulation en cours d'exploitation est modifiée afin de minimiser les effets de l'aléa.

Chapitre 2

État de l'art

Dans ce chapitre, nous présentons un court état de l'art des problématiques décrites précédemment. Nous nous intéressons d'abord à la littérature scientifique (section 2.1) du domaine, puis ensuite aux logiciels existants (section 2.2).

2.1 Littérature

2.1.1 Construction de graphiques de circulation

Dans [6, 7], les auteurs s'intéressent à la construction d'horaires périodiques pour une voie unidirectionnelle Une représentation sous forme de multigraphe orienté, dans lequel les nœuds correspondent aux dates d'arrivée et de départ des circulations, sert de base pour l'écriture d'un programme linéaire en nombres entiers. Ce modèle est résolu en y appliquant une relaxation lagrangienne.

Le problème décrit dans [12] consiste à déterminer un ordonnancement de circulations au niveau d'un terminus, où les rames partant du dépôt et celles faisant demi-tour sont en concurrence pour l'utilisation des voies. Ces rames doivent par ailleurs respecter des contraintes d'espacement entre deux circulations consécutives (*headway*) variant en fonction des périodes. En raison de la précision exigée par une telle problématique, la modélisation est faite au niveau des circuits de voie. Le problème, qui peut être vu comme la détermination d'un horaire et l'assignation d'une route pour chaque circulation, est résolu par programmation par contraintes.

Dans [30], l'auteur s'intéresse plus particulièrement à l'ordonnancement de circulations sur un point de jonction. Comme précédemment, les auteurs proposent une modélisation au niveau des circuits de voie par programmation par contraintes. Une variante relâchant les contraintes modélisant l'accélération et le freinage des circulations est aussi proposée. Les résultats présentées, basées sur une étude de cas réel, montrent que la méthode proposée peut être exploitée.

Dans [35, 34, 36], les auteurs proposent une modélisation basée sur le problème de *Node Packing*. La résolution est ensuite effectuée par *Branch-and-Cut*, après application de pré-traitements. Des résultats numériques sont présentés à partir d'instances issues du réseau ferroviaire Néerlandais.

Il faut aussi souligner que, si de nombreux modèles mathématiques ont été proposés (principalement de type programmation linéaire en nombres entiers), la

complexité réelle de ces problèmes ne permet pas la modélisation de l'ensemble des contraintes. Par exemple, la variabilité du matériel statique et roulant n'est pas considérée. Il est alors souvent nécessaire de procéder à une vérification et une correction du graphique obtenu, généralement à l'aide de simulations (voir par exemple SISYFE, LIPARI [17] et THOR, section 2.2).

Enfin, le problème de constructions d'horaires cycliques est traité dans [24] et [28]. Un modèle linéaire en nombres entiers est proposé dans [24]. Il est résolu par une décomposition du problème.

2.1.2 Gestion des aléas

Dans [10], les problèmes de construction d'un ordonnancement des circulations et de re-planification en présence d'aléas sont traités par un modèle unique. Ce modèle est résolu au moyen d'un algorithme génétique. Dans le cas de la re-planification, l'objectif considéré ici est la minimisation du temps d'attente des passagers.

La programmation par contraintes est utilisée dans [11] pour la modélisation du problème de re-planification. Deux heuristiques sont proposées pour guider la recherche.

Dans [14], les auteurs exploitent une représentation du problème sous forme de *job-shop* augmentée de contraintes *no-wait* et *no-store*. Dans [26], le problème est modélisé comme un graphe prenant en compte les temps de séparation entre circulations. Le modèle est résolu par séparation et évaluation progressive, et des règles de déduction sont mises à profit pour accélérer la résolution.

Une approche par programmation par contraintes est utilisée dans [15]. L'auteur se base sur une modélisation au niveau cantonnement, en prenant en compte la signalisation. Deux stratégies sont proposées pour la résolution des conflits.

L'heuristique HOAT proposée dans [31, 33] repose sur l'observation que les circulations directement affectées par la perturbation initiale ainsi que la circulation à l'origine de la perturbation sont les plus susceptibles d'être perturbées. Il est donc probable que leurs interactions avec les autres circulations doivent être modifiées afin de limiter les effets de la perturbation, tant dans les délais induits que dans le nombre de circulations affectées.
Les règles définies par HOAT sont les suivantes. L'affectation initiale des voies aux circulations n'est pas conservée. La séquence des circulations à chaque segment est, à quelques exceptions près, conservée. Ces exceptions consistent à permettre une modification partielle de la séquence, en autorisant le dépassement de la circulation à l'origine de la perturbation et celles directement affectées par une ou plusieurs autres circulations.

2.2 Logiciels disponibles

Le tableau 2.1 présente les caractéristiques de différents logiciels. À chaque ligne du tableau correspond un logiciel. La première colonne indique le nom du logiciel en question. Entre parenthèses se trouvent le numéro de section où une description plus détaillée peut être trouvée. Dans la mesure des informations que nous avons obtenues nous indiquons en deuxième colonne le caractère prospectif (étude en amont) ou opérationnel (utilisation « sur le terrain ») du logiciel, en

troisième la technique de résolution employée, et en quatrième colonne les principales fonctionnalités du logiciel. Le signe " ?" indique l'absence d'informations dans la colonne correspondante.

Une présentation détaillée de logiciels d'optimisation ferroviaire, enrichie de captures d'écran, peut être trouvée à l'adresse suivante : http://www.dsic.upv.es/docs/bib-dig/informes/etd-01152007-140458/AutomatedSystems.pdf[2].

2.2.1 Démiurge

http://recherche.sncf.com/approfondir_les_connaissances/l_optimisation_mathematique/projets/projets_art5_1.html

Démiurge est un outil d'aide à l'étude de capacité de réseau ferroviaire. Il est utilisé par la S.N.C.F pour :

- évaluer la capacité d'un réseau à absorber de nouvelles circulations ;
- situer les goulots d'étranglement ;
- apporter une aide à la décision sur les investissements liés à l'infrastructure ;
- optimiser les grilles horaires présentes et futures ;
- calculer la capacité résiduelle d'une grille horaire.

Le problème à résoudre est un problème d'optimisation sous contraintes résolu par un algorithme de programmation linéaire en variables mixtes. La résolution peut être exacte ou approchée, par décomposition du problème en sous-problèmes interdépendants.

2.2.2 SISYFE : SImulateur du SYstème Ferroviaire

(voir aussi [19])

SISYFE est utilisé pour la simulation de l'évolution d'un réseau ferroviaire en situation normale ou perturbée. Ses utilisations sont :

- évaluer la robustesse de diagrammes de circulations ;
- étudier sur les plans techniques et économiques la modification d'installations, l'utilisation de nouveaux systèmes de contrôle-commande, etc. ;
- définir des stratégies opérationnelles ;
- valider des systèmes de gestion opérationnelle et de supervision du trafic ;
- entraîner les combinateurs et les régulateurs.

SISYFE utilise pour cela la simulation par événements discrets, en reproduisant le fonctionnement des appareils de signalisation et des équipements mobiles.

2.2.3 LIPARI

(voir aussi [17])

Lipari sert à :

- évaluer, par simulation, la capacité d'un système de contrôle-commande à supporter un trafic extrêmement dense, et atteindre un seuil de ponctualité suffisant ;
- définir l'architecture cible d'un nouveau système de gestion opérationnelle de trafic basé sur le système de contrô-commande ERTMS 2 et des modules de fluidification et de contrôle de trafic semi-automatique ;

Nom	Cadre	Type de résolution	Fonctionnalités
Démiurge 2.2.1	prospectif	programmation linéaire mixte	outil d'optimisation et de mesure de la capacité du réseau ferroviaire
SISYFE 2.2.2	prospectif	simulation à événements discrets	simulation de l'évolution des trains sur le réseau ferroviaire
LIPARI 2.2.3	prospectif	simulation, programmation linéaire, heuristiques	simulation des systèmes de « contrôle-commande » et de gestion opérationnelle
VIRIATO 2.2.4	prospectif	?	modules de visualisation (netgraph, graphes de circulations, horaires) et de gestion des circulations
CAPRES 2.2.5	prospectif	?	élaboration d'horaires par saturation du réseau ferroviaire
Excalibur	opérationnel	?	collecte, visualisation et diffusion d'informations liées à l'apparition d'incidents, aide à la décision
AGORA	opérationnel	?	gestion des informations du chef d'escale aux acteurs nomades
Gédéon	opérationnel	?	suivi des circulations (fret)
Ecler	opérationnel	?	système d'aide à l'affectation des rames
Colt	opérationnel	?	pilotage et supervision de lignes
Sardaigne	prospectif	statistiques	évaluation statistique de la pertinence des simulations
PRaCoSy	prospectif	optimisation sous contraintes	réoptimisation

TAB. 2.1: Synthèse des différents logiciels ferroviaires.

– quantifier par simulation les gains en ponctualité d'un système cible par rapport à un système de référence.

Lipari utilise la programmation linéaire et des heuristiques, et s'interface avec SISYFE pour la simulation proprement dite.

2.2.4 VIRIATO

VIRIATO est principalement utilisé pour la planification stratégique. Il permet d'adapter l'infrastructure aux évolutions des services, et de coordonner les opérateurs et les utilisateurs d'une même infrastructure. Il peut servir à :

– définir une planification initiale de circulations cadencées ;
– aider à l'analyse de circulations uniques ;
– déterminer le taux de saturation d'une ligne ;
– compresser une grille horaire pour déterminer la saturation d'une ligne ou d'une portion de la ligne.

De plus, VIRIATO permet l'ajout de modules pour enrichir la gamme des problématiques traitées.

2.2.5 CAPRES

(voir aussi [25])

CAPRES est un module de VIRIATO destiné à concevoir et saturer des variantes de grilles horaires. Il permet ainsi différentes actions :

– évaluer la capacité non utilisée d'un réseau ferroviaire ;
– détecter les goulots d'étranglement de la capacité ;
– comparer la capacité de différentes structures de grilles horaires ;
– étudier l'impact de l'ajout de nouvelles lignes dans un réseau ;
– concevoir des variantes de grilles horaires ;
– estimer les effets de modifications d'infrastructures, du matériel roulant ou des paramètres de contrôle sur la capacité ;
– étudier la faisabilité d'une grille horaire pour un réseau donné.

2.2.6 Sardaigne (Statistiques appliquées à la rationalisation des décisions pour aider à gagner en efficacité)

http://recherche.sncf.com/approfondir_les_connaissances/l_optimisation_mathematique/projets/projets_art4_1.html

Sardaigne propose une démarche statistique pour le choix des simulations. Sardaigne est utilisé en amont et en aval d'un simulateur (SISYFE). En amont, il aide à définir les configurations et incidents à tester. En aval, il agrège et pondère les résultats et fournit des résultats statistiques sur la pertinence des simulations effectuées.

Chapitre 3

Modélisation par programmation par contraintes selon le formalisme ordonnancement

3.1 Le formalisme ordonnancement

Les problèmes d'ordonnancement constituent un des principaux domaines d'étude de la recherche opérationnelle. Dans cette section, nous commençons par définir section 3.1.1 les problèmes d'ordonnancement et les différents éléments qui les caractérisent. Puis, nous présentons section 3.2.1 la programmation par contraintes, paradigme de représentation et de résolution qui s'est souvent montré efficace pour la résolution des problèmes d'ordonnancement.

3.1.1 Définition des problèmes d'ordonnancement

On définit le plus souvent un problème d'ordonnancement comme étant formé d'un ensemble de tâches (ou opérations), dont on cherche à déterminer des dates d'exécution (dates de début et de fin). Ceci doit se faire en général en optimisant un certain critère, et en affectant éventuellement un ensemble de ressources (ou machines) aux tâches.

Une tâche est une entité définie par ses dates de début et de fin, sa durée d'exécution et sa demande pour chacune des ressources. Lorsque le problème est dit préemptif, la réalisation d'une tâche peut être faite en plusieurs fois. Sinon, le problème est dit non-préemptif, et la tâche doit être effectuée d'une seule traite.

Une ressource est caractérisée par son type et sa disponibilité. Selon la nature du problème, une ressource peut être renouvelable ou consommable. Elle est renouvelable quand la quantité de ressource allouée pour l'exécution d'une tâche est remise à disposition dans son intégralité à la fin de l'exécution (resources humaines par exemple), et elle est consommable quand la quantité de ressources n'est pas restituée en fin d'exécution (matières premières par exemple). Parmi

les ressources renouvelables, on distingue principalement les ressources disjonctives, de capacité unitaire. Elles ne permettent d'exécuter qu'une tâche à la fois. Les ressources cumulatives quant à elles autorisent la réalisation de plusieurs tâches simultanément dans la limite de leur capacité. On trouve dans certains problèmes des tâches dites génératrices, qui permettent la production d'une quantité donnée de ces ressources. Celle-ci est généralement associée à des ressources non-renouvelables. La disponibilité des différentes ressources peut varier au cours du temps, notamment dans le cas des ressources consommables.

La résolution du problème consiste à définir la date de début et de fin de toutes les opérations passant sur les machines. La séquence obtenue doit respecter des contraintes de succession, et des contraintes de disponibilité des ressources induites par la capacité de celles-ci.

Compte tenu de la diversité des applications des problèmes d'ordonnancement, de nombreuses fonctions objectifs ont été proposées. Les plus fréquentes sont la minimisation de la durée totale d'exécution (*makespan*) ou les retards (retard maximum, somme des retards, etc.)

Les problèmes d'ordonnancement sont dans leur grande majorité des problèmes fortement combinatoires, et NP-difficile au sens fort. Les nombreux travaux de recherche réalisés ont permis le développement d'une classification précise et de plusieurs nomenclatures.

Les deux classes de problèmes les plus connues et étudiées sont les problèmes d'ateliers. Plus particulièrement, les problèmes de *job-shop*, *flow-shop* et *open-shop*, et le problème d'ordonnancement de projets sous contraintes de ressources (*RCPSP*, pour *resource constrained project scheduling problem*). Ce dernier regroupe sous une forme généralisée de nombreuses variantes.

3.2 Programmation par contraintes et Ordonnancement

3.2.1 Programmation par contraintes

Dans ce document, nous utilisons par abus de langage l'appellation « programmation par contraintes » pour désigner à la fois le formalisme de représentation que sont les réseaux de contraintes et le paradigme de résolution qu'est la programmation par contraintes.

Réseaux de contraintes

Les réseaux de contraintes sont un des formalismes utilisables pour la représentation de problèmes d'optimisation combinatoire. Le problème est ainsi modélisé par un ensemble de variables disposant chacune d'un domaine de validité. Des relations nommées contraintes lient des sous-ensembles de variables, en limitant les affectations possibles pour ces variables. Une solution du problème est obtenue lorsqu'une valeur du domaine est affectée à chaque variable, et que l'affectation de l'ensemble des variables vérifie toutes les contraintes du modèle. Pour effectuer cette affectation, le paradigme couramment utilisé est la programmation par contrainte.

Programmation par contraintes

La programmation par contraintes s'appuie sur les contraintes du modèle pour la recherche d'une solution. La résolution fait intervenir trois mécanismes :
- le filtrage ;
- la propagation ;
- l'exploration.

Le filtrage consiste à éliminer du domaine d'une variable les valeurs inconsistantes. Différents algorithmes de filtrage ont été proposés dans la littérature. Mais il est nécessaire de trouver un compromis entre le temps nécessaire à l'application des règles de filtrage, et l'efficacité du filtrage.

Des mécanismes de propagation sont ensuite utilisés en complément du filtrage pour propager les éventuelles réductions de domaine, c'est-à-dire pour prendre en compte ces réductions avec les autres variables et contraintes du modèle.

Filtrage et propagation ne sont toutefois que rarement suffisantes pour trouver une solution au problème, ou prouver la non-existence de solution. On doit alors effectuer une exploration de l'espace de recherche, en réduisant arbitrairement le domaine d'une (ou plusieurs) variable(s) (phase d'affectation). A chaque exploration les opérations de filtrage et de propagation sont reitérées.

3.3 Modélisation au niveau voie

Nous montrons maintenant que les problèmes de gestion ferroviaire peuvent être représentés comme des problèmes d'ordonnancement.

Nous commençons section 3.3.1 par transposer le problème d'optimisation ferroviaire en un problème d'ordonnancement, en faisant la correspondance avec les tâches, ressources et contraintes qui le définissent. Nous détaillons ensuite la modélisation des contraintes du problème par programmation par contraintes (section 3.3.2).

3.3.1 Reformulation

La circulation d'un train dans un réseau ferroviaire peut être vue comme l'occupation successive de points du réseau par la circulation durant un intervalle de temps. Cette occupation est caractérisée par les dates d'entrée et de sortie de la circulation, autrement dit l'horaire de la circulation au point considéré. Le passage d'une circulation en un point du réseau définit donc une tâche. L'ordonnancement est ici non-préemptif.

Lorsqu'une circulation emprunte un point du réseau, il est nécessaire de lui affecter une voie. Un point du réseau peut être composé d'une ou plusieurs voies. S'il semble possible de représenter l'ensemble des voies comme une ressource cumulative, la nécessité de pouvoir distinguer les voies (granularité moyenne, voir section 1.2.2) et la non-homogénéité de celles-ci nous oblige à choisir une autre représentation. En effet, à chaque voie correspond une orientation, un temps de parcours et un espacement minimum. Chaque voie est donc assimilée à une ressource, et les voies de même orientation, auquel nous ajoutons les voies banalisées, forment des ensembles de ressources alternatives. Dans le cas des nœuds, les voies ne peuvent (sauf cas particulier des associations de trains) accueillir

qu'une seule circulation à la fois. Ce sont donc des ressources disjonctives, i.e. de capacité unitaire. Ce n'est pas le cas des voies au niveau des tronçons. En effet, dans la modélisation que nous avons choisi d'utiliser, la gestion des distances de sécurité entre circulations est effectuée par le biais de temps de séparation minimaux entre circulations empruntant une même voie. Les ressources ne sont alors pas disjonctives car plusieurs circulations peuvent se trouver simultanément sur la même voie, si leurs horaires respectent les temps de séparation . Ceci est illustré sur la figure 3.1. Dans cette figure les rectangles pleins représentent l'utilisation de plusieurs cantons par deux circulations (distingués par la profondeur du gris), et les zones hachurés représentent l'occupation d'un tronçon regroupant plusieurs de ces cantons. La présence simultanée des deux circulations dans le tronçon est mise en évidence par le chevauchement des zones hachurées, tandis que le non chevauchement des rectangles pleins illustrent le caractère disjonctif des cantons.

L'itinéraire d'un train se définit comme l'ordre des points parcourus par une circulation. L'entrée de la circulation en un point de l'itinéraire est directement consécutive à la sortie du point qui le précède. En se ramenant au formalisme ordonnancement, l'itinéraire peut donc être assimilé à une gamme de production, et l'ordonnancement est de type *no-wait*. Une contrainte *no-wait* est ainsi insérée pour toute paire d'opérations consécutives de la gamme, et impose que ces opérations soient traitées sans temps d'attente. On est aussi, dans le cas des ressources de capacité unitaire, en présence de contraintes de type *blocking no swap*. Une ressource n'est libérée que lorsque l'activité consécutive débute (*blocking*), et le caractère *no swap* traduit le fait que la permutation simultanée de deux activités sur deux ressources ne peut avoir lieu (du fait notamment des contraintes d'espacement).

Compte tenu du niveau de représentation choisi, les espacements minimaux entre circulations sont représentés par des temps de transition entre tâches. Ces temps de transition dépendent de plusieurs paramètres :

- voie affectée ;
- type des circulations ;
- sens des ciculations.

On est ainsi en présence de temps de transition dépendant de la séquence et des ressources (*resource and sequence dependant transition time*).

3.3.2 Modélisation

Nous nous intéressons dans cette partie à la modélisation selon le formalisme de la programmation par contraintes des principales contraintes des problèmes de gestion ferroviaire (c'est-à-dire des contraintes communes aux différentes problématiques que nous abordons sections 4.2, 4.3 et 4.4).

Une tâche t est représentée ci-dessous par les notations suivantes :

S_t : date de début de la tâche t ;

F_t : date de fin de la tâche t ;

p_t : durée de la tâche t, $p_t = F_t - S_t$, dépendant du choix de la ressource affectée à la tâche ;

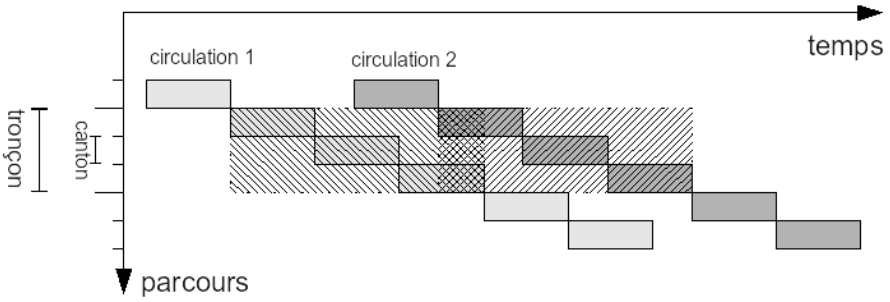

FIG. 3.1: Exemple de recouvrement des intervalles temporels d'utilisation d'un tronçon.

r_t : ressource affectée à la tâche t.

Contraintes de ressources

Elles permettent de tenir compte de la nécessité d'affecter une ressource à chaque tâche. Comme nous l'avons expliqué précédemment, nous associons à chaque point deux ensembles de ressources alternatives, un par orientation. On a alors, pour une tâche t nécessitant une ressource r_t d'un ensemble R_t de ressources alternatives, la relation $r_t \in R_t$.

Contraintes potentielles

Elles correspondent aux contraintes de parcours et aux circulations associées. Elles permettent de s'assurer que deux tâches consécutives t_1 et t_2 d'une circulation soient exécutées sans temps d'attente.

Dans le cas des contraintes de parcours, on a, pour t_1 précédant t_2, noté par la suite $t_1 \prec t_2$:

$$F_{t_1} = S_{t_2}$$

L'association d'une circulation t_2 à une circulation t_1 impose une durée minimale d'arrêt de la circulation t_2, et le départ simultané des deux circulations. Elle est exprimée comme suit :

$$p_{t_2} \geq tpsminstat(t_2) \quad \wedge \quad F_{t_1} = F_{t_2}$$

La contrainte peut être renforcée en imposant l'arrivée de la circulation t_2 après la circulation t_1 :

$$S_{t_1} < S_{t_2}$$

On va de plus limiter les affectations des ressources aux tâches :

$$r_{t_1}, r_{t_2} \in R_{t_1} \cap R_{t_2}$$

Contraintes disjonctives

Elles incluent les contraintes d'espacement aux nœuds (dans lesquels les voies sont assimilées à des ressources unitaires), et d'itinéraires incompatibles. Nous y ajoutons aussi les contraintes d'espacement dans les tronçons.

Soient t_1, t_2 deux tâches utilisant la même ressource r de capacité unitaire (i.e., une voie d'un nœud). On a alors :

$$S_{t_2} \geq F_{t_1} + espnoeud(t_1, t_2, r) \quad \vee \quad S_{t_1} \geq F_{t_2} + espnoeud(t_2, t_1, r)$$

Soient t_1 et t_2 deux tâches utilisant la même ressource r correspondant à une voie d'un tronçon. Si c_1 et c_2 (circulations associées à t_1 et t_2) sont de même sens, on a, pour $t_1 \prec t_2$:

$$S_{t_2} \geq S_{t_1} + tpsespmin(t_1, t_2, r) \quad \wedge \quad F_{t_2} \geq F_{t_1} + tpsespmin(t_1, t_2, r)$$

Si c_1 et c_2 circulent en sens contraire, on a, pour $t_1 \prec t_2$:

$$S_{t_2} \geq F_{t_1} + tpsespmin(t_1, t_2, r)$$

Le même type de contraintes se retrouve pour le cas de circulations t_1 et t_2 empruntent des itinéraires incompatibles dans un même nœud, on a, pour $c_1 \prec c_2$ les contraintes suivantes :

– nœud de type bifurcation

$$S_{t_2} \geq F_{t_1} + espitibif(t_1, t_2, r)$$

– nœud de type gare, itinéraires incompatibles en entrée

$$S_{t_2} \geq S_{t_1} + espitiEE(t_1, t_2, r)$$

– nœud de type gare, itinéraires incompatibles en sortie

$$F_{t_2} \geq F_{t_1} + espitiSS(t_1, t_2, r)$$

– nœud de type gare, itinéraires incompatibles en entrée/sortie

$$F_{t_2} \geq S_{t_1} + espitiES(t_1, t_2, r)$$

– nœud de type gare, itinéraires incompatibles en sortie/entrée

$$S_{t_2} \geq F_{t_1} + espitiSE(t_1, t_2, r)$$

Contraintes conjonctives

L'échange de voyageurs implique la mise en conjonction des tâches associées. On considère ici la relève comme une forme de conjonction « simplifiée » (elle s'apparente peut-être plus à une forme de contrainte potentielle).

La relève de voyageurs implique qu'une tâche doit respecter un délai (minimum et maximum) entre sa date de fin et la date de début de la tâche en conjonction.

$$tpscorrmin(t_1, t_2) \leq F_{t_2} - S_{t_1} \leq tpscorrmax(t_1, t_2)$$

L'échange des voyageurs impose que les deux tâches soient mises en conjonction, c'est-à-dire que les tâches doivent sur un intervalle de temps de longueur bornée être en cours d'exécution simultanément :

$$\begin{array}{rl} tpscorrmin(t_1, t_2) \leq & F_{t_2} - S_{t_1} \leq tpscorrmax(t_1, t_2) \\ & \vee \\ tpscorrmin(t_1, t_2) \leq & F_{t_1} - S_{t_2} \leq tpscorrmax(t_1, t_2) \end{array}$$

Contraintes de durée

Les contraintes de temps de parcours limite la durée d'une tâche. Soit t une tâche, et r la ressource affectée à cette tâche. On a :

$$p_t \geq tpsparcmin(t, r)$$

Dans le cas où la tâche correspond à la traversée d'un nœud sans arrêt, on a $p_t = 0$.

La desserte d'un nœud impose à la tâche correspondante une borne sur sa durée :

$$p_t \geq tpsdessmin(t, r) \quad \wedge \quad p_t \leq tpsdessmax(t, r)$$

Chapitre 4

Résolutions numériques

Nous avons présenté dans le chapitre précédent les modèlisations génériques comportant les contraintes communes aux problèmatiques que nous abordons aux sections 4.2, 4.3 et 4.4. Chacune de ces problèmatiques a ses propres objectifs, et eventuellement des contraintes supplémentaires comme dans le cas de la recherche d'ordonnancement cyclique. La résolution numérique exacte des modèles résultants peut s'envisager en utilisant un outil dédié à la résolution des problèmes de programmation par contraintes comme Ilog Solver, moyennant la conception d'algorithme de filtrage. Cependant, l'expérience numérique montre que la taille des problèmes en jeu rend inefficace la recherche de solution exacte même avec des outils logiciels aussi performant qu'Ilog Scheduler. Des approches heuristiques ou métaheuristiques ont alors été élaborées. Nous donnons dans cette section (section 4.1.1) une brève présentation de ces techniques. Nous reportons ensuite section 4.2, section 4.3, section 4.4 les algorithmes utilisés pour les différents types de problèmes traités.

4.1 Heuristiques et métaheuristiques

4.1.1 Heuristiques et métaheuristiques

Métaheuristiques

Les métaheuristiques forment une famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation NP-difficile, pour lesquels on ne connaît pas de méthode (exacte) efficace. Elles permettent de déterminer des solutions de bonne qualité en un temps raisonnable mais sans garanti d'optimalité.

Les métaheuristiques sont généralement des algorithmes itératifs, qui progressent vers un optimum local, que l'on espère proche de l'optimum global.

Il existe un grand nombre de métaheuristiques, allant de la simple recherche locale à des algorithmes complexes de recherche globale. Ces méthodes utilisent cependant un haut niveau d'abstraction, leur permettant d'être adaptées à une large gamme de problèmes différents.

Heuristiques

Comme pour les métaheuristiques, les heuristiques ont pour but de déterminer rapidement une solution réalisable, non nécessairement optimale, pour un problème d'optimisation NP-difficile. La différence réside dans le fait qu'une heuristique est conçue pour un problème particulier, en s'appuyant sur sa structure et ses spécificités propres.

4.1.2 Méthodes de voisinage

La plupart des heuristiques et métaheuristiques reposent sur la notion de voisinage d'une solution que l'on peut définir comme suit.

Soit P un problème d'optimisation NP-difficile à n variables, et S une solution de P. Un voisinage V de (P, S) est l'ensemble des solutions atteignables à partir de S en fixant $n - p$ variables à leur valeur courante dans S, et en faisant varier les valeurs affectées aux p variables non fixées.

Suivant la taille de p, on parlera de méthodes de voisinage (p petit) ou de grand voisinage (p grand). Dans le cas de voisinage de petite dimension, l'énumération exhaustive des solutions voisines de S peut être envisagée, généralement par application d'opérations de transformation. Dans le cas de grands voisinages, la taille du voisinage rend la recherche exhaustive des solutions voisines trop coûteuse. Des techniques d'énumération implicites ou des approches heuristiques sont alors utilisées pour la construction de V.

L'idée générale des heuristiques et métaheuristiques est que la valeur optimale recherchée peut être déterminée par exploration de voisinages successifs. Partant d'un point initial donné, un voisinage de ce point est considéré dans lequel une (ou plusieurs) nouvelles solutions sont choisis suivant un critére à déterminer. Ce ou ces nouveaux points constituent alors les nouveaux de points de départ à partir desquels d'autres voisinages sont explorés et ainsi de suite,... Le type de voisinage considéré et la manière d'explorer ces voisinages caractérisent les schémas heuristiques ou métaheuristiques ainsi que leur capacité à atteindre la solution optimale. Deux types d'exploration sont communément utilisés : l'intensification et la diversification.

L'intensification a pour rôle de privilégier l'exploration des voisinages des meilleures solutions connues. L'effort de calcul va être orienté principalement vers l'exploration du voisinage de la solution courante, et la détermination des meilleurs voisins. L'objectif est ici d'explorer de façon intensive les zones de l'espace qui semblent prometteuses.

À l'opposé, la diversification consiste à explorer d'autres parties de l'espace des solutions. La diversification est essentielle pour pouvoir s'extraire des optimums locaux dans lesquels les mécanismes d'intensification peuvent se retrouver bloquer. Elle consiste à perturber fortement la solution courante (voire à générer de façon aléatoire une nouvelle solution de départ), en s'autorisant généralement à dégrader la valeur de cette solution.

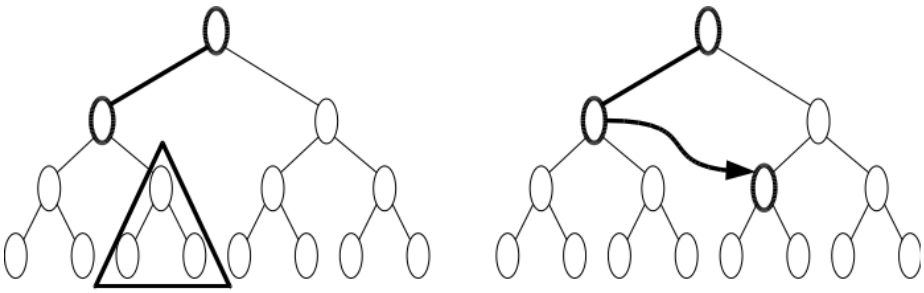

FIG. 4.1: Espace des solutions : intensification (gauche) vs diversification (droite).

Nous illustrons figure 4.1 les espaces de recherche explorés par intensification (à gauche) et diversification (à droite).

4.2 Construction du graphique de circulation

Comme nous avons déjà pu le dire, la construction du graphique de circulation consiste à déterminer les horaires d'un ensemble de circulations empruntant un réseau ferroviaire. Plus précisément, le problème revient à déterminer les dates d'entrée et de sortie des circulations aux différents points de leurs itinéraires, et la voie qui leur est affectée (voir aussi section 2.1.1 et [3]).

Nous nous intéressons dans cette section plus particulièrement aux stratégies de recherche, les contraintes composant le modèle étant celles présentées section 3. Nous commençons section 4.2.1 par présenter les algorithmes de filtrage, permettant la réduction des domaines des variables, puis les algorithmes d'affectation, qui interviennent dans le parcours de l'arbre de recherche (section 4.2.2). Nous nous intéressons enfin section 4.2.3 à des stratégies de recherche heuristiques pouvant être mis en place pour l'obtention de solutions de bonne qualité dans un temps limité.

4.2.1 Algorithmes de filtrage

Les algorithmes de filtrage (ou de propagation) ont pour but de réduire le domaine des variables en en supprimant les valeurs non consistantes.

L'utilisation d'une bibliothèque dédiée comme ILOG Solver et ILOG Scheduler [22] permet de masquer la complexité de l'implantation et la gestion des algorithmes de propagation. Nous présentons ici les différentes types de filtrage des contraintes et les algorithmes proposés par ILOG. Nous nous limitons ici aux types utilisés dans notre implantation. Nous précisons en **gras** le niveau actuellement utilisé.

Contraintes de capacité

Ces contraintes permettent de fixer et de gérer, pour les ressources discrètes leurs capacités minimales et maximales. Le tableau 4.1 indique les algorithmes utilisés selon le niveau de propagation demandé.

Niveau de traitement de la capacité	Algorithmes
Ressources discrètes :	
IloLow, IloMediumLow, **IloBasic**	Timetable
IloMediumHigh	Timetable + Disjunctive
IloHigh	Timetable + Disjunctive + Edge finder
IloExtended	Timetable + Disjunctive + Edge finder + Balance
Ressources unitaires :	
IloLow	Timetable
IloMediumLow	Light Precedence Graph
IloBasic	Light Precedence Graph + Disjunctive
IloMediumHigh	Light Precedence Graph + Disjunctive + Edge finder
IloHigh	Light Precedence Graph + Disjunctive + Edge finder
IloExtended	Light Precedence Graph + Disjunctive + Edge finder + Balance

TAB. 4.1: Filtrages des contraintes de capacité.

Contraintes de précédence

Le traitement des précédences est lié à l'effort fourni par le solveur pour analyser les relations de précédence entre les contraintes de ressource d'une ressource. ILOG Scheduler utilise pour cela le graphe de précédence. Dans le cas des ressources discrètes ou unitaires, des algorithmes spécifiques peuvent être appliqués pour prendre en compte simultanément les limites de capacité et la connaissance du graphe de précédence. Le tableau 4.2 indique les algorithmes utilisés selon le niveau de propagation demandé.

Le traitement des séquences permet de considérer les éventuelles synchronisations entre exécution de tâches et de prendre en compte si besoin les temps ou coûts de transition. Cela ne concerne que les ressources unitaires et impose l'existence d'un graphe de précédence ou de contraintes de séquence. Le tableau 4.3 indique les algorithmes utilisés selon le niveau de propagation demandé.

Contraintes horaires

Le traitement des temps de transition correspond à l'effort fourni par le solveur pour tenir compte des temps de transition entre tâches s'exécutant sur une même ressource. Le tableau 4.4 indique les algorithmes utilisés selon le niveau de propagation demandé.

Le traitement des durées permet la prise en compte des durées variables des tâches. Le tableau 4.5 indique les algorithmes utilisés selon le niveau de propagation demandé. L'algorithme de *edge-finding* a été présenté pour la première fois dans [29], on trouvera une application pour le problème de *job-shop* dans [8], et une présentation des algorithmes de *edge-finding* pour les problèmes d'ordonnancement est faite dans [1].

Après application des algorithmes de filtrage, trois cas de figure se présentent. Soit le domaine d'une des variables est réduit à l'ensemble vide, et dans ce cas le problème est irréalisable. Soit le domaine de chaque variable est réduit à un singleton, et on dispose dans ce cas d'une solution réalisable au problème. Soit le domaine d'au moins une variable n'est pas réduit à un singleton, et il est alors nécessaire d'effectuer une réduction arbitraire du domaine de cette variable afin de déclencher de nouvelles propagations. Le complémentaire est ensuite exploré afin d'assurer la complétude de la recherche. On se retrouve ainsi dans un mécanisme de recherche arborescente géré par des algorithmes d'affectation.

4.2.2 Algorithmes d'affectation

La sélection d'une variable et la réduction de son domaine sont gérées sous ILOG Solver et ILOG Scheduler par l'utilisation d'un classe appelée *IloGoal*. Là encore, les bibliothèques fournissent un ensemble d'algorithmes de sélection et d'affectation. Nous détaillons ici les algorithmes utilisés pour la recherche d'une solution.

Ressources Pour les ressources, le critère de choix principal réside dans l'affectation des tâches aux ressources. Nous utilisons pour cela la fonction

Niveau de traitement des précédences	Algorithmes
Ressources unitaires et discrètes :	
IloLow, IloMediumLow, **IloBasic**	No global constraint
IloMediumHigh	Precedence Graph
IloHigh	Precedence Graph + Level 1
IloExtended	Precedence Graph + Level 2

TAB. 4.2: Filtrages des relations de précédence.

Niveau de traitement des séquences	Algorithmes
Ressources unitaires :	
IloLow, IloMediumLow, **IloBasic**	No global constraint
IloMediumHigh	Precedence Graph
IloHigh, IloExtended	Sequence Constraint
Autres types de ressources :	
Does not apply.	

TAB. 4.3: Filtrages des séquences.

Niveau de traitement des transitions	Algorithmes
Ressources unitaires :	
IloLow, IloMediumLow, **IloBasic**	Light Precedence Graph
IloMediumHigh, IloHigh, IloExtended	Disjunctive Constraint
Autres types de ressources :	
IloLow, IloMediumLow, **IloBasic**, IloMediumHigh, IloHigh, IloExtended	Type Timetable

TAB. 4.4: Filtrages des temps de transition.

Niveau de traitement des durées	Algorithmes
Ressources discrètes :	
IloLow, IloMediumLow, **IloBasic**	No global constraint
IloMediumHigh, IloHigh, IloExtended	Extra timetable propagation
Autres types de ressources :	
IloLow, IloMediumLow, **IloBasic**, IloMediumHigh, IloHigh, IloExtended	No global constraint

TAB. 4.5: Filtrages des contraintes de durée.

`IloAssignAlternative`,

qui permet d'affecter à une activité une ressource prise parmi un ensemble de ressources alternatives. En l'absence d'informations sur des stratégies métiers ou des contraintes opérationnelles, nous utilisons le sélecteur de ressources par défaut, qui affecte à une circulation la première de ses ressources disponibles.

Un autre critère d'affectation concerne l'ordre d'utilisation des tâches affectées à une ressource, ordre qui peut être géré au moyen des fonctions

`IloRank{Forward,Backward}` et `IloSequence{Forward,Backward}`.

Toutefois, dans les problématiques que nous traitons ici, ce sont surtout les ordres des circulations elles-mêmes qui importent. Nous n'utilisons donc pas ce type d'algorithme, qui se sont montrés dans les tests préliminaires coûteux et sans efficacité notable.

Tâches Une fois les ressources allouées aux tâches, il nous faut encore définir les dates de début et de fin de chaque tâche, leur durée, et l'ordre des tâches utilisant la même ressource. Notons que si les dates de début et de fin sont fixées, alors les durées et l'ordre des tâches sont aussi connues.

Nous utilisons pour cela la fonction `IloSetTimesForward` qui permet d'attribuer une date de début aux tâches, en cherchant à minimiser un critère, ici la date de fin du projet. Ceci nous permet, d'une part de fixer de façon indirecte les dates de fin des tâches, d'autre part de ne pas nous préoccuper de leur durée d'exécution.

Ces algorithmes de filtrage et d'exploration se montrent sur les instances de test globalement inefficaces, dès que la taille des instances grandit. L'obtention de solutions dans un temps court, l'amélioration des solutions trouvées ou la preuve d'optimalité sont inconcevables avec de telles approches. De plus, la présence de voies banalisées augmente fortement l'aspect combinatoire du problème, et les instances, même de taille réduite, peuvent être extrêmement difficiles à résoudre. Des méthodes heuristiques doivent donc être développées pour la détermination de solutions de bonne qualité en temps raisonnable. Ces heuristiques se basent sur la construction de voisinages d'une solution courante, que nous présentons ci-dessous.

4.2.3 Stratégies heuristiques de recherche

L'utilisation d'heuristiques ou de méta-heuristiques n'est pas chose nouvelle . La plupart des articles cités précédemment en font usage [6, 7, 10, 12, 32], ce même lorsque la problématique considérée peut être supposée plus simple (planification des circulations sur un corridor par exemple). En effet, la nature fortement combinatoire des problèmes traités conduit à une augmentation significative du nombre de variables et contraintes. La résolution exacte de ces problèmes ne peut être alors envisagée dans un temps raisonnable.

En l'absence d'instances réelles, d'informations fournies ou d'études disponibles sur les pratiques métiers permettant le développement d'heuristiques basées sur celles-ci, nous nous intéressons ici à l'application d'heuristiques classiques. Nous présentons section 4.2.3 l'algorithme que nous avons utilisé pour la construction de notre solution initiale.

La recherche d'une meilleure solution se base sur la notion de voisinage, c'est-à-dire d'un sous-espace de l'espace des solutions, dans lequel va s'effectuer la recherche. Nous commençons par présenter le voisinage que nous avons utilisé, puis les différentes heuristiques que nous avons testées, respectivement une méthode gloutonne, un recuit simulé [18], ainsi qu'une application de la méthode Mimausa [27].

Construction de la solution initiale

L'utilisation de stratégies de recherche procédant par améliorations successives de la solution nécessite la mise à disposition d'une solution initiale. L'obtention de cette solution initiale doit être rapide, tout en étant de bonne qualité pour pouvoir faciliter la recherche.

Pour notre problème, la solution initiale est obtenue en plaçant l'une après l'autre les circulations. Les circulations et tâches sont considérées dans leur ordre d'insertion dans le modèle. Seule une solution réalisable est cherchée (pas de fonction objectif), afin de limiter le temps pris pour la construction de cette solution initiale.

Exploration de voisinage pour le problème de construction de graphiques de circulation

Avant d'aborder l'application des heuristiques, il convient de définir le voisinage que l'on va considérer à partir d'une solution du problème. Rappelons qu'une solution de notre problème est définie par :

- les dates de début et de fin de chaque tâche ;
- la durée d'exécution de chaque tâche ;
- la ressource affectée à chaque tâche ;
- l'ordre de passage des tâches affectées à une même ressource.

La construction du voisinage d'une solution courante va donc consister à fixer un certain nombre de ces affectations, et relâcher les autres affectations. La définition du voisinage repose sur deux critères de choix :

- quantitatif, correspondant au nombre d'éléments à relâcher ;
- qualitatif, correspondant aux types d'affectations à relâcher.

La structure du problème nous amène aux observations suivantes. Considérant une tâche, les affectations de ses dates de début et de fin, et de sa durée, dépendent étroitement de l'affectation des tâches qui la précédent et la suivent

Cette dépendence est valable aussi bien dans la gamme que constitue la circulation à laquelle elle appartient, qu'avec les tâches qui partagent la même ressource qu'elle. On peut donc supposer qu'un voisinage défini par la fixation de toutes les tâches sauf une sera de très petite taille. Nous orientons donc notre choix vers la non fixation d'un nombre plus important de tâches.

Nous proposons pour la sélection des tâches de relâcher l'ensemble des tâches d'une circulation. L'idée n'est pas originale en soi, puisqu'on la retrouve dans différents travaux portant sur les problèmes d'ordonnancement d'atelier. Le voisinage d'une solution est construit comme suit. Soit $\overline{c}$ une circulation du problème. Pour toute tâche correspondant à une circulation autre que $\overline{c}$, on conserve les prédécesseurs et successeurs de la tâche, la ressource qui lui a été affectée, ainsi que sa capacité. Les dates de début et de fin de la tâche, ainsi que sa durée d'exécution sont relâchées. Pour les tâches de $\overline{c}$, aucune affectation n'est conservée. Ce voisinage, dont la taille maximale est égale au nombre de circulation du problème, consiste donc à replacer une ou plusieurs circulations dans le graphique de circulation, en conservant l'ordre relatif des autres circulations.

Le voisinage ne permet que la caractérisation d'un sous-espace de l'espace des solutions du problème. Des méthodes heuristiques, présentées ci-dessous, faisant usage de ce voisinage sont donc développées pour la recherche de solutions au problème.

Méthode gloutonne

L'algorithme glouton suit le principe de faire, étape par étape, un choix optimum local, dans l'espoir d'obtenir un résultat optimum global. L'algorithme glouton que nous avons testé, à défaut de fournir des solutions de bonne qualité, a l'avantage d'être simple. Son fonctionnement est décrit dans l'algorithme 1.

Algorithme 1 Algorithme glouton.

S^* solution initiale du problème
$v^* \leftarrow v(S^*)$ valeur de la fonction objectif de S^*
répéter
  construire le voisinage V de S^*
  si $v(S) < v^*, S \in V$ **alors**
    $v^* \leftarrow v(S)$
    $S^* \leftarrow S_i$
  sinon
    FIN : aucune solution du voisinage n'améliore la solution courante
  fin si
fin répéter

Comme le montre la figure 4.2, une telle approche ne permet généralement pas l'obtention de solutions de bonne qualité. Ceci est dû à l'absence de mécanisme de diversification des solutions parcourues. Nous présentons ci-dessous une application du recuit simulé, qui dispose de mécanismes de diversification.

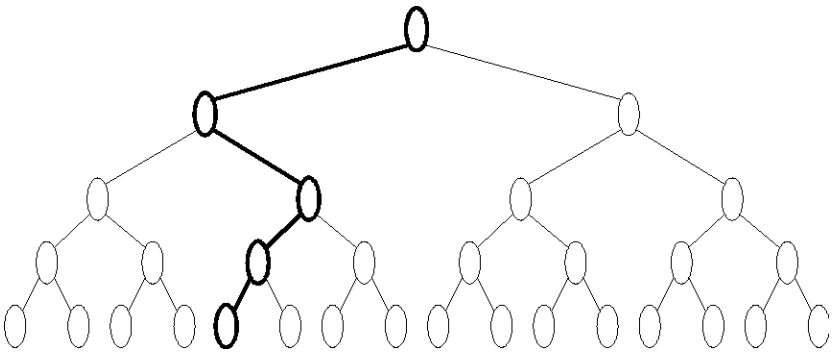

FIG. 4.2: Parcours de l'espace de recherche par une méthode glouton.

Recuit simulé

Le recuit simulé [23, 9] est une métaheuristique qui s'inspire d'un processus utilisé en métallurgie. Ce processus alterne des cycles de refroidissement lent et de réchauffage (recuit) qui tendent à minimiser l'énergie du matériau (recristallisation).

Le recuit simulé s'appuie sur l'algorithme de Metropolis, qui permet de décrire l'évolution d'un système thermodynamique. Par analogie avec le processus physique, la fonction à minimiser deviendra l'énergie E du système. On introduit également un paramètre fictif, la température T du système. Partant d'une solution donnée, en la modifiant, on en obtient une seconde. Soit celle-ci améliore le critère que l'on cherche à optimiser, on dit alors qu'on a fait baisser l'énergie du système, soit celle-ci le dégrade. Si on accepte une solution améliorant le critère, on tend ainsi à chercher l'optimum dans le voisinage de la solution de départ. L'acceptation d'une solution dégradant le critère permet quant à lui d'explorer une plus grande partie de l'espace des solutions, évitant ainsi de s'enfermer trop vite dans la recherche d'un optimum local. C'est le mécanisme de diversification. Nous renvoyons à la littérature sur le sujet pour une présentation détaillée du fonctionnement du recuit simulé.

Toute solution améliorante (en-dessous de la solution courante) est acceptée immédiatement, tandis qu'une solution dégradante est acceptée avec une probabilité d'acceptation $e^{-\frac{\Delta E}{T}}$, diminuant en s'éloignant de la valeur de la solution courante.

La figure 4.3 et l'algorithme 2 illustrent cette méthode. Sur la figure 4.3, le point rouge représente la solution courante, et la zone bleue représente le domaine d'acceptation d'une nouvelle solution : en largeur, l'espace de recherche, en intensité la probabilité d'acceptation.

L'utilisation de la méthode du recuit simulé [18] n'est pas nouvelle dans le domaine de l'optimisation ferroviaire, on pourra citer par exemple [32] pour le problème de re-planification des circulations en présence d'aléas.

Afin d'estimer les paramètres du recuit (température initiale, vitesse de refroidissement, et limite d'échecs pour la résolution du sous-problème), nous avons exécuté la résolution 5 fois sur 100 itérations avec différents jeux de paramètres. Le tableau 4.6 indique selon le jeu de paramètres utilisé (première ligne et première colonne) les valeurs minimales, moyennes et maximales des solu-

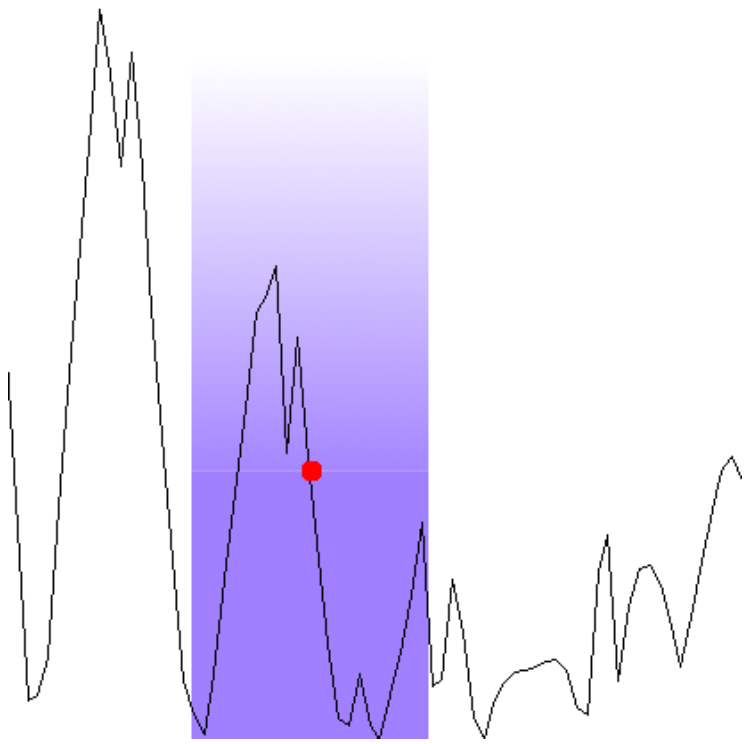

FIG. 4.3: Recuit simulé.

Algorithme 2 Schéma d'implantation du recuit simulé.

T température initiale
S solution initiale
E énergie initiale
k nombre de cuisson
f vitesse de refroidissement
répéter
  tant que $--k > 0$ et solution trouvée s d'énergie e **faire**
    si $e < E$ **alors**
      $S \leftarrow s$
      $E \leftarrow e$
    fin si
  fin tant que
  $T \leftarrow T \times f$
tant que $k = 0$ ou plus de voisins

initial tp°		limite du nombre d'échecs 10	100	1000
10	obj.	**425**/**458**/501	451/478/**497**	444/466/**497**
	temps	228/244/258	226/240/252	185/205/221
100	obj.	474/488/501	451/486/501	440/476/501
	temps	197/248/263	264/272/288	192/203/214
1000	obj.	463/481/501	481/490/501	448/470/501
	temps	218/225/300	259/266/272	203/212/226

TAB. 4.6: Influence des paramètres du recuit sur la valeur de la solution et le temps de résolution (en secondes).

tions trouvées (lignes 2, 4 et 6) et des temps de résolution (en secondes, lignes 3, 5 et 7). L'instance utilisée pour cette évaluation est formée de 100 nœuds (99 tronçons, i.e. réseau de densité minimale), chaque point étant formé de deux voies de sens contraire, et de 100 circulations (une représentation graphique est faite en annexe B).

Du tableau 4.6, nous pouvons faire les observations suivantes. Premièrement, une température initiale trop élevée ne permet pas l'obtention de bonnes solutions. Pour un nombre d'échecs équivalents, les meilleures valeurs moyennes de la fonction objectif sont obtenues avec une température initiale faible (10°). Une température élevée autorise en effet une plus grande dégradation de la solution, qui n'est ici semble-t-il pas compensée en raison du nombre d'échecs permis et du facteur de refroidissement (0.99) tous deux relativement faibles. De même, on peut observer que les valeurs maximales sont équivalentes (501 correspondant à la solution initiale), tandis que les valeurs minimales tendent à diminuer en même temps que la température initiale.
Deuxièmement, la variation du nombre d'échecs permis influe aussi sur la valeur de la fonction objectif. Sa valeur moyenne tout comme ses valeurs minimales et maximales semblent ici diminuer avec l'augmentation du nombre d'échecs permis, ce qui est compréhensible. Le cas 10-10 peut être probablement considéré comme un biais dû à la bonne solution trouvée lors d'une exécution (425, meilleure valeur du tableau).

Méthode Mimausa

Une autre heuristique que nous avons testée est une application de la méthode Mimausa [27], qui se base sur la théorie du *Referent Domain Optimization* [21]. Cette approche consiste à combiner recherche exacte et recherche locale selon le principe suivant. Une heuristique sélectionne K variables d'un problème d'optimisation $\mathcal{P}$, et fixe les autres variables. Nous appelons ici K-problème le problème d'optimisation ainsi obtenu. Une méthode exacte sert ensuite à déterminer la valeur optimale des K variables sélectionnées. Une procédure permet ensuite de reconstruire la solution complète de $\mathcal{P}$ à partir de la solution du K-problème. Ce processus est répété jusqu'à l'obtention d'une solution satisfaisante, et une stratégie de recherche permet la gestion des phases d'intensification et de diversification, et empêche les phénomènes de cycle.

La figure 4.4 illustre cette méthode. En haut, le point rouge représente la solution courante, et la zone bleue représente le domaine d'acceptation d'une nouvelle solution. La densité est cette fois constante sur l'ensemble du domaine, mais la largeur du domaine est amenée à varier en fonction du temps mis à déterminer une nouvelle solution (en bas, en rouge).

L'application faite ici se différencie de la description précédente sur plusieurs points. La différence principale repose sur la génération et la résolution du K-problème. Cette différence est due à la forte combinatoire du problème traité. Celle-ci rend parfois extrêmement difficile la résolution à l'optimum du K-problème, même lorsque K est choisi petit. Le K-problème est donc défini comme suit. Pour un entier K donné, avec $taille(\mathcal{P}) \geq K \geq 1$ (avec $taille(\mathcal{P})$ égal au nombre de circulations du problème), on construit un K-voisinage où chaque voisin est obtenu en sélectionnant K circulations, et en fixant les circulations restantes. On s'assure que chaque circulation est présente dans au moins un voisin, et les $K-1$ circulations restant à sélectionner le sont par tirage aléatoire[1]. On génère ainsi autant de K-voisins qu'il y a de circulations, et ces K-voisins sont ensuite résolus selon les mécanismes de programmation par contraintes décrits précédemment, en limitant arbitrairement le nombre d'échecs permis, tant qu'une meilleure solution n'a pas été trouvée, et qu'il reste des voisins non traités.

Nous avons défini la stratégie de recherche suivante. À chaque itération de la boucle principale, une solution meilleure que la solution mémorisée est recherchée. Si aucune solution n'est trouvée, et que le nombre de circulations non fixées est égal à 1, alors on permet une diversification dans la recherche en augmentant la valeur de la borne supérieure de $div\%$. La valeur de div est diminuée d'un pas fixé par avance, ce qui nous assure de l'arrêt de la procédure, et évite les problèmes de cyclage (ou du moins une partie). Quel que soit le résultat de la recherche à l'itération courante, la taille des sous-problèmes (K-voisins) est éventuellement adaptée :

- si le temps de résolution est inférieur à un seuil minimum t_{min}, alors on incrémente le nombre de circulations non fixées K, tant que K reste inférieur ou égal à $taille(\mathcal{P})$;
- si le temps de résolution est supérieur à un seuil maximum t_{max}, alors on décrémente K, tant que K reste supérieur ou égal à 1.

Cette boucle principale est répétée tant qu'un nombre d'échecs consécutifs n'est pas atteint, ou que l'on se retrouve dans une situation d'échec devant déclencher une diversification sans que celle-ci soit encore permise.

Le schéma de résolution général est repris dans l'algorithme 3.

Les tableaux 4.7 et 4.8 présentent les résultats obtenus pour l'instance décrite précédemment, de 100 nœuds et 100 circulations. Nous indiquons pour chaque jeu de paramètres testé (première ligne) les valeurs minimales, moyennes et maximales des solutions trouvées (deuxième ligne) et des temps de résolution (en secondes, troisième ligne). La quatrième ligne reporte la valeur moyenne du temps d'exécution hors itérations d'échec. Les paramètres qui n'ont pas varié lors des tests sont :

- $iter$, nombre d'itérations non améliorantes restant à effectuer avant échec, fixé à 10 ;

[1] D'autres stratégies peuvent bien entendu être utilisées, comme par exemple privilégier des circulations en interaction avec les circulations déjà sélectionnées.

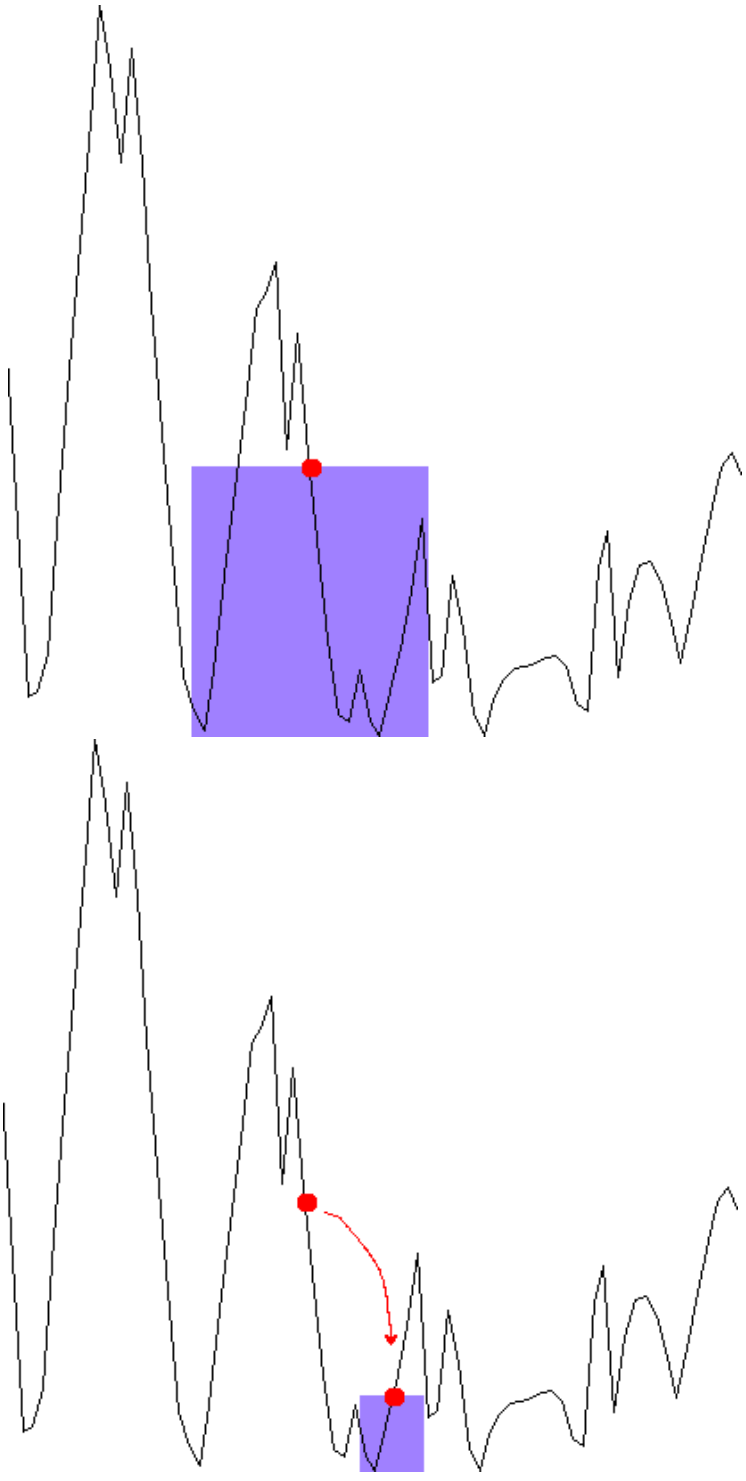

FIG. 4.4: Méthode Mimausa : en haut, à une itération donnée, en bas, après un mouvement.

Algorithme 3 Schéma d'implantation de Mimausa.

$\mathcal{P}$ problème d'optimisation combinatoire
$\overline{S}$ solution initiale de $\mathcal{P}$
t_{min}, t_{max} seuils inférieur et supérieur du temps de résolution
$iter \leftarrow iter_{init}$ nombre d'itérations non améliorantes restant à effectuer avant échec
$fail \leftarrow fail_{init}$ nombre d'échecs pour la recherche d'une solution dans le voisinage
$nf \leftarrow nf_{init}$ nombre de variables non fixées dans les sous-problèmes
$div \leftarrow div_{init}$ coefficient de divergence
tant que $iter > 0$ **faire**
  décrémenter $iter$
  construire le voisinage $V(nf)$ de $\overline{S}$
  chercher une solution S de $V(nf)$ (tps : temps de résolution)
  si S est trouvée **alors**
    {mémorisation}
    $\overline{S} \leftarrow S$
    $iter \leftarrow iter_{init}$
  sinon si nf=1 **alors**
    {diversification}
    si $div > 0$ **alors**
      augmenter la borne supérieure de div%
      diminuer div de div_{step}
    sinon
      FIN : on n'arrive pas à trouver d'autre zones d'exploration
    fin si
  fin si
  {ajustement des sous-problèmes}
  si $tps < t_{min}$ et $nf < taille(\mathcal{P})$ **alors**
    incrémenter nf
  sinon si $tps > t_{max}$ et $nf > 1$ **alors**
    décrémenter nf
  fin si
fin tant que

	limite du nombre d'échecs		
	10	100	1000
obj.	**322** / 341 / 356	332 / **338** / **350**	339 / 358 / 375
temps	1282/**1822**/2245	1515/1918/2107	1475/2708/3335
	(1011)	**(886)**	(1495)

TAB. 4.7: Valeurs minimales, moyennes (hors échec) et maximales de l'objectif et du temps de résolution en fonction des paramètres de Mimausa.

	limite du nombre d'échecs		
	10	100	1000
obj.	**340**/**359**/**384**	342/360/395	383/408/ 464
temps	145/**371**/544	146/476/929	68 /467/1001

TAB. 4.8: Valeurs minimales, moyennes et maximales de l'objectif et du temps de résolution en fonction des paramètres de Mimausa (au premier échec).

- t_{min}, seuil minimum de temps d'exécution, égal à 2s ;
- t_{max}, seuil maximum de temps d'exécution, égal à 10s ;
- nf_{init}, nombre de variables non fixées initialement, égal à 1 ;
- div_{init}, égal à 1,05 ;
- div_{step}, égal à 0,01.

Ces résultats sont obtenus sur 5 exécutions de l'algorithme.

Le tableau 4.7 donne ces valeurs après exécution complète de l'algorithme (c'est à dire après 10 itérations successives de la boucle principale sans amélioration). Les valeurs les plus intéressantes de la fonction objectif sont obtenues par une limite du nombre d'échecs permis fixée à 10 ou 100. Une plus grande limite semble ne pas être profitable, en raison de l'augmentation significative du temps de résolution alors que la valeur moyenne de la fonction objectif se dégrade.

Il est toutefois important de noter qu'une partie importante du temps de résolution total est obtenue lors des itérations en échec (de l'ordre de 90s (resp. 100s, 180s) par itération pour une limite de 10 (resp. 100, 1000) échecs). Comme le montre la troisième ligne du tableau, les itérations en échec représentent environ la moitié du temps total de résolution. Notons que ces mesures de temps ont été obtenues dans des conditions plus stables que pour le recuit, afin de pouvoir être exploités. Pour souligner le bon comportement de la méthode, nous présentons tableau 4.8 ces mêmes valeurs avant obtention du premier échec.

Les données du tableau 4.8 confortent les observations précédentes. Les résultats les plus intéressants sont là encore obtenus avec les limites de 10 et 100, et plus particulièrement à 10 avec un temps moyen sensiblement inférieur, tout en ayant des valeurs extrêmes et moyennes de la fonction objectif similaires. L'étude des résultats obtenus avec une limite fixée à 1000 semble montrer que l'algorithme se retrouve plus rapidement bloqué dans des « minimaux locaux », en raison notamment d'un nombre de variables non fixées plus faible.

Pour mieux illustrer l'évolution de la solution au cours de la résolution, nous représentons figures 4.5, 4.6, 4.7, 4.8, 4.9 et 4.10 l'évolution de la valeur de la

fonction objectif et du nombre de variables non fixées au cours des itérations pour une exécution de l'approche Mimausa et selon la limite du nombre d'échecs.

4.2.4 Conclusion

Nous venons de présenter l'application de trois heuristiques pour la résolution du problème de construction de graphique. Nous résumons brièvement ci-dessous les résultats numériques obtenus pour chacune d'elles, puis nous effectuons quelques observations.

La première heuristique testée est un algorithme glouton. Les résultats numériques obtenus ne sont, comme on peut s'y attendre, pas de bonne qualité. Toutefois, on pourrait par exemple envisager d'utiliser une telle approche, augmentée de stratégies de recherche adaptées au problème, pour la recherche rapide d'une solution. De plus, l'ajout de mécanismes de diversification pourrait sans difficulté permettre d'améliorer les résultats obtenues par cette approche.

Le recuit simulé permet déjà l'obtention de meilleures solutions (au sens de l'objectif), notamment par l'apport de procédés de diversification. Une telle métaheuristique présente toutefois l'inconvénient de devoir estimer de nombreux paramètres (température initiale, vitesse de refroidissement, critère(s) d'arrêt, longueur de palier). Une étude plus large s'appuyant sur un jeu d'instances significatives serait nécessaire.

Les meilleurs résultats ont été obtenus avec la méthode Mimausa. Cette métaheuristique qui consiste à fixer une partie de la solution de taille variable, et explorer ensuite le voisinage obtenu, semble bien adapté au problème. Ceci peut se comprendre par l'interaction des circulations sur le réseau, et conforte les études de Tornquist [31, 33]. Il reste nécessaire de travailler sur les mécanismes de sélection (fixation du sous-problème) et de diversification.

Plus généralement, et quelle que soit l'heuristique ou la métaheuristique utilisée (voire même dans le cas d'une méthode exacte), deux observations sont à faire.

Premièrement, il est nécessaire d'obtenir des informations sur les « règles métier ». afin de s'assurer que la modélisation prenne en compte les obligations et les préférences de l'utilisateur, et afin de limiter éventuellement l'espace des solutions à explorer.

Un autre point concerne les stratégies utilisées pour la définition des voisinages et l'exploration. La connaissance des interactions et de l'influence mutuelle des circulations semblent important pour pouvoir construire un voisinage significatif. Par exemple, la sélection d'un ensemble de circulations qui ont un lien est plus à même de fournir des voisinages réalisables, et donc susceptibles d'améliorer la solution courante. Les stratégies liées à l'exploration de l'espace de recherche sont, elles, nécessaires localement pour explorer rapidement le voisinage, et globalement pour la recherche d'une bonne solution.

De façon évidente, les deux points que nous venons de citer sont dépendantes de la connaissance du métier et de l'utilisation de données représentatives.

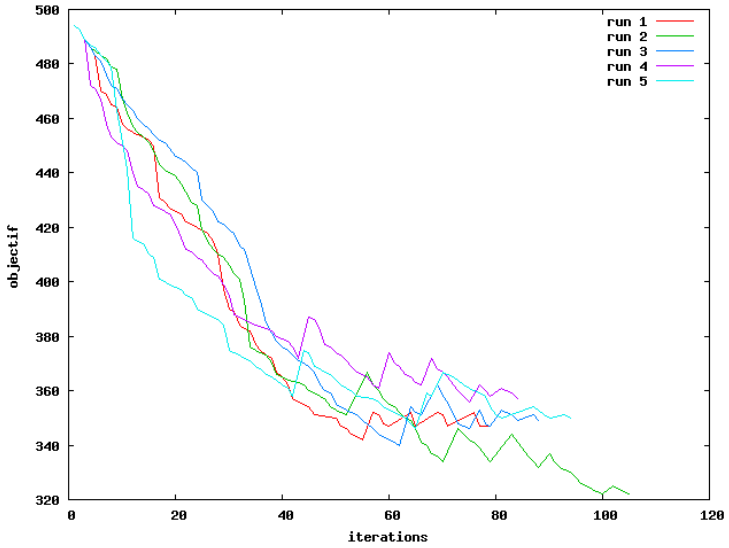

FIG. 4.5: Évolution de la valeur de la fonction objectif en fonction des itérations (nb. échecs permis : 10).

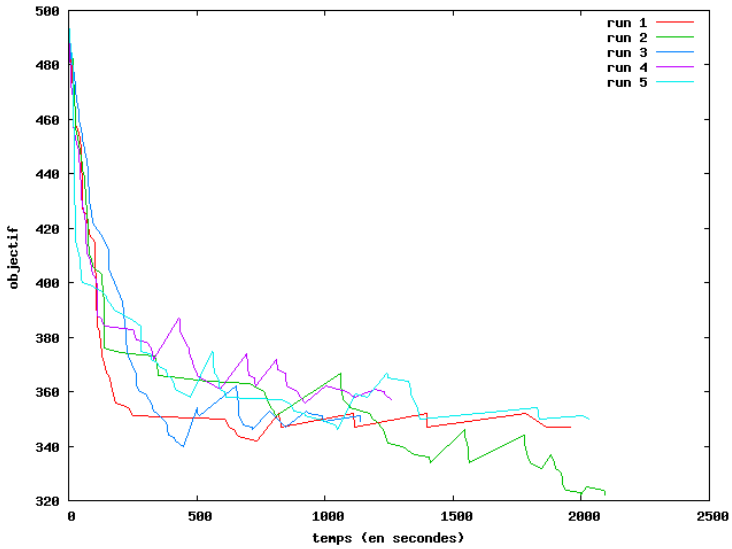

FIG. 4.6: Évolution de la valeur de la fonction objectif en fonction du temps (nb. échecs permis : 10).

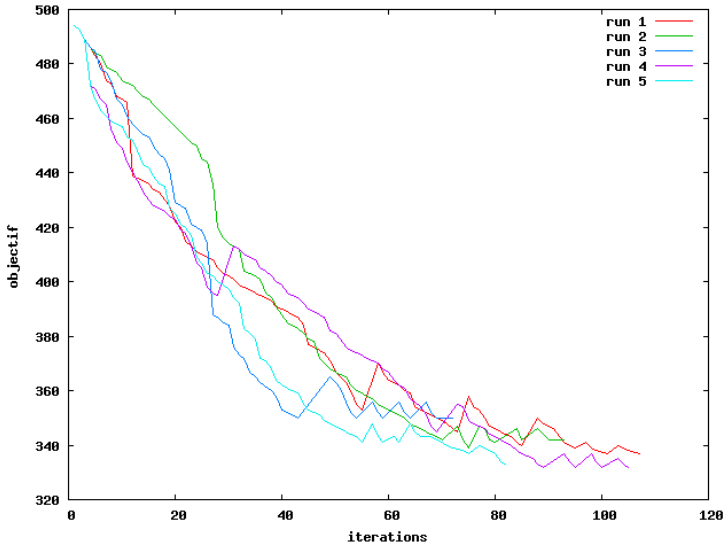

FIG. 4.7: Évolution de la valeur de la fonction objectif en fonction des itérations (nb. échecs permis : 100).

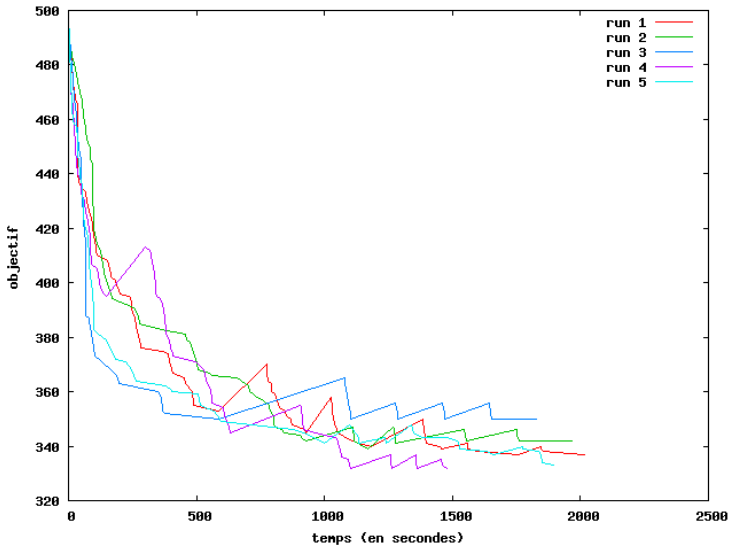

FIG. 4.8: Évolution de la valeur de la fonction objectif en fonction du temps (nb. échecs permis : 100).

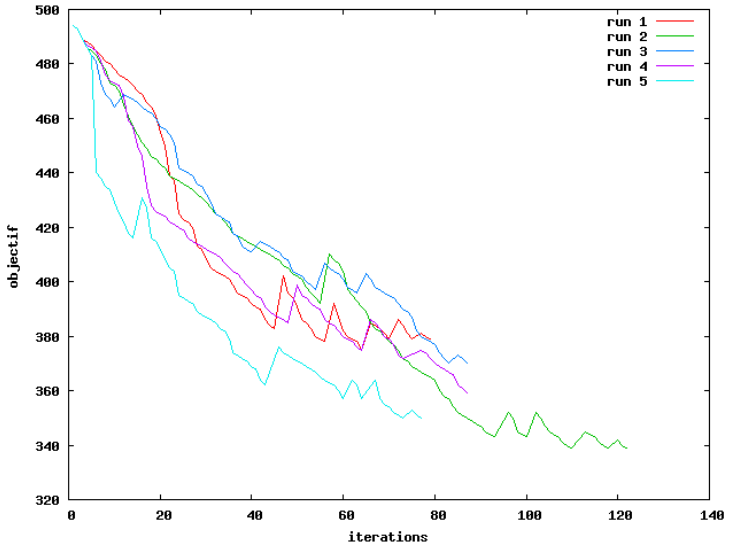

FIG. 4.9: Évolution de la valeur de la fonction objectif en fonction des itérations (nb. échecs permis : 1000).

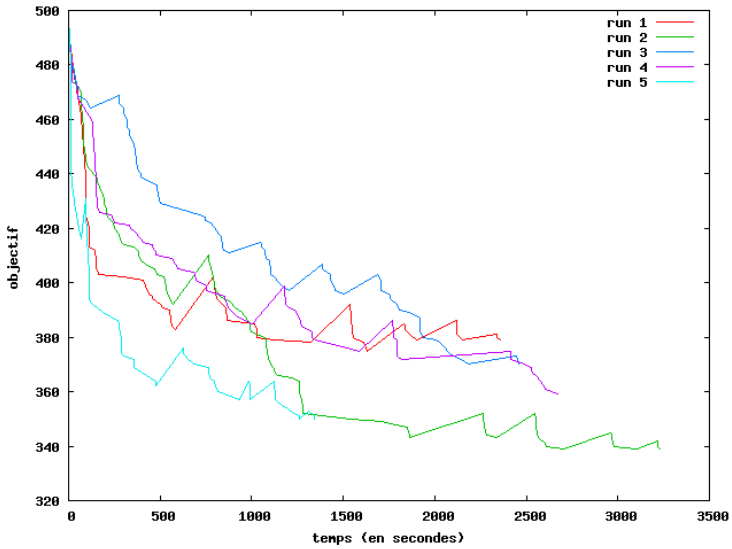

FIG. 4.10: Évolution de la valeur de la fonction objectif en fonction du temps (nb. échecs permis : 1000).

4.3 Réoptimisation en présence d'aléas

Nous avons présenté section 1.3 le problème de réoptimisation en présence d'aléas. Il s'agit ici d'une problématique opérationnelle, qui consiste à déterminer un nouvel ordonnancement des circulations par rapport à une planification initiale (cf. section 4.2) lorsqu'un aléa survient (retard, problème technique, etc.)

L'intérêt de la modélisation et des (méta)heuristiques présentées respectivement sections 3 et 4.2 est qu'elles permettent de considérer ce problème comme un cas particulier de construction de graphique. En effet, la réoptimisation des circulations considère une planification initiale (solution initiale), dans laquelle ne sont pas pris en compte les événements précédant l'aléa (fixation d'un sous-ensemble des événements). On retrouve ici une notion de voisinage par rapport à la solution correspondant à la planification initiale. Si le problème ainsi obtenu est de plus petite taille que le problème de construction du graphique de circulation associé (fixation de certains événements), sa taille reste généralement trop grande pour envisager sa résolution exacte. De plus, s'agissant d'une problématique opérationnelle, des contraintes fortes sur le temps de résolution doivent être respectées. Nous utilisons ainsi les heuristiques présentées précédemment.

Différentes fonctions objectifs peuvent être considérées. Nous considérons ici des minimisations du retard :

- retard maximal
- retard algébrique maximal
- somme des retards
- sommes des retards algébriques

Les retards sont ici pris en compte sur l'ensemble des événements du problème. Il peut être envisagé de ne considérer que les retards en gares qui ont une importance commerciale plus importante (ou mettre en place une pondération). Un autre aspect, non pris en compte ici, peut être de pénaliser le nombre de modifications (permutation de circulations, réaffectation de voies) par rapport à la planification initiale.

Il est intéressant de souligner que les voisinages utilisées dans les (méta)heuristiques proposées ici ont de grandes similarités avec des méthodes comme HOAT (restriction des modifications possibles aux événements voisins des événements perturbés [31, 33]), et peuvent aisément être adaptées pour en reprendre les caractéristiques.

4.4 Construction d'horaires cycliques

De nombreux pays européens proposent des grilles horaires cycliques. Si la structure en étoile du réseau ferroviaire français a probablement retardé le développement d'une telle approche, nous nous sommes tout de même intéressés à cette problématique. Nous considérons pour cela le problème de construction de graphiques de circulation « classique », auquel nous ajoutons la prise en compte des contraintes liées au caractère cyclique des tâches.

Nous illustrons figure 4.11 les contraintes supplémentaires qui apparaissent entre paires de tâches. Sur cette figure, nous représentons le placement temporel de deux tâches t_1 et t_2 que l'on supposera ici utilisant une même ressource disjonctive. La tâche t_1 précédant t_2 sur la ressource, un temps de transition d_{12} doit être respecté entre la fin de t_1 et le début de t_2. L'ordonnancement étant

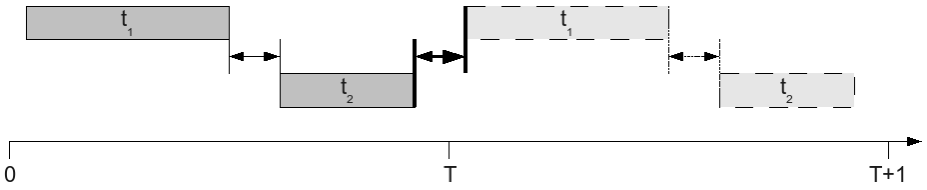

FIG. 4.11: contraintes liées à la cyclicité des tâches.

cyclique, de période T, on retrouve ces tâches dans une même configuration en déplaçant la fenêtre temporelle d'observation de T. Mais dans le cas où on déplace cette fenêtre de temps de $T/2$ unités de temps, nous pouvons alors voir que la tâche t_2 précéde t_1 sur la ressource. Il est alors nécessaire de respecter un temps de transition d_{21} entre la fin de t_2 et le début de t_1.

4.4.1 Gestion de l'aspect cyclique

La construction d'horaires cycliques impose donc de considérer, en plus des contraintes du problème de base, des temps de transition supplémentaires entre les tâches. La prise en compte de fonctions modulo dans les solveurs n'étant pas commune, nous utilisons une linéarisation du modulo (voir par exemple [28]).

Soit t_1, t_2 deux tâches, et S_1, F_1, S_2 et F_2 leurs dates de début et de fin respectives. Si une même ressource (que l'on supposera de capacité unitaire) est affectée à ces deux tâches, alors la prise en compte des contraintes d'espacement cycliques peut être obtenue au moyen des contraintes suivantes :

$$\begin{cases} S_2 - F_1 & \geq & delai_{12} - T & \times p \\ S_1 - F_2 & \geq & delai_{21} + T & \times (p-1) \end{cases} , p \in \mathbb{Z}$$

En effet, supposons $t_1 \prec t_2$. On a alors :

$$S_2 - F_1 \geq delai_{12} \tag{4.1}$$

En considérant l'aspect cyclique, (4.1) devient, pour s'assurer que $t_1 \prec t_2$:

$$S_2 - F_1 \geq delai_{12} - T \times p \tag{4.2}$$

De plus, si l'ordonnancement est cyclique, on doit aussi considérer le cas $t_2 \prec t_1$, modulo T.

$$S_1 - F_2 + T \geq delai_{21}$$

Soit, en tenant compte de l'aspect cyclique :

$$S_1 - F_2 + T \geq delai_{21} + T \times p$$

ou bien encore

$$S_1 - F_2 \geq delai_{21} + T \times (p-1) \tag{4.3}$$

La différence de signe devant $T \times p$ dans les équations (4.2) et (4.3) s'explique par le fait que si l'une des variations est positive, alors l'autre est négative.

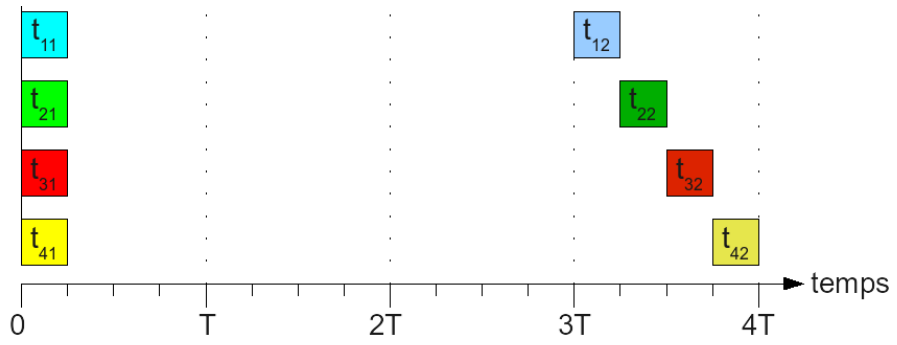

FIG. 4.12: Exemple pour le calcul des contraintes d'espacement cycliques.

4.4.2 Application numérique

Pour illuster notre propos, nous proposons de calculer les valeurs des équations (4.2) et (4.3) pour les paires de tâches représentées sur la figure 4.12. On supposera une gradation = une unité de temps, avec un espacement pour tous les cas de une unité de temps, la période T étant de quatre unités de temps.

Pour t_{11}, t_{12}, on a :

$$\begin{array}{lrcll} (4.2): & 12-1 & \geq & 1-4p & \Rightarrow \quad 10 \geq -4p \\ (4.3): & 0-13 & \geq & 1+4(p-1) & \Rightarrow \quad -10 \geq 4p \end{array}$$

En prenant $p = -3$ pour satisfaire (4.3), (4.2) devient $10 \geq 12$, ce qui est faux. La paire de tâches (t_{11}, t_{12}) ne vérifie donc pas la contrainte d'espacement cyclique. En effet, on peut voir qu'en décalant de trois périodes vers la gauche la tâche t_{32}, les deux tâches sont superposées.

Pour t_{21}, t_{22}, on a :

$$\begin{array}{lrcll} (4.2): & 13-1 & \geq & 1-4p & \Rightarrow \quad 11 \geq -4p \\ (4.3): & 0-14 & \geq & 1+4(p-1) & \Rightarrow \quad -11 \geq 4p \end{array}$$

En prenant $p = -3$ pour satisfaire (4.3), (4.2) devient $11 \geq 12$, ce qui est faux. La paire de tâches (t_{21}, t_{22}) ne vérifie donc pas la contrainte d'espacement cyclique. En effet, on peut voir qu'en décalant de trois périodes vers la gauche la tâche t_{32}, l'espacement d'une unité de temps entre t_{31} et t_{32} n'est pas respecté, les deux tâches étant accolées.

Pour t_{31}, t_{32}, on a :

$$\begin{array}{lrcll} (4.2): & 14-1 & \geq & 1-4p & \Rightarrow \quad 12 \geq -4p \\ (4.3): & 0-15 & \geq & 1+4(p-1) & \Rightarrow \quad -12 \geq 4p \end{array}$$

En prenant $p = -3$ pour satisfaire (4.3), (4.2) devient $12 \geq 12$, ce qui est vrai. La paire de tâches (t_{31}, t_{32}) vérifie donc la contrainte d'espacement cyclique. En effet, on peut voir qu'en décalant de trois périodes vers la gauche la tâche t_{32} il y a bien un espacement d'une unité de temps entre t_{31} et t_{32} et, en décalant t_{31} de quatre périodes vers la droite, il y a bien un espacement d'une unité de temps entre t_{32} et t_{31}.

Pour t_{41}, t_{42}, on a :

$$\begin{array}{lrcll} (4.2): & 15-1 & \geq & 1-4p & \Rightarrow \quad 13 \geq -4p \\ (4.3): & 0-16 & \geq & 1+4(p-1) & \Rightarrow \quad -13 \geq 4p \end{array}$$

En prenant $p = -4$ pour satisfaire (4.3), (4.2) devient $13 \geq 16$, ce qui est faux. La paire de tâches (t_{41}, t_{42}) ne vérifie donc pas la contrainte d'espacement cyclique. En effet, on peut voir qu'en décalant t_{31} de quatre périodes vers la droite, l'espacement d'une unité de temps entre t_{32} et t_{31} n'est pas respecté, les deux tâches étant accolées.

4.4.3 Résolution

Concernant la résolution, nous utilisons les mêmes approches que pour les problèmes de construction de graphiques et de réoptimisation en présence d'aléas.

Chapitre 5

Méthodes de décomposition

Nous nous intéressons maintenant à l'application de méthodes de décomposition pour le problème de réoptimisation en présence d'aléas. Plus particulièrement, nous appliquons la méthode de Benders à une modélisation sous forme de programme linéaire en variables mixtes. Nous rappelons brièvement section 5.1 le principe de la décomposition de Benders. Puis, après avoir décrit le modèle utilisé dans [**?**] (section 5.2), nous donnons section 5.3 la formulation des sous-problèmes obtenus par la décomposition, et faisons quelques observations à partir des premiers résultats numériques obtenus. Ces derniers nous conduisent ensuite à la présentation d'heuristiques (section **??**)

5.1 Méthode de Benders

Nous présentons ici la méthode de Benders dans son application à un programme linéaire en variables mixtes. Nous renvoyons à la littérature [4, 20] pour une présentation plus détaillée de la méthode et de ses généralisations.

5.1.1 Problème initial

Nous considérons comme problème de départ le programme linéaire en variables mixtes suivant :

$$\begin{aligned} (P): \quad & \min \langle c, x \rangle + \langle d, y \rangle \\ \text{sous les contraintes :} & \\ & Ax + By \geq e \\ & y \in \mathbb{R}^+ \\ & x \in \mathcal{X} \end{aligned}$$

où $\mathcal{X}$ est un ensemble quelconque, ici un sous-ensemble de $\mathbb{N}$.

5.1.2 Reformulation

La décomposition de Benders consiste à séparer la composante difficile liée aux variables x (problème maître) de la composante facile correspondant aux variables y (sous-problème). Nous obtenons ainsi :

$$(P): \quad \min \langle c, x \rangle + f(x)$$

$$\begin{array}{c} \text{sous les contraintes :} \\ x \in \mathcal{X} \end{array}$$

avec :

$$\begin{array}{l} f(x) = \min\langle d, y\rangle \\ \text{sous les contraintes :} \\ By \geq e - Ax \\ y \in \mathbb{R}^+ \end{array}$$

Afin d'éviter que le domaine réalisable de f ne dépende du choix de x, on utilise la forme duale :

$$\begin{array}{ll} (SP): & f(x) = \max\langle e - AX, u\rangle \\ \text{sous les contraintes :} & \\ & B^t u \leq d \\ & u \in \mathbb{R}^+ \end{array}$$

5.1.3 Méthode de Benders : génération de coupes

On peut alors remplacer le domaine réalisable par l'expression de ses points et rayons extrêmes.

$$\begin{array}{lll} (PM): & \min\langle c, x\rangle + x_0 & (5.1) \\ \text{sous les contraintes :} & & \\ & \langle e - Ax, u^r\rangle \leq 0 \quad u^r \in R & (5.2) \\ & x_0 \geq \langle e - Ax, u^p\rangle \quad u^p \in P & (5.3) \\ & x \in \mathcal{X} & (5.4) \end{array}$$

où P (resp. R) est l'ensemble des points (resp. rayons) extrêmes du domaine de $f(x)$. On appellera contraintes de type I (resp. type II) les contraintes (5.2) (resp. (5.3)).

La détermination de l'ensemble des points et rayons extrêmes ne pouvant être raisonnablement envisagée (ne serait-ce qu'en raison de leur nombre très élevé), la méthode de Benders consiste à les générer de façon itérative. On commence pour cela par résoudre (PM) avec un sous-ensemble (éventuellement vide) de points et rayons extrêmes. La solution obtenue permet de fixer la formulation de (SP), qui est ensuite résolue afin de déterminer un nouveau point ou rayon extrême. On répète ensuite cet algorithme jusqu'à ce qu'une nouvelle coupe générée soit vérifiée par la solution courante (obtention d'une solution optimale pour le problème initial), ou que (PM), et par conséquent (P), ne soit pas réalisable.

5.2 Programmation linéaire du problème de réoptimisation en présence d'aléa

Nous nous proposons d'appliquer la décomposition présenté à la formulation linéaire du problème de réoptimisation en présence d'aléas issue de nos travaux sur ce thème (voir [**?**]).

5.2.1 Notations

Les notations utilisées dans [**?**] sont les suivantes :

Données

T **:** ensemble des circulations ;
F **:** ensemble des sections ;
B **:** ensemble des cantons ;
E **:** ensemble des événements ;
E_{sec} **:** ensemble des événements en section ;
E_{st} **:** ensemble des événements en gare ;
R_s **:** ensemble des voies de la section s ;
G_s **:** ensemble des cantons de la section s ;
H_s **:** ensemble des circulations empruntant la section s ;
$I_{s,i}$ **:** ensemble ordonné des événements du train i dans la section s ;
K_i **:** ensemble ordonné des événements du train i ;
L_j **:** ensemble ordonné des événements du canton j ;
P_j **:** ensemble des voies du canton j ;
$N_{k,t}$ **:** ensemble des voies non utilisables après l'événement k sur la voie t ;
C **:** ensemble des paires d'événements $(k, \hat{k})$, où k est en correspondance avec $\hat{k}$;
d_k^{min} **:** durée minimale de l'événement k ;
d_k^{max} **:** durée maximale de l'événement k ;
d_k^{fre} **:** durée de freinage de l'événement k ;
d_k^{acc} **:** durée d'accélération de l'événement k ;
b_k^{ini} **:** date de début de l'événement k dans la planification initiale ;
e_k^{ini} **:** date de fin de l'événement k dans la planification initiale ;
b_k^{inc} **:** date de début de l'événement k dans la planification réelle ;
e_k^{inc} **:** date de fin de l'événement k dans la planification réelle ;
h_k **:** $\begin{cases} 1 \text{ si l'événement } k \text{ a un arrêt planifié, } k \in E_{st} \\ 0 \text{ sinon} \end{cases}$;
r_k **:** $\begin{cases} 1 \text{ si } k \in E_{st} \\ 0 \text{ sinon} \end{cases}$;
w_j **:** $\begin{cases} 1 \text{ si } j \in G_s, s \in E_{sec} \\ 0 \text{ si } j \in G_s, s \in E_{st} \end{cases}$;
c_t **:** $\begin{cases} 1 \text{ si la voie } t \text{ est bidirectionnelle} \\ 0 \text{ sinon} \end{cases}$;
f_t **:** $\begin{cases} 1 \text{ si les événements utilisant la voie } t \text{ ne peuvent changer de voie} \\ 0 \text{ sinon} \end{cases}$;
o_k^E **:** sens de l'événement k ;
o_t^T **:** sens de la voie (unidirectionnelle) t ;
$q_{k,t}^0$ **:** $\begin{cases} 1 \text{ si l'événement } k \text{ utilise la voie } t \text{ dans la planification initiale, } k \in L_j, t \in P_j, j \in B \\ 0 \text{ sinon} \end{cases}$;

On note également $k \prec \hat{k}$ le fait que l'évenement k précéde $\hat{k}$.

Variables de décision

Les variables de décision du modèle étaient les suivantes :

$q_{k,t}$ **:** $\begin{cases} 1 \text{ si l'événement } k \text{ utilise la voie } t,\ k \in L_j, t \in P_j, j \in B \\ 0 \text{ sinon} \end{cases}$;

x_k^{beg} **:** date de début de l'événement $k \in E$;

x_k^{end} **:** date de fin de l'événement $k \in E$;

z_k **:** retard de l'événement $k \in E$;

y_k **:** $\begin{cases} 1 \text{ si l'événement } k \text{ fait un arrêt imprévu } k \in E \\ 0 \text{ sinon} \end{cases}$;

$\gamma_{k,\hat{k}}$ **:** $\begin{cases} 1 \text{ si l'événement } k \text{ précède } \hat{k},\ k, \hat{k} \in L_j, j \in B, k \prec \hat{k} \\ 0 \text{ sinon} \end{cases}$;

$\lambda_{k,\hat{k}}$ **:** $\begin{cases} 1 \text{ si l'événement } k \text{ est replanifié après } \hat{k},\ k, \hat{k} \in L_j, j \in B, k \prec \hat{k} \\ 0 \text{ sinon} \end{cases}$;

5.2.2 Modélisation

Le modèle linéaire permettant de réoptimiser les horaires des trains en minimisant les retard, les changements de voies et arrêts imprévus est alors le suivant :

$$\begin{aligned} \min\ & CD \sum_{\substack{i \in T \\ k \in K_i}} h_k z_k + CFD \sum_{i \in T} z_{k_i^n} \\ & + CCT \sum_{\substack{t \in P_j, j \in B \\ k \in L_j \cap E_{sec}}} q_{k,t}(1 - q_{k,t}^0) + CCP \sum_{\substack{t \in P_j, j \in B \\ k \in L_j \cap E_{st}}} q_{k,t}(1 - q_{k,t}^0) \\ & + CUS \sum_{\substack{k \in E_{st} \\ h_k = 0}} y_k \end{aligned} \tag{5.5}$$

sous les contraintes :

$$x_k^{end} = x_{k+1}^{beg} \qquad \forall k \in K_i, i \in T : k \neq k_i^n \tag{5.6}$$
$$x_k^{end} \geq x_k^{beg} + d_k^{min} + d_k^{fre} y_{k+1} + d_k^{acc} y_{k-1} \qquad \forall k \in K_i, i \in T : r_k = 0 \tag{5.7}$$
$$x_k^{end} \geq x_k^{beg} + d_k^{min} \qquad \forall k \in E : h_k = 1 \wedge r_k = 1 \tag{5.8}$$
$$x_k^{end} \geq x_k^{beg} + y_k d_k^{min} \qquad \forall k \in E : h_k = 0 \wedge r_k = 1 \tag{5.9}$$
$$x_k^{end} \leq x_k^{beg} + y_k d_k^{max} \qquad \forall k \in E : h_k = 0 \wedge r_k = 1 \tag{5.10}$$
$$x_k^{end} \leq x_k^{beg} + d_k^{max} \qquad \forall k \in E : b_k^{inc} = e_k^{inc} = 0 \tag{5.11}$$
$$x_k^{beg} \geq b_k^{ini} \qquad \forall k \in E : h_k = 1 \tag{5.12}$$
$$x_k^{beg} \geq b_k^{inc} \qquad \forall k \in E : b_k^{inc} > 0 \tag{5.13}$$
$$x_k^{end} \geq e_k^{inc} \qquad \forall k \in E : e_k^{inc} > 0 \tag{5.14}$$
$$x_k^{beg} - b_k^{ini} \leq z_k \qquad \forall k \in E \tag{5.15}$$

$$\sum_{t \in P_j} q_{k,t} = 1 \qquad \forall k \in L_j, j \in B \tag{5.16}$$

$$q_{k,t} + q_{\hat{k}t} - 1 \leq \lambda_{k,\hat{k}} + \gamma_{k,\hat{k}} \qquad \forall k, \hat{k} \in L_j, t \in P_j, j \in B : k \prec \hat{k} \tag{5.17}$$

$$\lambda_{k,\hat{k}} + \gamma_{k,\hat{k}} \leq 1 \qquad \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k} \tag{5.18}$$

$$x_{\hat{k}}^{beg} - x_k^{end} \geq \Delta_j^M \gamma_{k,\hat{k}} - M(1 - \gamma_{k,\hat{k}}) \qquad \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E \neq o_{\hat{k}}^E \tag{5.19}$$

$$x_k^{beg} - x_{\hat{k}}^{end} \geq \Delta_j^M \lambda_{k,\hat{k}} - M(1 - \lambda_{k,\hat{k}}) \qquad \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E \neq o_{\hat{k}}^E \tag{5.20}$$

$$x_{\hat{k}}^{beg} - x_k^{beg} \geq SP_{k,\hat{k}} \gamma_{k,\hat{k}} - M(1 - \gamma_{k,\hat{k}}) \qquad \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 \tag{5.21}$$

$$x_k^{beg} - x_{\hat{k}}^{beg} \geq SP_{\hat{k},k} \lambda_{k,\hat{k}} - M(1 - \lambda_{k,\hat{k}}) \qquad \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 \tag{5.22}$$

$$x_{\hat{k}}^{end} - x_k^{end} \geq SP_{k,\hat{k}} \gamma_{k,\hat{k}} - M(1 - \gamma_{k,\hat{k}}) \qquad \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 \tag{5.23}$$

$$x_k^{end} - x_{\hat{k}}^{end} \geq SP_{\hat{k},k} \lambda_{k,\hat{k}} - M(1 - \lambda_{k,\hat{k}}) \qquad \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 \tag{5.24}$$

$$x_{\hat{k}}^{beg} - x_k^{end} \geq \Delta_j^F \gamma_{k,\hat{k}} - M(1 - \gamma_{k,\hat{k}}) \qquad \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 1 \tag{5.25}$$

$$x_k^{beg} - x_{\hat{k}}^{end} \geq \Delta_j^F \lambda_{k,\hat{k}} - M(1 - \lambda_{k,\hat{k}}) \qquad \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 1 \tag{5.26}$$

$$q_{k,t} = q_{k+1,t} \qquad \forall k \in I_{s,i} : k \neq k_i^n, i \in H_s, t \in R_s : f_t = 1, s \in F \tag{5.27}$$

$$q_{k,t} = 0 \qquad \forall k \in L_j, t \in P_j, j \in B : o_t^T \neq o_k^E \wedge c_t = 0 \tag{5.28}$$

$$q_{k,t} + \sum_{t_{inc} \in N_{k,t}} q_{k+1,t_{inc}} \leq 1 \qquad \forall k \in I_{s,i} : k \neq k_i^n, i \in H_s, t \in R_s : f_t = 0, s \in F \tag{5.29}$$

$$x_{\hat{k}}^{end} \geq x_k^{beg} + g_{k,\hat{k}}^{min} \qquad \forall (k, \hat{k}) \in C \tag{5.30}$$

$$x_{\hat{k}}^{end} \leq x_k^{beg} + g_{k,\hat{k}}^{max} \qquad \forall (k, \hat{k}) \in C \tag{5.31}$$

$$y_k = 0 \qquad \forall k \in E : h_k = 1 \wedge r_k = 1 \tag{5.32}$$

$$y_k = 0 \qquad \forall k \in E : r_k = 0 \tag{5.33}$$

$$x_k^{beg}, x_k^{end}, z_k \geq 0 \qquad \forall k \in E$$

$$\gamma_{k,\hat{k}}, \lambda_{k,\hat{k}} \in \{0; 1\} \qquad \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k}$$

$$q_{k,t} \in \{0; 1\} \qquad \forall k \in L_j, t \in P_j, j \in B$$

$$y_k \in \{0; 1\} \qquad \forall k \in E$$

Les contraintes (5.6) imposent qu'une circulation quittant un canton entre immédiatement dans le canton suivant. Les contraintes (5.7) représentent le temps minimum de parcours d'un canton en section, en tenant compte si nécessaire de phases d'accélération (si un arrêt non planifié a lieu dans le canton précédent) ou de freinage (si un arrêt non planifié a lieu dans le canton suivant). Les contraintes (5.8) représentent le temps minimum de parcours d'un canton en station, avec arrêt planifié. Les contraintes (5.9) et (5.10) représentent le temps minimum de parcours d'un canton en station, avec arrêt non planifié, et tiennent compte de la durée de desserte minimale et maximale. Les contraintes (5.11) limitent la durée des événements hors incident. Les contraintes (5.12) imposent, en présence d'arrêt planifié, que la circulation ne quitte pas la gare avant la date prévue. Les contraintes (5.13) et (5.14) insèrent les modifications

imposées par l'incident dans la planification. Les contraintes (5.15) servent à quantifier le retard des événements.

Les contraintes (5.16) imposent l'affectation d'une (et une seule) voie à chaque événement. Les contraintes (5.17) servent à lier les variables q d'affectation des voies aux variables d'ordonnancement γ et λ : pour tout paire d'événements, un des évenements doit être fixée à 1 si les deux utilisent la même voie. Les contraintes (5.19) et (5.20) imposent un délai minimal entre deux événements ordonnés (même canton, même voie) de sens opposé, compte tenu de leur ordonnancement relatif. Les contraintes (5.21) à (5.24) imposent un délai minimal entre les dates de début d'une part, et les dates de fin d'autre part, de deux événements ordonnés (même canton, même voie) de même sens en section, compte tenu de leur ordonnancement relatif. Les contraintes (5.25) et (5.26) imposent un délai minimal entre les dates de début et de fin de deux événements ordonnés (même canton, même voie) de même sens en station, compte tenu de leur ordonnancement relatif.

Les contraintes (5.27) imposent à une circulation de ne pas changer de voie dans une section si la voie affectée ne le permet pas. Les contraintes (5.28) interdisent l'affectation d'une voie de sens contraire au sens de l'événement. Les contraintes (5.29) interdisent l'affectation à une circulation de voies incompatibles avec une voie qui lui est affectée. Les contraintes (5.30) et (5.29) imposent le respect de durées de recouvrement minimales et maximales pour toute paire de circulations en correspondance. Les contraintes (5.31) et (5.32) fixent à 0 les variables d'arrêt imprévu pour tout événement ayant déjà un arrêt planifié.

5.2.3 Observations

Les contraintes (5.28), (5.32) et (5.33) ont pour rôle de fixer à 0 un sous-ensemble des variables q et y compte tenu des données du problème. Ces variables peuvent ainsi être remplacées par leur valeur dans l'implantation.

D'autres variables peuvent être fixées compte tenu du fait que le modèle utilisé ici autorise que plusieurs événements d'une même circulation soient associés à un même canton. Si $k, \hat{k}$ sont deux événements d'une même circulation empruntant un même point , alors elles ne peuvent être affectées à la même voie :

$$\begin{aligned} \lambda_{k,\hat{k}} &= 0 \quad \forall k, \hat{k} \in K_i \cap L_j, i \in T, j \in B : k \prec \hat{k} \\ \gamma_{k,\hat{k}} &= 0 \quad \forall k, \hat{k} \in K_i \cap L_j, i \in T, j \in B : k \prec \hat{k} \end{aligned}$$

Dans le cas où $\lambda_{k,\hat{k}}, \gamma_{k,\hat{k}}$ peuvent être fixées à 0, les contraintes (5.18) à (5.26) correspondantes peuvent être supprimées.

5.3 Application de la décomposition de Benders

L'application de la décomposition de Benders au modèle ci-dessous conduit aux problèmes maître et sous-problème ci-dessous.

5.3.1 Décomposition de Benders

Problème maître

$$\begin{aligned}\min CCT & \sum_{\substack{t\in P_j, j\in B\\ k\in L_j\cap E_{sec}}} q_{k,t}(1-q^0_{k,t})\\ +CCP & \sum_{\substack{t\in P_j, j\in B\\ k\in L_j\cap E_{st}}} q_{k,t}(1-q^0_{k,t})\\ +CUS & \sum_{\substack{k\in E_{st}\\ h_k=0}} y_k\\ +f(y,\gamma,\lambda)&\end{aligned} \tag{5.34}$$

sous les contraintes :

$$\sum_{t\in P_j} q_{k,t} = 1 \qquad \forall k\in L_j, j\in B \tag{5.35}$$

$$q_{k,t}+q_{\hat{k}t}-1 \le \lambda_{k,\hat{k}}+\gamma_{k,\hat{k}} \qquad \forall k,\hat{k}\in L_j, t\in P_j, j\in B : k\prec\hat{k} \tag{5.36}$$

$$\lambda_{k,\hat{k}}+\gamma_{k,\hat{k}} \le 1 \qquad \forall k,\hat{k}\in L_j, j\in B : k\prec\hat{k} \tag{5.37}$$

$$q_{k,t} = q_{k+1,t} \qquad \forall k\in I_{s,i} : k\neq k^n_i, i\in H_s, t\in R_s : f_t=1, s\in F \tag{5.38}$$

$$q_{k,t} = 0 \qquad \forall k\in L_j, t\in P_j, j\in B : o^T_t \neq o^E_k \wedge c_t = 0 \tag{5.39}$$

$$q_{k,t}+\sum_{t_{inc}\in N_{k,t}} q_{k+1,t_{inc}} \le 1 \qquad \forall k\in I_{s,i} : k\neq k^n_i, i\in H_s, t\in R_s : f_t=0, s\in F \tag{5.40}$$

$$y_k = 0 \qquad \forall k\in E : h_k=1\wedge r_k=1 \tag{5.41}$$

$$y_k = 0 \qquad \forall k\in E : r_k=0 \tag{5.42}$$

coupes de type I (5.43)

coupes de type II (5.44)

$$\gamma_{k,\hat{k}},\lambda_{k,\hat{k}}\in\{0;1\} \qquad \forall k,\hat{k}\in L_j, j\in B : k\prec\hat{k}$$

$$q_{k,t}\in\{0;1\} \qquad \forall k\in L_j, t\in P_j, j\in B$$

$$y_k\in\{0;1\} \qquad \forall k\in E$$

Ajouts à la formulation initiale

Le fait de décomposer le problème en un problème maître et un problème esclave entraîne une perte d'information dans le problème maître. Cette perte d'information est compensée au fur et à mesure des itérations par l'ajout de coupes de type I et II identifiées (contraintes (5.43) et (5.44)), mais il peut être utile d'insérer aussi des contraintes supplémentaires.

On peut ainsi ajouter un ensemble d'inégalités triangulaires sur λ, γ, suppléant au transfert dans le sous-problème des contraintes (5.19) à (5.26) (entre

parenthèses, l'ordonnancement résultant de la contrainte).

$$\begin{array}{lll}
\gamma_{k_1,k_2} + \gamma_{k_2,k_3} - 1 \leq \gamma_{k_1,k_3} & \forall k_1,k_2,k_3 \in L_j, j \in B : k_1 \prec k_2 \prec k_3 & (\Rightarrow k_1 \prec k_2 \prec k_3) \\
\gamma_{k_1,k_3} + \lambda_{k_2,k_3} - 1 \leq \gamma_{k_1,k_2} & \forall k_1,k_2,k_3 \in L_j, j \in B : k_1 \prec k_2 \prec k_3 & (\Rightarrow k_1 \prec k_3 \prec k_2) \\
\lambda_{k_1,k_2} + \gamma_{k_1,k_3} - 1 \leq \gamma_{k_2,k_3} & \forall k_1,k_2,k_3 \in L_j, j \in B : k_1 \prec k_2 \prec k_3 & (\Rightarrow k_2 \prec k_1 \prec k_3) \\
\lambda_{k_1,k_3} + \gamma_{k_2,k_3} - 1 \leq \lambda_{k_1,k_2} & \forall k_1,k_2,k_3 \in L_j, j \in B : k_1 \prec k_2 \prec k_3 & (\Rightarrow k_2 \prec k_3 \prec k_1) \\
\gamma_{k_1,k_2} + \lambda_{k_1,k_3} - 1 \leq \lambda_{k_3,k_2} & \forall k_1,k_2,k_3 \in L_j, j \in B : k_1 \prec k_2 \prec k_3 & (\Rightarrow k_3 \prec k_1 \prec k_2) \\
\lambda_{k_1,k_2} + \lambda_{k_2,k_3} - 1 \leq \lambda_{k_1,k_3} & \forall k_1,k_2,k_3 \in L_j, j \in B : k_1 \prec k_2 \prec k_3 & (\Rightarrow k_3 \prec k_2 \prec k_1)
\end{array}$$

On notera que si la variable du terme de droite est fixée à 1, alors la contrainte peut être ignorée, car implicitement vérifiée.

Sous-problème (forme primale)

$$f(y,\gamma,\lambda) = \min CD \sum_{\substack{i \in T \\ k \in K_i}} h_k z_k + CFD \sum_{i \in T} z_{k_i^n}$$

sous les contraintes :

$$\begin{array}{lll}
x_k^{end} - x_{k+1}^{beg} = 0 & \forall k \in K_i, i \in T : k \neq k_i^n & (5.45) \\
x_k^{end} - x_k^{beg} \geq d_k^{min} + d_k^{fre} y_{k+1} + d_k^{acc} y_{k-1} & \forall k \in K_i, i \in T : r_k = 0 & (5.46) \\
x_k^{end} - x_k^{beg} \geq d_k^{min} & \forall k \in E : h_k = 1 \wedge r_k = 1 & (5.47) \\
x_k^{end} - x_k^{beg} \geq d_k^{min} y_k & \forall k \in E : h_k = 0 \wedge r_k = 1 & (5.48) \\
x_k^{beg} - x_k^{end} \geq -d_k^{max} y_k & \forall k \in E : h_k = 0 \wedge r_k = 1 & (5.49) \\
x_k^{beg} - x_k^{end} \geq -d_k^{max} & \forall k \in E : b_k^{inc} = e_k^{inc} = 0 & (5.50) \\
x_k^{beg} \geq b_k^{ini} & \forall k \in E : h_k = 1 & (5.51) \\
x_k^{beg} \geq b_k^{inc} & \forall k \in E : b_k^{inc} > 0 & (5.52) \\
x_k^{end} \geq e_k^{inc} & \forall k \in E : e_k^{inc} > 0 & (5.53) \\
z_k - x_k^{beg} \geq -b_k^{ini} & \forall k \in E & (5.54)
\end{array}$$

$$\begin{array}{lll}
x_{\hat{k}}^{beg} - x_k^{end} \geq \Delta_j^M \gamma_{k,\hat{k}} - M(1 - \gamma_{k,\hat{k}}) & \forall k,\hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E \neq o_{\hat{k}}^E & (5.55) \\
x_{\hat{k}}^{beg} - x_k^{end} \geq \Delta_j^M \lambda_{k,\hat{k}} - M(1 - \lambda_{k,\hat{k}}) & \forall k,\hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E \neq o_{\hat{k}}^E & (5.56) \\
x_{\hat{k}}^{beg} - x_k^{beg} \geq SP_{k,\hat{k}} \gamma_{k,\hat{k}} - M(1 - \gamma_{k,\hat{k}}) & \forall k,\hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 & (5.57) \\
x_k^{beg} - x_{\hat{k}}^{beg} \geq SP_{\hat{k},k} \lambda_{k,\hat{k}} - M(1 - \lambda_{k,\hat{k}}) & \forall k,\hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 & (5.58) \\
x_{\hat{k}}^{end} - x_k^{end} \geq SP_{k,\hat{k}} \gamma_{k,\hat{k}} - M(1 - \gamma_{k,\hat{k}}) & \forall k,\hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 & (5.59) \\
x_k^{end} - x_{\hat{k}}^{end} \geq SP_{\hat{k},k} \lambda_{k,\hat{k}} - M(1 - \lambda_{k,\hat{k}}) & \forall k,\hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 & (5.60) \\
x_{\hat{k}}^{beg} - x_k^{end} \geq \Delta_j^F \gamma_{k,\hat{k}} - M(1 - \gamma_{k,\hat{k}}) & \forall k,\hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 1 & (5.61) \\
x_k^{beg} - x_{\hat{k}}^{end} \geq \Delta_j^F \lambda_{k,\hat{k}} - M(1 - \lambda_{k,\hat{k}}) & \forall k,\hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 1 & (5.62)
\end{array}$$

$$x_{\hat{k}}^{end} - x_k^{beg} \geq g_{k,\hat{k}}^{min} \qquad \forall (k,\hat{k}) \in C \qquad (5.63)$$

$$
\begin{array}{ll}
x_k^{beg} - x_{\hat{k}}^{end} \geq -g_{k,\hat{k}}^{max} & \forall (k,\hat{k}) \in C \\
x_k^{beg}, x_k^{end}, z_k \geq 0 & \forall k \in E
\end{array}
\tag{5.64}
$$

Sous-problème (forme duale)

$$
\begin{array}{l}
f(q,y,\gamma,\lambda) = \max \\
\langle 0, u_k^{(5.45)} \rangle \\
+\langle d_k^{min} + d_k^{fre} y_{k+1} + d_k^{acc} y_{k-1}, u_k^{(5.46)} \rangle + \langle d_k^{min}, u_k^{(5.47)} \rangle + \langle d_k^{min} y_k, u_k^{(5.48)} \rangle \\
+\langle -d_k^{max} y_k, u_k^{(5.49)} \rangle + \langle -d_k^{max}, u_k^{(5.50)} \rangle \\
+\langle b_k^{ini}, u_k^{(5.51)} \rangle + \langle b_k^{inc}, u_k^{(5.52)} \rangle + \langle e_k^{inc}, u_k^{(5.53)} \rangle + \langle -b_k^{ini}, u_k^{(5.54)} \rangle \\
+\langle \Delta_j^M \gamma_{k,\hat{k}} - M(1-\gamma_{k,\hat{k}}), u_{k,\hat{k}}^{(5.55)} \rangle + \langle \Delta_j^M \lambda_{k,\hat{k}} - M(1-\lambda_{k,\hat{k}}), u_{k,\hat{k}}^{(5.56)} \rangle \\
+\langle SP_{k,\hat{k}} \gamma_{k,\hat{k}} - M(1-\gamma_{k,\hat{k}}), u_{k,\hat{k}}^{(5.57)} \rangle + \langle SP_{\hat{k},k} \lambda_{k,\hat{k}} - M(1-\lambda_{k,\hat{k}}), u_{k,\hat{k}}^{(5.58)} \rangle \\
+\langle SP_{k,\hat{k}} \gamma_{k,\hat{k}} - M(1-\gamma_{k,\hat{k}}), u_{k,\hat{k}}^{(5.59)} \rangle + \langle SP_{\hat{k},k} \lambda_{k,\hat{k}} - M(1-\lambda_{k,\hat{k}}), u_{k,\hat{k}}^{(5.60)} \rangle \\
+\langle \Delta_j^F \gamma_{k,\hat{k}} - M(1-\gamma_{k,\hat{k}}), u_{k,\hat{k}}^{(5.61)} \rangle + \langle \Delta_j^F \lambda_{k,\hat{k}} - M(1-\lambda_{k,\hat{k}}), u_{k,\hat{k}}^{(5.62)} \rangle \\
+\langle g_{k,\hat{k}}^{min}, u_{k,\hat{k}}^{(5.63)} \rangle + \langle -g_{k,\hat{k}}^{max}, u_{k,\hat{k}}^{(5.64)} \rangle \\
\text{sous les contraintes :}
\end{array}
$$

$$
\begin{array}{ll}
-u_{k-1}^{(5.45)} - u_k^{(5.46)} - u_k^{(5.47)} - u_k^{(5.48)} + u_k^{(5.49)} + u_k^{(5.50)} + u_k^{(5.51)} + u_k^{(5.52)} - u_k^{(5.54)} & \\
+\sum_{\hat{k}} u_{\hat{k},k}^{(5.55)} + \sum_{\hat{k}} u_{k,\hat{k}}^{(5.56)} - \sum_{\hat{k}} u_{k,\hat{k}}^{(5.57)} + \sum_{\hat{k}} u_{\hat{k},k}^{(5.57)} + \sum_{\hat{k}} u_{k,\hat{k}}^{(5.58)} - \sum_{\hat{k}} u_{\hat{k},k}^{(5.58)} & \\
+\sum_{\hat{k}} u_{\hat{k},k}^{(5.61)} + \sum_{\hat{k}} u_{k,\hat{k}}^{(5.62)} & \\
-\sum_{\hat{k}} u_{k,\hat{k}}^{(5.63)} + \sum_{\hat{k}} u_{k,\hat{k}}^{(5.64)} \leq 0 & \forall k \in E, u \text{ défini}
\end{array}
$$

$$
\begin{array}{ll}
+u_k^{(5.45)} + u_k^{(5.46)} + u_k^{(5.47)} + u_k^{(5.48)} - u_k^{(5.49)} - u_k^{(5.50)} + u_k^{(5.53)} & \\
-\sum_{\hat{k}} u_{k,\hat{k}}^{(5.55)} - \sum_{\hat{k}} u_{\hat{k},k}^{(5.56)} - \sum_{\hat{k}} u_{k,\hat{k}}^{(5.59)} + \sum_{\hat{k}} u_{\hat{k},k}^{(5.59)} + \sum_{\hat{k}} u_{k,\hat{k}}^{(5.60)} - \sum_{\hat{k}} u_{\hat{k},k}^{(5.60)} & \\
-\sum_{\hat{k}} u_{k,\hat{k}}^{(5.61)} - \sum_{\hat{k}} u_{\hat{k},k}^{(5.62)} & \\
+\sum_{\hat{k}} u_{\hat{k},k}^{(5.63)} - \sum_{\hat{k}} u_{\hat{k},k}^{(5.64)} \leq 0 & \forall k \in E, u \text{ défini}
\end{array}
$$

$$
\begin{array}{ll}
u_k^{(5.54)} \leq CD \cdot h_k & \forall k \in K_i, i \in T : k \neq k_i^n \\
u_k^{(5.54)} \leq CD \cdot h_k + CFD & \forall k \in K_i, i \in T : k = k_i^n
\end{array}
$$

$$
\begin{array}{ll}
u_k^{(5.45)} \in \mathbb{R} & \forall k \in K_i, i \in T : k \neq k_i^n \\
u_k^{(5.46)} \geq 0 & \forall k \in K_i, i \in T : r_k = 0 \\
u_k^{(5.47)} \geq 0 & \forall k \in E : h_k = 1 \wedge r_k = 1 \\
u_k^{(5.48)}, u_k^{(5.49)} \geq 0 & \forall k \in E : h_k = 0 \wedge r_k = 1 \\
u_k^{(5.50)} \geq 0 & \forall k \in E : b_k^{inc} = e_k^{inc} = 0 \\
u_k^{(5.51)} \geq 0 & \forall k \in E : h_k = 1 \\
u_k^{(5.52)} \geq 0 & \forall k \in E : b_k^{inc} > 0
\end{array}
$$

$$
\begin{array}{ll}
u_k^{(5.53)} \geq 0 & \forall k \in E : e_k^{inc} > 0 \\
u_k^{(5.54)} \geq 0 & \forall k \in E \\
u_{k,\hat{k}}^{(5.55)}, u_{k,\hat{k}}^{(5.56)} \geq 0 & \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E \neq o_{\hat{k}}^E \\
u_{k,\hat{k}}^{(5.57)}, u_{k,\hat{k}}^{(5.58)}, u_{k,\hat{k}}^{(5.59)}, u_{k,\hat{k}}^{(5.60)} \geq 0 & \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 0 \\
u_{k,\hat{k}}^{(5.61)}, u_{k,\hat{k}}^{(5.62)} \geq 0 & \forall k, \hat{k} \in L_j, j \in B : k \prec \hat{k}, o_k^E = o_{\hat{k}}^E, r_k = 1 \\
u_{k,\hat{k}}^{(5.63)}, u_{k,\hat{k}}^{(5.64)} \geq 0 & \forall (k, \hat{k}) \in C
\end{array}
$$

Coupes

Les coupes générées sont obtenus à partir de l'expression de la fonction objectif du sous-problème dual, dans laquelle les variables u sont remplacées par leur affectation dans la solution courante. Si le sous-problème dual est réalisable, le vecteur u correspond à un point extrême, et la coupe définit une borne inférieure à la valeur de $f(y, \gamma, \lambda)$. Si le sous-problème dual est non borné, le vecteur u définit un rayon extrême que la coupe interdit.

5.3.2 Résultats numériques

L'exécution sur une petite instance de test (3 circulations, 43 gares (totalisant 119 voies), 42 sections (totalisant 98 voies) pour un total de 220 événements) prend un temps considérable (plusieurs heures), à comparer à la seconde nécessaire à ILOG Cplex pour résoudre le modèle MIP correspondant). Nous représentons sur la figure 5.1 les temps de résolution cumulés pour le problème maître et le sous-problème en fonction des itérations. On pourra noter que la différence entre les temps d'exécution des problème maître et sous-problème est telle qu'il est nécessaire d'utiliser une échelle logarithmique.

On notera les quelques indicateurs suivants :

- la première solution réalisable trouvée à l'itération 186 (valeur : 53458),
- la solution réalisable de valeur optimale est trouvée à l'itération 365,
- le nombre total d'itérations est de 2413,
- 778 coupes de type I ont été générées,
- 1635 coupes de type II ont été générées.

La figure 5.3 représente l'évolution du nombre de coupes de type I et II en fonction du nombre d'itérations. On voit qu'initialement, ce sont principalement des coupes de type I qui sont générées, interdisant des configurations irréalisables. Les coupes de type II prennent ensuite le dessus. Toutefois, les coupes générées ne contribuent pas à faire augmenter significativement ma borne inférieure au cours des itérations (voir figure 5.2).

Pistes d'amélioration

À la vue des résultats obtenus, il peut paraître utile de chercher des pistes d'amélioration. Nous nous intéressons ici plus particulièrement à l'identification

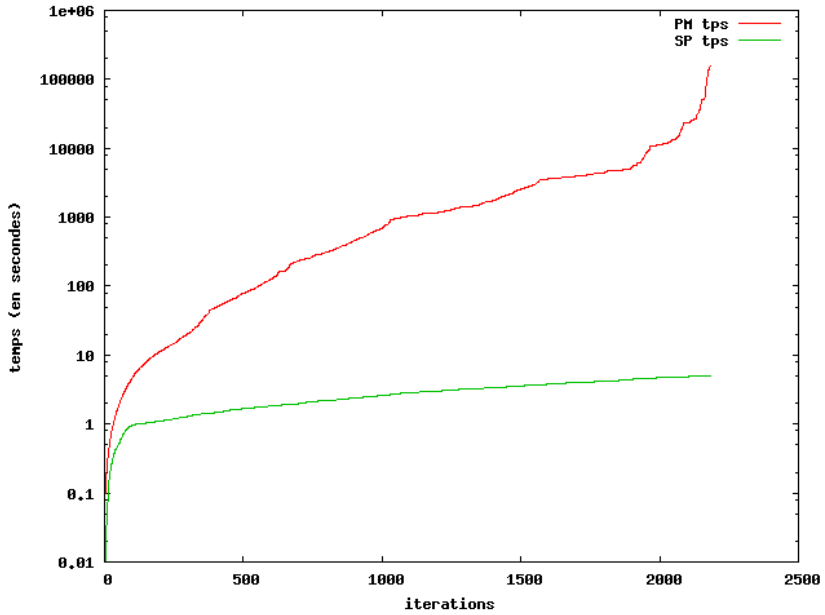

FIG. 5.1: Évolution des temps de résolution (cumul) des problème maître et sous-problème en fonction des itérations.

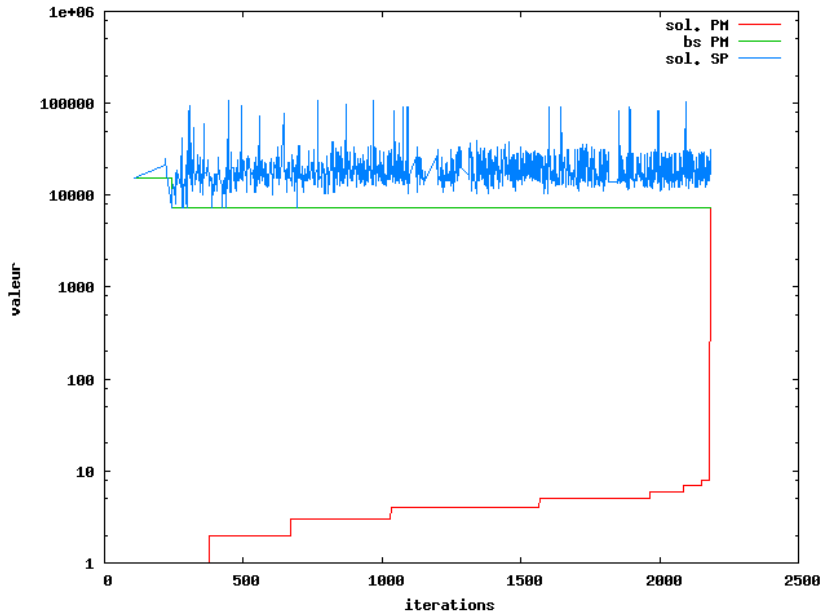

FIG. 5.2: Évolution de la valeur optimale et des bornes inférieures et supérieures des problème maître et sous-problème en fonction des itérations.

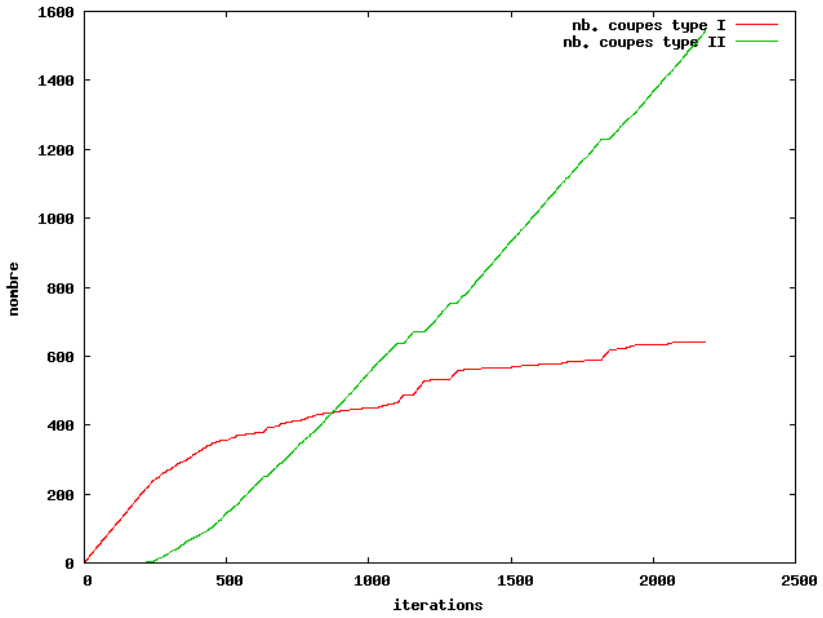

FIG. 5.3: Évolution du nombre des coupes réalisables et irréalisables en fonction des itérations.

des coupes.

Le cas d'un sous-problème non borné peut être dû globalement à deux raisons :

- soit l'affectation des variables λ et γ rend l'ordonnancement des trains irréalisable ;
- soit des arrêts non prévus (variables y) doivent être effectués, mais ne peuvent compte tenu des relations avec les autres circulations.

Dans le premier cas, il est possible de s'intéresser à la structure du graphe obtenu en considérant l'ensemble des relations de précédence entre événements, qu'il s'agisse de l'ordre entre événements d'une même circulation, ou des relations entre événements de circulations différentes, par l'affectation des variables λ et γ. Dans ce graphe, la non-faisabilité du sous-problème va se traduire par la présence d'un ou plusieurs cycles, qu'il sera nécessaire d'éliminer. Dans le cas des variables y, il paraît plus difficile de déterminer des coupes analytiquement. La non-faisabilité met en jeu l'ordre des événements, leurs durées minimales et maximales, et les arrêts imposés. On pourrait de plus se poser la question du rôle que jouent les variables q dans la recherche d'une solution optimale, et s'il est possible de les exploiter plus spécifiquement. On peut aussi se demander comment mettre à profit une borne inférieure (obtenue par exemple par relaxation linéaire du modèle agrégé). La difficulté vient alors de la présence du terme $f(y, \gamma, \lambda)$, représentant une borne inférieure de la valeur du sous-problème. La

méthode reste toutefois utilisable dans le cadre d'heuristiques, en limitant par exemple la combinatoire (nombre de modifications permises autour d'un événement pertubé).

Bibliographie

[1] Philippe Baptiste and Claude Le Pape. Edge-finding constraint propagation algorithms for disjunctive and cumulative scheduling. *Proceedings of the Fifteenth Workshop of the UK Planning Special Interest Group*, 335 :339–345, 1996.

[2] F. Barber, M. Abril, M.A. Salido, L.P. Ingolotti, P. Tormos, and A. Lova. Survey of automated Systems for Railway Management. Technical report, Universidad Politécnica de Valencia, Department of Information Systems and Computation, 2007.

[3] Jean-Luc Barnagaud. La construction du graphique. *Revue Générale des Chemins de Fer*, pages 123–138, 2002.

[4] J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4 :238–252, 1962.

[5] M.R. Bussieck, T. Winter, and U.T. Zimmermann. Discrete optimization in public rail transport. *Mathematical Programming*, 79(1) :415–444, 1997.

[6] Alberto Caprara, Matteo Fischetti, P.L. Guida, M. Monaci, G. Sacco, and Paolo Toth. Solution of real-world train timetabling problems. *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, page 10, 2001.

[7] Alberto Caprara, Matteo Fischetti, and Paolo Toth. Modeling and Solving the Train Timetabling Problem. *Operations Research*, 50(5) :851–861, 2002.

[8] Jacques Carlier and Éric Pinson. A Practical Use of Jackson's Preemptive Schedule for Solving the Job-Shop Problem. *Annals of Operations Research*, 26 :269–287, 1990.

[9] V. Černỳ. Thermodynamical approach to the traveling salesman problem : An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1) :41–51, 1985.

[10] S.C. Chang and Y.C. Chung. From timetabling to train regulation : a new train operation model. *Information and Software Technology*, 47(9) :575–585, 2005.

[11] C.K. Chiu, C.M. Chou, J.H.M. Lee, H.F. Leung, and Y.W. Leung. A Constraint-Based Interactive Train Rescheduling Tool. *Constraints*, 7 :167–198, 2002.

[12] H.W. Chun. Train timetable and route generation using a constraint-based approach. In *Proceedings of the 10th international conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 441–448. Goose Pond Press, 1997.

[13] Jean-François Cordeau, Paolo Toth, and Daniele Vigo. A Survey of Optimization Models for Train Routing and Scheduling. *Transportation Science*, 32(4) :380–404, 1998.

[14] Andrea D'ariano, Dario Pacciarelli, and Marco Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183 :643–657, 2007.

[15] E.S. de Oliveira. *Solving Single-Track Railway Scheduling Problem Using Constraint Programming*. PhD thesis, The University of Leeds, 2001.

[16] Xavier Delorme, Joaquín Rodriguez, and Xavier Gandibleux. Heuristics for railway infrastructure saturation. *Electronic Notes in Theoretical Computer Science*, 50(1) :1–15, 2004.

[17] Gilles Dessagne. Module de régulation : Spécification fonctionnelle du prototype ; spécification des algorithmes d'optimisation. Note interne SNCF, Projet LIPARI, janvier 2003.

[18] Johann Dréo, Alan Pétrowski, Patrick Siarry, and Éric Taillard. *Métaheuristiques pour l'optimisation difficile*. Eyrolles, 2002.

[19] M. Fontaine and D. Gauyacq. SISYFE : a toolbox to simulate the railway network functioning for many purposes. some cases of application. *Proceedings of the World Congress on Railway Research (WCRR 2001)*, 2001.

[20] Arthur M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10(4) :237–260, 1972.

[21] F.W. Glover and M. Laguna. *Tabu Search*. Springer, 1997.

[22] ILOG. Optimization Decision Management System (ODMS), 2006. http://www.ilog.com/products/optimization/.

[23] S. Kirkpatrick, C.D. Gelatt Jr, and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598) :671, 1983.

[24] T. Lindner and U.T. Zimmermann. Cost optimal periodic train scheduling. *Mathematical Methods of Operations Research (ZOR)*, 62(2) :281–295, 2005.

[25] L. Lucchini, R. Rivier, and D. Emery. CAPRES network capacity assessment for Swiss North-South rail freight traffic. *International conference on computers in railways*, (7) :221–230, 2000.

[26] A. Mascis and D. Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3) :498–517, 2002.

[27] Thierry Mautor and Philippe Michelon. Mimausa : An Application of Referent Domain Optimization. Technical report, Technical Report, Laboratoire d'Informatique d'Avignon, 2001.

[28] L.W.P. Peeters. *Cyclic Railway Timetable Optimization*. EPS ; EPS-2003-022-LIS, 2003.

[29] Éric Pinson. *Le problème de job-shop*. PhD thesis, Université Paris VI, 1988.

[30] Joaquín Rodriguez. A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B : Policy and Practice*, 41(2) :231–245, 2007.

[31] Johanna Törnquist. Railway traffic disturbance management - An experimental analysis of disturbance complexity, management objectives and limitations in planning horizon. *Transportation Research Part A : Policy and Practice*, 41(3) :249–266, 2007.

[32] Johanna Törnquist and Jan A. Persson. Train Traffic Deviation Handling Using Tabu Search and Simulated Annealing. In *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)-Track 3-Volume 03*. IEEE Computer Society Washington, DC, USA, 2005.

[33] Johanna Törnquist and Jan A. Persson. N-tracked railway traffic rescheduling during disturbances - Methodological. *Transportation Research Part B : Policy and Practice*, 41(3) :249–266, 2007.

[34] P.J. Zwaneveld. *Railway Planning : Routing of Trains and Allocation of Passenger Lines*. PhD thesis, Erasmus Universiteit Rotterdam, 1997.

[35] P.J. Zwaneveld, L.G. Kroon, H.E. Romeijn, M. Salomon, Stéphane Dauzère-Péres, S.P.M. Van Hoesel, and H.W. Ambergen. Routing trains through railway stations : Model formulation and algorithms. *Transportation science*, 30(3) :181–194, 1996.

[36] P.J. Zwaneveld, L.G. Kroon, and S.P.M. van Hoesel. Routing trains through a railway station based on a node packing model. *European Journal of Operational Research*, 128(1) :14–33, 2001.

Liste des tableaux

Table des figures

Liste des algorithmes

Table des matières

Annexe A

Définitions

A.1 Réseau ferroviaire

A.1.1 Point

Dans une représentation macroscopique d'un réseau ferroviaire, on appelle point les composantes de ce réseau. Les points sont constitués par les noeuds et les tronçons du réseau.

A.1.2 Tronçon (*segment* ou *section*)

Un tronçon est une portion homogène d'un réseau ferroviaire, dans laquelle une circulation ne peut que circuler d'une de ses extrémités à son autre extrémité opposé.

A.1.3 Nœud

Un noeud est une composante macroscopique du réseau ferroviaire permettant la jonction de deux ou plusieurs tronçons. Éventuellement, il représente la terminaison d'un tronçon. Les noeuds sont formés par les gares et les point de jonction/bifurcation du réseau.

A.1.4 Gare (ou *station*)

Type de noeud permettant l'arrêt d'une circulation, le plus souvent pour desserte.

A.1.5 Bifurcation

Type de noeud permettant un changement de voie d'une circulation au moyen d'aiguilles. Les arrêts sont généralement interdits sur ces nœuds.

A.1.6 Voie

Une voie ferrée est un chemin de roulement pour les circulations ferroviaires, constitué d'une ou plusieurs files de rails dont l'écartement est maintenu par

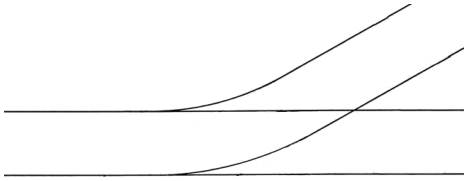

FIG. A.1: Branchement simple (ici, à gauche).

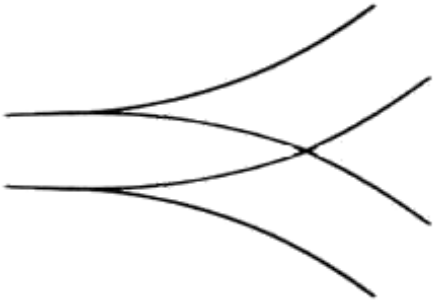

FIG. A.2: Branchement double.

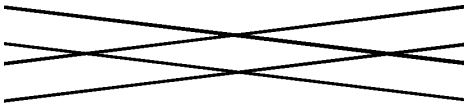

FIG. A.3: Traversée.

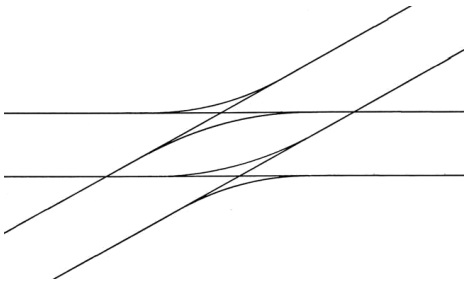

FIG. A.4: Traversée-croisement.

une fixation sur des traverses, reposant sur du ballast.. Tout point du réseau est constitué d'une ou plusieurs voies.

Les voies sont classées en plusieurs grandes catégories, chacune sous-entendant une vitesse maximale et une charge à l'essieu. On distingue ainsi :

- les voies principales, affectées à la circulation des trains,
- les voies de circulation affectées à desserte interne des grands complexes ferroviaires ;
- les voies de service, affectées aux man½uvres, qui peuvent être d'anciennes voies principales déclassées,
- les voies de garage sont des voies de service affectées au stationnement du matériel roulant.

A.1.7 Canton

Le cantonnement est le moyen généralement employé pour assurer l'espacement des trains circulant dans le même sens sur une même voie. Par principe, on n'admet que la présence d'un seul train dans un canton donné. Lorsqu'un train pénètre dans un canton, le signal d'entrée du canton est fermé. Lorsque le train poursuivant sa marche entre dans le canton suivant, le signal d'entrée de ce dernier est fermé tandis que celui du canton précédent est ouvert. Cela se fait soit manuellement par échange d'informations entre postes de cantonnement, soit dans les chemins de fer modernes de manière automatique, grâce aux systèmes dits de « block automatique » utilisant les circuits de voie. On parle de « cantonnement absolu » quand cette règle est appliquée strictement.

A.1.8 Circuit de voie

Un circuit de voie est un circuit électrique, empruntant les rails d'une voie ferrée, utilisé pour détecter la présence d'un train dans la section de voie considérée (qui constitue un canton) et commander automatiquement les signaux de protection.

A.2 Circulations

A.2.1 Circulation

Une circulation (ou convoi ferroviaire) est un ensemble de véhicules tractés (voitures, wagons, parfois groupés en rames) et de véhicules tracteurs (locomotive ou automotrice) attelés ensemble.

A.2.2 Itinéraire

Succession des points empruntés par une circulation.

A.2.3 Desserte

Arrêt d'une circulation en gare afin de permettre le transfert de personnes ou marchandises.

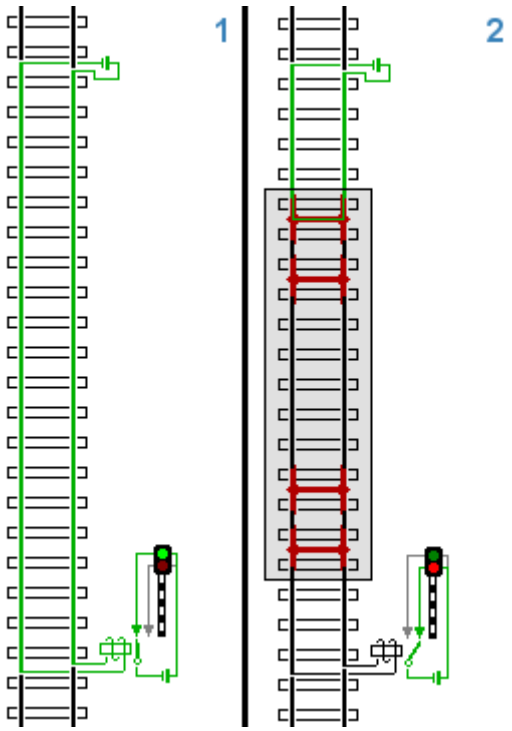

FIG. A.5: Fonctionnement d'un circuit de voie.

A.2.4 Circulation associée et navette

Deux circulations à l'origine indépendantes que l'on attelle ensemble pour n'en former plus qu'une. La circulation origine est appelée navette, et la circulation restante circulation associée.

A.2.5 Correspondance

Deux circulations sont en correspondance si une partie des passagers d'une des circulations est supposée emprunter l'autre circulation. Ceci implique que cette deuxième circulation ne peut partir qu'après l'arrivée de la première (et en assurant un temps mimimum de desserte, afin de permettre aux voyageurs le changement de circulation).

A.2.6 Echange

On parle d'échange lorsque deux circulations sont en correspondance l'une avec l'autre.

A.3 Divers

A.3.1 Graphique de circulation

un graphique de circulation est un document espace-temps, qui traduit graphiquement, sous forme de vecteur, la marche de chacun des trains sur une section de ligne donnée.

Annexe B

Représentation de l'instance de test

La figure B.1 représente la structure du réseau (généré aléatoirement) utilisée pour les tests numériques de la section 4. Les longueurs des arêtes ne sont pas représentatives des données, dans lesquelles les durées de parcours sont identiques quel que soit le tronçon considéré.

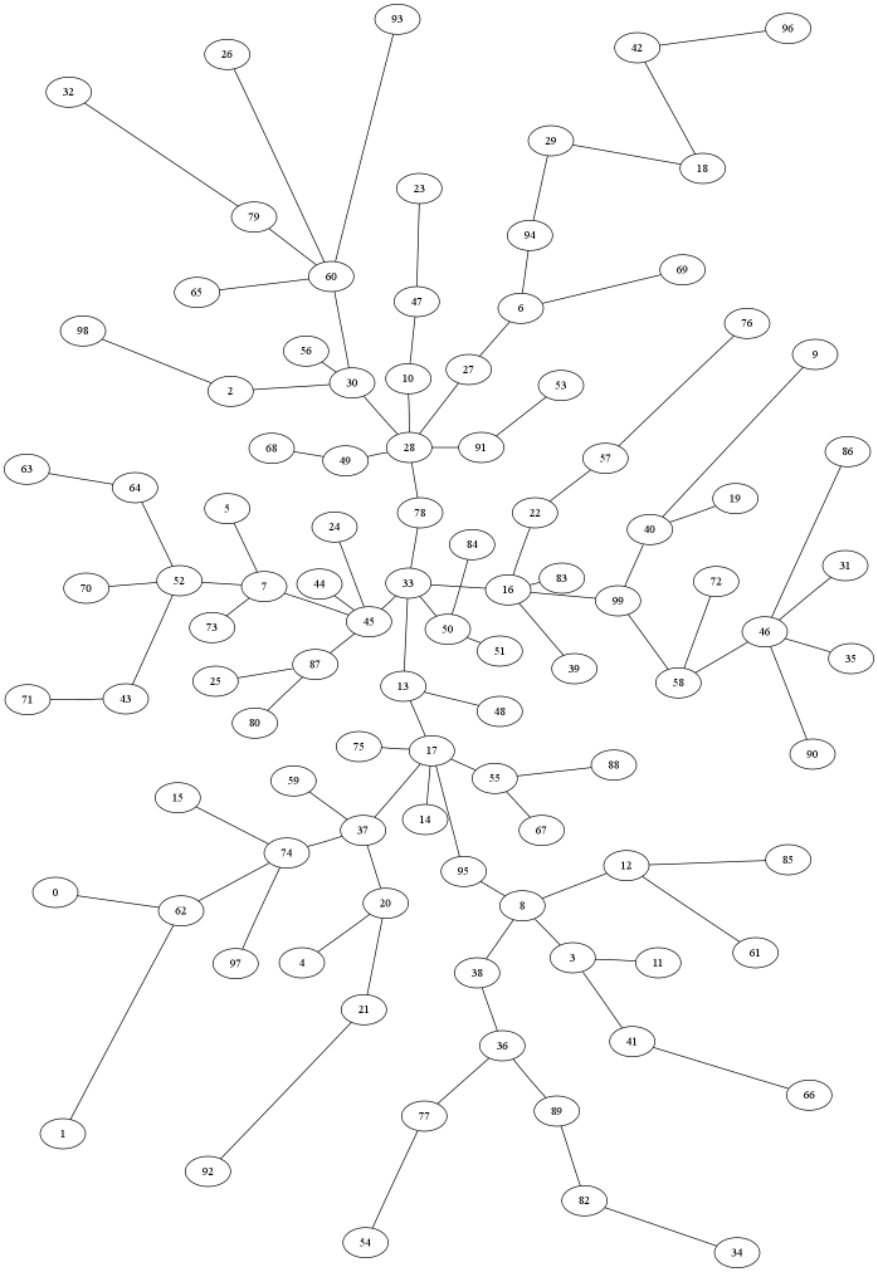

FIG. B.1: Représentation de l'instance de test.