Université de Valenciennes et du Hainaut-Cambrésis

Numéro d'ordre : 03-35

THÈSE

pour l'obtention du

Doctorat de l'université de Valenciennes et du Hainaut-Cambrésis

Spécialité Automatique et Informatique des Systèmes Industriels et Humains

Discipline : Informatique

Modélisation et résolution de problèmes liés à l'exploitation d'infrastructures ferroviaires

présentée par

Xavier DELORME

Soutenue publiquement le 10 décembre 2003 devant le jury composé de :

Daniel GAUYACQ	Docteur, SNCF	Président
Stéphane DAUZÈRE-PÉRÈS	Professeur, École des Mines de Nantes	Rapporteur
Celso RIBEIRO	Professeur, University of Rio de Janeiro	Rapporteur
Daniel VANDERPOOTEN	Professeur, LAMSADE (Paris IX)	Rapporteur
Arnaud FRÉVILLE	Professeur, Conseil régional Nord-Pas-de-Calais	Directeur
Xavier GANDIBLEUX	Maître de Conférences, LAMIH (UVHC)	Co-directeur
Joaquín RODRIGUEZ	Directeur de Recherche, INRETS-ESTAS	Co-directeur

Je dédie ce mémoire à ma femme, sans son amour et son soutien quotidien ce travail n'aurait jamais pu aboutir.

Avant-propos

Je tiens tout d'abord à témoigner toute ma reconnaissance à Messieurs Arnaud Fréville, Professeur et Directeur des transports au conseil régional du Nord-Pas-de-Calais, Xavier Gandibleux, Maître de Conférences à l'Université de Valenciennes, et Joaquín Rodriguez, Directeur de Recherche à l'Institut National de Recherche sur les Transports et leur Sécurité (INRETS), qui ont accepté de diriger cette thèse. Leurs nombreux conseils, et les longues discussions que nous avons eues, m'ont guidés tout au long de ce travail.

J'exprime également ma profonde gratitude à Messieurs Stéphane Dauzère-Pérès, Professeur à l'École des Mines de Nantes, Celso Ribeiro, Professeur à l'Université de Rio de Janeiro, et Daniel Vanderpooten, Professeur à l'Université de Paris-Dauphine, qui m'ont fait l'honneur d'être rapporteurs de ce mémoire.

Je remercie aussi Monsieur Daniel Gauyacq, Responsable de l'unité de recherche Génie décisionnelle appliqué de la Direction de la Recherche et de la Technologie de la SNCF, pour avoir accepté de participer à mon jury.

Ce travail n'aurait pas connu le même aboutissement sans la collaboration de Madame Corinne Carton et de Monsieur Roman Bartusiak, Chargés d'Études à la SNCF (Délégation Régionale Infrastructures de Lille), qui ont accepté de partager leur expérience.

De même, j'ai aussi pu bénéficier de l'aide de Messieurs Fabien Degoutin, Doctorant de l'Université de Valenciennes, et Grégory Marlière, Assistant-Ingénieur à l'INRETS. Qu'ils en soient ici remerciés.

J'adresse également tous mes remerciements à l'ensemble des personnes que j'ai cotoyé durant ces trois années au sein de l'INRETS et de l'Université de Valenciennes. Je pense plus particulièrement à ceux ayant travaillé avec moi au sein de l'unité de recherche Évaluation des Systèmes de Transports Automatisés et de leur Sécurité dirigée par Monsieur Gérard Couvreur, et de l'équipe Recherche Opérationnelle et Informatique du Laboratoire d'Automatique et d'Informatique industrielles et Humaines dirigée par le Professeur Frédéric Semet. Je tiens notamment à saluer David, Dorian, Georges, Olivier, Samuel et Philippe pour leur

aide et leur bonne humeur. Chacuns à leur manière, ils ont contribué à la réalisation de ce mémoire.

Enfin, je remercie ma femme, ma famille et mes amis pour la patience qu'ils ont eue à mon égard et les moments de joie et de détente qu'ils ont su m'apporter.

Table des matières

Table des figures **ix**

Liste des tableaux **xi**

Table des algorithmes **xiii**

Table des définitions **xv**

Introduction **1**

1 Contexte de modélisation **5**

1.1 Principes de base 5

1.1.1 Qu'est-ce qu'un modèle linéaire ? 6

1.1.2 Résolution exacte d'un modèle linéaire 7

1.1.3 Résolution approchée d'un modèle linéaire 10

1.2 Un problème facile : le problème de plus court chemin 11

1.2.1 Définition du problème de plus court chemin 12

1.2.2 Résolution du problème de plus court chemin 13

1.3 Un problème difficile : le problème de set packing 13

1.3.1 Définition du problème de set packing 14

1.3.2 Similarités avec d'autres problèmes en variables binaires 15

1.3.3 Résolution du problème de set packing 19

1.4 Optimisation de problèmes multiobjectifs 20

1.4.1 Qu'est-ce qu'un problème multiobjectif ? 20

1.4.2 Résolution d'un problème multiobjectif 22

1.5 Conclusion 26

2 Évaluation de la capacité d'infrastructures ferroviaires **29**

2.1 Qu'est-ce qu'une infrastructure ferroviaire ? 29

2.2 Pourquoi évaluer la capacité ? 32

2.2.1 Positionnement du problème 32

2.2.2 Importance du problème 33
2.3 Définition des problèmes de capacité 33
2.4 Comment évaluer la capacité : travaux existants 37
2.4.1 DONS 38
2.4.2 CAPRES 42
2.4.3 DÉMIURGE 43
2.4.4 Synthèse des travaux existants 44
2.5 Conclusion 45

3 Description du modèle ferroviaire 47
3.1 Notations et définitions 47
3.1.1 Ensembles de trains 47
3.1.2 Parcours des trains 48
3.1.3 Horaires des trains 49
3.1.4 Caractéristiques de l'infrastructure 51
3.1.5 Variables de décision 51
3.2 Faisabilité et saturation d'une grille horaire 52
3.2.1 Objectifs du problème 52
3.2.2 Contraintes du problème 54
3.2.3 Modèle complet 61
3.3 Évaluation de la stabilité du routage 64
3.4 Conclusion 67

4 Résolution du problème de set packing 71
4.1 Résolution exacte et bornes 71
4.1.1 Pré-traitements 71
4.1.2 Résolution exacte dans le cas mono-objectif 75
4.1.3 Recherche dichotomique dans le cas biobjectif 76
4.2 Résolution approchée 78
4.2.1 La métaheuristique GRASP 79
4.2.2 GRASP pour le problème de set packing 80
4.2.3 Extension de GRASP au cas biobjectif 86
4.3 Conclusion 87

5 Résultats des expérimentations 89
5.1 Conditions d'expérimentation 89
5.1.1 Instances mono-objectifs générées aléatoirement 90
5.1.2 Instances ferroviaires issues du nœud de Pierrefitte-Gonesse 91
5.1.3 Instances biobjectifs 92

5.2 Résolution exacte et borne supérieure 94
5.2.1 Impact des pré-traitements 94
5.2.2 Résultats obtenus dans le cas mono-objectif 100
5.2.3 Détermination de la frontière efficace dans le cas biobjectif 103
5.3 Résolution approchée 104
5.3.1 Impact de chaque stratégie considérée pour GRASP 105
5.3.2 Impact de chaque phase de GRASP 107
5.3.3 Solutions obtenues avec notre procédure GRASP complète 110
5.3.4 Approximation de la frontière efficace dans le cas biobjectif 110
5.4 Conclusion 114

6 Description de la plate-forme expérimentale 117
6.1 Le projet RECIFE 117
6.2 Déroulement du projet 118
6.2.1 Recueil des données du noeud Lillois 119
6.2.2 Modules d'analyse et de simulation 120
6.2.3 Validation sur le noeud Lillois 123
6.3 Architecture logicielle 123
6.4 Conclusion 124

Conclusion générale et perspectives 127

Glossaire 133

Bibliographie 135

Index 144

Table des figures

1.1 Points caractéristiques d'un problème de maximisation biobjectif 22

2.1 Éléments d'une infrastructure ferroviaire 30
2.2 Exemple d'infrastructure 31
2.3 Exemple de graphique de circulation 31
2.4 Exemple de diagramme de Gantt 31

3.1 Utilisation des zones 52
3.2 Conflit entre deux trains sur une zone 55
3.3 Cadencement d'un train 55
3.4 Quai au passage 57
3.5 Quai en cul-de-sac 57
3.6 Correspondance entre deux trains 58
3.7 Couplage sur un quai en cul-de-sac 60
3.8 Exemple de matrice des contraintes 63
3.9 Graphe des conflits potentiels directs 68

4.1 Cliques propres au problème de capacité 77
4.2 Test de dominance entre variables 77
4.3 Instance réduite 77
4.4 Principe du path relinking 85
4.5 Schéma de l'utilisation de GRASP dans le cas biobjectif 86

5.1 Plan de voies du nœud de Pierrefitte-Gonesse 91
5.2 Impact des pré-traitements sur le nombre de contraintes 96
5.3 Impact des pré-traitements sur la densité des contraintes 97
5.4 Impact des pré-traitements sur la valeur de la relaxation linéaire 99
5.5 Impact des pré-traitements sur le gap obtenu avec Cplex 101
5.6 Résolution avec Cplex et borne supérieure 102
5.7 Exemple de frontière efficace et de bornes 104
5.8 Impact de chaque stratégie utilisée pour GRASP 106

5.9 Impact de chaque phase de notre procédure GRASP 108
5.10 Répartition par phase du temps utilisé par notre procédure GRASP 109
5.11 Synthèse des résultats obtenus avec notre procédure GRASP complète 111
5.12 Exemple d'approximations obtenues 115

6.1 Schéma de la gare de Lille-Flandres 119
6.2 Édition de l'infrastructure 120
6.3 Visualisation sous la forme d'un diagramme de Gantt 121
6.4 Visualisation des parcours sur le plan de voies 122
6.5 Visualisation de la position des trains lors d'une simulation 122
6.6 Projets INRETS-ESTAS sur le thème de l'exploitation ferroviaire 123
6.7 Architecture de la plate-forme du projet RECIFE 125

Liste des tableaux

2.1 Niveaux de décisions en gestion prévisionnelle des modes de transport en commun 32
2.2 Synthèse des différentes études de capacité 44

5.1 Caractéristiques des instances mono-objectifs générées aléatoirement 93
5.2 Caractéristiques des instances ferroviaires 93
5.3 Pourcentage de variables supprimées pour les instances aléatoires 95
5.4 Pourcentage de variables supprimées pour les instances ferroviaires 95
5.5 Pourcentages moyens de solutions efficaces trouvées 113
5.6 Distances moyennes à la frontière efficace 114
5.7 Pourcentages moyens de l'hypervolume de la frontière efficace 114

Liste des algorithmes

4.1 Calcul de cliques maximales 73
4.2 Test de dominance entre variables 74
4.3 Test de dominance entre contraintes 75
4.4 GRASP 80
4.5 Construction gloutonne aléatoire greedy1 81
4.6 Construction gloutonne aléatoire greedy2 81
4.7 Construction gloutonne aléatoire greedy3 82
4.8 Reactive GRASP 83
4.9 Recherche locale 84
4.10 Path relinking 85

Liste des définitions

1.1 Heuristique selon Reeves 10
1.2 Métaheuristique selon Osman et Laporte 11
1.3 Métaheuristique selon Pirlot 11
1.4 Chemin dans un graphe 12
1.5 Clique dans un graphe 18
1.6 Dominance entre deux solutions 21
1.7 Pareto optimalité 21

2.1 Capacité d'une infrastructure ferroviaire selon Bellaiche 33
2.2 Capacité d'une infrastructure ferroviaire selon Hachemane 34
2.3 Problème de saturation 35
2.4 Problème de faisabilité 35
2.5 Problème d'optimisation des préférences 36
2.6 Problème de fluidification 36

Introduction

Ce mémoire synthétise mes travaux de recherche réalisés, durant ces trois dernières années, au sein de l'INRETS[1] (unité de recherche ESTAS[2]) et du LAMIH[3] (équipe ROI[4]) dans le cadre de mon doctorat. Ils s'inscrivent dans l'une des thématiques principales de ces deux laboratoires : la gestion et l'optimisation des systèmes de transport. Plus précisément, le problème traité ici relève du domaine de la planification et de la programmation de l'exploitation ferroviaire. Parmi les très nombreuses problématiques issues de ce domaine, nous nous intéressons à la question de **l'évaluation de la capacité des infrastructures**.

Cette question s'inscrit dans le cadre d'un processus de gestion ferroviaire prévisionnelle à long et moyen terme. Pour mettre en œuvre une stratégie d'offre, il est impératif de se doter d'outils permettant de situer les limites d'un réseau (existant ou futur) par rapport à une, ou plusieurs, offre(s) possible(s). En outre, ces outils doivent aussi faciliter l'étude précise de chaque variante d'infrastructure pour en évaluer l'intérêt, la construction de nouvelles infrastructures nécessitant la mobilisation de moyens financiers très importants. L'évaluation de la capacité d'une infrastructure ferroviaire permet ainsi de concevoir et d'évaluer une offre répondant à la demande future sans surdimensionner les investissements et en utilisant au mieux l'existant.

Cette question s'avère cruciale dans le contexte actuel. En effet la demande, en terme de transport ferroviaire, est appelée à évoluer (concurrence des autres modes de transport, séparation entre le gestionnaire et l'exploitant de l'infrastructure, ...) et à croître fortement (besoins accrus en terme de mobilité, substitution souhaitée à la route, ...) dans les prochaines années. Dans ce cadre, cette problématique se révèle particulièrement importante dans une région comme le Nord-Pas-de-Calais, celle-ci étant placée en position de carrefour au niveau européen. Le **projet régional RECIFE**, dans lequel s'inscrit cette thèse, tente de répondre à ce besoin croissant. Il est l'objet d'une collaboration en cours entre l'INRETS, le LAMIH et la SNCF[5] et doit donner lieu à une validation sur la gare de Lille-Flandres.

[1] Institut National de REcherche sur les Transports et leur Sécurité

[2] Évaluation des Systèmes de Transport Automatisés et de leur Sécurité

[3] Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines, UMR CNRS 8530

[4] Recherche Opérationnelle et Informatique

[5] Société Nationale des Chemins de fer Français

Le champ de questions soulevées lors d'une étude de capacité est vaste. Il inclut notamment la détermination du nombre de trains pouvant circuler dans l'infrastructure. Cependant, d'autres questions se posent aussi très souvent :

- La capacité de l'infrastructure est-elle suffisante pour faire face aux évolutions de l'offre ?
- Quels sont les investissements offrant le meilleur débit pour les offres à venir ?
- Quelle est la meilleure façon de répondre à l'offre ?

De plus, si ces études peuvent être réalisées sans difficulté particulière pour des infrastructures simples, la combinatoire induite par la complexité de certaines infrastructures rend nécessaire l'utilisation de méthodes algorithmiques. Cela est notamment le cas lors de l'étude d'une portion importante d'un réseau ou d'une zone comprenant de nombreux croisements comme un nœud. L'accroissement des circulations transforme ainsi certains nœuds du réseau ferroviaire en véritables goulets d'étranglement. Cette situation y rend très critique la gestion du trafic. Ces nœuds se comportent alors comme de véritables amplificateurs de retards, en transformant une petite perturbation en une avalanche de retards.

Face à ce genre de problèmes, la recherche opérationnelle a engendré un grand nombre de méthodes permettant de répondre de manière efficace, du moins lorsque les objectifs ne sont pas flous ou trop ambitieux. Parmi ces problèmes, ceux pouvant s'exprimer sous la forme d'un modèle linéaire, et présentant des structures particulières, sont les plus faciles à aborder quantitativement. **L'objectif de ce travail est de proposer une modélisation linéaire du problème de capacité ferroviaire à l'échelle d'un nœud, ainsi que des algorithmes permettant de le résoudre**. Notre modèle conduit ainsi à deux structures caractéristiques correspondant à des problèmes d'optimisation classiques appelés « plus court chemin » et « set packing ». Il s'inspire de celui proposé par Zwaneveld et al. [ZKR+96, Zwa97, ZKVH01] dans le cadre d'une étude réalisée pour les chemins de fer néerlandais.

Si le problème de plus court chemin est facile à résoudre et qu'il existe plusieurs algorithmes très performants pour cela, ce n'est pas le cas pour celui de set packing qui est connu comme NP-difficile. Nous proposons donc différents algorithmes pour résoudre le problème de set packing, dont certains tirent profit des caractéristiques spécifiques du problème de capacité. Ceux-ci peuvent entrer dans le cadre d'une résolution exacte permettant de déterminer la solution optimale. Cependant, en raison de la taille des problèmes considérés, et afin de pouvoir traiter des cas de figures complexes dans un contexte de temps limité, une résolution approchée générant de «bonnes» solutions a aussi été développée. Celle-ci est basée sur les principes de la métaheuristique GRASP proposée par Féo et Resende [FR89]. Enfin, pour prendre en compte les objectifs multiples (et souvent contradictoires) soulevés par les nombreuses questions liées à une étude de capacité, des travaux sur des algorithmes biobjectifs

ont aussi été engagés. Des expérimentations numériques, menées sur des instances correspondant au problème ferroviaire étudié mais aussi sur d'autres générées aléatoirement, montrent l'efficacité de ces algorithmes pour obtenir la solution optimale, ou au moins une solution de très bonne qualité dans des temps raisonnables.

Dans la suite, ce mémoire s'organise de la manière suivante :

Le **chapitre 1** fournit les éléments de recherche opérationnelle nécessaires à la compréhension de la suite du mémoire, et notamment du modèle et des algorithmes proposés. Les caractéristiques des problèmes classiques de plus court chemin et de set packing, dont la structure se retrouve dans notre modélisation, y sont décrits plus en détail. De plus, les caractéristiques principales des problèmes multiobjectifs et de leurs méthodes de résolution sont indiquées.

Le problème de capacité d'infrastructures ferroviaires que nous avons étudié est, lui, décrit dans le **chapitre 2**. Sa position dans le domaine de la gestion ferroviaire prévisionnelle et son importance y sont soulignés. Les principaux modèles et méthodes de résolution utilisés actuellement pour ce type de problème sont aussi présentées.

Le **chapitre 3** développe les différents éléments de la modélisation que nous proposons. Cette modélisation couvre notamment les problèmes de la faisabilité du passage d'un groupe de trains dans un nœud, de sa saturation, et de l'évaluation de la stabilité des solutions obtenues.

Les différents algorithmes que nous proposons pour résoudre le problème de set packing sont présentés au **chapitre 4**. La méthode de résolution exacte considérée est basée sur des algorithmes de pré-traitements mettant à profit la structure de set packing et les spécificités du problème ferroviaire, et sur le logiciel Cplex [ILO02]. La méthode de résolution approchée, quant à elle, est une adaptation de la métaheuristique GRASP. En outre, une première extension de ces méthodes dans le cas biobjectif est indiquée.

Dans le **chapitre 5**, nous rapportons l'ensemble des résultats obtenus lors d'expérimentations numériques avec les différents algorithmes présentés au chapitre 4. L'impact des pré-traitements y est observé. Ils permettent ainsi d'améliorer de manière significative les résultats obtenus avec Cplex, même si toutes les instances ne peuvent pas être résolues. De même, l'efficacité des différentes stratégies proposées pour l'algorithme GRASP, et l'intérêt de chaque composant, sont mesurés. La capacité de cet algorithme à générer des solutions de bonne qualité en un temps raisonnable pour l'ensemble des instances est mis en évidence. Enfin, son extension au cas biobjectif est comparée avec la métaheuristique multiobjectif SPEA.

Pour finir, le **chapitre 6** présente le projet RECIFE dans lequel s'inscrivent l'ensemble des travaux présentés dans les chapitres précédents. Les différents modules développés dans le cadre de ce projet sont notamment décrits.

Chapitre 1

Contexte de modélisation

L'objet de la recherche opérationnelle est l'amélioration du fonctionnement des entreprises et des organismes publics par l'application de l'approche scientifique. Reposant sur l'utilisation de méthodes scientifiques, de techniques spécialisées et des ordinateurs, elle permet d'obtenir une évaluation quantitative des politiques, stratégies et actions possibles dans le cours des opérations d'une organisation ou d'un système. Elle est apparue en tant que telle en Grande-Bretagne durant la seconde guerre mondiale, lorsqu'on décida d'employer des méthodes scientifiques pour étudier divers aspects des opérations militaires (d'où son nom). Depuis lors, la recherche opérationnelle est devenue un élément important du processus de prise de décision dans de nombreux contextes commerciaux, industriels et gouvernementaux, car elle permet d'appréhender de façon systématique la complexité toujours grandissante des problèmes de gestion auxquels sont confrontés les décideurs.

Située au carrefour de l'économie, des mathématiques et de l'informatique, la recherche opérationnelle s'appuie sur la représentation des problèmes concrets par des modèles mathématiques dont la résolution entraine presque toujours l'usage d'un ordinateur. Dans ce chapitre et dans le reste de ce mémoire, nous nous intéressons plus particulièrement à des problèmes pouvant s'exprimer sous la forme d'un modèle linéaire. Nous verrons ainsi leurs caractéristiques et les principales méthodes utilisées pour les résoudre. Parmi ces problèmes, nous en présenterons plus en détail deux classiques appelés plus court chemin et set packing. Enfin, nous présenterons le cas des problèmes à objectifs multiples.

1.1 Principes de base

Généralement, le problème de décision à traiter peut se ramener à un choix de valeurs sur un ensemble de variables de décision. Ce choix de valeurs devra alors être fait en fonction d'un objectif à optimiser et de contraintes à respecter. Lorsque l'objectif et les contraintes d'un problème peuvent être modélisés entièrement à l'aide de fonctions (équations ou inéquations)

linéaires affines des variables de décision, ce problème sera appelé programme linéaire. La plupart des éléments généraux présentés dans cette section ont été décrits dans de nombreux ouvrages (citons à titre d'exemple l'ouvrage de Gondran et Minoux [GM95] ou encore de Nemhauser et Wolsey [NW99]).

1.1.1 Qu'est-ce qu'un modèle linéaire ?

Un problème d'optimisation exprimé sous la forme d'un programme linéaire en variables continues (dans la suite nous utiliserons la notation LP pour Linear Program) peut s'écrire sous la forme générale suivante (équation 1.1) :

$$\left[\begin{array}{lll} Opt\ z_{\text{LP}}(X) = & \sum_{i \in I} c_i x_i & \\ sc & \sum_{i \in I} t_{i,j} x_i \, \Delta_j \, d_j & , \forall j \in J \\ & l_i \leq x_i \leq u_i & , \forall i \in I \\ & \Delta_j \in \{\leq ; = ; \geq\} & , \forall j \in J \end{array}\right] \tag{1.1}$$

avec :

- *Opt* pouvant signifier Minimiser ou Maximiser selon le problème traité,
- un ensemble de variables de décision $I = \{1, \ldots, n\}$ représenté par un vecteur $X = (x_i)$,
- un vecteur $C = (c_i)$ définissant les coefficients appliqués à chaque variable de décision dans la fonction objectif $z_{\text{LP}}(X)$,
- un ensemble de contraintes $J = \{1, \ldots, m\}$ représenté par une matrice $T = (t_{i,j})$,
- un ensemble (l_i) (resp. (u_i)) de bornes inférieures (resp. supérieures) sur les valeurs des variables de décision.

Notons que les contraintes peuvent aussi être exprimées sous forme matricielle. Dans ce cas $\sum_{i=1}^{n} t_{i,j} x_i \, \Delta_j \, d_j, \forall j \in J$ devient $TX \, \Delta \, d$. Dans la suite, les deux notations seront utilisées indifféremment.

Les variables utilisées dans ce problème prennent des valeurs réelles (quantité de produit ou temps par exemple). Le système de contraintes et les bornes définissant un domaine éventuellement vide de solutions admissibles $\mathcal{D}$[1] , l'objectif est d'obtenir la ou les solution(s) optimale(s) (quand il en existe) vis à vis de la fonction objectif.

Cependant, dans de nombreuses situations réelles, des variables peuvent être astreintes au domaine des valeurs discrètes (nombre de pièces à produire par exemple). Ces problèmes sont dits en variables mixtes (Mixed Integer Linear Problem). Lorsque toutes les variables du

[1] La notation $X \in \mathcal{D}$ signifie que le vecteur X représente une solution admissible vis-à-vis du système de contraintes et des bornes

problème sont entières, on parle d'un problème en variables entières (Integer Linear Program) qui se présente sous la forme suivante (1.2) :

$$\left[\begin{array}{lll} Opt\ z_{\text{ILP}}(X) = & \sum_{i=1}^{n} c_i x_i & \\ sc & \sum_{i=1}^{n} t_{i,j} x_i\, \Delta_j\, d_j & ,\forall j \\ & l_i \leq x_i \leq u_i & ,\forall i \\ & x_i \in \mathbb{Z} & ,\forall i \\ & \Delta_j \in \{\leq ; = ; \geq\} & ,\forall j \end{array}\right] \tag{1.2}$$

Lorsque les contraintes d'intégrité des variables d'un MILP ou d'un ILP sont relachées, on parle de relaxation linéaire du problème. Cette relaxation donne une borne supérieure (resp. inférieure) à la valeur de la fonction objectif pour les problèmes de maximisation (resp. minimisation).

Enfin, certains problèmes comportent uniquement des variables de type booléen (pour des décisions par exemple). La formulation algébrique d'un problème en variables binaires (0-1ILP) est la suivante (1.3) :

$$\left[\begin{array}{lll} Opt\ z_{\text{0-1ILP}}(X) = & \sum_{i=1}^{n} c_i x_i & \\ sc & \sum_{i=1}^{n} t_{i,j} x_i\, \Delta_j\, d_j & ,\forall j \\ & x_i = \{0\,;1\} & ,\forall i \\ & \Delta_j \in \{\leq ; = ; \geq\} & ,\forall j \end{array}\right] \tag{1.3}$$

Les problèmes n'utilisant que des variables booléennes sont nombreux (par exemple des problèmes d'affectation, de transport, de transbordement). Même si une variable binaire peut-être considérée comme une variable entière aux valeurs restreintes à l'intervalle discret $[0; 1]$, ces problèmes méritent une attention particulière en raison de leurs caractéristiques spécifiques et des méthodes de résolution qui leur sont appliquées.

1.1.2 Résolution exacte d'un modèle linéaire

Bien évidemment, une fois le problème modélisé, se pose la question de la détermination de sa (ou ses) solution(s) optimale(s) et donc de sa résolution. Nous parlerons de résolution exacte d'un problème lorsque l'algorithme utilisé permettra de trouver au moins une so-

lution optimale et de démontrer son optimalité. Les méthodes de résolution exactes sont nombreuses. Comme pour tout algorithme, une manière de comparer ces méthodes est de considérer leur complexité mathématique dans le pire des cas. Ainsi, on parlera d'un algorithme de complexité :

- polynomiale lorsque son temps de calcul est borné par un polynôme de la taille du problème (c-à-d que la complexité dans le pire des cas de l'algorithme est en $O(n^p m^q)$ avec p et q constants),
- exponentielle lorsque son temps de calcul ne peut pas être borné par un polynôme de la taille du problème (c-à-d que la complexité dans le pire des cas de l'algorithme est par exemple en $O(p^n)$ ou en $O(p^m)$ avec p constant).

Par extension, la théorie de la complexité (voir l'ouvrage de Garey et Johnson [GJ79] pour une description détaillée) s'intéresse à la complexité des modèles de programmation linéaire. Celle-ci se détermine en fonction de la complexité des algorithmes susceptibles de le résoudre exactement. Ces modèles peuvent ainsi être classés en deux catégories :

- ceux pour lesquels un algorithme de résolution exacte en temps polynomial est connu. On parlera alors de problèmes faciles.
- ceux pour lesquels on ne connait pas d'algorithme de résolution exacte en temps polynomial. On parlera alors de problèmes difficiles.

Parmi cette deuxième catégorie, les problèmes pour lesquels il ne peut pas exister d'algorithme de résolution exacte en temps polynomial, à moins que la conjecture $\mathcal{P} \neq \mathcal{NP}$ ne soit fausse, sont appelés NP difficiles.

1.1.2.1 Cas des problèmes en variables continues

Pour les problèmes en variables continues, le nombre de solutions admissibles est infini lorsque leur domaine n'est pas vide. Heureusement, des résultats d'algèbre linéaire ont montré que le domaine des solutions admissibles (ou espace de recherche) d'un programme linéaire en variables continues est soit un polytope convexe (cas d'un domaine non borné), soit un polyhèdre convexe (cas d'un domaine borné). De plus, si le domaine est borné et non vide, la solution optimale est un sommet du domaine (ou une face s'il y a plusieurs solutions optimales). Ainsi, l'algorithme du simplexe de G.B. Dantzig permet d'obtenir la solution optimale d'un problème en parcourant la fermeture convexe de l'espace de recherche et ce en passant de sommet en sommet. Malgré une complexité théorique dans le pire des cas exponentielle, il permet de résoudre la plupart des problèmes rapidement. D'autres algorithmes en temps polynomial ont été proposés pour résoudre ces problèmes. Certains peuvent ainsi se révéler très compétitifs sur certains types de problèmes. La méthode du simplexe reste malgré tout largement utilisée dans la pratique.

1.1.2.2 Cas des problèmes en variables discrètes

Pour les problèmes en variables discrètes, le nombre de solutions admissibles est fini lorsque leur domaine est borné. Pourtant, le nombre de ces solutions grandissant généralement de manière exponentielle (on parle d'explosion combinatoire), et la propriété de convexité du domaine des solutions n'étant en général plus valable, la résolution de ces problèmes n'est souvent pas aisée. Une exception existe lorsque la matrice des contraintes T est totalement unimodulaire (c-à-d que tous les déterminants de ses sous-matrices carrées soient égaux à -1, 0 ou 1). Tous les sommets du domaine des solutions admissibles de la relaxation linéaire du problème sont alors entiers. Le problème peut donc être résolu comme un problème en variables continues (et donc en temps polynomial, voir section 1.1.2.1). D'une manière générale, la résolution de ces problèmes n'est cependant pas aisée. Beaucoup de ces problèmes sont ainsi NP difficiles.

Dans le cas des problèmes en variables binaires, en plus des difficultés liées aux problèmes en variables entières ou mixtes, la principale difficulté provient du grand nombre de variables nécessaires pour modéliser des situations réelles. Cependant certains problèmes présentent des structures particulières. Ils font partie d'un domaine appelé Optimisation Combinatoire. Il est alors possible de tirer profit de l'information liée à cette structure pour faciliter leur résolution.

Les méthodes de résolution exacte pour les problèmes en variables discrètes ou mixtes sont nombreuses, cependant trois familles principales peuvent être distinguées :

- la programmation dynamique consiste à placer le problème dans une famille de problèmes de même nature mais de difficulté décroissante, puis à trouver une relation de récurrence liant les solutions optimales de ces problèmes,

- les méthodes dites par séparation et évaluation (Branch & Bound par exemple) consistent à faire une énumération implicite en séparant le problème en sous-problèmes (branchements) et en évaluant ceux-ci à l'aide d'une relaxation (continue ou lagrangienne principalement) jusqu'à ne plus avoir que des problèmes faciles à résoudre ou dont on sait avec certitude qu'ils ne peuvent pas contenir de solution optimale,

- les méthodes polyhédrales consistent à ajouter progressivement des contraintes supplémentaires (ou coupes) afin de ramener le domaine des solutions admissibles à un domaine convexe (sans en enlever la ou les solutions optimales bien évidemment). Ces coupes sont appelées inégalités valides lorsqu'elles ne suppriment aucune solution admissible du problème. Lorsqu'une inégalité valide fait partie de l'enveloppe convexe de la relaxation linéaire dont tous les sommets sont entiers, on parlera de facette.

De plus, certaines méthodes essayent de combiner les avantages de différentes familles. Ainsi par exemple, la méthode du Branch & Cut, dérivée du Branch & Bound, intégre une recherche de coupes dans les nœuds de l'arbre de recherche afin d'améliorer la valeur de la relaxation linéaire guidant le processus.

Enfin, une méthode issue de l'intelligence artificielle et baptisée programmation par contraintes (PPC) reprend un schéma de séparation, mais en utilisant des algorithmes dits de propagation de contraintes afin d'obtenir une solution réalisable. Cette méthode peut notamment être efficace pour les problèmes fortement contraints et lorsque l'obtention d'une solution admissible s'avère difficile.

1.1.3 Résolution approchée d'un modèle linéaire

Dans certaines situations, il est nécessaire de disposer d'une solution de bonne qualité (c'est-à-dire assez proche de l'optimum) dans un contexte de ressources (temps de calcul et/ou mémoire) limitées. Dans ce cas l'optimalité de la solution ne sera pas garantie, ni même l'écart avec la valeur optimale. Cependant, le temps nécessaire pour obtenir cette solution sera maitrisé, généralement de façon à être plus faible, et pourra même être fixé (bien évidemment dans ce cas la qualité de la solution obtenue dépendra fortement du temps laissé à l'algorithme pour l'obtenir). On parlera alors de résolution approchée ou d'heuristique[2]. Une définition communément acceptée de ce qu'est une heuristique a été proposée par Reeves [Ree95] (définition 1.1).

Définition 1.1 (Heuristique selon Reeves)
Une heuristique est une technique trouvant de bonnes solutions (c-à-d proches de l'optimum) pour un coût de calcul raisonnable, sans pouvoir garantir l'admissibilité ou l'optimalité, ou même dans de nombreux cas, préciser la distance à l'optimum d'une solution particulière.

Typiquement ce type de méthodes est particulièrement utile pour les problèmes nécessitant une solution en temps réel (ou très court) ou pour résoudre des problèmes difficiles sur des instances numériques de grande taille. Elles peuvent aussi être utilisées afin d'initialiser une méthode exacte (Branch & Bound par exemple).

À côté des méthodes heuristiques, sont apparues des méthodes qualifiées de métaheuristiques. Plusieurs définitions ont été proposées pour décrire leurs particularités par rapport aux heuristiques. Osman et Laporte [OL96] (voir la définition 1.2) comparent une métaheuristique à un processus maître pilotant une heuristique (parfois même plusieurs) à l'aide d'une stratégie permettant d'approcher la solution optimale de manière plus efficace.

[2] du grec heuriskein = trouver

Définition 1.2 (Métaheuristique selon Osman et Laporte)
Une métaheuristique est un processus itératif de génération guidant une heuristique subordonnée en combinant intelligemment différents concepts pour explorer et exploiter l'espace de recherche en utilisant des stratégies pour structurer l'information de manière à trouver efficacement des solutions proches de l'optimum.

Cependant, cette définition ne fait pas clairement apparaître l'indépendance des métaheuristiques actuelles vis-à-vis des problèmes à traiter. Nous préférerons la définition de Pirlot [Pir02] (définition 1.3) qui assimile les métaheuristiques à des stratégies de recherche. Ainsi, il faut distinguer les heuristiques ciblées sur un problème particulier et les métaheuristiques plus générales et adaptables pour résoudre un grand nombre de problèmes. Cependant, pour être suffisamment performante sur un problème donné, une métaheuristique nécessitera une adaptation intelligente aux caractéristiques de ce problème. Une même métaheuristique peut d'ailleurs être à l'origine d'heuristiques différentes pour un même problème.

Définition 1.3 (Métaheuristique selon Pirlot)
Les métaheuristiques ne sont pas à proprement parler des heuristiques, mais des schémas généraux, des «moules» à heuristiques ; le schéma général doit être adapté à chaque type particulier de problème.

Malgré la grande diversité de métaheuristiques (algorithmes génétiques, méthode de bruitage, méthode GRASP, recherche à voisinage variable, recherche dispersée, recherche tabou, recuit simulé, systèmes de fourmis, ...), leurs principes sont souvent assez proches. Taillard [Tai02] distingue trois éléments principaux constitutifs de toutes les métaheuristiques :
- la structure de voisinage permettant de modifier une solution,
- la mémoire des solutions déjà obtenues,
- la construction de solutions entièrement nouvelles.

Ainsi, chaque métaheuristique utilise au moins l'un de ces trois éléments. De plus, l'hybridation de différentes métaheuristiques est une pratique de plus en plus courante permettant d'obtenir des heuristiques performantes. Taillard [Tai98] propose d'ailleurs une unification de la plupart des métaheuristiques telles qu'elles sont implantées actuellement sous la dénomination de programmation à mémoire adaptative.

1.2 Un problème facile : le problème de plus court chemin

Maintenant que nous avons vu les concepts généraux liés aux modèles linéaires et à leur résolution, nous allons nous intéresser plus en détail à deux problèmes particuliers. Dans cette section, nous allons présenter rapidement les caractéristiques du problème dit de plus

court chemin dans un graphe. Ce problème a été abondamment décrit dans de nombreux ouvrages (par exemple celui de Faure [Fau91] ou de Gondran et Minoux [GM95]).

1.2.1 Définition du problème de plus court chemin

Ce problème, comme de nombreux problèmes de recherche opérationnelle, est basé sur la notion de graphe. Un graphe $G(V, E)$ est composé d'un ensemble V de sommets (ou nœuds), et d'un ensemble E d'arcs reliant des couples de sommets ($E \subseteq V \times V$). Les arcs d'un graphe peuvent être orientés ou pas. Lorsqu'ils sont orientés, un élément $e = (i, j) \in E$ représente un arc d'extrémité initiale i et d'extrémité terminale j. Un sommet j est alors appelé successeur (resp. prédécesseur) d'un sommet i si $(i, j) \in E$ (resp. $(j, i) \in E$). La partie de V correspondant à l'ensemble des successeurs (resp. prédécesseurs) d'un sommet i dans un graphe G est notée $\Gamma^+_G(i)$ (resp. $\Gamma^-_G(i)$).

L'objectif du problème de plus court chemin est de trouver un chemin (voir la définition 1.4) de valeur totale minimum d'un sommet du graphe à un autre (ou éventuellement d'un sommet à tous les autres voire même entre tous les couples de sommets).

Définition 1.4 (Chemin dans un graphe)
Soit un graphe $G(V, E)$ et une fonction $c : E \rightarrow I\!R$ de valuation des arcs de G, un chemin entre deux sommets i et j est une séquence d'arcs $\{e_1, e_2, \ldots, e_q\}$ telle que :

$$\begin{aligned} e_1 &= (i, k_1), \\ e_2 &= (k_1, k_2), \\ &\vdots \\ e_q &= (k_{q-1}, j). \end{aligned}$$

En outre, le problème du plus court chemin d'un sommet a à un sommet b peut aussi être formulé avec le modèle mathématique suivant (équation 1.4) :

$$\left[\begin{array}{lll} Min\ z(X) = \displaystyle\sum_{(i,j)\in E} c_{i,j}x_{i,j} & & \\ sc & \displaystyle\sum_{i\in\Gamma^-_G(a)} x_{i,a} - \sum_{j\in\Gamma^+_G(a)} x_{a,j} = -1 & \\ & \displaystyle\sum_{i\in\Gamma^-_G(j)} x_{i,j} - \sum_{i\in\Gamma^+_G(j)} x_{j,i} = 0 & ,\forall j \in V \setminus \{a, b\} \\ & \displaystyle\sum_{i\in\Gamma^-_G(b)} x_{i,b} - \sum_{j\in\Gamma^+_G(b)} x_{b,j} = 1 & \\ & x_{i,j} \in \{0, 1\} & ,\forall (i, j) \in E \end{array}\right] \tag{1.4}$$

avec :

- une matrice $X = (x_{i,j})$ telle que $x_{i,j} = \begin{cases} 1 \text{ si l'arc } (i,j) \text{ fait parti du chemin} \\ 0 \text{ sinon} \end{cases}$
- une matrice $C = (c_{i,j})$ telle que $c_{i,j} =$ valuation de l'arc (i,j)

Les applications pratiques de ce problème en tant que problème principal ou sous-problème sont très nombreuses, la valuation l pouvant représenter des grandeurs physiques diverses (longueur, coût de transport, temps de trajet, ...).

1.2.2 Résolution du problème de plus court chemin

Le problème de plus court chemin est considéré comme facile car des algorithmes permettant de le résoudre de manière exacte en temps polynomial sont connus. Ces algorithmes se différencient par les propriétés des graphes étudiés. Ainsi par exemple, certains algorithmes (celui de Moore-Dijkstra, dont la complexité dans le pire des cas est en $O(n^2)$, est le plus connu) ne traitent que les graphes pour lesquels toutes les valuations des arcs sont positives ou nulles ($c_{i,j} \geq 0, \forall (i,j) \in E$), voire même où toutes les valuations sont unitaires ($c_{i,j} = 1, \forall (i,j) \in E$). D'autres algorithmes (comme ceux proposés par Berge, Moore, Bellman ou Ford) permettent de traiter les graphes ne vérifiant pas cette propriété.

Il est intéressant de noter que tous ces algorithmes calculent en fait, non pas le plus court chemin d'un sommet d'un graphe à un autre, mais le plus court chemin d'un sommet du graphe à tous les autres sommets. De plus, d'autres algorithmes (comme ceux proposés par Floyd ou Dantzig) calculent le plus court chemin entre tous les couples de sommets d'un graphe (on parlera alors d'algorithmes matriciels puisque leur résultat est une matrice de plus courts chemins). Bien évidemment, ce calcul peut aussi être effectué en utilisant les précédents algorithmes successivement pour chaque sommet du graphe.

Pour finir, même si nous avons vu que le problème de plus court chemin pouvait se modéliser sous la forme d'un 0-1ILP, les méthodes de résolution les plus usuelles (dont celles que nous avons citées) n'utilisent pas cette modélisation. Il est malgré tout possible de le résoudre en temps polynomial à l'aide de ce modèle. En effet, la matrice des contraintes de ce problème vérifie la propriété de totale unimodularité (voir section 1.1.2.2). Celui-ci peut donc être résolu comme un problème en variables continues.

1.3 Un problème difficile : le problème de set packing

Nous allons maintenant détailler les caractéristiques d'un autre problème particulier appelé problème de set packing. Là encore, ce problème a été décrit dans de nombreux ouvrages (par exemple celui de Gondran et Minoux [GM95] ou de Nemhauser et Wolsey [NW99]).

1.3.1 Définition du problème de set packing

Soit un ensemble fini d'éléments valués $I = \{1, \ldots, n\}$ et $\{T_j\}, j \in J = \{1, \ldots, m\}$ une collection de sous-ensembles de I, un couplage est un sous-ensemble $P \subseteq I$ d'éléments disjoints (c-à-d tel que $\nexists (p_1, p_2) \in P^2, \exists j \in J, (p_1, p_2) \in T_j^2$). L'ensemble J peut aussi être vu comme un ensemble de contraintes entre les éléments de l'ensemble I. L'objectif du problème de set packing est de maximiser la valeur totale du couplage obtenu. Le nom de problème de couplage généralisé est ainsi parfois utilisé. Ce problème peut être formulé avec le modèle mathématique suivant (équation 1.5) :

$$\left[\begin{array}{ll} Max\ z(X) = & \sum_{i \in I} c_i x_i \\ sc & \sum_{i \in I} t_{i,j} x_i \leq 1, \forall j \in J \\ & x_i \in \{0, 1\} \quad , \forall i \in I \end{array}\right] \tag{1.5}$$

avec :

- un vecteur $X = (x_i)$ tel que $x_i = \begin{cases} 1 \text{ si } i \in P \\ 0 \text{ sinon} \end{cases}$
- un vecteur $C = (c_i)$ tel que $c_i =$ valeur de l'élément i
- une matrice $T = (t_{i,j})$ telle que $t_{i,j} = \begin{cases} 1 \text{ si } i \in T_j \\ 0 \text{ sinon} \end{cases}$

De manière surprenante pour un problème aussi classique et étudié, très peu d'applications pratiques modélisées par un problème de set packing sont présentées dans la littérature. En fait, si des inégalités de type set packing apparaissent dans de nombreux problèmes, elles ne permettent que rarement de représenter un problème à elles seules. Nous pouvons malgré tout citer six applications pratiques récentes (indiquées par ordre chronologique) :

- Rönnqvist [Rön95] a travaillé sur un problème de cutting stock qu'il a formulé comme un problème de set packing et résolu en utilisant une relaxation lagrangienne combinée à une optimisation de sous-gradient,
- Zwaneveld et al. [ZKR+96, Zwa97, ZKVH01] ont formulé un problème de faisabilité de passage de trains dans un nœud ferroviaire en problème de set packing et l'ont résolu de manière exacte à l'aide de tests de réduction et d'un algorithme de Branch & Cut,
- Kim et Lee [KL97] ont représenté un problème d'ordonnancement de bateaux par un set packing et ont utilisé le logiciel LINDO pour le résoudre,
- Mingozzi et al. [MMRB98] ont utilisé une formulation de type set packing afin de calculer des bornes pour un problème d'ordonnancement de projet avec contraintes de ressource (RCPSP) à l'aide d'un algorithme glouton,
- Rossi et Smriglio [RS01] ont considéré une formulation de type set packing pour un

problème de «ground holding» et l'ont résolu exactement avec un algorithme de Branch & Cut,
- Ladanyi et al. [LJE03] ont présenté très récemment un travail sur un problème lexicographique (voir section 1.4) de set packing qu'ils ont résolu exactement avec un algorithme de Branch & Cut & Price.

1.3.2 Similarités avec d'autres problèmes en variables binaires

Le problème de set packing présente des similarités avec de nombreux autres problèmes en variables binaires classiques. Nous allons nous intéresser dans cette section aux problèmes suivants :
- node packing
- set covering
- set partitioning
- clique maximum

Cependant, cette liste n'est pas exhaustive et d'autres problèmes dont nous ne parlerons pas ici présentent aussi des similarités avec le problème de set packing. Celui-ci peut, par exemple, être vu comme un cas particulier du problème de sac-à-dos multi-dimensionnels en variables binaires (voir Fréville [Fré03] pour un état de l'art). En pratique cependant, cette propriété est rarement utilisée pour une résolution en raison notamment du nombre de contraintes beaucoup plus grand dans les instances du problème de set packing.

1.3.2.1 Problème de node packing ou maximum independant set

Soit un graphe $G(V, E)$ et une fonction $c : E \rightarrow I\!R$ de valuation des arcs de G, le problème de node packing consiste à trouver un sous-ensemble de sommets indépendants $P \subseteq V$ (c-à-d non reliés par un arc) de poids maximum. Ce problème peut être formulé avec le modèle mathématique suivant (équation 1.6) :

$$\left[\begin{array}{ll} Max\ z(X) = \sum_{i \in I} c_i x_i & \\ sc & x_i + x_j \leq 1, \forall (i,j) \in E \\ & x_i \in \{0,1\}\ , \forall i \in V \end{array}\right] \tag{1.6}$$

avec :
- un vecteur $X = (x_i)$ tel que $x_i = \begin{cases} 1 \text{ si } i \in P \\ 0 \text{ sinon} \end{cases}$
- un vecteur $C = (c_i)$ tel que $c_i =$ valeur du sommet i

Le problème de node packing est en fait un cas particulier du problème de set packing

dans lequel les sous-ensembles de T_j sont tous de cardinalité 2 (c-à-d que les contraintes concernent uniquement des paires d'éléments, $\sum_{i \in I} t_{i,j} = 2, \forall j \in J$).

Tous les problèmes de set packing peuvent être formulés en problèmes de node packing. Pour cela, il suffit de substituer chaque contrainte par les contraintes entre toutes les paires d'éléments apparaissant dans la contrainte initiale. Cependant, cette reformulation augmente de manière importante le nombre de contraintes ; chaque contrainte concernant n éléments est remplacée par C_n^2 contraintes entre deux éléments. De plus, la relaxation linéaire du problème ainsi obtenue est beaucoup plus faible que celle du problème initial.

1.3.2.2 Problème de set covering

Soit un ensemble fini d'éléments valués $I = \{1, \ldots, n\}$ et $\{T_j\}, j \in J = \{1, \ldots, m\}$ une collection de sous-ensembles de I, une couverture est un sous-ensemble $P \subseteq I$ couvrant tous les sous-ensembles T_j (c-à-d telle que $|T_j \cap P| \geq 1, \forall j \in J$). L'objectif du problème de set covering est de minimiser le coût total de la couverture obtenue. La formulation mathématique de ce problème (équation 1.7) est très proche de celle du problème de set packing puisque seuls la fonction objectif et le sens des inégalités sont changés.

$$\left[\begin{array}{ll} Min\ z(X) = & \sum_{i \in I} c_i x_i \\ sc & \sum_{i \in I} t_{i,j} x_i \geq 1, \forall j \in J \\ & x_i \in \{0,1\} \quad , \forall i \in I \end{array}\right] \tag{1.7}$$

avec :

- un vecteur $X = (x_i)$ tel que $x_i = \begin{cases} 1 \text{ si } i \in P \\ 0 \text{ sinon} \end{cases}$
- un vecteur $C = (c_i)$ tel que $c_i =$ coût de l'élément i
- une matrice $T = (t_{i,j})$ telle que $t_{i,j} = \begin{cases} 1 \text{ si } i \in T_j \\ 0 \text{ sinon} \end{cases}$

Il est aussi intéressant de noter que tous les problèmes de node packing peuvent être exprimés en tant que problème de set covering grâce au changement de variable $x'_i = 1 - x_i, \forall i \in I$. L'opération inverse n'est pas possible dans le cas général.

1.3.2.3 Problème de partitionnement d'ensembles

Soit un ensemble fini d'éléments valués $I = \{1, \ldots, n\}$ et $\{T_j\}, j \in J = \{1, \ldots, m\}$ une collection de sous-ensembles de I, une partition est un sous-ensemble $P \subseteq I$ étant à la fois un couplage et une couverture (c-à-d telle que $|T_j \cap P| = 1, \forall j \in J$). L'objectif du problème de set partitioning est de minimiser le coût total de la partition obtenue. Là encore, la formulation

mathématique de ce problème (équation 1.8) est très proche de celles des problèmes de set packing et de set covering.

$$\left[\begin{array}{lll} Min\ z(X) = & \sum_{i \in I} c_i x_i & \\ sc & \sum_{i \in I} t_{i,j} x_i = 1, & \forall j \in J \\ & x_i \in \{0,1\} & ,\forall i \in I \end{array}\right] \tag{1.8}$$

avec :

- un vecteur $X = (x_i)$ tel que $x_i = \begin{cases} 1 \text{ si } i \in P \\ 0 \text{ sinon} \end{cases}$
- un vecteur $C = (c_i)$ tel que $c_i =$ coût de l'élément i
- une matrice $T = (t_{i,j})$ telle que $t_{i,j} = \begin{cases} 1 \text{ si } i \in T_j \\ 0 \text{ sinon} \end{cases}$

Là encore, il est intéressant de noter que tout problème de set packing peut être exprimé comme un problème de set partitioning en ajoutant des variables d'écarts x_j^e de valeur nulle dans chacune des contraintes (équation 1.9).

$$\left[\begin{array}{lll} Max\ z(X) = & \sum_{i \in I} c_i x_i & + \sum_{j \in J} 0\ x_j^e \\ sc & \sum_{i \in I} t_{i,j} x_i & + x_j^e = 1 \quad ,\forall j \in J \\ & x_i \in \{0,1\} & ,\forall i \in I \\ & x_j^e \in \{0,1\} & ,\forall j \in J \end{array}\right] \tag{1.9}$$

De même, tout problème de set partitioning peut être transformé en un problème de set packing ayant la même solution optimale. Pour cela, il faut d'abord ajouter des variables artificielles x_j^a de coût prohibitif (servant de pénalité de type big-M) dans chacune des contraintes (équation 1.10).

$$\left[\begin{array}{lll} Min\ z(X) = & \sum_{i \in I} c_i x_i & + \sum_{j \in J} \theta x_j^a \\ sc & \sum_{i \in I} t_{i,j} x_i & + x_j^a = 1 \ ,\forall j \in J \\ & x_i \in \{0,1\} & ,\forall i \in I \\ & x_j^a \in \{0,1\} & ,\forall j \in J \end{array}\right] \tag{1.10}$$

On peut ensuite remplacer les variables artificielles x_j^a dans la fonction objectif :

$$\begin{aligned}\sum_{i\in I} c_i x_i + \sum_{j\in J} \theta x_j^a &= \sum_{i\in I} c_i x_i + \theta \sum_{j\in J}(1 - \sum_{i\in I} t_{i,j} x_i) \\ &= \sum_{i\in I} c_i x_i + \theta |J| - \theta \sum_{j\in J}\sum_{i\in I} t_{i,j} x_i \\ &= \sum_{i\in I}(c_i - \theta \sum_{j\in J} t_{i,j}) x_i + \theta |J|\end{aligned}$$

Enfin, le fait de supprimer les variables artificielles x_j^a (équation 1.11) des contraintes correspond à une relaxation du problème et augmente le domaine des solutions admissibles. Mais cela ne change pas l'ensemble des solutions optimales grâce aux pénalités qu'il y a à ne pas respecter une égalité.

$$\left[\begin{array}{lll} Min\ z(X) = & \sum_{i\in I}(c_i - \theta \sum_{j\in J} t_{i,j}) x_i + \theta |J| & \\ sc & \sum_{i\in I} t_{i,j} x_i \leq 1 & ,\forall j \in J \\ & x_i \in \{0,1\} & ,\forall i \in I \end{array}\right] \tag{1.11}$$

1.3.2.4 Problème de clique maximum

Ce problème est basé sur la notion, courante dans les graphes, de clique (voir la définition 1.5).

Définition 1.5 (Clique dans un graphe)
Soit un graphe $G(V, E)$, une clique $\mathcal{C}$ est un sous-ensemble de sommets qui sont tous reliés deux à deux par un arc :

$$\mathcal{C} \subseteq V, \forall (i,j) \in \mathcal{C}^2, (i,j) \in E$$

Soit un graphe $G(V, E)$ et une fonction $c : E \rightarrow I\!R$ de valuation des arcs de G. L'objectif du problème de clique maximum consiste à trouver une clique de poids maximum, le poids d'une clique étant la somme des poids des sommets la composant.

Ce problème peut être formulé avec le modèle mathématique suivant (équation 1.12) :

$$\left[\begin{array}{lll} Max\ z(X) = & \sum_{i\in I} c_i x_i & \\ sc & x_i + x_j \leq 1, \forall (i,j) \notin E & \\ & x_i \in \{0,1\}\ ,\forall i \in I & \end{array}\right] \tag{1.12}$$

avec :

- un vecteur $X = (x_i)$ tel que $x_i = \begin{cases} 1 \text{ si } i \in \mathcal{C} \\ 0 \text{ sinon} \end{cases}$
- un vecteur $C = (c_i)$ tel que $c_i =$ valeur du sommet i

Le problème de clique maximum est en fait équivalent au problème de node packing au sens où trouver une clique maximum dans un graphe $G(V, E)$ revient à trouver un ensemble indépendant maximum dans le graphe complémentaire $G(V, \bar{E})$.

1.3.3 Résolution du problème de set packing

1.3.3.1 Résolution exacte

Le problème de set packing est connu comme NP-difficile au sens fort d'après Garey et Johnson [GJ79]. Ainsi, nous ne connaissons pas d'algorithme polynomial pour le résoudre dans le cas général. Il existe cependant des cas particuliers. Ainsi, si la matrice des contraintes T est totalement unimodulaire, il peut être résolu en temps polynomial. Par ailleurs, lorsque la matrice des contraintes T contient au plus deux 1 par colonne et que toutes les valuations sont unitaires ($\sum_{j \in J} t_{i,j} \leq 2, \forall i \in I$ et $c_i = 1, \forall i \in I$, ce qui revient au calcul d'un ensemble d'arrêtes indépendantes de cardinalité maximum), le problème peut être résolu en temps polynomial d'après Garey et Johnson [GJ79]. Sakarovitch [Sak84], par exemple, présente une méthode de marquage pour résoudre ce problème.

De plus, de nombreux travaux ont été réalisés sur sa structure particulière. Ainsi, les travaux de Padberg sur les cliques et les cycles de longueur impaire [Pad73] ont initié de nombreuses recherches liées à l'étude polyhédrale du problème de set packing. Borndörfer [Bor98] recence un grand nombre de familles de facettes et d'inégalités valides spécifiques au problème de set packing. En pratique pourtant, la détermination de l'ensemble de l'enveloppe convexe est impossible à part pour des problèmes de petite taille. De ce fait, la meilleure méthode connue actuellement pour résoudre exactement le problème de set packing est le Branch & Cut. Deux implémentations de cette méthode ont été récemment proposées par Borndörfer [Bor98] et Esö [Esö99] pour résoudre un problème de set partitioning en recherchant des familles de coupes spécifiques au problème de set packing.

Enfin, Nemhauser et Wolsey [NW99] indiquent une propriété très intéressante du problème de set packing et plus particulièrement de sa formulation en problème de node packing. En effet, nous avons vu dans la section 1.3.2.1 que tout problème de set packing peut être formulé en un problème de node packing. Or lorsque la solution optimale de la relaxation linéaire d'un problème de node packing contient des variables dont les valeurs sont entières, ces valeurs font obligatoirement parties d'une solution optimale entière. Le lecteur intéressé pourra trouver la démonstration de cette propriété dans l'ouvrage pré-cité. Cette dernière peut se révéler très intéressante lorsque le nombre de variables entières dans la solution opti-

male de la relaxation linéaire est important. Cependant, cet intérêt est limité par le fait que la relaxation linéaire d'un problème de set packing est souvent beaucoup plus forte que celle de sa formulation en node packing. En effet, cette propriété ne s'applique pas au problème de set packing.

1.3.3.2 Résolution approchée

Contrairement à la plupart des autres problèmes cités ci-dessus, à notre connaissance, et selon l'état de l'art d'Osman et Laporte [OL96], personne n'a pour le moment développé de métaheuristique pour la résolution du problème de set packing. Ceci est certainement dû au très faible nombre d'applications pratiques de grande taille rapportées dans la littérature pour ce problème.

1.4 Optimisation de problèmes multiobjectifs

Dans les sections précédentes, nous n'avons considéré que les cas où le problème à traiter possédait un objectif unique à optimiser. Pour de nombreuses applications pratiques, ce n'est en fait pas le cas et il faut optimiser en même temps plusieurs objectifs pour un même problème. La fonction objectif n'est alors plus scalaire mais vectorielle. On parle de problème multiobjectif. La plupart des éléments généraux présentés dans cette section ont été décrits dans de nombreux ouvrages (par exemple ceux de Steuer [Ste86] ou d'Ehrgott et Gandibleux [EG02b]).

1.4.1 Qu'est-ce qu'un problème multiobjectif ?

Un programme linéaire multiobjectif en variables continues (MOLP) peut s'écrire sous la forme générale suivante (équation 1.13) :

$$\left[\begin{array}{lll} Opt^q\ z^q_{\text{MOLP}}(X) = & \sum_{i\in I} c^q_i x_i & ,\forall q \in Q \\ sc & \sum_{i\in I} t_{i,j} x_i\, \Delta_j\, d_j & ,\forall j \in J \\ & l_i \leq x_i \leq u_i & ,\forall i \in I \\ & \Delta_j \in \{\leq ; = ; \geq\} & ,\forall j \in J \end{array}\right] \tag{1.13}$$

avec :

- un ensemble de fonctions objectifs $Q = \{1, \ldots, p\}$
- Opt^q pouvant signifier Minimiser ou Maximiser selon l'objectif considéré,
- un ensemble de variables de décision $I = \{1, \ldots, n\}$ représenté par un vecteur $X = (x_i)$,

- une matrice $C = (c_i^q)$ définissant les coefficients appliqués à chaque variable de décision dans chaque fonction objectif $z^q(X)$,
- un ensemble de contraintes $J = \{1, \ldots, m\}$ représenté par une matrice $T = (t_{i,j})$,
- un ensemble (l_i) (resp. (u_i)) de bornes inférieures (resp. supérieures) sur les valeurs des variables de décision.

Toute solution X définie dans l'espace de décision ($I\!R^n$) peut être projetée en un point de l'espace des objectifs ($I\!R^p$). Bien évidemment, deux solutions différentes dans l'espace de décision peuvent correspondre à un même point dans l'espace des objectifs. Ces solutions seront appelées équivalentes.

La difficulté de ces problèmes vient du fait que, contrairement aux problèmes mono-objectifs, il n'existe plus de relation d'ordre entre toutes les solutions admissibles et donc plus de notion d'optimalité (à moins qu'une solution soit optimale pour toutes les fonctions objectifs, ce qui est rarement le cas). Dans le cas des problèmes multiobjectifs, elles sont remplacées par les notions de dominance et de Pareto optimalité (voir respectivement les définitions 1.6 et 1.7) :

Définition 1.6 (Dominance entre deux solutions)
Une solution A domine une solution B pour un problème de maximisation (resp. minimisation) si :

$$z^q(A) \geq z^q(B) \text{ (resp. } z^q(A) \leq z^q(B) \text{)}, \forall q \in Q$$

$$\text{et } \exists q' \in Q, z^{q'}(A) > z^{q'}(B) \text{ (resp. } z^{q'}(A) < z^{q'}(B) \text{)}$$

Dans la suite, la notation $A \succ B$ signifiera que la solution A domine la solution B. De même, la notation $A \succeq B$ signifiera que la solution A domine ou est équivalente (c-à-d $z^q(A) = z^q(B), \forall q \in Q$) à la solution B.

Définition 1.7 (Pareto optimalité)
$X \in \mathcal{D}$ est un optimum de Pareto strict si $\nexists X' \in \mathcal{D}, X' \succ X$

Une solution Pareto optimale est aussi appelée solution non-dominée. Nous ne parlerons donc plus de solution optimale, mais d'un ensemble de solutions Pareto optimales (aussi appelées solutions efficaces). L'ensemble des solutions efficaces est noté E. La projection dans l'espace des objectifs de cet ensemble E décrit une frontière communément appelée frontière efficace. En outre, deux points caractéristiques ne correspondant généralement à aucune solution admissible sont aussi souvent utilisés :

- le point Idéal ($\bar{z}^q = \max_{X \in E} z^q(X)$ (resp. $\min_{X \in E} z^q(X)$), $\forall q \in Q$ pour un problème de maximisation (resp. minimisation)),
- et le point Nadir ($\eta^q = \min_{X \in E} z^q(X)$ (resp. $\max_{X \in E} z^q(X)$), $\forall q \in Q$ pour un problème de maximisation (resp. minimisation)).

De la même manière, les problèmes multiobjectifs peuvent être en variables mixtes (MOMILP), entières (MOILP), binaires (0-1MOILP) ou encore relever du domaine de l'Optimisation Combinatoire (MOCO, Ehrgott et Gandibleux [EG02a] proposent une bibliographie annotée spécifique à ces problèmes). Dans ce cas les solutions efficaces regroupent des solutions dites supportées (correspondants aux sommets de la fermeture convexe de la frontière efficace et notées SE), et des solutions non-supportées (n'appartenant pas à cette fermeture convexe et notées NE). La figure 1.1 illustre ces différents points caractéristiques dans l'espace des objectifs sur un exemple de problème de maximisation biobjectif.

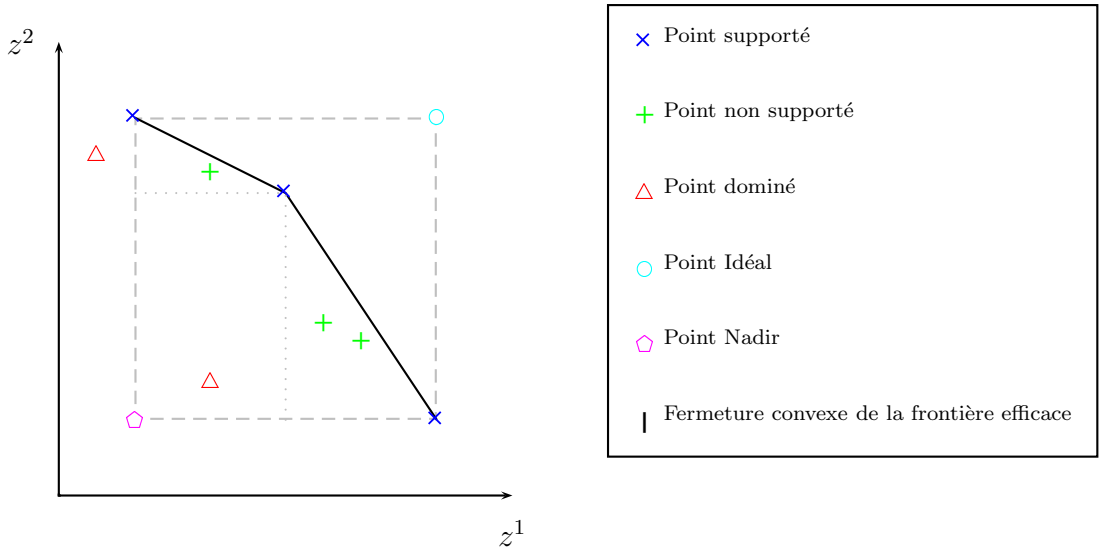


FIG. 1.1 – Points caractéristiques d'un problème de maximisation biobjectif

1.4.2 Résolution d'un problème multiobjectif

De nombreuses méthodes de résolution existent pour les problèmes multiobjectifs. Elles correspondent à des situations et des problématiques différentes. Toutes ces méthodes peuvent être classées en trois grandes catégories suivant le but recherché :

- celles visant à obtenir une (ou plusieurs) solution(s) représentant un bon compromis entre les différents objectifs,
- celles visant à déterminer l'ensemble de la frontière efficace,
- et celles visant à calculer une bonne approximation de cette frontière.

Notons qu'une autre classification, basée sur le type d'interactivité avec le décideur (a priori, interactive ou a posteriori), est parfois utilisée.

1.4.2.1 Recherche d'un compromis

La première catégorie regroupe des méthodes ramenant la résolution d'un problème multiobjectif à la résolution d'un (ou de plusieurs) problème(s) mono-objectif(s). Leur but est de trouver une (ou plusieurs) solution(s) qui correspondra(ont) à un bon compromis entre les différents objectifs en fonction des préférences exprimées par le décideur à l'aide des paramètres. Ces méthodes peuvent être utilisées soit avec des paramètres fixés a priori, soit de manière interactive en modifiant les paramètres durant la recherche. De nombreuses méthodes peuvent entrer dans cette catégorie, nous en présentons ci-dessous quatre couramment utilisées :

- Aggrégation des objectifs : une première méthode est d'aggréger les différents objectifs en un seul. Pour cela il faut que tous les objectifs représentent des grandeurs d'unités comparables. De plus, il est nécessaire de déterminer des poids respectifs pour chacun de ces objectifs (notés $\lambda_q, \forall q \in Q$). Il est ainsi possible de se ramener à un problème mono-objectif en faisant la somme pondérée des différents objectifs :

  $$z_{\text{Somme pondérée}} = \sum_{q \in Q} \lambda_q \sum_{i \in I} c_i^q x_i$$

  Le problème mono-objectif obtenu conserve la structure de la matrice des contraintes (ce qui peut être intéressant lorsque cette structure correspond à un problème caractéristique facile à résoudre). La solution obtenue par une résolution exacte de ce problème sera l'une des solutions de E (et même une des solutions de SE).

- Résolution lexicographique : une deuxième méthode consiste à considérer un ordre de priorité (dit lexicographique) entre chacun des objectifs. Le problème sera alors formulé avec l'objectif suivant :

  $$lex\ Opt(z^1, z^2, \ldots, z^p)$$

  Dans ce cas, la résolution pourra s'effectuer en résolvant le problème successivement sur chacun des objectifs pris par ordre de priorité décroissante. Les valeurs obtenues sur un objectif sont ensuite intégrées comme contraintes pour la résolution sur des objectifs moins prioritaires (cet ajout de contraintes peut casser la structure de la matrice des contraintes). La solution obtenue par une résolution exacte successive de ces problèmes mono-objectifs sera l'une des solutions extrémales de E (et donc une des solutions de SE).

- Goal programming : une troisième méthode revient à s'approcher au maximum de valeurs cibles (aussi appelées niveaux d'aspiration et notées $\widehat{z^q}, \forall q \in Q$) sur chacun

des objectifs. Pour cela, des poids sont affectés à chaque objectif (notés $\lambda_q, \forall q \in Q$), et la somme pondérée des écarts en excès (notés d_q^+) comme en défaut (notés d_q^-) par rapport aux valeurs cibles doit être minimisée. Le problème sera alors formulé de la manière suivante (équation 1.14) :

$$\left[\begin{array}{lll} Min\ z_{\text{Goal Programming}} = & \sum_{q \in Q} \lambda^q (d_q^+ + d_q^-) & \\ sc & X \in \mathcal{D} & \\ & \sum_{i \in I} c_i^q\ x_i + d_q^- - d_q^+ = \widehat{z^q} & ,\forall q \in Q \\ & d_q^+, d_q^- \geq 0 & ,\forall q \in Q \end{array}\right] \tag{1.14}$$

La structure de la matrice des contraintes est alors cassée. La solution obtenue par une résolution exacte de ce problème mono-objectif ne sera pas obligatoirement l'une des solutions de E.

– Max-ordering : une quatrième méthode correspond au cas où il faut optimiser non pas tous les objectifs, mais uniquement le moins bon. Dans ce cas, le problème reviendra à optimiser l'objectif suivant pour un problème de maximisation (resp. minimisation) :

$$Max\ z_{\text{Max-Ordering}} = \min_{q \in Q} z^q \text{ (resp. } Min\ z_{\text{Max-Ordering}} = \max_{q \in Q} z^q)$$

La structure de la matrice des contraintes est alors cassée. La solution obtenue par une résolution exacte de ce problème mono-objectif ne sera pas obligatoirement l'une des solutions de E.

1.4.2.2 Détermination de la frontière efficace

La deuxième catégorie rassemble les méthodes déterminant l'ensemble complet des solutions Pareto optimales du problème, ou plus souvent seulement l'ensemble minimum complet (nom donné par Hansen [Han79] au sous-ensemble des solutions efficaces ne contenant pas les solutions équivalentes). Ces méthodes permettent au décideur de sélectionner a posteriori la (ou les) solution(s) qui l'intéresse parmi l'ensemble des solutions efficaces. Différentes méthodes peuvent être utilisées, nous présentons ci-dessous quatre méthodes courantes :

– la méthode en deux phases est un principe de résolution général applicable aux problèmes biobjectifs. Elle consiste à déterminer dans un premier temps l'ensemble SE des solutions supportées, puis à rechercher l'ensemble des solutions non supportées (ensemble NE) localisées dans les triangles définis par deux points supportés adjacents.

- la méthode du ranking est, elle aussi, applicable uniquement aux problèmes biobjectifs. Elle consiste à rechercher l'ensemble des solutions situées entre le point Nadir et le point Idéal. Ceux-ci sont obtenus en calculant les points extrèmes de la frontière efficace (par deux résolutions lexicographiques successivement sur chaque objectif). Ensuite, en partant de l'une des solutions extrèmes, la deuxième meilleure est cherchée, puis la troisième, ... puis la k^{ième} jusqu'à atteindre la valeur du point Nadir.

- la recherche dichotomique est, là encore, seulement applicable aux problèmes biobjectifs. Elle consiste à effectuer une optimisation dans un sous-domaine de recherche borné par deux solutions non dominées de références (initialement les deux points extrèmes) pour trouver une autre solution efficace. Le sous-domaine est ainsi divisé en deux nouveaux sous-domaines jusqu'à ce que ceux-ci ne contiennent plus de solution efficace.

- la méthode ϵ-contrainte est valable quelque soit le nombre d'objectifs. Elle consiste à optimiser le problème sur l'un des objectifs en bornant tous les autres objectifs, ce qui revient pour l'objectif $q \in Q$ d'un problème de maximisation à résoudre le problème suivant (équation 1.15) :

$$\left[\begin{array}{lll} Max & z^q(X) & \\ sc & X \in \mathcal{D} & \\ & z^{q'}(X) \geq \epsilon^q_{q'} & ,\forall q' \in Q \setminus \{q\} \end{array}\right] \tag{1.15}$$

Toutes ces méthodes font appel à de nombreuses résolutions exactes de problèmes monoobjectifs (dont certains pour lesquels la structure de la matrice des contraintes est cassée) et peuvent donc demander beaucoup de temps.

Une autre méthode parfois utilisée consiste à faire une recherche paramétrique sur la somme pondérée des objectifs (la structure de la matrice des contraintes est donc conservée). Cependant, cette technique présente l'inconvénient de ne permettre de trouver que les solutions de SE ; cette méthode n'est donc valide que pour les problèmes pour dont toutes les solutions efficaces sont supportées ($E = SE$). Pour les autres problèmes, cette méthode peut cependant être utilisée pour obtenir un sous-ensemble de la frontière efficace.

1.4.2.3 Approximation de la frontière efficace

Une dernière catégorie réunit les méthodes ne cherchant qu'une approximation de la frontière efficace. Celles-ci peuvent être utilisées soit dans le cadre d'une méthode interactive, soit pour effectuer une sélection a posteriori. Ces méthodes sont particulièrement utiles pour les problèmes de grande taille ou avec de nombreux objectifs dans un contexte de ressources (temps de calcul et/ou mémoire) limitées. La plupart des méthodes heuristiques utilisées pour les problèmes multiobjectifs sont dérivées d'heuristiques ou de métaheuristiques pour des problèmes mono-objectifs. Elles permettent de générer un ensemble PE de solutions dites potentiellement efficaces. Une solution est potentiellement efficace si elle n'est dominée par aucune autre solution de PE ($A \in PE$ potentiellement efficace si $\nexists B \in PE, B \succ A$).

Les solutions ainsi générées peuvent appartenir à E, mais rien ne permet de le garantir. Le problème de l'évaluation d'une approximation se pose alors. Pour répondre à ce problème, plusieurs solutions sont possibles :

- utiliser des ensembles de bornes (supérieures et inférieures) comme proposé par Ehrgott et Gandibleux [EG01] lorsque ces dernières sont de bonne qualité ;
- détecter une relation de dominance entre deux approximations mais cela est rarement possible ;
- calculer une (ou plusieurs) métrique(s) pour mesurer la qualité d'un ensemble de solutions, mais le choix de la métrique peut influencer l'évaluation. Certaines de ces mesures nécessitent la connaissance d'une solution ou d'un ensemble de solutions de référence (par exemple l'ensemble E, un ensemble PE connu, le point idéal, ...). De nombreuses mesures ont ainsi été proposées. Hansen et Jaszkiewicz [HJ98, Jas01, Jas04] proposent un cadre général pour ces mesures.

1.5 Conclusion

Dans ce chapitre, nous avons présenté l'ensemble des notions de recherche opérationnelle nécessaires à la compréhension du travail rapporté dans ce mémoire. Nous nous sommes ainsi plus particulièrement intéressés à la modélisation linéaire des problèmes. De même, les principales méthodes utilisées pour résoudre ces problèmes, de manière exacte ou approchée, ont été indiquées. En particulier les différences, en terme de difficulté de résolution et de technique employée, découlant du type de variables, du nombre d'objectifs ou encore de la structure des problèmes, ont été mises en évidence.

En outre, nous avons décrit plus en détail deux problèmes classiques appelés plus court chemin et set packing. Ainsi, si la résolution exacte du premier est facile, le second appartient à la classe des problèmes NP difficiles. De plus, si de nombreux travaux ont exploré ses propriétés dans le cadre d'une résolution exacte, sa résolution approchée n'a pour le moment quasiment pas été étudiée.

Dans la suite, nous allons nous intéresser au problème de l'évaluation de la capacité d'infrastructures ferroviaires, puis à la modélisation que nous en proposons. Nous verrons notamment que les structures des deux problèmes classiques présentés dans ce chapitre se retrouvent dans notre modélisation.

Chapitre 2

Évaluation de la capacité d'infrastructures ferroviaires

Dans ce chapitre, nous allons nous intéresser au problème de l'évaluation de la capacité d'infrastructures ferroviaires qui est l'objet de ce mémoire. Comme tout service de transport collectif guidé, le transport ferroviaire requiert trois types de ressources :

- un matériel roulant (trains) avec des caractéristiques différentes (longueur, vitesse, freinage, ...) qui permet de prendre en charge les passagers ou les marchandises,
- un personnel ayant diverses qualifications (*mécanicien*, contrôleurs, agents de maintenance, ...),
- une infrastructure sur laquelle circule le matériel roulant.

L'infrastructure ferroviaire étant la principale ressource associée au problème étudié ici, nous indiquerons ses caractéristiques. Nous verrons aussi l'intérêt de ce problème de gestion prévisionnelle. Les travaux les plus importants publiés sur ce sujet seront ensuite présentés. Le glossaire (page 133) précise la définition des termes techniques du domaine ferroviaire employés dans ce chapitre et dans les suivants. Ceux-ci sont indiqués en italique dans le texte lors de leur première utilisation.

2.1 Qu'est-ce qu'une infrastructure ferroviaire ?

Une infrastructure ferroviaire peut présenter différentes typologies, pouvant aller d'une simple voie à un *réseau* complet en passant par des ensembles de voies parallèles (*tronçons*) ou sécantes (jonctions ou *nœuds*) et des zones d'arrêt (quais). Ces typologies peuvent être classées en deux catégories :

- une zone homogène correspond à une partie de l'infrastructure où le nombre et l'ordre des trains ne peut pas changer, c'est-à-dire ne contenant pas d'aiguillage et ne permettant ni l'arrêt ni le changement du sens de circulation des trains,

– une zone hétérogène correspond aux autres éléments d'une infrastructure. Une zone englobant une zone hétérogène est forcément hétérogène aussi.

La figure 2.1 illustre ces principales typologies.

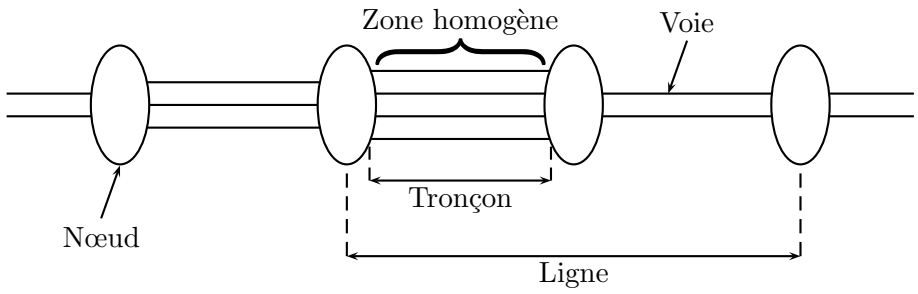


FIG. 2.1 – Éléments d'une infrastructure ferroviaire

La *marche* d'un (ou plusieurs) train(s) sur une infrastructure donnée est habituellement décrite à l'aide de deux modes de représentations principaux :

– le graphique de circulation représente par un trait l'évolution d'un train dans un graphique où le temps et l'espace sont représentés de manière continue respectivement en abscisse et en ordonnée. Il permet de donner une bonne vue de trains se succédant et du *sillon* qu'ils occupent. Il est donc bien adapté pour la représentation d'un tronçon ou d'une *ligne*. Par contre, il peut rendre difficile la visualisation des conflits dûs à un croisement entre deux trains dans des nœuds complexes. La figure 2.3 en propose un exemple correspondant à une situation comprenant quatre trains, dont deux se succédent en *batterie* (t2 et t3). De plus, le train t4 circule dans l'autre sens et ralentit au cours de la période observée. Les *zones de détection* et les deux parcours de l'infrastructure utilisés pour cet exemple sont représentés dans la figure 2.2. En outre, les zones z2 et z3 appartiennent à un même *canton*.
– le diagramme de Gantt représente par des rectangles le temps d'occupation (et éventuellement de réservation et de dégagement) d'un train sur les différentes zones de détection d'une infrastructure dans un graphique où l'espace est représenté de manière discrète (toujours en ordonnée). Il permet ainsi de visualiser tous les conflits de ressources liés à l'infrastructure. Même s'il est peu utilisé actuellement (voir Rodriguez [Rod00b]), nous l'estimons donc plus adapté pour représenter un nœud. La figure 2.4 en propose un exemple correspondant à la même situation que la figure 2.3. Notons dans cet exemple la synchronisation des dates de début d'occupation des zones z2 et z3 puisqu'elles appartiennent à un même canton.

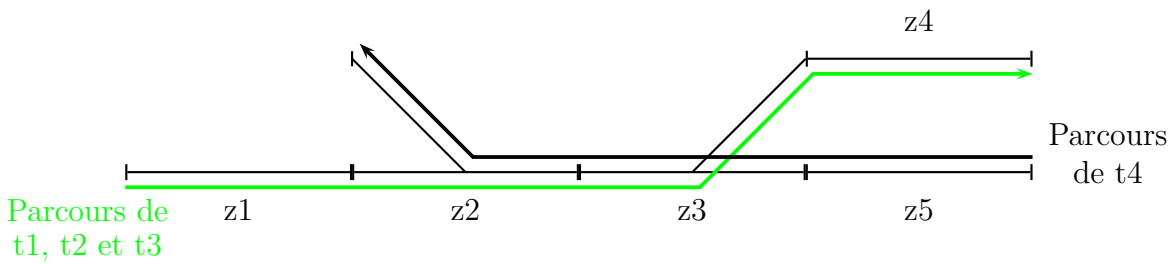


FIG. 2.2 – Exemple d'infrastructure

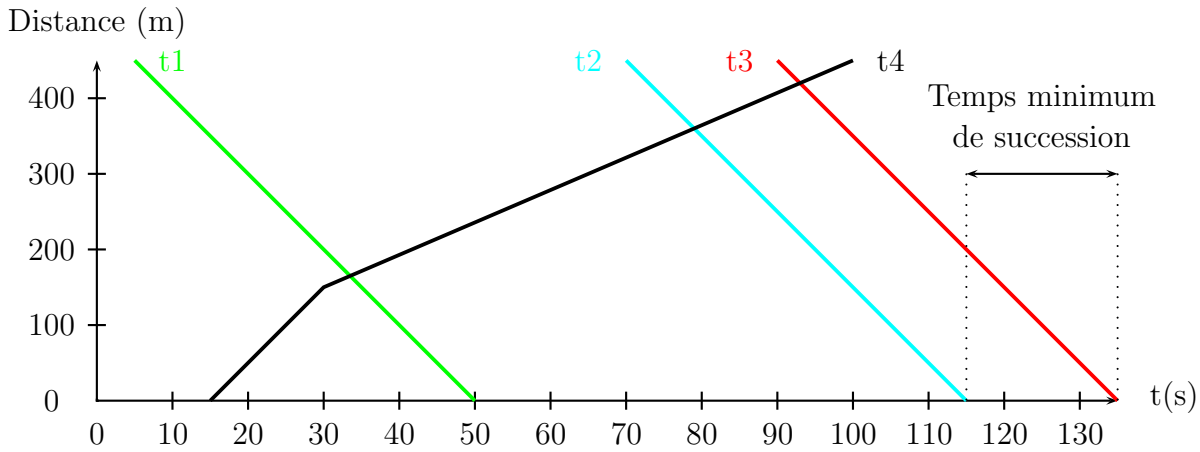


FIG. 2.3 – Exemple de graphique de circulation

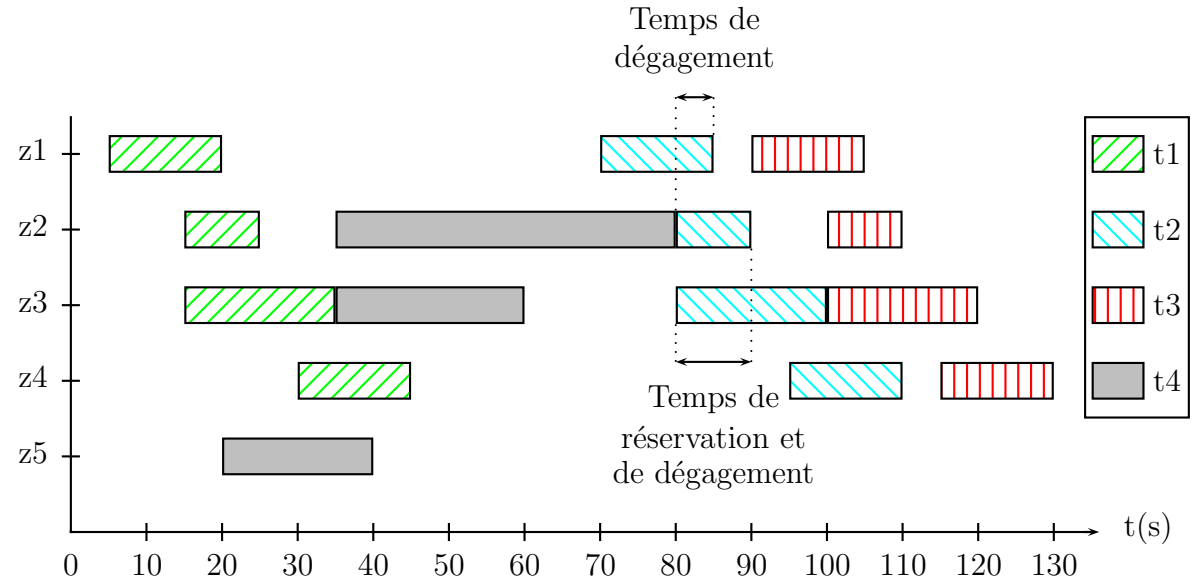


FIG. 2.4 – Exemple de diagramme de Gantt

2.2 Pourquoi évaluer la capacité ?

2.2.1 Positionnement du problème

Comme dans d'autres domaines de gestion de production (voir par exemple l'ouvrage de Giard [Gia03] ou celui de Tahon [Tah92]), les problèmes de gestion prévisionnelle dans les transports ferroviaires peuvent se décomposer suivant trois niveaux de décision en fonction de leur horizon temporel (voir le tableau 2.1) :

1. La construction de nouvelles infrastructures dont la durée de vie est très longue (de l'ordre de cinquante ans), l'abandon d'une partie des installations, l'engagement de personnels et l'achat de matériels roulants engendrent des coûts très importants. Par conséquent, ces projets d'aménagements ne peuvent être conçus, évalués et décidés que dans une perspective à long terme, et se situent au niveau stratégique.
2. Faisant suite à la configuration des ressources, la programmation de l'exploitation consiste à concevoir un *plan de transport* répartissant les ressources dans le temps et dans l'espace. Ces décisions sont qualifiées de tactiques.
3. Enfin, l'exploitation reste une phase tout aussi complexe que les précédentes. Elle correspond à la mise en œuvre des plans de transport conçus en phase tactique. En effet, ceux-ci ne sont jamais complètement appliqués en phase opérationnelle, et les aléas de disponibilité des ressources provenant de problèmes ponctuels (par exemple les *blanc-travaux*) impliquent une adaptation de ces horaires établis.

Notons que les problèmes liés à la gestion des aléas survenant lors de l'exploitation, comme par exemple les retards de certains trains, n'entrent pas dans ce cadre mais dans celui du pilotage en temps réel.

Domaine	Planification	Programmation	Ordonnancement
Niveaux de décisions	Stratégique (long terme)	Tactique (moyen terme)	Opérationnel (court terme)
Horizon temporel	années	mois	journées
Objet	– Infrastructures – Personnel – Offre de transport	– Horaires – Roulements du personnel – Roulements du matériel	– Gestion des dépôts – Gestion des circulations – Maintenance

Tab. 2.1 – Niveaux de décisions en gestion prévisionnelle des modes de transport en commun

Pour mettre en œuvre une stratégie d'offre dans le cadre d'un processus de gestion ferroviaire prévisionnelle à long et moyen terme (planification et programmation), il est impératif de se doter d'outils permettant de situer les limites d'un réseau (existant ou futur) par rapport à une ou plusieurs offres possibles, et facilitant l'étude précise de chaque variante

d'infrastructure pour en évaluer l'intérêt. L'évaluation de la capacité d'une infrastructure ferroviaire permet ainsi de concevoir et d'évaluer une offre répondant à la demande future sans surdimensionner les investissements, et d'utiliser au mieux les infrastructures existantes.

2.2.2 Importance du problème

Dans le contexte actuel, la demande en terme de transport ferroviaire est appelée à évoluer et à croître fortement dans les prochaines années :

- Le mode ferroviaire fait l'objet d'un renouveau d'intérêt afin qu'il se positionne comme mode de substitution à la route pour faire face à la croissance ininterrompue de la mobilité des personnes et des biens. Cette volonté d'augmenter le report modal vers le fer conduit inévitablement à des questions ayant trait à son redéploiement. Une de ces questions concerne les infrastructures et leur capacité d'absorption d'un trafic toujours plus dense et diversifié (TGV, grande ligne, régional, urbain, fret).
- La rude concurrence avec les autres modes de transport conduit les opérateurs qui exploitent les réseaux à faire des efforts permanents pour améliorer la *qualité de service* offerte et adapter l'offre aux attentes des usagers.
- Sur le plan français, la séparation récente entre le gestionnaire de l'infrastructure et le ou les exploitant(s) (voir directive européenne [Eur91]) n'a fait que renforcer ce besoin.

L'accroissement des circulations transforme ainsi certains nœuds, et même certains tronçons très saturés, du réseau ferroviaire en véritables goulets d'étranglement. Cette situation y rend très critique la gestion du trafic. Ces nœuds se comportent alors comme de véritables amplificateurs de retards, en transformant une petite perturbation en une avalanche de retards. Cette question se révèle donc cruciale.

2.3 Définition des problèmes de capacité

Il n'existe pas de définition univoque de la capacité d'une infrastructure ferroviaire. Cependant, la plupart des définitions reposent sur un cadre d'hypothèses définissant les éléments concernés. Les résultats d'une étude de capacité reposent donc sur la validité de ce cadre d'hypothèses. Bellaiche [Bel97] (voir la définition 2.1) distingue ainsi une capacité théorique ne dépendant que de l'infrastructure et des caractéristiques techniques du matériel (la capacité maximale atteignable techniquement), et une capacité pratique prenant en compte les aléas d'une exploitation réelle.

Définition 2.1 (Capacité d'une infrastructure ferroviaire selon Bellaiche)
La capacité théorique est le nombre maximum de trains que l'on peut faire circuler théoriquement sur une ligne ou à travers un nœud pendant une période donnée. La notion de

capacité pratique quant à elle, tient compte d'une marge dite de souplesse afin d'éviter la saturation *totale et les retards en cascade en cas d'incident.*

Cette définition montre ainsi la nécessité de définir le périmètre, spatial et temporel, de l'étude. Elle n'exprime cependant qu'une capacité intrinsèque à l'infrastructure. En fait, beaucoup d'autres paramètres peuvent influer sur la capacité d'une infrastructure. Bellaiche [Bel97] et Schneider [Sch97] décrivent ainsi les paramètres suivants et présentent une étude de leur impact sur la capacité dans des cas réels :

- les infrastructures (nombre de voies, signalisation, vitesse autorisée, maintenance, cisaillements de voies, ...),
- le plan de transport (ordonnancement des trains, hétérogénéité des circulations, contraintes horaires dues aux correspondances ou aux dessertes de voyageurs, ...),
- la qualité de l'exploitation qu'ils assimilent à la stabilité de l'horaire vis à vis des perturbations (travaux, retards, ...) et lient donc à l'importance des marges prévues lors de l'élaboration de la *grille horaire* (afin d'éviter les effets «boule de neige»),
- les caractéristiques techniques des trains (vitesse, accélération, freinage, longueur, ...).

Une notion prenant en compte le point de vue de l'utilisateur comme la qualité de service (qui englobe d'ailleurs la qualité de l'exploitation) peut aussi être considérée. Nous retiendrons donc la définition, plus complète, proposée par Hachemane [Hac97] (définition 2.2).

Définition 2.2 (Capacité d'une infrastructure ferroviaire selon Hachemane)
La notion de capacité peut se définir comme le nombre maximum de trains pouvant circuler dans un intervalle de temps donné dans des conditions pratiques d'exploitation[1] *et pour une structure de lignes, une structure d'horaire et une qualité de service données.*

Ces définitions expriment la capacité d'une infrastructure ferroviaire en nombre de trains par unité de temps. Bien que cette pratique soit très courante, la capacité pourrait aussi être exprimée dans d'autres unités (nombre de passagers, quantité de marchandise, ...).

Une étude de capacité peut se faire sur l'ensemble des parties d'un réseau ferroviaire. En fait, devant l'importance croissante que pourrait prendre le trafic entre les différents pays européens, il pourrait même s'envisager sur plusieurs réseaux. Suivant la typologie de l'infrastructure dont on cherche à évaluer la capacité, la difficulté de cette évaluation peut énormément varier. Basiquement, la capacité C d'une voie ferroviaire dans un sens et pour un intervalle de temps u donné est définie par l'expression théorique suivante (équation 2.1) :

$$C = \frac{u}{h} \tag{2.1}$$

[1] Incluant des marges pour tenir compte des variations inévitables dans la pratique (tension électrique, patinage des roues), du mécanicien (temps de réaction, conduite) et des normes de sécurité.

avec h le temps minimum de séparation entre deux trains successifs. Ce temps de séparation dépend du système de *signalisation* installé sur la voie considérée. Des expressions plus précises peuvent être utilisées pour considérer des situations plus complexes (voir la norme *UIC* [U.I78]). De même, la capacité d'une zone homogène comme un tronçon peut être facilement calculée en effectuant la somme des capacités des différentes voies parallèles qui la composent

Par contre, l'additivité n'est plus applicable dès que l'on considère des zones hétérogènes. Il n'est donc pas possible de calculer la capacité d'une ligne ou d'un nœud, et a fortiori d'un réseau complet, en combinant à l'aide d'une fonction quelconque (comme le minimum ou le maximum par exemple) les capacités des différents éléments qui la composent. Ainsi, il est nécessaire de construire des modèles plus complexes. La capacité d'une jonction peut être définie comme la solution d'un problème d'optimisation appelé «problème de saturation » (voir la définition 2.3).

Définition 2.3 (Problème de saturation)
Le problème de saturation consiste à introduire le maximum de circulations supplémentaires dans une grille horaire établie (éventuellement vide). Les trains ainsi ajoutés représentent la marge de capacité disponible (c-à-d la capacité résiduelle, ou la capacité absolue lorsque la grille horaire établie est vide) de l'infrastructure par rapport à une offre. Les nouveaux trains sont habituellement choisis parmi un ensemble de trains saturants.

Plus généralement, une étude de capacité ne se résume pas à un seul problème d'optimisation. En fait, parmi les autres questions souvent soulevées lors d'une analyse de capacité ferroviaire, celles qui reviennent le plus souvent sont :

1. La capacité de l'infrastructure est-elle suffisante pour faire face aux évolutions de l'offre ?
2. Quels sont les investissements offrant le meilleur débit pour les offres à venir ?
3. Quel est la meilleure façon de répondre à l'offre pour une infrastructure donnée ?

À la première question correspond un autre problème d'optimisation appelé «problème de faisabilité » (voir la définition 2.4).

Définition 2.4 (Problème de faisabilité)
Le problème de faisabilité consiste à vérifier la faisabilité du passage d'une combinaison donnée de trains (point et date d'entrée-sortie fixés) dans une infrastructure considérée. La question est alors de savoir s'il existe un routage *(affectation d'un itinéraire et éventuellement d'un quai) de ces trains permettant de les faire tous passer sans retard.*

La deuxième question, dont la finalité est d'étudier l'impact de modifications d'un ou plusieurs élément(s) physique(s) du système d'exploitation ferroviaire, reprend les problématiques des problèmes de faisabilité et de saturation. Elle y ajoute l'intérêt de déterminer les

points bloquants de l'infrastructure, c'est-à-dire ceux qui limitent la capacité en empêchant d'ajouter des trains saturants supplémentaires.

Enfin, la dernière question est encore plus large puisqu'elle correspond à la notion difficile à définir de qualité de service. Elle couvre ainsi plusieurs problèmes d'optimisation du passage des trains dans l'infrastructure. Cela concerne notamment l'optimisation des préférences émises par le décideur en fonction des attentes supposées ou exprimées des usagers (voir la définition 2.5).

Définition 2.5 (Problème d'optimisation des préférences)
Le problème d'optimisation des préférences consiste à déterminer un routage maximisant la somme des préférences sur des choix de parcours et/ou de dates lors du passage d'une combinaison donnée de trains (point et date d'entrée-sortie fixés) dans une infrastructure considérée. Les préférences considérées dans ce cas sont intrinsèques à chaque train et ne dépendent pas des choix retenus pour les autres trains.

Dans ce cadre se situe également le problème de la fluidification du passage de ces trains lorsqu'il n'est pas possible de les faire passer sans aucun retard (voir la définition 2.6). Cependant, ce problème relève d'un niveau de décision opérationnelle et ne concerne donc pas notre étude.

Définition 2.6 (Problème de fluidification)
Le problème de fluidification consiste à déterminer un routage minimisant la somme des retards générés lors du passage d'une combinaison donnée de trains (point et date d'entrée-sortie fixés) dans une infrastructure considérée.

Enfin, la stabilité d'un routage est une notion très importante soulevée par cette question. Elle représente la capacité de ce routage à absorber les retards de certains trains. Une solution avec une ponctualité dégradée peut entraîner un mécontentement des usagers et donc une réduction de la demande. Cela peut ainsi rendre inutiles tous les efforts réalisés pour augmenter l'offre. Il n'existe cependant pas de définition générale de la stabilité. S'il apparaît clairement qu'il peut exister une dépendance entre le niveau d'utilisation de la capacité (*taux de saturation*) et la stabilité, il n'existe pas dans le cas général de corrélation négative entre ces deux valeurs. La capacité maximale d'une infrastructure n'est pas forcément atteinte lorsque la stabilité est minimale. Une façon (mais pas la seule) d'évaluer la stabilité d'un routage est de calculer les retards générés sur les autres trains du routage par le retard d'un ou plusieurs trains.

2.4 Comment évaluer la capacité : travaux existants

Nous allons maitenant nous intéresser aux différents travaux existants portant sur l'étude de la capacité d'infrastructures ferroviaires. Les problèmes soulevés dans le domaine ferroviaire et les modèles d'optimisation proposés sont nombreux ; Bussieck et al. [BWZ97] et Cordeau et al. [CTV98] présentent ainsi les principaux problèmes de routage et d'ordonnancement rencontrés dans le domaine ferroviaire et leurs modèles. Pourtant, peu d'études scientifiques portent sur le problème de l'analyse de la capacité d'une infrastructure ferroviaire. Ce constat vaut aussi pour les outils commercialisés permettant de faire des analyses de capacité.

Les méthodes pouvant servir à évaluer la capacité sont habituellement classées en quatre grandes catégories :

1. Les formules analytiques qui se basent sur une évaluation de la moyenne des temps minimums de succession entre les différents trains (déterminés grâce aux caractéristiques techniques des trains et des infrastructures et majorés en fonction des normes de sécurité et des niveaux de qualité souhaités). Elles permettent ainsi d'obtenir une valeur théorique représentant la capacité de l'infrastructure étudiée. La plupart de ces méthodes sont dérivées de la formule UIC. Elles ont l'avantage d'être simples et rapides à mettre en œuvre mais imposent l'utilisation d'hypothèses souvent trop réductrices pour l'étude des zones hétérogènes.
2. Les méthodes probabilistes qui se basent sur une évaluation probabiliste de la répartition des trains et font des hypothèses sur la distribution de circulation. Là encore, le résultat obtenu est une valeur théorique. Si elles sont assez faciles à utiliser, ces méthodes imposent des hypothèses souvent trop contraignantes.
3. Les méthodes de simulation qui n'effectuent aucun calcul théorique mais se contentent de simuler la circulation des différents trains connus et des différents événements survenant sur le réseau afin de se rendre compte «de visu» du niveau de qualité et de robustesse d'une grille. Ces méthodes permettent une bonne évaluation de la plupart des cas mais nécessitent un volume d'informations très important sur les caractéristiques de l'infrastructure, et surtout se limitent à l'évaluation des grilles horaires (notamment pour observer l'impact d'une perturbation ou d'un retard) et non pas de la capacité en elle-même.
4. Les méthodes de construction d'horaires qui partent d'une grille horaire donnée pour élaborer la grille la plus dense possible qui correspond à la situation où le niveau de saturation est maximum. Cette catégorie réunit à la fois des méthodes graphiques parfois anciennes et des méthodes algorithmiques plus récentes. Elles peuvent être assez efficaces même sur des zones assez complexes. Leur inconvénient principal est soit de ne pas gérer les conflits éventuels dans les nœuds, soit de nécessiter un volume

d'informations important.

Parmi ces catégories, les méthodes basées sur la construction d'horaires sont les seules à permettre d'effectuer une évaluation de la capacité tout en garantissant l'existence de grilles horaires correspondant aux valeurs calculées. Dans la suite nous nous intéressons donc aux méthodes algorithmiques appartenant à cette catégorie. Nous présentons ainsi (par ordre chronologique) trois méthodes développées dans le cadre de projets réalisés dans différents pays européens.

2.4.1 DONS

Le projet DONS [2] a été développé par les chemins de fer néerlandais[3]. Il a été présenté initialement par Van den Berg et Odijk [vdBO94]. Zwaneveld [Zwa97] en a ensuite donné une description plus complète. Son objectif est d'aider à planifier une ou plusieurs grille(s) horaire(s) pour l'ensemble du réseau hollandais afin d'évaluer la capacité de différents projets d'aménagement à absorber un trafic prévu.

Le résultat de ce projet est un logiciel comprenant une interface graphique et deux modules de calcul complémentaires. Le premier, appelé CADANS, tente d'établir une grille horaire *cadencée* à une heure pour le réseau complet, alors que le deuxième, appelé STATIONS, aide à résoudre les problèmes de routage dans chaque gare afin de vérifier la faisabilité de la grille horaire (dates d'arrivée et de départ de chaque train) proposée par CADANS. Dans le cas où le routage de tous les trains ne peut pas être réalisé par STATIONS, il met en évidence les trains bloquants pour que CADANS génère une autre grille horaire. Lorsque CADANS ne parvient pas à trouver de grille horaire, de nouveaux projets d'aménagements doivent être intégrés ou les estimations de trafic doivent être revues à la baisse.

2.4.1.1 CADANS

L'objectif du module CADANS est de déterminer une grille horaire cadencée à une heure pour l'ensemble du réseau. Cette grille horaire doit être réalisable uniquement en dehors des gares. Par contre, elle peut ne pas être réalisable dans l'une des gares du réseau. Elle ne prend donc en compte que les lignes situées entre ces gares.

Une grille horaire est caractérisée par les dates d'arrivées et de départs des trains dans chaque gare. Les variables de décisions entières $x_{t,s}$ considérées représentent donc la date de départ du train t depuis la gare s. Les dates d'arrivée peuvent ensuite être déduites car le temps passé dans la gare est supposé constant. Ces variables sont soumises à trois familles de contraintes :

[2] Design Of Network Schedules
[3] Nederlandse Spoorwegen (NS)

- sur les temps de parcours, l'écart entre les dates de départ d'un train pour deux gares successives où il doit passer devant être cohérent avec le temps de parcours entre ces deux gares et le temps d'arrêt prévu dans la deuxième gare,
- pour les connexions entre certaines paires de trains, les temps de présence au quai de deux trains dans une même gare devant être suffisamment proches pour permettre une correspondance lorsqu'il y a lieu,
- et sur l'utilisation de l'infrastructure, deux trains utilisant une même partie de l'infrastructure devant le faire avec un temps d'écart suffisant.

Toutes ces contraintes peuvent être exprimées sous la forme générale suivante (équation 2.2) :

$$(x_{t,s} - x_{t',s'}) \in [a, b] + 60p \tag{2.2}$$

avec la variable p modélisant le caractère cyclique sur 60 minutes de l'horaire, l'ensemble des autres variables prenant une valeur dans l'intervalle $[0, 59]$. Elles peuvent aussi être représentées sous forme d'un graphe dans lequel les variables sont représentées par des nœuds, et les intervalles de compatibilité par des arcs orientés valués entre ces nœuds.

De nombreuses approches ont été considérées pour résoudre ce modèle (Lindner [Lin00] présente les principales). Ces méthodes font appel à des techniques très diverses :
- programmation par contraintes,
- algorithme de Branch & Bound après une reformulation en un MILP,
- méthode de coupes.

La plupart des instances de ce problème sont ainsi résolues dans des temps raisonnables. Il existe cependant encore des cas de figure où la résolution exacte de ce problème n'est pas possible.

2.4.1.2 STATIONS

L'objectif des travaux autour de STATIONS, présentés par Zwaneveld et al. [ZKR+96, Zwa97, ZKVH01] et par Kroon et al. [KRZ97], est de réaliser le routage d'un nombre maximum de trains dans un nœud ferroviaire sur une période de temps donnée. Le fait de chercher un routage valide pour un nombre maximum de trains et non directement pour tous les trains permet d'identifier un sous-ensemble de trains bloquants lorsque le routage de tous les trains n'est pas possible.

Dans cette étude, un nœud ferroviaire est caractérisé par un certain nombre de points d'entrée, de quais et de points de sortie. Un train parcourant ce nœud emprunte donc un itinéraire d'entrée, composé de plusieurs sections, depuis son point d'entrée (dépendant de la provenance du train) jusqu'à un quai. Puis, il emprunte un itinéraire de sortie de ce quai

jusqu'à son point de sortie (dépendant de sa direction de voyage). Un itinéraire complet correspond à la combinaison d'un itinéraire d'entrée et d'un itinéraire de sortie utilisant le même quai[4].

Les heures d'arrivée et de départ considérées pour un train correspondent aux moments où il s'arrête et où il quitte le quai et non pas le nœud. En fait, les heures d'arrivée et de départ du nœud sont directement liées à ces horaires. En effet, le modèle est basé sur l'hypothèse que tous les itinéraires possibles pour un train nécessitent exactement le même temps de parcours. Notons aussi que dans le cas d'un train au passage, les heures d'arrivée et de départ sont égales.

Ainsi, ce problème est modélisé en considérant l'ensemble des trains T, l'ensemble des itinéraires R (l'ensemble $R_t \in R$ représentant les itinéraires possibles[5] pour un train $t \in T$) et l'ensemble des variables de décision $x_{t,r} = \begin{cases} 1 \textit{ si le train } t \in T \textit{ utilise l'itinéraire } r \in R_t \\ 0 \textit{ dans les autres cas} \end{cases}$.
L'ensemble $F_{t,t'}$ contient toutes les combinaisons d'itinéraires possibles pour deux trains t et t'. Il peut donc se formuler comme le problème en variables binaires suivant (2.3) :

$$\left[\begin{array}{lll} Max\ z(X) = & \displaystyle\sum_{t\in T}\sum_{r\in R_t} x_{t,r} & \\ & \displaystyle\sum_{r\in R_t} x_{t,r} \leq 1 \quad ,\forall t \in T & (a) \\ & x_{t,r} + x_{t',r'} \leq 1 \quad ,\forall (t,t') \in T^2, r \in R_t, r' \in R_{t'}, (r,r') \notin F_{t,t'} & (b) \\ & x_{t,r} \in \{0,1\} \quad ,\forall t \in T, r \in R_t & \end{array}\right] \tag{2.3}$$

où les contraintes (a) modélisent le fait qu'un train ne peut prendre qu'un seul parcours et les contraintes (b) le fait que le passage de deux trains sur deux parcours non compatibles ne peut pas être réalisé en même temps. C'est donc un problème de set packing.

Deux extensions ont aussi été proposées. La première permet d'autoriser un léger décalage des horaires d'arrivée et de départ des trains. Ces deux décalages sont exprimés par une déviation $\delta = (\delta_a, \delta_d)$. Ainsi, l'ensemble $F_{t,t'}$ contient toutes les combinaisons d'itinéraires-déviations possibles pour les trains t et t', et R_t les itinéraires-déviations possibles pour le train t.

[4]La décomposition des itinéraires en itinéraires d'entrée et en itinéraires de sortie permet ainsi de modéliser le couplage ou le découplage de certains trains.

[5]La décomposition des itinéraires permet donc ainsi de diminuer le nombre d'itinéraires : si un train a respectivement n_1 et n_2 itinéraires d'entrée et de sortie et $|P|$ quais possibles, le nombre d'itinéraires complets possibles passera de $n_1 * n_2 * |P|$ à $(n_1 + n_2) * |P|$.

La seconde permet d'exprimer les préférences de certains trains pour un itinéraire particulier, comme par exemple des choix de quais pour des correspondances ou des raisons de convenance des usagers qui préfèrent qu'un même type de train parte toujours du même quai. Dans ce cas, seule la fonction économique du modèle est modifiée par l'introduction d'un paramètre $\rho_{t,r}$ qui vient pondérer les différentes variables $x_{t,r}$.

La méthode utilisée pour résoudre de manière exacte ce problème pour une grille horaire cadencée à l'heure est composée de trois phases :

1. L'initialisation sert à déterminer toutes les variables (itinéraires pour un train) ainsi que les itinéraires compatibles. Pour cela, les seules sections considérées sont celles contenant un aiguillage, une intersection, un point d'entrée, un quai ou un point de sortie. Ce choix de ne considérer que les sections dites déterminantes pour détecter les conflits potentiels n'est valide que sous des hypothèses très fortes. Il suppose notamment que les infrastructures étudiées ne contiennent aucune *voie banalisée* et qu'aucun croisement n'a lieu. De même, il requiert l'hypothèse d'égalité des temps de parcours évoquée précédemment, et exclue les cas de dépassement d'un train par un autre dans le nœud. Cette dernière hypothèse s'avère surtout restrictive lorsque le trafic est hétérogène et comprend des trains circulant à des vitesses très différentes.
2. Une phase de réduction utilise successivement un test sur la pertinence des variables puis quatre autres tests valides pour tout problème de node packing [6] :
   (a) «Detour routes» supprimant les variables dont l'itinéraire correspond à un détour par rapport à un autre itinéraire possible,
   (b) «Node dominance» supprimant les variables dominées par une autre,
   (c) «Combining nodes» combinant des couples de variables en une seule si la sélection de l'une dans une solution optimale entraîne la sélection de l'autre,
   (d) «Set dominance» supprimant des variables dominées par un ensemble de variables,
   (e) «Iterative set dominance» supprimant des ensembles de variables dominé par un autre ensemble.

   Ces tests de réduction permettent de diminuer très fortement la taille des problèmes considérés. Ainsi, dans les résultats rapportés, le nombre de variables des problèmes est ramené en moyenne à 7.7% du nombre initial de variables. Les trois premiers tests sont les plus efficaces (de très loin, surtout pour les deux premiers).

[6] Ils utilisent pour cela la transformation du problème de set packing en node packing décrite à la section 1.3.2.1 (page 15)

3. Un algorithme de Branch & Cut résout le problème en utilisant une relaxation linéaire et en ajoutant des cliques spécifiques. Cependant, celui-ci pouvant nécessiter un temps important, le problème à résoudre ne doit plus être de trop grande taille.

2.4.2 CAPRES

Le projet CAPRES [7] a été développé par les chemins de fer suisses[8]. Il est basé sur les travaux d'Hachemane [Hac97]. Le logiciel résultant, décrit par Curchod et Lucchini [CL01], est aussi utilisé dans d'autres pays et notamment en France pour certaines études menées par la SNCF. Son objectif est de fournir un système d'aide à l'évaluation de la capacité d'infrastructures complexes (réseaux) par construction d'horaires. Pour cela, il traite les deux problèmes suivants :

1. élaborer des variantes d'horaires (c-à-d d'ordonnancement et non pas de grille horaire) cadencées d'un ensemble de trains permettant de les faire circuler dans l'infrastructure étudiée,
2. mesurer la capacité de cette infrastructure en saturant une de ces variantes avec des trains supplémentaires.

La partie du réseau étudiée est représentée sous forme de nœuds et de tronçons. Les nœuds peuvent être modélisés selon trois niveaux de détail différents : conflits dans le nœud non pris en compte par le modèle, conflits de cisaillement uniquement pris en compte, conflits de cisaillement et d'attribution des quais pris en compte. Les contraintes à respecter dans la construction de l'horaire peuvent être rangées en deux catégories :

- Les contraintes dites potentielles modélisent les intervalles horaires imposés aux trains, leurs temps de parcours, leurs durées d'arrêt et leurs correspondances. Elles expriment le fait qu'un événement B doit se produire dans une fourchette de temps dépendant d'un autre événement A. Il s'agit en fait de contraintes de localisation d'un événement par rapport à un autre exprimés par les deux inégalités de potentiel suivantes (équation 2.4) :

$$h_A + t^- \leq h_B \text{ et } h_B \leq h_A + t^+ \tag{2.4}$$

  avec h_A (resp. h_B) l'heure de l'événement A (resp. B) et $t^- \leq t^+$,
- Les contraintes dites disjonctives modélisent l'espacement et les croisements entre les trains. Elles expriment le fait qu'un événement B doit se produire avec un écart de temps minimal par rapport à un autre événement A (voir l'équation 2.5) :

$$h_B \geq h_A + t^+ \text{ OU } h_B \leq h_A - t^- \tag{2.5}$$

[7] système d'aide à l'analyse de la CAPacité des RÉSeaux ferroviaires
[8] Chemins de Fer Fédéraux suisses (CFF)

Ces contraintes sont en fait très proches de celles exprimées dans le modèle CADANS décrit dans la section 2.4.1.1.

La résolution utilisée s'inspire directement des travaux de Fukumori et Sano [FS87]. Elle se base sur des méthodes de propagation des contraintes potentielles. L'algorithme effectue sa recherche en branchant sur les contraintes disjonctives. Il peut ainsi positionner tous les trains en commençant par les plus contraints. Pour le problème de saturation, le même algorithme est utilisé de manière itérative en introduisant successivement les trains (seuls ou par groupes selon la stratégie utilisée) de la liste saturante ordonnée.

2.4.3 DÉMIURGE

Le projet DÉMIURGE est actuellement développé par les chemins de fer français[9]. Il est encore à l'état de prototype et a, pour le moment, été uniquement présenté de manière succinte par Labouisse et Djellab [LD01, LD02]. Son objectif est de mettre au point un outil d'aide à la décision permettant de mesurer et d'optimiser la capacité de toutes les parties d'un réseau (pour le moment les grandes lignes). Cet outil serait commun à tous les services de la SNCF, et devrait donc remplacer l'utilisation du logiciel CAPRES. Il doit notamment permettre de considérer les points suivants :

- vérifier dans quelle mesure le réseau actuel peut absorber l'accroissement prévu de différents types de trafic,
- identifier les points bloquants du réseau,
- évaluer différentes hypothèses d'investissements pour modifier l'infrastructure au niveau des points bloquants,
- et optimiser les grilles horaires.

La partie du réseau étudiée est représentée sous forme de nœuds et de tronçons. Là encore, les trois niveaux de détail utilisés pour les nœuds dans CAPRES (voir la section 2.4.2) peuvent être considérés. Les trains considérés sont, eux, répartis en trois catégories : ceux de l'horaire de base, ceux demandés à court ou moyen terme et ceux demandés à long terme. Les variables de décision utilisées sont les heures d'arrivée et de départ de ces trains dans les nœuds. Les valeurs possibles de ces variables sont définies par des contraintes techniques liées à l'infrastructure telles que l'espacement entre les trains successifs ou le temps de séparation entre des trains utilisant des itinéraires incompatibles, et par des contraintes commerciales comme les parcours et les horaires des trains, les temps d'arrêt en gare, le cadencement ou encore les correspondances. La solution recherchée pour le MILP ainsi modélisé peut alors optimiser différents critères :

[9] SNCF

- maximiser le nombre de trains,
- minimiser les temps de parcours,
- minimiser les modifications des horaires des trains de base (si elles sont autorisées),
- minimiser la violation de certaines contraintes techniques (si cela est autorisé pour rechercher les points bloquants).

La résolution est effectuée à l'aide du logiciel commercial Cplex (basé sur une résolution de type Branch & Cut). L'ajout de pré-traitements et la recherche de coupes a ainsi permis d'obtenir des résultats dans des temps satisfaisants pour les cas de figures testés. Cependant, une méthode de résolution par décomposition est actuellement envisagée. De plus, plusieurs autres améliorations (principalement pour optimiser le routage des trains et pour évaluer la robustesse des grilles horaires vis-à-vis de petits aléas d'exploitation) pourraient être développées.

2.4.4 Synthèse des travaux existants

Ces trois projets considèrent des problèmes assez proches puisqu'ils s'intéressent essentiellement à la capacité des infrastructures ferroviaires à l'échelle d'un réseau (ou au moins d'un sous-ensemble important d'un réseau). Leurs objectifs sont d'ailleurs assez proches ; le fait que la SNCF envisage de remplacer le logiciel CAPRES par DÉMIURGE montre bien la similitude entre les deux projets. De même, ils présentent des caractéristiques communes, notamment pour la formulation des contraintes. Malgré tout, plusieurs différences peuvent être relevées. Le tableau 2.2 récapitule ainsi les caractéristiques principales des différentes études de capacité présentées dans les sections précédentes.

Projet	DONS		CAPRES	DÉMIURGE
Type de problème	Faisabilité ou optimisation des préférences		Faisabilité et saturation	Faisabilité, saturation ou fluidification
Type d'horaire	Cadencé		Cadencé	Quelconque
Infrastructure étudiée	Réseau (CADANS)	Gare (STATIONS)	Réseau	Réseau, lignes
Modélisation	MILP	0-1ILP	MILP	MILP
Méthode(s) de résolution	PPC, Branch & Bound, ou méthodes de coupes	Branch & Cut	PPC	Pré-traitements, coupes et Branch & Cut (Cplex)

Tab. 2.2 – Synthèse des différentes études de capacité

Tout d'abord, si ces projets considèrent tous le problème de la capacité ferroviaire, ils ne traitent pas toujours le même champ de questions. S'ils permettent tous de répondre au problème de faisabilité, les problèmes de saturation et surtout d'optimisation des préférences et de fluidification ne sont pas toujours considérés. En outre, hormis CAPRES, ils considèrent ces problèmes de manière indépendante et non pas selon une approche multiobjectif. De plus, la notion de stabilité n'est jamais traitée à l'intérieur de ces projets. Ils renvoient ainsi l'utilisateur vers des méthodes de simulation pour effectuer cette évaluation[10].

Ensuite, chacun de ces projets a été réalisé dans une logique liée aux contraintes d'exploitation propres au pays où il a été développé. Le type d'horaire considéré est ainsi caractéristique. L'utilisation plus rare, jusqu'à maintenant, en France d'horaires cadencés a amené la SNCF à ne considérer cela que comme une possibilité.

Enfin, la diversité des méthodes de résolution considérées montrent bien la difficulté de ces problèmes. Notons à ce sujet que les problèmes traités étant fort contraints, l'utilisation de méthodes de propagations de contraintes ou basées sur des ajouts de coupes semblent plus performantes. Bien évidemment, la pertinence des résultats (et les temps de réponse) de ces méthodes (et la faisabilité réelle des grilles horaires construites) dépendra grandement de la précision des données fournies en entrée. Ceci peut parfois amener à représenter les nœuds d'un réseau d'une manière trop simplifiée. L'approche du projet DONS est dans cette optique très intéressante, puisqu'elle décompose le problème, ce qui permet de traiter à part le cas des gares.

2.5 Conclusion

Dans ce chapitre, nous avons présenté la problématique d'une étude de capacité d'infrastructures ferroviaires. Après avoir indiqué son positionnement dans le cadre des problèmes de transport ferroviaire, nous avons montré son importance dans le contexte actuel. De plus, nous avons vu que de nombreuses questions, et donc de nombreux problèmes, se posent lors de ce type d'étude. Dans notre cas, nous nous intéressons plus particulièrement aux problèmes de faisabilité, de saturation, d'optimisation des préférences et à la stabilité des routages correspondants, pour des infrastructures à l'échelle d'un nœud ou d'une gare.

Pour répondre à ces questions, de nombreuses méthodes sont possibles pour des infrastructures simples, mais la combinatoire induite par la complexité de certaines infrastructures comme les nœuds rend nécessaire l'utilisation de méthodes algorithmiques. Parmi celles existantes à l'heure actuelle, aucune ne répond complètement au champ de questions nous intéressant, et aucune à la question de la stabilité. De même, l'aspect multiobjectif du problème n'est jamais envisagé. En outre, une seule de ces méthodes considère l'échelle

[10] L'évaluation s'effectue alors habituellement en observant les effets d'une ou plusieurs perturbations sur la grille horaire

d'un nœud, mais retient des hypothèses trop fortes pour des nœuds complexes comprenant des voies banalisées et accueillant un trafic hétérogène. Les autres, elles, s'intéressent à des échelles plus macroscopiques et représentent les nœuds de façon trop simplifiée pour garantir la faisabilité des grilles horaires générées.

Dans la suite, nous allons donc développer la modélisation linéaire, puis les méthodes de résolution, que nous proposons pour traiter complètement le problème considéré. Nous verrons notamment comment nous pouvons considérer un niveau de détail plus approprié à une infrastructure telle qu'un nœud complexe. De plus, nous montrerons de qu'elle manière l'évaluation de la stabilité des grilles horaires générées peut être intégrée dans ce modèle.

Chapitre 3

Description du modèle ferroviaire

Après avoir présenté notre problème, nous allons maintenant développer dans ce chapitre les différents éléments de la modélisation que nous proposons. En effet, aucune des méthodes présentées dans le chapitre précédent ne répond complétement à nos attentes, notamment à cause de l'échelle considérée et des questions traitées. Cette modélisation s'inspire cependant des travaux réalisés par Zwaneveld et al. [ZKR+96, Zwa97, ZKVH01] dans le cadre du projet STATIONS (voir la section 2.4.1.2) qui considérait une échelle équivalente. Avant de la présenter, il est nécessaire d'introduire quelques notations et définitions. Ensuite, nous nous intéresserons aux problèmes, décrits dans la section 2.3 (page 33), de faisabilité, de saturation et d'optimisation des préférences d'une grille horaire, puis à l'évaluation de la stabilité des solutions obtenues.

3.1 Notations et définitions

Une instance de notre problème correspond à un scénario se déroulant sur un horizon temporel fixé. Cet horizon temporel est défini par les horaires de début et de fin du scénario appelés respectivement *debut* et fin.

3.1.1 Ensembles de trains

Durant ce scénario, nous travaillons avec un ensemble pré-défini de trains. Ce choix présente l'avantage de ne pas considérer des trains ne correspondant à aucune demande, générant des problèmes hors du nœud considéré ou n'apparaissant pas pertinents aux yeux du décideur. Cet ensemble de trains peut être divisé en plusieurs sous-ensembles. Nous avons ainsi les ensembles de trains suivants :

T Ensemble de tous les trains

T_{Fais} Sous-ensemble des trains de la grille horaire initiale (problème de faisabilité) :

$$T_{Fais} \subseteq T$$

T_{Sat} Sous-ensemble des trains de la liste saturante (problème de saturation) :

$$T_{Sat} = T \setminus T_{Fais}$$

Les trains saturants peuvent être séparés en plusieurs types ou catégories définies par l'ensemble $Types$. La séparation entre les trains voyageurs et les trains de marchandise est la plus courante mais d'autres séparations peuvent être considérées.

T^q_{Sat} Sous-ensemble des trains saturants de type $q \in Types$:

$$T_{Sat} = \bigcup_{q \in Types} T^q_{Sat}, \bigcap_{q \in Types} T^q_{Sat} = \emptyset$$

T_{Cycle} Sous-ensemble des trains devant effectuer leur parcours de manière cyclique ou cadencée avant et après l'horizon temporel du scénario :

$$T_{Cycle} \subseteq T$$

T_{Coupl} Sous-ensemble des trains devant être couplés à un autre train :

$$T_{Coupl} \subset T$$

La fonction $f_{Coupl}(t) : T_{Coupl} \rightarrow T$ associe à chaque train t devant être couplé, le train auquel il se couple.

$T_{Decoupl}$ Sous-ensemble des trains devant être découplés d'un autre train :

$$T_{Decoupl} \subset T$$

La fonction $f_{Decoupl}(t) : T_{Decoupl} \rightarrow T$ associe à chaque train t devant être découplé, le train duquel il se découple.

3.1.2 Parcours des trains

Chacun de ces trains peut être routé sur différents parcours. Nous considérons uniquement un ensemble fini de parcours entièrement définis à l'avance (voir l'hypothèse 3.1).

Hypothèse 3.1
Les marches des trains associées à chacun des parcours considérés correspondent à des marches types pré-définies (par exemple des marches en voie libre*) et indépendantes des autres trains ou de l'horaire.*

Ces parcours peuvent être partitionnés en itinéraires. Ce partitionnement a priori peut permettre de diminuer la combinatoire sur le nombre de parcours possibles lorsque le train fait un arrêt dans le nœud. Les itinéraires peuvent être classés en deux catégories : les routes dans lesquels le train ne s'arrête pas et les quais dans lesquels le train fait un arrêt. Les ensembles ci-dessous définissent ces parcours et leurs partitions :

I_t Ensemble des partitions considérées pour les parcours du train $t \in T$.

$R_{t,i}$ Ensemble des itinéraires possibles pour un train $t \in T$ sur la partition $i \in I_t$

$Comp_{t,i,r}$ Ensemble des itinéraires possibles pour un train $t \in T$ sur la partition $(i+1) \in I_t$ et compatibles (au sens de la continuité physique) avec le choix de l'itinéraire $r \in R_{t,i}$ sur la partition i ($Comp_{t,i,r} \subset R_{t,i+1}$).

Ainsi par exemple, le parcours d'un train dans une gare peut être séparé en une route d'entrée, un arrêt au quai et une route de sortie. En notant a (resp. c) le nombre de routes d'entrée (resp. de sortie) et b le nombre de quais possibles, on obtient $a + b + c$ itinéraires en partitionnant les parcours au lieu de $a * b * c$ parcours (si toutes les combinaisons sont possibles). Notons que pour un parcours partitionné, toutes les combinaisons d'itinéraires physiquement compatibles sont possibles.

En outre, la fonction $f'_{Coupl}(t) : T_{Coupl} \rightarrow I_{f_{Coupl}(t)}$ (resp. $f'_{Decoupl}(t) : T_{Decoupl} \rightarrow I_{f_{Decoupl}(t)}$) associe à chaque train devant être couplé (resp. découplé) à un autre train, la partition du parcours de cet autre train sur laquelle le couplage (resp. découplage) a lieu.

3.1.3 Horaires des trains

De même, chacun de ces trains doit respecter des horaires de référence. Un nombre fini de décalages temporels peut éventuellement être considéré pour chaque train de manière indépendante. Ils permettent ainsi de définir un ensemble fini d'horaires possibles (voir l'hypothèse 3.2).

Hypothèse 3.2
Les horaires possibles pour chacun des trains considérés correspondent à un ensemble discret pré-défini d'horaires compatibles avec :

- *les disponibilités en matériel roulant et en personnel,*

- *la capacité et le trafic des infrastructures extérieures,*
- *et l'offre que l'exploitant souhaite proposer.*

Certains couples de trains peuvent aussi être mis en correspondances. Les éléments ci-dessous définissent ces horaires :

H_t Horaire de référence du train $t \in T$. Dans le cas où les parcours de t ne sont pas partitionnés ($|I_t| = 1$) et que t traverse le nœud, H_t correspond à l'heure d'arrivée du train dans le nœud. Dans les autres cas ($|I_t| > 1$), H_t est un ($|I_t| - 1$)-uplet $(H_t^1, \ldots, H_t^{|I_t|-1})$ qui correspond aux dates d'intersections entre les partitions. Par exemple, pour des parcours en trois parties (route d'entrée, arrêt au quai, route de sortie) H_t représente les heures d'arrêt et de redémarrage du quai.

Δ_t Ensemble des décalages temporels autorisés pour un train $t \in T$ par rapport à l'horaire de référence H_t. Dans le cas où les parcours de t sont partitionnés ($|I_t| > 1$), un élément de Δ_t est un ($|I_t|-1$)-uplet $(\delta_1, \ldots, \delta_{|I_t|-1})$ qui correspond aux décalages de chacune des dates d'intersections entre les partitions.

$corr_{t,r,t',r'}$ Temps minimum requis pour la correspondance du train $t \in T$ arrêté au quai $r \in R_{t,i}$ (avec $i \in I_t$) vers le train $t' \in T$ arrêté au quai $r' \in R_{t',i'}$ (avec $i' \in I'_t$). Une valeur du paramètre $corr_{t,r,t',r'}$ supérieure à la durée de l'horizon temporel (c-à-d $corr_{t,r,t',r'} \geq (fin - debut)$) signifie que la correspondance du train t vers le train t' ne doit pas se faire entre les quais r et r'.

$coupl_{t,t'}$ Temps minimum requis pour le couplage du train $t \in T_{Coupl}$ au train $t' \in T$.

$decoupl_{t,t'}$ Temps minimum requis pour le découplage du train $t \in T_{Decoupl}$ du train $t' \in T$.

Par définition, les horaires de référence et les décalages temporels de tous les trains sont dans l'horizon temporel défini par les horaires de début et de fin du scenario.

$$debut \leq H_t^i + \delta_i \leq fin, \forall t \in T, \forall i \in I_t, \forall \delta \in \Delta_t \tag{3.1}$$

Toutes ces définitions correspondent au cas de figure classique où les trains entrent et sortent de l'infrastructure par les itinéraires prévus (ou alors sont présents depuis le début du scénario ou jusqu'à la fin). Si un train t doit «disparaître» de (resp. «apparaître» dans) l'infrastructure au cours du scénario, une composante supplémentaire $H_t^{|I_t|}$ (resp. H_t^0) doit être ajoutée à l'horaire de référence H_t pour indiquer l'horaire de cet événement. De même,

une composante $\delta_{|I_t|}$ (resp. δ_0) doit être ajoutée à l'ensemble des décalages temporels Δ_t autorisés. Cette particularité sert essentiellement lors du couplage ou du découplage de deux trains (voir section 3.2.2.7).

3.1.4 Caractéristiques de l'infrastructure

Enfin, il est nécessaire de définir les caractéristiques de l'infrastructure étudiée. Les caractéristiques utiles pour notre modèle sont les zones de détection et leur utilisation par chacun des trains :

Zones Ensemble de ressources unaires propres à l'infrastructure considérée. Ces ressources peuvent être physiques (zones de détections) ou logiques (découpage de zones de détection à capacité n-aire comme certains quais).

$h^a_{z,t,r,\delta}$ Heure à partir de laquelle la zone $z \in Zones$ est utilisée par le train $t \in T$ empruntant l'itinéraire $r \in R_{t,i}$ (avec $i \in I_t$) avec un décalage temporel $\delta \in \Delta_t$. Cette utilisation peut être soit physique (présence sur la zone), soit «logique» (respect des règles de sécurité et d'espacement entre les trains induites par le cantonnement de l'infrastructure et sa signalisation). Elle correspond à un parcours en voie libre du train, et peut être déterminée soit par des relevés de terrain, soit par une simulation.

$h^d_{z,t,r,\delta}$ Heure à partir de laquelle la zone $z \in Zones$ n'est plus utilisée par le train $t \in T$ empruntant l'itinéraire $r \in R_{t,i}$ (avec $i \in I_t$) avec un décalage temporel $\delta \in \Delta_t$ et est donc disponible pour d'autres trains.

À la différence des horaires de références, les heures d'utilisation des zones peuvent être en dehors de l'horizon temporel du scénario afin de modéliser le routage complet du parcours du train.

Le diagramme de Gantt de la figure 3.1 présente un exemple d'utilisation des zones dépassant l'horaire de fin du scénario par un train s'arrêtant sur un quai en cul-de-sac.

3.1.5 Variables de décision

À partir de ces définitions, notre problématique est de déterminer un sous-ensemble de trains et de fixer leur parcours et leur horaire. Nous considérons donc pour notre modèle les variables de décision binaires suivantes :

$$x_{t,r,\delta} = \begin{cases} 1 \text{ si le train } t \text{ est routé sur l'itinéraire } r \\ \qquad\qquad \text{avec un décalage temporel } \delta \\ 0 \text{ sinon} \end{cases} \quad ,\forall t \in T, i \in I_t, r \in R_{t,i}, \delta \in \Delta_t$$

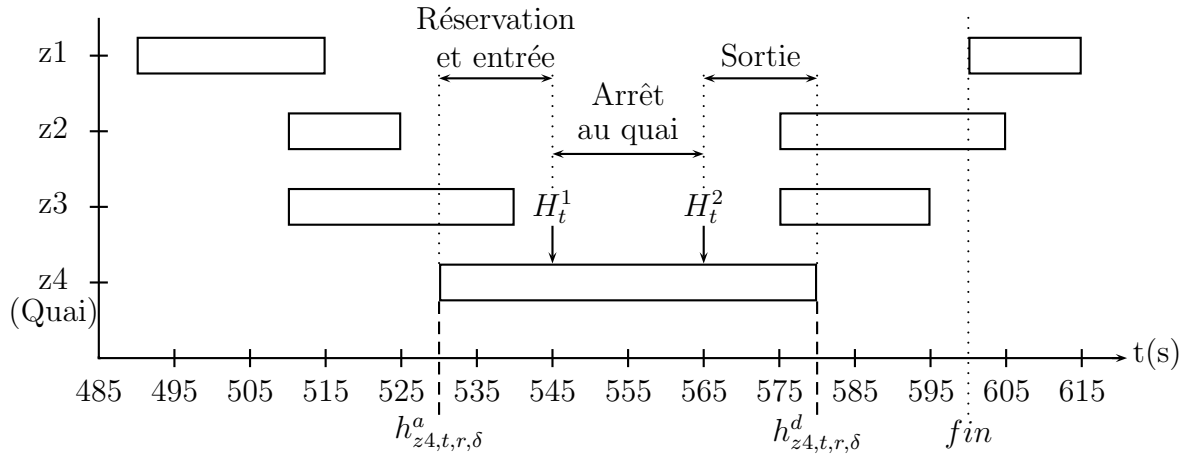


FIG. 3.1 – Utilisation des zones

En outre, le décideur peut définir une pondération $w_{t,r,\delta} \in I\!N$ pour chacune des variables $x_{t,r,\delta}$. Celle-ci est utilisée dans le cadre du problème d'optimisation des préférences (voir la section 3.2.1.3).

3.2 Faisabilité et saturation d'une grille horaire

Nous allons maintenant nous intéresser au modèle que nous proposons pour les problèmes de faisabilité, de saturation, et d'optimisation des préférences, présentés dans la section 2.3 (page 33). Ce modèle est une évolution de celui proposé par Delorme et al. [DRG01].

3.2.1 Objectifs du problème

Le choix des valeurs à affecter à chacune de ces variables doit se faire en fonction de plusieurs critères.

3.2.1.1 Faisabilité

Le premier objectif de notre problème (équation 3.2) consiste à faire circuler un nombre maximum de trains de la grille horaire initiale. Il correspond au problème de faisabilité (voir la définition 2.4, page 35). Les parcours étant partitionnés, cet objectif correspond à la somme, pondérée en fonction du nombre de partitions, des variables représentant un des trains de cette grille horaire. En effet, lorsque le parcours d'un train est partitionné, une variable associée à ce train ne représente que son affectation à un itinéraire et non pas à un parcours complet.

$$max\ z_{Fais} = \sum_{t \in T_{Fais}, i \in I_t, r \in R_{t,i}, \delta \in \Delta_t} \frac{1}{|I_t|} x_{t,r,\delta} \tag{3.2}$$

Ainsi, par exemple, pour un train dont le parcours est partitionné en une route d'entrée, un arrêt au quai et une route de sortie ($|I_t| = 3$), une variable représentant un itinéraire possible pour la route d'entrée de ce train aura une pondération d'$\frac{1}{3}$ dans la fonction objectif. Cela signifie que la sélection de cette seule variable ne permet pas d'assurer le routage complet du train. Pour que le train soit routé entièrement, il faut aussi que deux autres variables représentant ce train soit sélectionnées.

Cet objectif est prioritaire par rapport aux suivants.

3.2.1.2 Saturation

Ensuite, nous avons une famille d'objectifs (équation 3.3) correspondant au problème de saturation (voir la définition 2.3, page 35) pour chacune des catégories de trains. Pour chaque catégorie, il s'agit de maximiser le nombre de trains sélectionnés dans la solution. Le principe est le même que pour l'objectif de faisabilité mais concerne les trains contenus dans les listes saturantes.

$$max\ z^q_{Sat} = \sum_{t \in T^q_{Sat}, i \in I_t, r \in R_{t,i}, \delta \in \Delta_t} \frac{1}{|I_t|} x_{t,r,\delta}, \forall q \in Types \tag{3.3}$$

Dans le cas général, il n'y a pas de priorité fixée a priori entre ces objectifs de saturation.

Pour exprimer la capacité dans une autre unité que le nombre de trains par unités de temps (voir section 2.3, page 33), un coefficient multiplicateur représentant la quantité associée à chaque train peut être intégré dans les équations 3.2 et 3.3.

3.2.1.3 Préférences du décideur

Enfin, le dernier objectif (équation 3.4) correspond au problème d'optimisation des préférences (voir la définition 2.5, page 36) émises par le décideur. Il s'agit de maximiser la somme, pondérée en fonction du nombre de partitions et de ces préférences, de l'ensemble des variables du problème (c-à-d tous les trains de la grille horaire et des listes saturantes). Ce niveau de préférence peut éventuellement être aussi utilisé pour modéliser une capacité exprimée dans une autre unité que le nombre de trains (voir la section 2.3). Dans ce cas $w_{t,r,\delta}$ représentera l'apport du train t vis-à-vis de cette unité.

$$max\ z_{Pref} = \sum_{t \in T, i \in I_t, r \in R_{t,i}, \delta \in \Delta_t} w_{t,r,\delta} \frac{1}{|I_t|} x_{t,r,\delta} \tag{3.4}$$

Là encore, il n'y a pas de priorité fixée a priori entre cet objectif et ceux de saturation (équation 3.3).

3.2.2 Contraintes du problème

Le choix des valeurs à affecter à chacune des variables doit respecter plusieurs contraintes afin de vérifier que la solution obtenue correspond bien à un routage réalisable.

3.2.2.1 Unicité du parcours d'un train

Une première famille de contraintes permet de s'assurer qu'un parcours unique est choisi pour chacun des trains sur une même partition (équation 3.5). En effet, plusieurs variables du modèle correspondent aux différents itinéraires et décalages temporels de ce train. Une seule de ces variables peut être sélectionnée dans une solution.

$$\sum_{r \in R_{t,i}, \delta \in \Delta_t} x_{t,r,\delta} \leq 1, \forall t \in T, \forall i \in I_t \tag{3.5}$$

3.2.2.2 Utilisation des zones

Une deuxième famille permet de s'assurer du respect des règles de sécurité liées à l'utilisation des zones (équation 3.6). La plupart des zones sont des ressources unaires. Ainsi, deux trains différents ne peuvent pas utiliser en même temps une même zone.

$$\begin{aligned} x_{t,r,\delta} + x_{t',r',\delta'} \leq 1, \forall t, t' \in T^2, t \neq t', \forall i, i' \in I_t \times I_{t'}, \forall r, r' \in R_{t,i} \times R_{t',i'}, \forall \delta, \delta' \in \Delta_t \times \Delta_{t'} \\ , \exists z \in Zones, [h^a_{z,t,r,\delta}, h^d_{z,t,r,\delta}] \cap [h^a_{z,t',r',\delta'}, h^d_{z,t',r',\delta'}] \neq \emptyset \end{aligned} \tag{3.6}$$

Le diagramme de Gantt de la figure 3.2 présente le passage de deux trains t1 et t2 sur quatres zones (z1, z2, z3 et z4). Ces deux trains sont en conflit puisqu'ils utilisent la zone z3 au même moment ($[h^a_{z3,t1,r,\delta}, h^d_{z3,t1,r,\delta}] \cap [h^a_{z3,t2,r',\delta'}, h^d_{z,t2,r',\delta'}] = [45, 60] \neq \emptyset$).

Pour les problèmes cadencés, cette vérification doit être faite pour les cycles précédent et suivant des trains cadencés (équation 3.7).

$$\begin{aligned} x_{t,r,\delta} + x_{t',r',\delta'} \leq 1, \forall t, t' \in T \times T_{Cycle}, t \neq t', \forall i, i' \in I_t \times I_{t'} \\ , \forall r, r' \in R_{t,i} \times R_{t',i'}, \forall \delta, \delta' \in \Delta_t \times \Delta_{t'}, \exists z \in Zones \\ , [h^a_{z,t,r,\delta}, h^d_{z,t,r,\delta}] \cap [h^a_{z,t',r',\delta'} + (fin - debut), h^d_{z,t',r',\delta'} + (fin - debut)] \neq \emptyset \\ \text{ou } [h^a_{z,t,r,\delta}, h^d_{z,t,r,\delta}] \cap [h^a_{z,t',r',\delta'} - (fin - debut), h^d_{z,t',r',\delta'} - (fin - debut)] \neq \emptyset \end{aligned} \tag{3.7}$$

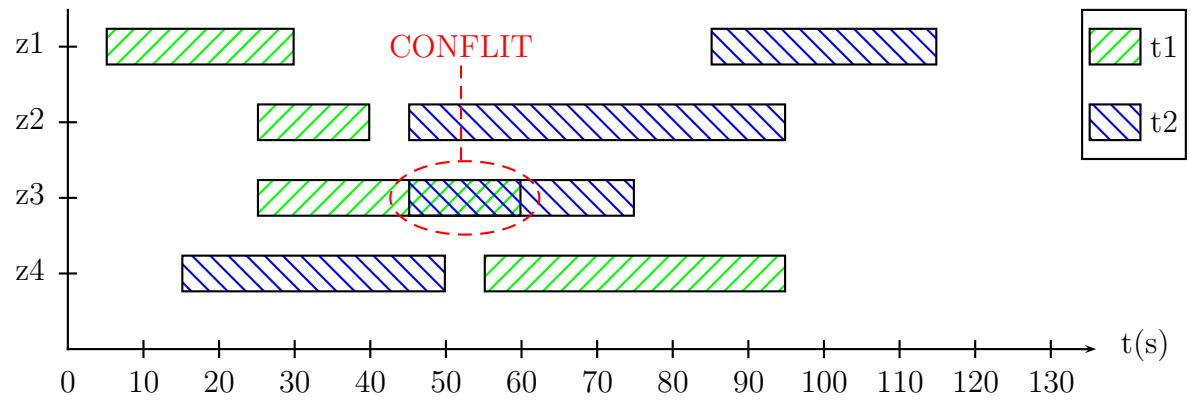


FIG. 3.2 – Conflit entre deux trains sur une zone

Le diagramme de Gantt de la figure 3.3 présente un exemple de train cadencé sur un horizon temporel d'une heure. L'utilisation des zones par ce train y est représentée en trait plein pour l'horizon temporel de référence et en trait discontinu pour les cycles précédent et suivant.

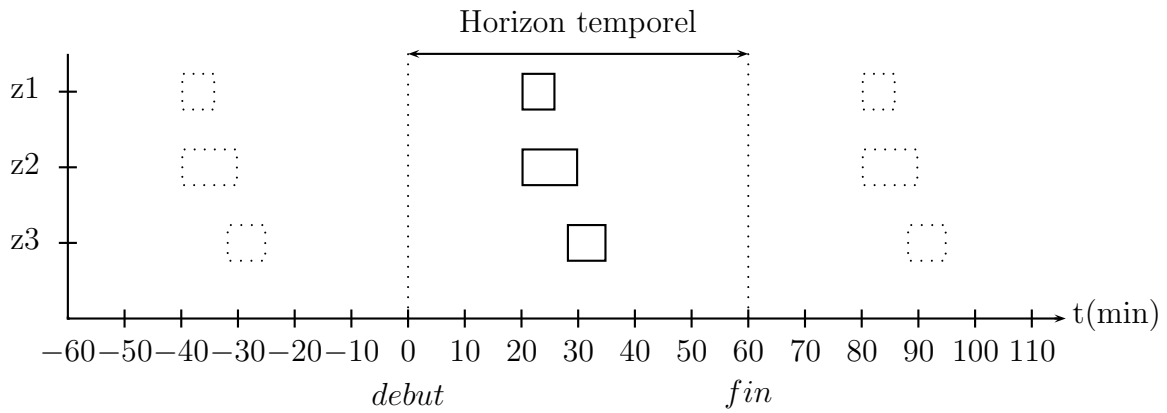


FIG. 3.3 – Cadencement d'un train

3.2.2.3 Cas particulier des quais

Certains quais présentent la particularité de pouvoir accueillir plusieurs trains en même temps. Ce cas de figure correspond à des zones à ressource n-aire, la capacité de ces zones dépendant en pratique de la longueur du quai et de celle des trains. Dans notre cas, nous ne prenons pas en compte les différences de longueur entre les trains (voir l'hypothèse 3.3). Ainsi, une zone de capacité n sera remplacée par n zones fictives successives de capacité unaire. L'arrêt d'un train sera alors possible sur chacune de ces zones. Ce train utilisera

alors cette zone pendant tout le temps de son arrêt. Les contraintes concernant l'utilisation de ces zones sont donc celles définies dans la section 3.2.2.2.

Hypothèse 3.3
La place occupée dans un quai est uniquement considérée en nombre de trains pouvant être accueillis en même temps sur ce quai, et non pas en fonction de la longueur réelle de ces trains.

De plus, afin de gérer l'ordre d'arrivée et de départ des trains sur ce quai (le dépassement de train par un autre n'étant pas possible sur le quai), le train utilisera aussi pendant une durée forfaitaire (dans notre cas nous avons considéré une durée d'une seconde) les autres zones fictives qu'il traversera. La date de début d'utilisation de toutes les zones traversées lors de l'arrivée sera la date de début d'utilisation de la zone d'arrêt, et la date de fin d'utilisation de toutes les zones traversées lors du départ sera la date de fin d'utilisation de la zone d'arrêt. Il faut distinguer deux cas :

- pour les quais au passage, le train traverse les zones précédant la zone d'arrêt lors de son arrivée et les zones suivantes lors de son départ. On obtient ainsi un comportement de type FIFO (premier entrée, premier sorti). La figure 3.4 représente le diagramme de Gantt d'un exemple dans lequel un train s'arrête sur la deuxième des trois zones fictives (z1, z2 et z3) d'un quai au passage. Dans ce cas, il traversera la zone z1 lors de son arrivée et la zone z3 lors de son départ.
- pour les quais en cul-de-sac, le train traverse les zones précédant la zone d'arrêt lors de son arrivée et retraverse les même zones lors de son départ (dans ce cas, certaines zones peuvent ne pas être traversées). On obtient ainsi un comportement de type LIFO (dernier entrée, premier sorti). La figure 3.5 représente le diagramme de Gantt d'un exemple dans lequel un train s'arrête sur la dernière des trois zones fictives (z1, z2 et z3) d'un quai en cul-de-sac. Dans ce cas, il traversera les zones z1 et z2 lors de son arrivée et de son départ.

3.2.2.4 Compatibilité des partitions

Une troisième famille permet de s'assurer du respect de la compatibilité physique des partitions successives des parcours d'un même train (équation 3.8). Il faut que l'itinéraire choisi pour une partition du parcours d'un train permette d'accéder à l'itinéraire choisi pour la partition suivante.

$$\begin{aligned} x_{t,r,\delta} + x_{t,r',\delta'} \leq 1, \forall t \in T, \forall i \in \{1, \ldots, |I_t| - 1\}, \forall \delta, \delta' \in \Delta_t^2 \\ , \forall r, r' \in R_{t,i} \times R_{t,i+1}, r' \notin Comp_{t,i,r} \end{aligned} \tag{3.8}$$

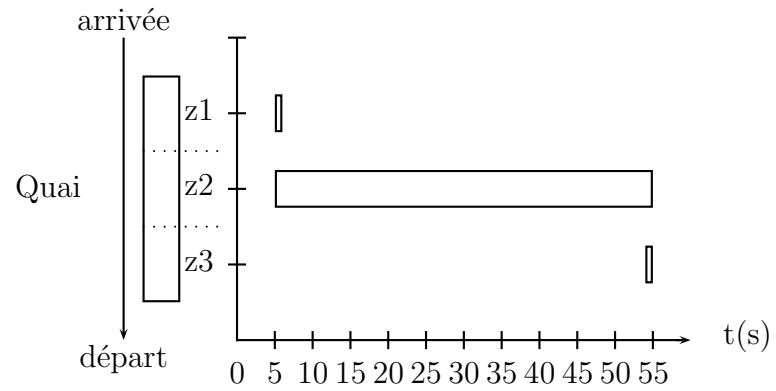


FIG. 3.4 – Quai au passage

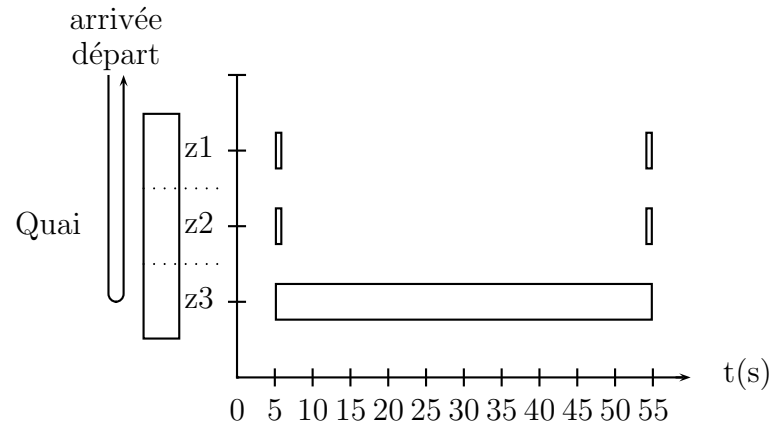


FIG. 3.5 – Quai en cul-de-sac

3.2.2.5 Concordance temporelle des partitions successives

Une quatrième famille permet de la même manière de s'assurer de la concordance temporelle des partitions de parcours successives d'un même train (équation 3.9). Ainsi, un train ne peut pas parcourir deux partitions successives avec un décalage temporel différent. Cela reviendrait soit à faire disparaître ce train de l'infrastructure durant cette période ($\delta_{i'} > \delta_i$), soit à le considérer comme étant sur les deux partitions en même temps ($\delta_{i'} < \delta_i$).

$$\begin{aligned} x_{t,r,\delta} + x_{t,r',\delta'} \leq 1, \forall t \in T, \forall i \in \{1, \ldots, |I_t| - 1\} \\ , \forall r, r' \in R_{t,i} \times R_{t,i+1}, \forall \delta, \delta' \in \Delta_t^2, \delta'_i \neq \delta_i \end{aligned} \tag{3.9}$$

3.2.2.6 Temps minimum pour les correspondances

Une cinquième famille permet de s'assurer du respect des temps minimums requis pour les correspondances (équation 3.10).

$$\begin{aligned} x_{t,r,\delta} + x_{t',r',\delta'} \leq 1, \forall t \in T, \forall t' \in T_{Fais}, t \neq t', \forall i, i' \in I_t \times I_{t'}, i \text{ et } i' \text{ quais} \\ , \forall r, r' \in R_{t,i} \times R_{t',i'}, \forall \delta, \delta' \in \Delta_t \times \Delta_{t'} \\ , (corr_{t,r,t',r'} > 0 \text{ et } (H_{t'}^{i'} + \delta'_{i'}) - (H_t^{i-1} + \delta_{i-1}) < corr_{t,r,t',r'}) \\ \text{ou } (corr_{t',r',t,r} > 0 \text{ et } (H_t^i + \delta_i) - (H_{t'}^{i'-1} + \delta'_{i'-1}) < corr_{t',r',t,r}) \end{aligned} \tag{3.10}$$

Nous avons décidé de ne pas permettre de demander une correspondance entre deux trains saturants dans notre modèle. En effet, les futurs utilisateurs de ce modèle estiment que cela ne présente pas d'intérêt en pratique. Dans le cas où une correspondance impliquerait un train t traversant le nœud dans sa première (resp. dernière) partition, on considérera que $H_t^0 = debut$ (resp. $H_t^{|I_t|} = fin$) et que $\delta_0 = 0$ (resp. $\delta_{|I_t|} = 0), \forall \delta \in \Delta_t$.

Le diagramme de Gantt de la figure 3.6 présente un exemple de la correspondance d'un train t1 vers un train t2. Ce cas de figure présente deux trains dont les parcours sont divisés en trois partitions ($|I_{t1}| = |I_{t2}| = 3$), la deuxième partition correspondant à l'arrêt au quai. Les variables sélectionnées dans cet exemple correspondent à un décalage temporel nul pour les deux trains ($\delta \in \Delta_{t1}$ (resp Δ_{t2}), $\delta = (0, 0)$). Le temps disponible pour la correspondance est donc dans ce cas égal au temps séparant l'arrivée au quai du premier train du départ du quai du second ($H_{t2}^2 - H_{t1}^1$). Afin de respecter la contrainte, ce temps disponible doit être supérieur ou égal au temps minimum requis pour cette correspondance étant donné les itinéraires choisis.

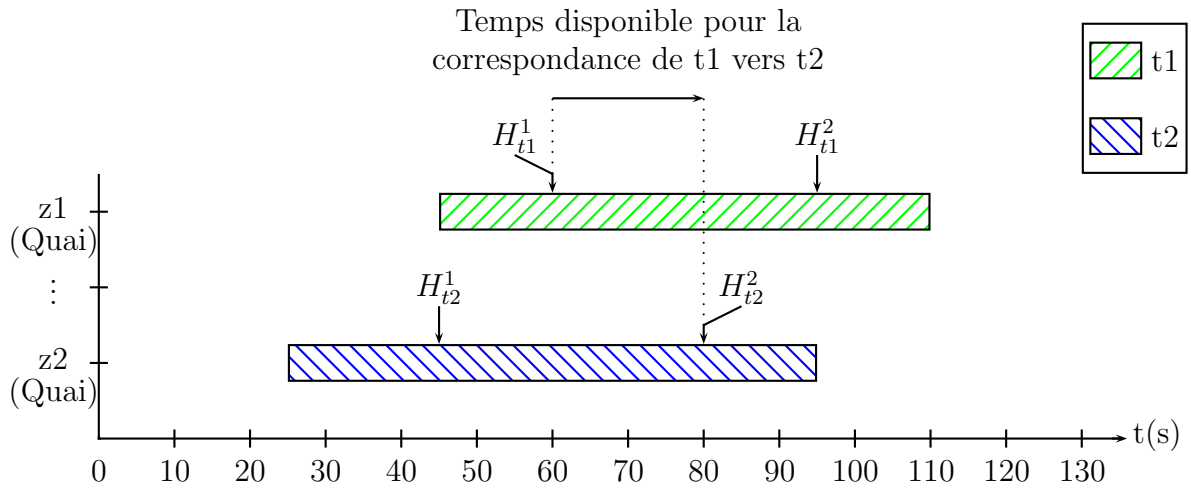


FIG. 3.6 – Correspondance entre deux trains

3.2.2.7 Couplage/Découplage de trains

Une sixième famille concerne le couplage et le découplage de certains trains. Lorsqu'un train «disparaît» de l'infrastructure en se couplant avec un autre train (équation 3.11), ces deux trains doivent être arrêtés sur deux zones fictives successives (et donc contigües) d'un même quai. De plus, la date d'arrivée au quai du train couplé doit être postérieure à celle du train auquel il se couple, et suffisamment antérieure à sa date de départ pour laisser le temps nécessaire au couplage. Enfin, la date de disparition du train couplé doit être identique à la date de départ du quai de l'autre train (le couplage étant considéré comme effectif au moment du départ du train).

$$\begin{aligned} x_{t,r,\delta} + x_{t',r',\delta'} \leq 1, & \forall t \in T_{Coupl}, t' = f_{Coupl}(t), i' = f'_{Coupl}(t) \\ & , \forall r, r' \in R_{t,|I_t|} \times R_{t',i'}, \forall \delta, \delta' \in \Delta_t \times \Delta_{t'} \\ & , (r, r' \text{ zones de quai fictives non successives}) \\ & \text{ou } (H_t^{|I_t|-1} + \delta_{|I_t|-1} \notin [H_{t'}^{i'-1} + \delta_{i'-1}; H_{t'}^{i'} + \delta_{i'} - coupl_{t,t'}]) \\ & \text{ou } (H_t^{|I_t|} + \delta_{|I_t|} \neq H_{t'}^{i'} + \delta_{i'}) \end{aligned} \tag{3.11}$$

Le diagramme de Gantt de la figure 3.7 présente un exemple du couplage d'un train t1 à un train t2. Ce cas de figure présente deux trains dont les parcours sont divisés en deux partitions ($|I_{t1}| = |I_{t2}| = 2$), la deuxième partition correspondant à l'arrêt au quai et donc à la partition où a lieu le couplage ($f_{Coupl}(t1) = 2$). Les variables selectionnées dans cet exemple correspondent à un décalage temporel nul pour les deux trains ($\delta \in \Delta_{t1}$ (resp Δ_{t2}), $\delta = (0, 0)$). Le temps disponible pour le couplage est donc dans ce cas égal au temps séparant l'arrivée au quai du premier train du départ du quai du second ($H_{t2}^2 - H_{t1}^1$). Afin de respecter la contrainte, ce temps disponible doit être supérieur ou égal au temps minimum requis pour ce couplage ($coupl_{t1,t2}$).

De même, lorsqu'un train «apparaît» dans l'infrastructure en se découplant d'un autre train (équation 3.12), ces deux trains doivent être arrêtés sur deux zones fictives successives (et donc contigües) d'un même quai. De plus, la date de départ du quai du train découplé doit être antérieure à celle du train duquel il se découple, et suffisamment postérieure à sa date d'arrivée pour laisser le temps nécessaire au découplage. Enfin, la date d'apparition du train découplé doit être identique à la date d'arrivée au quai de l'autre train (le découplage étant considéré comme effectif au moment de l'arrivée du train).

$$\begin{aligned} x_{t,r,\delta} + x_{t',r',\delta'} \leq 1, & \forall t \in T_{Decoupl}, t' = f_{Decoupl}(t), i' = f'_{Decoupl}(t) \\ & , \forall r, r' \in R_{t,1} \times R_{t',i'}, \forall \delta, \delta' \in \Delta_t \times \Delta_{t'} \\ & , (r, r' \text{ zones de quai fictives non successives}) \\ & \text{ou } (H_t^1 + \delta_1 \notin [H_{t'}^{i'-1} + \delta_{i'-1} + decoupl_{t,t'}; H_{t'}^{i'} + \delta_{i'}]) \\ & \text{ou } (H_t^0 + \delta_0 \neq H_{t'}^{i'-1} + \delta_{i'-1}) \end{aligned} \tag{3.12}$$

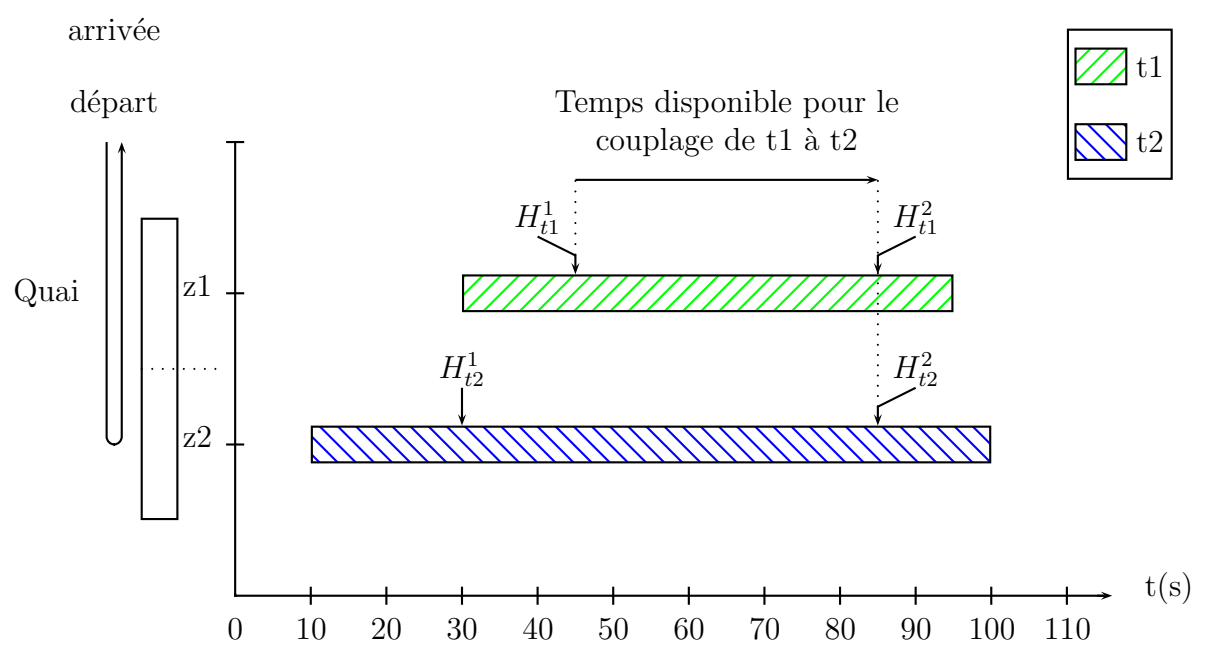


FIG. 3.7 – Couplage sur un quai en cul-de-sac

3.2.2.8 Généralisation au concept d'incompatibilités

Dans un souci de simplification des notations, on peut aussi considérer un ensemble Inc reprenant les paires de variables incompatibles dans les cinq familles de contraintes précédentes (équations 3.6, 3.7, 3.8, 3.9, 3.10, 3.11 et 3.12), ce qui nous donne la formulation suivante (équation 3.13) :

$$\begin{aligned} x_{t,r,\delta} + x_{t',r',\delta'} \leq 1, \forall t, t' \in T^2, \forall i, i' \in I_t \times I_{t'}, \forall r, r' \in R_{t,i} \times R_{t',i'} \\ , \forall \delta, \delta' \in \Delta_t \times \Delta_{t'}, (x_{t,r,\delta}, x_{t',r',\delta'}) \in Inc \end{aligned} \tag{3.13}$$

Dans la suite, nous utiliserons cette dernière notation.

3.2.2.9 Complétude des parcours

Enfin, une septième famille permet de garantir la complétude des parcours des trains sélectionnés dans la solution (équation 3.14). Pour cela, il faut avoir un nombre de parcours choisis égal pour tous les couples de partitions successives d'un même train.

$$\sum_{r \in R_{t,i}, \delta \in \Delta_t} x_{t,r,\delta} = \sum_{r \in R_{t,i+1}, \delta \in \Delta_t} x_{t,r,\delta}, \forall t \in T, \forall i \in \{1, \ldots, |I_t| - 1\} \tag{3.14}$$

De plus, les trains devant être couplés (resp. découplés) ne peuvent être routés que si le train auquel ils doivent se coupler (resp. découpler) est lui aussi routé (équations 3.15 (resp. 3.16)). Pour cela, il faut avoir un nombre de parcours choisis égal pour une partition quelconque de ces deux trains (par exemple la partition 1).

$$\sum_{r \in R_{t,1}, \delta \in \Delta_t} x_{t,r,\delta} = \sum_{r \in R_{t',1}, \delta \in \Delta_{t'}} x_{t',r,\delta}, \forall t \in T_{Coupl}, t' = f_{Coupl}(t) \tag{3.15}$$

$$\sum_{r \in R_{t,1}, \delta \in \Delta_t} x_{t,r,\delta} = \sum_{r \in R_{t',1}, \delta \in \Delta_{t'}} x_{t',r,\delta}, \forall t \in T_{Decoupl}, t' = f_{Decoupl}(t) \tag{3.16}$$

Il est aussi intéressant de noter que ces contraintes peuvent être supprimées pour les trains de la grille horaire initiale quand l'objectif est de faire circuler tous ces trains. La contrainte d'unicité des parcours (équation 3.5) limitant à un le nombre de variables d'un même train et d'une même partition dans une solution, une condition nécessaire pour obtenir $z_{Fais} = |T_{Fais}|$ est en effet de vérifier l'équation suivante (3.17) :

$$\sum_{r \in R_{t,i}, \delta \in \Delta_t} x_{t,r} = 1 = \sum_{r \in R_{t,i+1}, \delta \in \Delta_t} x_{t,r}, \forall t \in T_{Fais}, \forall i \in \{1, \ldots, |I_t| - 1\} \tag{3.17}$$

3.2.3 Modèle complet

L'ensemble des éléments indiqués dans les parties précédentes peut être rassemblé dans un modèle complet. Celui-ci est présenté dans l'équation 3.18. La structure principale de ce modèle, hormis les contraintes de complétude des parcours (équation 3.14), correspond à un problème de set packing (décrit dans la section 1.3, page 13). De plus, c'est un problème multiobjectif avec un objectif prioritaire (z_{Fais}). Il n'y a, par contre, pas d'ordre de priorité entre les autres objectifs (z^q_{Sat} et z_{Pref}).

Ainsi, par exemple, la figure 3.8 représente la matrice des contraintes obtenue dans un scénario à six trains. Dans un souci de clarté, seuls les éléments non nuls y ont été indiqués. Dans cet exemple, les parcours ne sont pas partitionnés ($|I_t| = 1, \forall t \in T$). Le nombre de parcours possibles varie de deux à trois suivant les trains, et l'horaire de référence est le seul autorisé pour chacun des trains ($|\Delta_t| = 1, \forall t \in T$).

La matrice est scindé en deux parties. La partie (a) correspond aux contraintes d'unicité de parcours (équation 3.5), alors que la partie (b) correspond aux contraintes d'incompatibilités (équation 3.13).

$$
\left[
\begin{array}{lll}
lex\ max(z_{Fais}, z^q_{Sat} | z_{Pref}) & & \\
sc \quad z_{Fais} = \displaystyle\sum_{t \in T_{Fais}, i \in I_t, r \in R_{t,i}, \delta \in \Delta_t} \frac{1}{|I_t|} x_{t,r,\delta} & & \\
\quad z^q_{Sat} = \displaystyle\sum_{t \in T^q_{Sat}, i \in I_t, r \in R_{t,i}, \delta \in \Delta_t} \frac{1}{|I_t|} x_{t,r,\delta} & & , \forall q \in Types \\
\quad z_{Pref} = \displaystyle\sum_{t \in T, i \in I_t, r \in R_{t,i}, \delta \in \Delta_t} w_{t,r,\delta} \frac{1}{|I_t|} x_{t,r,\delta} & & \\
\quad \displaystyle\sum_{r \in R_{t,i}, \delta \in \Delta_t} x_{t,r,\delta} \leq 1 & & , \forall t \in T, \forall i \in I_t \\
\quad x_{t,r,\delta} + x_{t',r',\delta'} \leq 1 & & , \forall t, t' \in T^2, \forall i, i' \in I_t \times I_{t'} \\
 & & , \forall r, r' \in R_{t,i} \times R_{t',i'} \\
 & & , \forall \delta, \delta' \in \Delta_t \times \Delta_{t'} \\
 & & , (x_{t,r,\delta}, x_{t',r',\delta'}) \in Inc \\
\quad \displaystyle\sum_{r \in R_{t,i}, \delta \in \Delta_t} x_{t,r,\delta} = \sum_{r \in R_{t,i+1}, \delta \in \Delta_t} x_{t,r,\delta} & & , \forall t \in T, \forall i \in \{1, \ldots, |I_t| - 1\} \\
\quad x_{t,r,\delta} \in \{0, 1\} & & , \forall t \in T, \forall i \in I_t, \forall r \in R_{t,i}, \delta \in \Delta_t
\end{array}
\right]
\tag{3.18}
$$

1	1	1												
			1	1										
					1	1								
							1	1	1					
										1	1			
												1	1	1

(a)

1			1											
1				1										
1					1									
1						1								
	1		1											
	1			1										
	1				1									
	1					1								
		1	1											
		1		1										
				1	1									
				1		1								
					1		1							
					1			1						
						1			1					
						1						1		
						1							1	
						1								1
							1			1				
								1		1				
											1	1		
											1		1	
											1			1

(b)

Fig. 3.8 – Exemple de matrice des contraintes

3.3 Évaluation de la stabilité du routage

Une fois obtenu un routage, il est important de pouvoir évaluer la stabilité de cette solution, et éventuellement de la comparer avec d'autres solutions (voir la section 2.3, page 33). Idéalement, la stabilité pourrait être considérée comme l'un des objectifs de notre problème d'optimisation. Malheureusement, nous ne disposons pas pour le moment d'un modèle linéaire permettant d'optimiser à la fois la faisabilité, la saturation et les préférences d'une grille horaire, et la stabilité du routage correspondant.

Notre calcul de stabilité se base sur les retards potentiellement générés par le retard initial de l'un des trains du routage. Le retard initial d'un train signifie que toutes les dates d'utilisations de zones de ce train sont décalées de ce retard initial ($h^a_{z,t,r,\delta}$ (resp. $h^d_{z,t,r,\delta}$) $\rightarrow$ $h^a_{z,t,r,\delta}$ (resp. $h^d_{z,t,r,\delta}$) $+ retardInitial, \forall z \in Zones$). Il ne s'agit pas là de ré-optimiser l'ensemble du routage, mais d'observer l'impact en terme de retard sur les autres trains. Nous considérons donc un calcul de retards basé sur des hypothèses de conservation des parcours, de l'ordre des trains et des temps d'arrêt (voir l'hypothèse 3.4).

Hypothèse 3.4
Lorsqu'un train est en retard par rapport à l'horaire initial prévu, l'ensemble des choix de parcours et d'ordre entre les différents trains fait dans le routage et les temps d'arrêt au quai prévus ne sont pas remis en cause.

Notre évaluation de la stabilité correspond à la somme des retards générés par chacun des trains du routage soumis à un retard initial, sur tous les autres trains. Elle correspond à celle proposée par Delorme et al. [DGR03a]. Il est important de noter que les retards générés peuvent l'être directement ou indirectement (effet «boule-de-neige»). Bien évidemment, cette évaluation dépendra fortement de la valeur du retard initial considéré. Il peut donc être intéressant de considérer non pas une mais plusieurs évaluations de stabilité pour des valeurs de retard initial différentes. Nous allons maintenant introduire quelques notations nécessaires à la compréhension du modèle d'évaluation de la stabilité.

X^* Ensemble des variables sélectionnées dans la solution à évaluer :

$$X^* = \{(t, i, r, \delta), t \in T, i \in I_t, r \in R_{t,i}, \delta \in \Delta_t, x_{t,r,\delta} = 1\}$$

Par définition, deux variables de cet ensemble sont compatibles ($(x1, x2) \notin Inc, \forall (x1, x2) \in X^*$) puisqu'elles font parties d'une même solution réalisable.

$retardInitial$ Retard initial considéré pour chacune des variables $x \in X^*$.

P — Ensemble des conflits potentiels directs entre deux variables de X^* :

$$P = \{((t,i,r,\delta),(t,i+1,r',\delta')) \in X^{*2}\}$$

$$\cup\{((t,i+1,r,\delta),(t,i,r',\delta')) \in X^{*2}\}$$

$$\cup\{((t,i,r,\delta),(t',i',r',\delta')) \in X^{*2}, t \neq t', corr_{t,r,t',r'} > 0 \text{ ou}$$

$$\exists d > 0, \exists z \in Zones, ([h^a_{z,t,r,\delta} + d, h^d_{z,t,r,\delta} + d] \cap [h^a_{z,t',r',\delta'}, h^d_{z,t',r',\delta'}] \neq \emptyset))\}$$

Il y a un conflit potentiel dans quatre cas de figure :
- entre chaque paire de variables correspondant à deux partitions successives d'un même train,
- entre deux trains devant être couplés ou découplés,
- entre chaque paire de variables ayant une correspondance entre elles,
- entre une paire de variables utilisant une zone commune lorsqu'il existe un retard strictement positif de la première variable provoquant un conflit sur l'utilisation de cette zone avec la deuxième variable.

$G(X^*, P)$ — Graphe orienté des conflits potentiels directs. Dans ce graphe, chaque sommet représente une des variables sélectionnées dans la solution à évaluer. Chaque paire de sommets est reliée par un arc orienté lorsque le retard du train associé au premier sommet peut provoquer un retard de celui associé au deuxième.

$w : P \rightarrow I\!N$ — Fonction de valuation des arcs du graphe orienté $G(X^*, P)$. La valuation w_e de l'arc e (équation 3.19) correspond à la valeur de retard du train associé au premier sommet au delà de laquelle il y a propagation à celui associé au deuxième sommet (c-à-d à la marge de temps disponible entre ces deux trains). Cette valuation est soit :
- nulle lorsque les deux sommets représentent un même train sur deux partitions successives, le retard du train sur l'une des partitions entrainant automatiquement son retard sur l'autre partition (de même pour les trains devant être couplés ou découplés),
- la plus grande valeur de retard possible pour le premier train sans générer un conflit sur une zone quelconque ou empêcher une correspondance avec le deuxième train.

$$w_e = \begin{cases} 0 \quad ,\forall((t,i,r,\delta),(t,i+1,r',\delta')) \in X^{*2} \\ \\ 0 \quad ,\forall((t,i+1,r,\delta),(t,i,r',\delta')) \in X^{*2} \\ \\ min\ d \geq 0, (\exists z \in Zones, [h^a_{z,t,r,\delta} + d + 1, h^d_{z,t,r,\delta} + d + 1] \cap [h^a_{z,t',r',\delta'}, h^d_{z,t',r',\delta'}] \neq \emptyset) \\ \qquad\qquad \text{ou } (corr_{t,r,t',r'} > 0 \text{ et } (H^{i'}_{t'} + \delta'_{i'}) - (H^{i-1}_t + \delta_{i-1} + d + 1) < corr_{t,r,t',r'}) \\ \quad ,\forall((t,i,r,\delta),(t',i',r',\delta')) \in X^{*2}, t \neq t' \end{cases} \tag{3.19}$$

À l'aide du graphe $G(X^*, P)$ et de la fonction de valuation w, on peut calculer l'ensemble des retards générés par le retard d'un des trains de la solution sur chacun des autres trains en prenant en compte les retards générés par des conflits directs et indirects.

La marge $marge_{x1,x2}$ de temps disponible entre les trains associés aux deux sommets $x1 = (t1, i1, r1, \delta1)$ et $x2 = (t2, i2, r2, \delta2)$ du graphe correspond au retard maximum possible pour la variable $x1$ sans générer de retard sur la variable $x2$. Elle peut être obtenue par un calcul de plus court chemin entre ces deux sommets (équation 3.20).

$$\left[\begin{array}{ll} marge_{x1,x2} = min & \sum_{i \in X^*, j \in \Gamma^-_G(i)} w_{i,j} x_{i,j} \\ sc & \sum_{i \in \Gamma^-_G(x2)} x_{i,j} = 1 \\ & \sum_{i \in \Gamma^-_G(j)} x_{i,j} - \sum_{i \in \Gamma^+_G(j)} x_{j,i} = 0, \forall j \in X^* \setminus \{x1, x2\} \\ & (x1, x2) \in X^*, x1 \neq x2 \end{array}\right] \tag{3.20}$$

Le calcul de cette marge correspond donc à la résolution d'un problème de plus court chemin (décrit dans la section 1.2, page 11) dans le graphe $G(X^*, P)$.

Une fois cette marge connue, le retard $retard_{x1,x2}$ généré par un retard initial de la variable $x1$ sur la variable $x2$ peut être calculé (équation 3.21) et notre évaluation de la stabilité (équation 3.22) peut en être déduite. Les différentes partitions d'un même train étant synchronisées par des arcs de valuations nulles, elles ont forcément le même retard. Nous pouvons donc ne considérer que les variables correspondant à la première partition du parcours du train pour ce calcul.

$$retard_{x1,x2} = \max(0, retardInitial - marge_{x1,x2}), \forall(x1, x2) \in X^{*2}, x1 \neq x2 \tag{3.21}$$

$$z_{Stabilite} = \sum_{x1=(t1,1,r1,\delta1)\in X^*} \left(\sum_{x2=(t2,1,r2,\delta2)\in X^*\setminus\{x1\}} retard_{x1,x2} \right) \tag{3.22}$$

De la même manière, on peut calculer la stabilité d'un routage pour différentes valeurs de retard initial (équation 3.23).

$$z_{Stabilite}^{retardInitial} = \sum_{x1=(t1,1,r1,\delta1)\in X^*} \left(\sum_{x2=(t2,1,r2,\delta2)\in X^*\setminus\{x1\}} \max\left(0, retardInitial - marge_{x1,x2}\right) \right) \tag{3.23}$$

Le graphe $G(X^*, P)$ ne prend pas en compte le cadencement éventuel de certains trains. Les retards ne sont calculés que pour les trains de l'horizon temporel considéré.

La figure 3.9 présente un exemple de graphe des conflits potentiels directs pour un routage contenant quatre trains (avec trois partitions pour les parcours des trains $t1$ et $t3$ et une seule pour ceux de $t2$ et $t4$). Les valeurs des itinéraires et des décalages temporels correspondant à chacune des variables ne sont pas indiquées. Dans cet exemple, pour un retard initial de 20 s, on obtient les retards suivants :

- $t1$ génère un retard de 10 s de $t4$ ($marge_{(t1,1,\ldots),(t4,1,\ldots)} = 10$),
- $t2$ génère un retard de 15 s de $t1$ ($marge_{(t2,1,\ldots),(t1,1,\ldots)} = 5$), de 15 s de $t3$ ($marge_{(t2,1,\ldots),(t3,1,\ldots)} = 5$) et de 5 s de $t4$ ($marge_{(t2,1,\ldots),(t4,1,\ldots)} = 15$),
- $t3$ ne génère aucun retard,
- $t4$ génère un retard de 10 s de $t3$ ($marge_{(t4,1,\ldots),(t3,1,\ldots)} = 10$).

Nous avons donc une évaluation de stabilité pour un retard initial de 20 s égale à 55 s ($z_{Stabilite}^{20} = 55$).

3.4 Conclusion

Dans ce chapitre, nous avons présenté le modèle que nous proposons pour le problème de l'évaluation et de l'analyse de la capacité d'infrastructures ferroviaires complexes. C'est un modèle linéaire multiobjectif en variables binaires. Sa structure principale correspond à celle de deux problèmes classiques de recherche opérationnelle appelés plus court chemin et set packing.

L'échelle considérée ici est celle d'un nœud ou d'une gare. Le principal avantage de cette modélisation est de considérer un niveau de détail suffisamment fin pour pouvoir obtenir des grilles horaires réalisables dans la pratique sur ce type d'infrastructure. De plus, elle intègre l'ensemble des questions se posant dans notre étude :

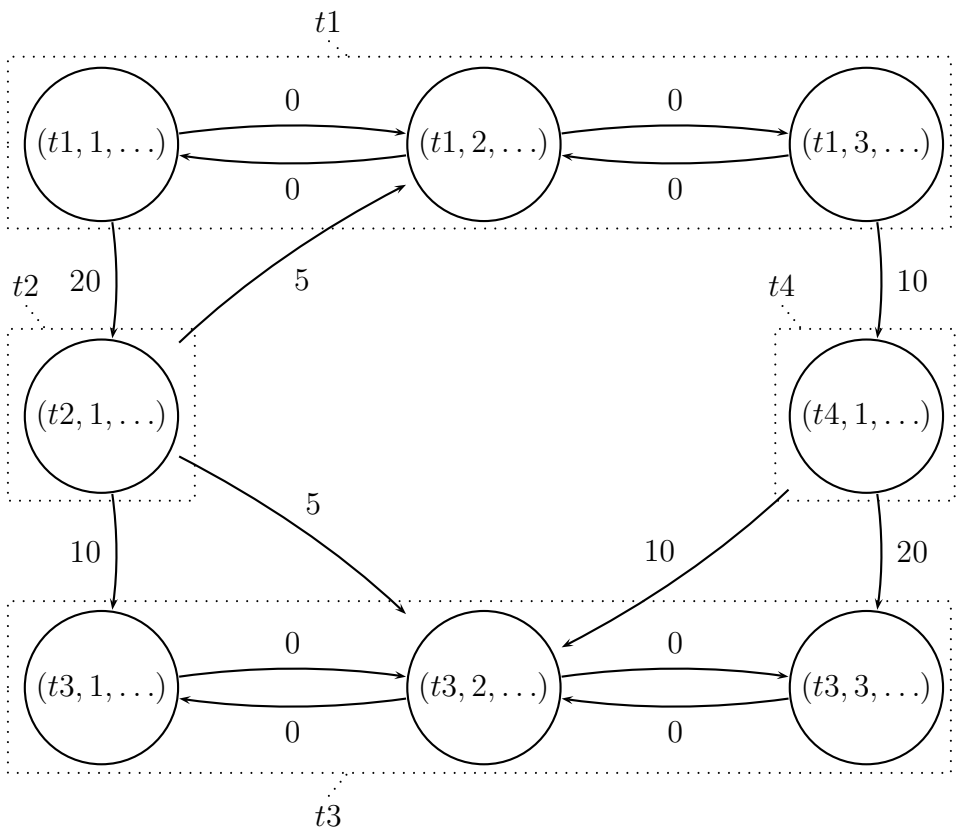


FIG. 3.9 – Graphe des conflits potentiels directs

- la vérification de la faisabilité du routage d'un ensemble de trains selon une grille horaire établie,
- la détermination du nombre maximum de trains pouvant circuler pour une configuration d'horaire donnée par saturation d'une grille horaire initiale à l'aide de listes saturantes non ordonnées permettant de considérer séparément différentes catégories de trains,
- l'optimisation d'une grille horaire en fonction de préférences émises par le décideur,
- et l'évaluation de la stabilité des grilles horaires générées.

En outre, notre modélisation permet de traiter des scénarios variés, incluant des correspondances, des couplages, des découplages et la présence de quais pouvant accueillir plusieurs trains. Enfin, les seules hypothèses simplificatrices posées sont liées à la discrétisation du problème, et ne nous semblent pas trop réductrices dans le cadre d'un problème de gestion prévisionnelle. En contrepartie, la plupart des notations et des équations définissant ce modèle s'avèrent relativement complexes.

Dans la suite, nous allons nous intéresser à la résolution de ce modèle. Nous avons vu au chapitre 1 que le problème de plus court chemin était facile à résoudre. Au contraire, celui

de set packing est NP-difficile. Nous nous focaliserons donc sur les différentes méthodes de résolution, exactes et approchées, que nous proposons pour le problème de set packing.

Chapitre 4

Résolution du problème de set packing

Dans le chapitre précédent, nous avons décrit la modélisation retenue pour notre problème d'évaluation de la capacité d'infrastructures ferroviaires. Cette modélisation fait apparaître deux structures principales correspondants aux problèmes de plus court chemin et de set packing. Nous avons vu au chapitre 1 que la résolution du premier ne présentait pas de difficulté avec les algorithmes existants (section 1.2.2, page 13), mais que celle du deuxième pouvait se révéler difficile (section 1.3.3, page 19). Dans ce chapitre, nous allons donc décrire les méthodes de résolution (exactes et approchées) du problème de set packing que nous avons utilisées dans les cas mono et biobjectifs.

4.1 Résolution exacte et bornes

Afin de résoudre le problème de set packing dans le cas mono-objectif de manière exacte, voire obtenir une borne supérieure lorsque la résolution exacte n'est pas possible dans le temps imparti, nous avons utilisé le logiciel commercial Cplex [ILO02]. De plus, plusieurs pré-traitements, développés pour améliorer les performances, sont présentés. Le schéma global de résolution est ensuite décrit. Enfin, un algorithme de recherche dichotomique, utilisé pour déterminer la frontière efficace dans le cas biobjectif, est indiqué.

4.1.1 Pré-traitements

4.1.1.1 Cliques propres au problème de capacité

Nous avons vu au chapitre 1 (section 1.3.3, page 19) que les cliques d'un problème de set packing correspondaient à des inégalités valides pour ce problème (et même à des facettes lorsqu'elles sont maximales). La détermination de cliques (même non maximales) permet

ainsi d'améliorer la valeur de la borne supérieure obtenue par relaxation linéaire. Dans cette optique, la formulation du modèle peut être modifiée. En effet, les contraintes entre paires de variables incompatibles (équation 3.13, page 60) peuvent être remplacées par les cliques suivantes (équation 4.1) :

$$x_{t,r,\delta} + \sum_{\substack{r' \in R_{t',i'}, \delta' \in \Delta_{t'}, \\ (x_{t,r,\delta}, x_{t',r',\delta'}) \in Inc}} x_{t',r',\delta'} \leq 1, \forall t, t' \in T^2, \forall i, i' \in I_t \times I_{t'}, \forall r \in R_{t,i}, \forall \delta \in \Delta_t \tag{4.1}$$

En effet, les variables $x_{t',r',\delta'}$ forment une clique en raison de la contrainte d'unicité de parcours (équation 3.5, page 54). La variable $x_{t,r,\delta}$ étant en conflit avec chacune d'elles, elle forme une clique avec elles.

Ce principe peut être étendu aux variables, notées $x_{t,r'',\delta''}$, faisant partie de la même clique d'unicité de parcours que la variable $x_{t,r,\delta}$, à condition qu'elles soient aussi en conflit avec toutes les variables $x_{t',r',\delta'}$. Nous obtenons ainsi les cliques suivantes (équation 4.2) :

$$x_{t,r,\delta} + \sum_{\substack{r' \in R_{t',i'}, \delta' \in \Delta_{t'}, \\ (x_{t,r,\delta}, x_{t',r',\delta'}) \in Inc}} x_{t',r',\delta'} + \sum_{\substack{r'' \in R_{t,i}, \delta'' \in \Delta_t, (r'',\delta'') \neq (r,\delta), \forall r' \in R_{t',i'}, \forall \delta' \in \Delta_{t'}, \\ ((x_{t,r,\delta}, x_{t',r',\delta'}) \in Inc) \Rightarrow ((x_{t,r'',\delta''}, x_{t',r',\delta'}) \in Inc)}} x_{t,r'',\delta''} \leq 1$$
$$, \forall t, t' \in T^2, \forall i, i' \in I_t \times I_{t'}, \forall r \in R_{t,i}, \forall \delta \in \Delta_t \tag{4.2}$$

En fait, ces cliques font partie d'une famille de cliques spécifique au problème de capacité (voir l'équation 4.3). Cette famille regroupe toutes les cliques concernant les variables associées à un couple de trains (t, t'). Il n'est cependant pas possible de générer toutes les cliques de cette famille car leur nombre (et le temps nécessaire pour les obtenir) croît de manière exponentielle par rapport à la taille du problème. Nous nous contentons donc de générer celles définies par l'équation 4.2.

$$\sum_{r \in A_r, \delta \in A_\delta} x_{t,r,\delta} + \sum_{\substack{r' \in R_{t',i'}, \delta' \in \Delta_{t'}, \forall r \in A_r \\ , \forall \delta \in A_\delta, (x_{t,r,\delta}, x_{t',r',\delta'}) \in Inc}} x_{t',r',\delta'} \leq 1, \forall t, t' \in T^2, \forall i, i' \in I_t \times I_{t'}$$
$$, \forall A_r \subseteq R_{t,i}, \forall A_\delta \subseteq \Delta_t \tag{4.3}$$

Notons aussi que la famille définie par l'équation 4.3 est une généralisation de la famille de cliques présentée par Zwaneveld [Zwa97] (voir l'équation 4.4). En effet celle-ci ne concernait

que les couples de trains sur un même itinéraire.

$$\sum_{r \in A_r, \delta \in A_\delta} x_{t,r,\delta} + \sum_{\substack{r' \in R_{t',i}, \delta' \in \Delta_{t'}, \forall r \in A_r \\ , \forall \delta \in A_\delta, (x_{t,r,\delta}, x_{t',r',\delta'}) \in Inc}} x_{t',r',\delta'} \leq 1, \forall t, t' \in T^2, \forall i \in I_t \cap I_{t'} \\ , \forall A_r \subseteq R_{t,i}, \forall A_\delta \subseteq \Delta_t \tag{4.4}$$

Nous avons considéré cette généralisation car les conflits entre trains sur des partitions différentes (par exemple entre la route d'entrée d'un train et la route de sortie d'un autre train) sont très nombreux dans les instances sur lesquelles nous avons travaillé. Ceci est notamment dû au grand nombre de voies banalisées dans les infrastructures que nous avons étudiées.

4.1.1.2 Calcul de cliques maximales

Cependant, rien ne garantit que les cliques définies dans la section précédente soient maximales[1]. Nous avons donc développé l'algorithme 4.1 afin de compléter les contraintes du problème de sorte qu'elles correspondent toutes à des cliques maximales.

```
pour j ∈ J faire
    | I_0 ← {i ∈ I, t_{i,j} = 0}
    | pour i ∈ I_0 faire
    |     | si (∀k ∈ I \ I_0, ∃j' ∈ J, (t_{k,j'} = 1) et (t_{i,j'} = 1)) alors
    |     |     | t_{i,j} ← 1
    |     |     | I_0 ← I_0 \ {i}
    |     | finSi
    | finPour
finPour
```

Alg. 4.1 – Calcul de cliques maximales

Cet algorithme complète chacune des contraintes en y ajoutant des variables en conflit avec toutes celles déjà présentes dans la contrainte. C'est un algorithme de type glouton. Il effectue ainsi cette opération successivement pour toutes les variables. Notons qu'il n'est pas spécifique au problème de capacité et peut être utilisé pour tout problème de set packing. Sa complexité théorique est en $O(n^2m)$ dans le pire des cas.

[1]Une clique est maximale si aucune variable ne peut y être ajoutée pour former une nouvelle clique. Cette notion ne doit pas être confondue avec celle de clique maximum (décrite dans la section 1.3.2.4, page 18) dont la détermination est un problème NP-difficile.

4.1.1.3 Test de dominance entre variables

Une autre catégorie de pré-traitement permet de diminuer la taille du problème. Le test de dominance entre variables (voir algorithme 4.2) compare chaque paire de variables. Nous considérons qu'une variable i_1 est dominée par une autre variable i_2 si :

- i_1 et i_2 appartiennent à une même contrainte (c-à-d qu'elles ne peuvent pas être toutes les deux égales à 1 dans une solution réalisable),
- toutes les variables appartenant à une même contrainte que i_2 appartiennent aussi à une contrainte commune avec i_1,
- la valeur de l'élément i_1 est inférieure ou égale à celle de l'élément i_2.

Une telle variable i_1 peut être supprimée du problème (c-à-d fixée à 0) sans en changer la solution optimale. De plus, les contraintes n'ayant plus qu'un seul élément non nul après la suppression de la variable i_1 peuvent aussi être supprimées. La complexité théorique de cet algorithme est en $O(n^4m)$ dans le pire des cas. Cependant, en l'occurence le pire des cas correspondrait à un problème pour lequel toutes les variables sauf une seraient dominées, la résolution exacte du problème réduit devenant ainsi évidente.

```
répéter
    | I_t ← I
    | pour i_1 ∈ I faire
    |     | si ∃i_2 ∈ I \ {i_1}, (∃j ∈ J, t_{i_1,j} = 1 et t_{i_2,j} = 1) et (∀i_3 ∈ I \ {i_1, i_2},
    |     |        (∃j ∈ J, t_{i_2,j} = 1 et t_{i_3,j} = 1) ⇒ (∃j ∈ J, t_{i_1,j} = 1 et t_{i_3,j} = 1))
    |     |                                                                  et (c_{i_1} ≤ c_{i_2}) alors
    |     |     | I ← I \ {i_1}
    |     |     | J ← J \ {j : t_{i_1,j} = 1 et ∑_{i∈I} t_{i,j} ≤ 2}
    |     |     | sortie(finPour)
    |     | finSi
    | finPour
jusqu'à I_t = I
```

Alg. 4.2 – Test de dominance entre variables

Ce test de dominance correspond à celui proposé par Zwaneveld [Zwa97] sous le nom de «Node dominance» (voir section 2.4.1.2, page 39). Parmi les autres pré-traitements qu'ils avaient proposés, nous avons écarté celui nommé «Detour routes» car il n'est pas valide pour notre problème. En effet, ce type de routes peut, par exemple, permettre l'évitement de deux trains devant se croiser, ou le dépassement d'un train par un autre. De plus, nous n'avons pas retenu les autres pré-traitements en raison de leur faible efficacité. Des expérimentations préliminaires ont ainsi montré que le test appelé «Combining nodes» n'avait aucun effet sur les instances que nous avons considérées, car il n'était pas possible d'identifier des couples de variables vérifiant cette propriété.

4.1.1.4 Test de dominance entre contraintes

Enfin, certaines contraintes dominées ou redondantes peuvent être retirées du problème (voir l'algorithme 4.3). Dans le cas du set packing, une contrainte j est considérée comme dominée (ou redondante) par une contrainte j' si toutes les variables de la contrainte j sont aussi présentes dans la contrainte j'. Ces suppressions ne modifient en rien la structure du problème considéré, mais permettent de diminuer sa taille.

pour $j \in J$ faire
  si $\exists j' \in J \setminus \{j\}, \{i \in I, t_{i,j} = 1\} \subseteq \{i \in I, t_{i,j'} = 1\}$ alors
    $J \leftarrow J \setminus \{j\}$
  finSi
finPour

Alg. 4.3 – Test de dominance entre contraintes

La complexité théorique de cet algorithme est en $O(nm^2)$ dans le pire des cas.

4.1.2 Résolution exacte dans le cas mono-objectif

Nous allons maintenant voir comment ces pré-traitements ont été utilisés. Pour déterminer la solution optimale (ou au moins une borne supérieure) pour le problème de set packing, nous avons considéré la séquence suivante composée de sept étapes :

1. Reformulation des contraintes entre paires de variables incompatibles (équation 4.2),
2. Suppression des contraintes dominées ou redondantes (algorithme 4.3),
3. Suppression des variables dominées (algorithme 4.2),
4. Suppression des contraintes dominées ou redondantes,
5. Complétion des contraintes pour en faire des cliques maximales (algorithme 4.1),
6. Suppression des contraintes dominées ou redondantes,
7. Résolution du problème réduit avec le solveur en nombre entier de Cplex 8.0 [ILO02].

Ainsi, le calcul des cliques liées au problème ferroviaire s'effectue au début car il n'est possible que tant que la structure des contraintes n'a pas été modifiée. Ensuite, le test de dominance entre paires de variables est utilisé avant l'heuristique de calcul de cliques maximales. En effet, si le calcul de nouvelles cliques ne peut pas permettre de détecter une variable dominée supplémentaire, la suppression de certaines variables peut rendre des cliques non maximales. Il ne servirait à rien d'ajouter des variables dans certaines contraintes pour les supprimer à l'étape suivante. Enfin, après chacun de ces pré-traitements, les contraintes devenues inutiles sont supprimées. Si cette suppression n'a aucun impact sur les variables supprimées grâce au test de dominance, elle peut théoriquement supprimer des contraintes qui, une fois

complétées pour en faire des cliques maximales, auraient pu devenir des contraintes non dominées. Cependant, nous avons constaté que cette situation est rare en pratique.

Un exemple de ce processus est présenté dans les figures 4.1, 4.2 et 4.3. Il illustre les étapes 1 à 6 de la résolution d'un problème ferroviaire avec une fonction économique dont les coefficients sont tous égaux à 1.

La figure 4.1, composée de deux matrices, correspond à l'étape 1. La matrice de gauche est la matrice initiale des contraintes du problème ; elle est identique à celle présentée dans l'exemple de la section 3.2.3 (page 61). Pour ne pas surcharger la figure, seuls les coefficients égaux à 1 sont indiqués, la case étant laissée vide lorsque le coefficient est nul. Chaque ligne de la matrice correspond à une contrainte. Notons la distinction de deux blocs de contraintes : celles d'unicité de parcours (définies par l'équation 3.5, page 54) puis celles d'incompatibilités (définies par l'équation 3.13, page 60). Dans la matrice de droite, les contraintes du deuxième bloc sont reformulées en utilisant les cliques ferroviaires (définies par l'équation 4.2, page 72). Ces dernières sont indiquées en vis-à-vis de celles de la matrice de gauche. Lorsqu'une contrainte dans la matrice de droite correspond à plusieurs contraintes (celles-ci sont forcément successives), elle est alignée sur la première contrainte et les lignes correspondantes suivantes sont laissées blanches.

La figure 4.2 illustre, elle, l'étape 3. La matrice de gauche représente l'état de la matrice après la suppression des contraintes redondantes (étape 2). Les lignes blanches ont aussi été supprimées. La matrice de droite est la matrice obtenue en supprimant les variables dominées (algorithme 4.2). Les colonnes de ces variables sont marquées par des tirets . De plus, les contraintes devenues inutiles (celles ne contenant plus qu'un seul élément à 1) sont aussi enlevées (là encore, une ligne blanche est mise à leur place pour conserver l'alignement). Ainsi, lorsqu'une colonne est entièrement blanche, cela signifie que la variable n'est pas supprimée mais n'intervient plus dans aucune contrainte.

Enfin, la figure 4.3 représente la matrice obtenue à la fin de l'étape 4. Dans cet exemple, la matrice obtenue à l'issue de l'étape 6 est strictement identique puisque les cliques sont déjà maximales. C'est donc cette matrice qui devra être résolue par Cplex.

4.1.3 Recherche dichotomique dans le cas biobjectif

Dans le cas biobjectif, tous les pré-traitements indiqués dans la section 4.1.1 restent valides. La seule différence porte sur le test de dominance entre paires de variables (algorithme 4.2) dans lequel la condition ($c_{i_1} \leq c_{i_2}$) doit être remplacée par la condition ($c^1_{i_1} \leq c^1_{i_2}$ et $c^2_{i_1} \leq c^2_{i_2}$). Pour déterminer la frontière efficace du problème ainsi réduit, nous avons utilisé un schéma algorithmique simple, basé sur une recherche dichotomique, qui a servi pour des études préliminaires présentées par Degoutin et Gandibleux [Deg02, DG02]. Le logiciel Cplex [ILO02] est alors utilisé pour résoudre le problème paramétrique mono-objectif suivant (voir

1	1	1											
			1	1									
					1	1							
							1	1	1				
										1	1		
												1	1	1

1			1										
1				1									
1					1								
1						1							
	1		1										
	1			1									
	1				1								
	1					1							
		1	1										
		1		1									
				1	1								
				1		1							
					1		1						
					1			1					
						1			1				
						1						1	
						1							1
							1			1			
								1			1		
										1	1		
										1		1	
										1			1

1	1	1											
			1	1									
					1	1							
							1	1	1				
										1	1		
												1	1	1

1	1	1	1	1									

1	1				1	1							

1	1	1	1	1									

1	1				1	1							

1	1	1	1	1									

				1	1	1							

					1		1	1					

						1			1				
						1						1	1	1

							1	1		1				
							1	1		1				
											1	1	1	1

Fig. 4.1 – Cliques propres au problème de capacité

							1	1	1					
										1	1			

1	1				1	1								
1	1	1	1	1										
				1	1	1								
					1		1	1						
						1			1					
						1						1	1	1
							1	1		1				
											1	1	1	1

-	-	-		-			-	1	1			-	-	
-	-	-		-			-			1	1	-	-	

-	-	-		-	1	1	-					-	-	

-	-	-		-	1	1	-					-	-	
-	-	-		-	1		-	1				-	-	
-	-	-		-		1	-		1			-	-	
-	-	-		-		1	-					-	-	1
-	-	-		-			-	1		1		-	-	
-	-	-		-			-				1	-	-	1

Fig. 4.2 – Test de dominance entre variables

			1	1			
					1	1	

	1	1					
	1		1				
		1		1			
		1					1
			1		1		
						1	1

Fig. 4.3 – Instance réduite

équation 4.5) :

$$\left[\begin{array}{lll} Max\ z_\lambda(X) = & \sum_{q=1}^{2} \lambda_q z^q(X) & \\ sc & X \in \mathcal{D} & \\ & c^1 X > z^1(X^{(B)}) & \\ & c^2 X > z^2(X^{(A)}) & \\ & 0 \leq \lambda_q \leq 1 & ,\forall q \in Q \\ & \sum_{q=1}^{2} \lambda_q = 1 & \end{array}\right] \tag{4.5}$$

avec $X^{(A)}$ et $X^{(B)}$ les deux solutions optimales respectivement pour z^1 et z^2 du problème scalairisé.

L'ajout de contraintes d'inégalité stricte sur la valeur des fonctions objectifs permet ainsi de calculer les solutions non supportées (NE). Cet algorithme génère donc l'ensemble minimum complet des solutions efficaces. L'ensemble complet peut aussi être obtenu par cette méthode si toutes les solutions optimales des différents problèmes paramétriques sont recherchées (et non une seule solution optimale par problème).

4.2 Résolution approchée

Nous allons maintenant nous intéresser à la résolution approchée du problème de set packing. La plupart des métaheuristiques actuelles pourraient être adaptées à ce problème. Notre choix s'est arrêté sur la métaheuristique GRASP. En effet, plusieurs études ont montré que cette métaheuristique permettait d'obtenir des solutions de bonne qualité pour des problèmes d'optimisation combinatoire difficiles, et notamment sur des problèmes proches du problème de set packing :

- le problème de set covering (voir Féo et Resende [FR95] ou Delorme [Del00] dans le cas mono-objectif et Gandibleux et al. [GVT98] dans le cas biobjectif),
- le problème de node packing (voir Féo et al. [FRS94, FR95]),
- le problème de clique maximum (voir Abello et al. [APR99]).

Après avoir présenté les principes généraux de GRASP nous décrirons l'adaptation de ces principes pour le problème de set packing que nous avons effectuée.

4.2.1 La métaheuristique GRASP

La métaheuristique GRASP a été proposée initialement par Féo et Resende [FR89]. C'est une méthode multi-départ en deux phases pour la résolution approchée de problèmes difficiles de l'optimisation combinatoire. La première phase correspond à la construction d'une solution initiale à l'aide d'une procédure gloutonne aléatoire. L'incorporation d'une composante aléatoire permet d'obtenir des solutions dans des zones diverses de l'espace des solutions. La seconde phase correspond à une recherche locale améliorant ces solutions. Ce processus est répété un grand nombre de fois. Une présentation générale de GRASP a été proposée par Pitsoulis et Resende [PR02].

De plus, plusieurs nouveaux composants sont venus compléter le schéma de base de GRASP comme par exemple :

- l'évolution, appelée reactive GRASP, permettant de déterminer de manière dynamique la valeur du paramètre α pilotant l'importance de la composante aléatoire,
- l'utilisation de techniques, à base de mémoire et d'apprentissage, pour profiter de l'information apportée par les solutions générées aux itérations précédentes,
- le couplage avec des méthodes de path relinking pour rechercher des solutions de bonne qualité en recomposant des chemins entre les solutions trouvées par GRASP.

En outre, de nombreuses hybridations avec d'autres métaheuristiques ont été proposées. Resende et Ribeiro [RR02] présentent en détail et discutent tous ces composants.

Une implémentation de GRASP pour un problème particulier va être caractérisée par six éléments principaux :

- l'algorithme glouton utilisé,
- l'importance de la composante aléatoire (fixée par un paramètre $\alpha \in [0; 1]$, une valeur $\alpha = 1$ correspondant à un glouton pur et une valeur $\alpha = 0$ correspondant à un algorithme aléatoire),
- le type de recherche locale et le voisinage considéré,
- la phase d'intensification éventuelle,
- le critère d'arrêt,
- et la phase de post-optimisation éventuelle.

L'algorithme 4.4 décrit ainsi le schéma complet de la métaheuristique GRASP (les phases optionnelles sont indiquées entre crochets).

L'adaptation de GRASP pour résoudre un problème particulier est assez facile lorsqu'il existe des algorithmes de construction et de recherche locale pour ce problème. GRASP a ainsi été appliqué avec succès à un grand nombre de problèmes d'optimisation comme des problèmes d'ordonnancement, de routage ou encore des problèmes théoriques dans les graphes. Une bibliographie annotée récente des travaux réalisés sur GRASP a été proposée par Festa et Resende [FR02].

$Solutions \leftarrow \emptyset$
répéter
  $initialSol \leftarrow greedyRandomized(problem, \alpha)$
  $improvedSol \leftarrow localSearch(initialSol)$
  $[improvedSol \leftarrow intensification(improvedSol)]$
  $Solutions \leftarrow Solutions \cup \{improvedSol\}$
jusqu'à critère d'arrêt
$[Solutions \leftarrow postOptimization(Solutions)]$
$finalSol \leftarrow best(Solutions)$

Alg. 4.4 – GRASP

4.2.2 GRASP pour le problème de set packing

Nous allons maintenant présenter l'adaptation de GRASP que nous proposons pour le problème de set packing mono-objectif. L'algorithme décrit dans cette section correspond à celui proposé par Delorme et al. [DGR04]. Il ne comprend pas de phase de post-optimisation. Dans la suite, nous allons indiquer les cinq autres éléments principaux le caractérisant.

4.2.2.1 La phase gloutonne aléatoire

Pour la phase gloutonne de notre algorithme GRASP, nous avons considéré trois algorithmes gloutons différents (appelés greedy1, greedy2 et greedy3). Leurs caractéristiques sont détaillées ci-dessous :

L'algorithme greedy1 Notre procédure greedy1 (voir l'algorithme 4.5) est inspirée par l'adaptation de GRASP pour le problème de set covering proposée par Féo et Resende [FR95]. Elle construit une solution en partant de la solution non-réalisable[2] triviale, $x_i = 1, \forall i \in I$. Les valeurs de plusieurs variables sont mises à 0 jusqu'à ce que la solution soit réalisable. Ces changements ne concernent à chaque fois qu'une seule variable par itération. Afin de conserver une valeur maximale pour la fonction objectif, les variables sont évaluées. Celles faisant parties d'un nombre maximum de contraintes et avec une valeur minimum (c-à-d avec c_i minimum) sont privilégiées. Une liste restreinte de variables candidates (RCL pour Restricted Candidate List) est ainsi constituée avec les meilleures variables. Un paramètre $\alpha \in [0, 1]$ appliqué comme coefficient multiplicateur sur la meilleure évaluation permet de fixer un seuil en dessous duquel les variables ne sont pas conservées dans la RCL. Le choix de la variable devant être fixée à 0 est fait de manière aléatoire parmi toutes les variables de la RCL. Ainsi, avec une valeur $\alpha = 0$, l'algorithme correspond à une construction entièrement aléatoire ; avec une valeur $\alpha = 1$, il correspond à un algorithme glouton pur.

La complexité théorique de cet algorithme est en $O(n^2m)$ dans le pire des cas.

[2] À l'exception du cas de figure évident à résoudre dans lequel il n'y a aucune contrainte.

$I_t \leftarrow I$
$x_i \leftarrow 1, \forall i \in I_t$
$J_t \leftarrow J \setminus \{j : \sum_{i \in I_t} t_{i,j} x_i \leq 1\}$
tant que $(J_t \neq \emptyset)$ faire
  $Eval_i \leftarrow \sum_{j \in J_t} t_{i,j}/c_i, \forall i \in I_t$
  $Limit \leftarrow \min_{i \in I_t}(Eval_i) + \alpha * (\max_{i \in I_t}(Eval_i) - \min_{i \in I_t}(Eval_i))$
  $RCL \leftarrow \{i \in I_t, Eval_i \geq Limit\}$
  $i^* \leftarrow RandomSelect(RCL)$
  $x_{i^*} \leftarrow 0$
  $I_t \leftarrow I_t \setminus \{i^*\}$
  $J_t \leftarrow J_t \setminus \{j : \sum_{i \in I_t} t_{i,j} x_i \leq 1\}$
finTant

Alg. 4.5 – Construction gloutonne aléatoire greedy1

L'algorithme greedy2 Notre procédure greedy2 (voir l'algorithme 4.6) est inspirée par l'adaptation de GRASP pour le problème de node packing proposée par Féo et al. [FRS94, FR95]. Elle construit une solution en partant de la solution réalisable triviale, $x_i = 0, \forall i \in I$. Les valeurs de plusieurs variables sont mises à 1 tant que la solution reste réalisable. Ces changements ne concernent à chaque fois qu'une seule variable par itération. Afin d'obtenir une valeur maximale pour la fonction objectif, les variables sont évaluées. Celles faisant parties d'un nombre minimum de contraintes et avec une valeur maximum (c-à-d avec c_i maximum) sont privilégiées. Comme pour la procédure greedy1, le choix de la variable devant être fixée à 1 est fait de manière aléatoire parmi les variables sélectionnées dans une RCL.

$I_t \leftarrow I$
$x_i \leftarrow 0, \forall i \in I_t$
$Eval_i \leftarrow c_i / \sum_{j \in J} t_{i,j}, \forall i \in I_t$
tant que $(I_t \neq \emptyset)$ faire
  $Limit \leftarrow \min_{i \in I_t}(Eval_i) + \alpha * (\max_{i \in I_t}(Eval_i) - \min_{i \in I_t}(Eval_i))$
  $RCL \leftarrow \{i \in I_t, Eval_i \geq Limit\}$
  $i^* \leftarrow RandomSelect(RCL)$
  $x_{i^*} \leftarrow 1$
  $I_t \leftarrow I_t \setminus \{i^*\}$
  $I_t \leftarrow I_t \setminus \{i : \exists j \in J, t_{i,j} + t_{i^*,j} > 1\}$
finTant

Alg. 4.6 – Construction gloutonne aléatoire greedy2

La complexité théorique de cet algorithme est en $O(\beta nm)$ dans le pire des cas, avec β représentant le nombre maximum d'itérations. Ce nombre est égal au nombre maximum de variables pouvant être fixées à 1 dans une solution réalisable. Cela signifie que sa complexité théorique est en $O(n^2m)$ ($\beta \leq n$), mais en pratique β est souvent très petit par rapport à n.

Ainsi, la complexité de l'algorithme greedy2 est très souvent largement inférieure à celle de greedy1.

L'algorithme greedy3 Notre procédure greedy3 est une évolution de la procédure greedy2 décrite précedemment. Elle y inclue un processus d'apprentissage permettant d'améliorer l'évaluation des variables pour les itérations suivantes de l'algorithme GRASP (voir l'algorithme 4.4). Ainsi, un poids est associé à chaque contrainte en fonction de la fréquence avec laquelle elle est saturée[3] dans l'ensemble des solutions générées au cours des itérations précédentes. Dans l'algorithme 4.7, $SaturationFrequency(j)$ représente le rapport entre le nombre de fois que la contrainte j a été saturée et le nombre de solutions générées. Ce calcul est réalisé par la fonction $Update()$. $SaturationFrequency$ est initialisée à 0 pour la première itération de GRASP.

$I_t \leftarrow I$
$x_i \leftarrow 0, \forall i \in I_t$
$Eval_i \leftarrow c_i / \sum_{j \in J} (t_{i,j} * (1 + SaturationFrequency(j))), \forall i \in I_t$
tant que $(I_t \neq \emptyset)$ faire
  $Limit \leftarrow \min_{i \in I_t} (Eval_i) + \alpha * (\max_{i \in I_t} (Eval_i) - \min_{i \in I_t} (Eval_i))$
  $RCL \leftarrow \{i \in I_t, Eval_i \geq Limit\}$
  $i^* \leftarrow RandomSelect(RCL)$
  $x_{i^*} \leftarrow 1$
  $I_t \leftarrow I_t \setminus \{i^*\}$
  $I_t \leftarrow I_t \setminus \{i : \exists j \in J, t_{i,j} + t_{i^*,j} > 1\}$
finTant
$Update(SaturationFrequency(j)), \forall j \in J$

Alg. 4.7 – Construction gloutonne aléatoire greedy3

Comme pour l'algorithme greedy2, la complexité théorique de cet algorithme est en $O(\beta nm)$ dans le pire des cas.

4.2.2.2 Le choix du paramètre α : reactive GRASP

Des expérimentations préliminaires ne nous ont pas permis d'identifier une valeur particulière du paramètre α pouvant être considérée comme une recommandation pour toutes les instances (ou au moins une grande partie). Cette observation avait déjà été rapportée par Resende et Ribeiro [RR02]. En conséquence, nous avons décidé d'utiliser plusieurs valeurs pour le paramètre α. Notre première stratégie fut de choisir la valeur de α de manière aléatoire parmi un ensemble discret pré-défini *alphaSet* de valeurs possibles avec une distribution de probabilités uniforme. Cependant, nous avons aussi considéré une seconde stratégie

[3]Une contrainte j est considérée comme saturée par une solution si une variable i, fixée à 1 dans cette solution, apparaît dans cette contrainte (c-à-d $\exists i \in I, x_i = 1$ et $t_{i,j} = 1$).

appelée reactive GRASP. Le paramètre α est alors déterminé automatiquement en fonction de la qualité des solutions obtenues aux itérations précédentes de GRASP. L'algorithme que nous proposons (voir l'algorithme 4.8) est une évolution de l'agorithme 4.4. Il est inspiré par l'agorithme GRASP proposé par Prais et Ribeiro [PR00].

$Solutions \leftarrow \emptyset$
$proba_\alpha \leftarrow 1/|alphaSet|, \forall \alpha \in alphaSet$
répéter
  $\alpha^* \leftarrow RandomSelect(alphaSet, proba)$
  $initialSol \leftarrow greedyRandomized(problem, \alpha^*)$
  $improvedSol \leftarrow localSearch(initialSol)$
  $[improvedSol \leftarrow intensification(improvedSol)]$
  $Solutions \leftarrow Solutions \cup \{improvedSol\}$
  $Pool_{\alpha^*} \leftarrow Pool_{\alpha^*} \cup \{improvedSol\}$
  si condition $probaUpdate$ est vrai alors
    $valuation_\alpha \leftarrow \left(\frac{mean_{s \in Pool_\alpha}(z(s)) - z(worst(Solutions))}{z(best(Solutions)) - z(worst(Solutions))}\right)^\delta, \forall \alpha \in alphaSet$
    $proba_\alpha \leftarrow valuation_\alpha / \left(\sum_{\alpha' \in alphaSet} valuation_{\alpha'}\right), \forall \alpha \in alphaSet$
  finSi
jusqu'à critère d'arrêt
$[Solutions \leftarrow postOptimization(Solutions)]$
$finalSol \leftarrow best(Solutions)$

Alg. 4.8 – Reactive GRASP

En partant d'une distribution de probabilités uniforme pour les valeurs de l'ensemble discret pré-défini $alphaSet$, $proba_\alpha$ est mise-à-jour selon une périodicité dépendant de la condition $probaUpdate$. Les nouvelles probabilités sont calculées à partir de la valeur moyenne des meilleures solutions obtenues pour chaque valeur du paramètre α (ces meilleures solutions sont mémorisées dans un ensemble $Pool_\alpha$ de taille limitée). Un paramètre δ est introduit pour atténuer les écarts entre les probabilités des différentes valeurs.

4.2.2.3 La phase de recherche locale

Le voisinage $\mathcal{N}$ utilisé pour notre procédure de recherche locale (voir l'algorithme 4.9) est basé sur des échanges $k-p$. Le voisinage, défini par les échanges $k-p$, d'une solution x est l'ensemble des solutions obtenues à partir de x en changeant la valeur de k variables de 1 à 0, et en changeant la valeur de p variables de 0 à 1. En raison de l'explosion combinatoire du nombre d'échanges possibles lorsque les valeurs de k et de p augmentent, nous avons seulement considéré des échanges $0-1$, $1-1$, $2-1$ et $1-2$. De plus, la procédure de recherche a été implémentée en utilisant une stratégie de choix de type première solution améliorante trouvée (c-à-d que nous sélectionnons le premier voisin trouvé dont la valeur est meilleure que celle de la solution courante). Chaque fois qu'un échange est réalisé, la

recherche locale est relancée à partir de la nouvelle solution obtenue. Elle s'arrête lorsqu'il n'y a plus d'échange améliorant possible.

fonction $searchNeighbourhood(s, \mathcal{N})$
  tant que s non localement optimal faire
    Trouver $s' \in \mathcal{N}(s)$ avec $z(s') > z(s)$
    $s \leftarrow s'$
  finTant
  renvoyer s
fin $searchNeighbourhood$
répéter
  $Solution \leftarrow searchNeighbourhood(Solution, 0-1exchanges)$
  $Solution \leftarrow searchNeighbourhood(Solution, 1-1exchanges)$
  $Solution \leftarrow searchNeighbourhood(Solution, 2-1exchanges)$
  $Solution \leftarrow searchNeighbourhood(Solution, 1-2exchanges)$
jusqu'à $Solution$ non amélioré
renvoyer $Solution$

Alg. 4.9 – Recherche locale

La complexité théorique de l'algorithme $searchNeighbourhood$ est en $O(n^4m)$ dans le pire des cas (pour les échanges $1-2$ et $2-1$). En pratique cependant, le temps nécessaire peut être diminué de manière importante quand l'algorithme de recherche locale est appliqué sur une solution initiale de bonne qualité.

4.2.2.4 La phase d'intensification : path relinking

La procédure d'intensification que nous proposons est basée sur le principe du path relinking (recomposition de chemins en français). Ce principe a été proposé à l'origine pour la recherche tabou par Glover et Laguna [GL97]. Le path relinking vise à générer un chemin reliant deux solutions $X^{(i)}$ et $X^{(f)}$ de bonne qualité (aussi appelées solutions d'élite), et de l'explorer pour trouver de meilleures solutions (voir figure 4.4). La notion de chemin utilisée ici ne correspond pas à celle de la définition 1.4 (chapitre 1, page 12), mais à une succession de solutions (admissibles ou non) $X^{(t_1)}, \ldots, X^{(t_k)}$ telle que le passage d'une solution à la suivante peut se faire à l'aide d'un mouvement élémentaire. Il a été utilisé pour la première fois comme phase d'intensification d'une procédure GRASP par Laguna et Martí [LM99].

Il existe plusieurs variantes de cet algorithme. Celles-ci concernent notamment le sens d'exploration du chemin, le fait de n'explorer qu'une partie du chemin, et l'utilisation éventuelle d'une recherche locale à chaque itération. L'algorithme que nous proposons pour

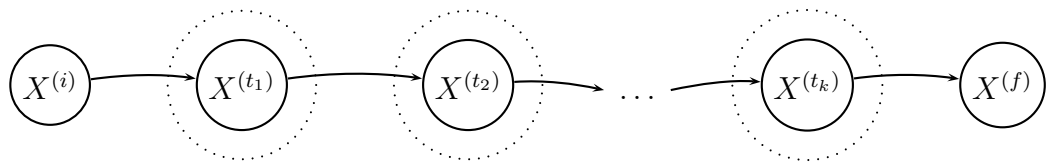


FIG. 4.4 – Principe du path relinking

```
Solution ← initialSolution
pour i ∈ I faire
    si initialSolution(i) ≠ targetSolution(i) alors
        Solution(i) ← targetSolution(i)
        currentSolution ← Solution
        si currentSolution non réalisable alors
            currentSolution ← repairing(currentSolution)
        finSi
        currentSolution ← saturation(currentSolution)
        Pool ← Pool ∪ {currentSolution}
    finSi
finPour
```

ALG. 4.10 – Path relinking

le problème de set packing (voir l'algorithme 4.10) est inspiré de celui proposé par Resende et Ribeiro [RR03].

Les meilleures solutions obtenues avec GRASP sont mémorisées dans un ensemble *Pool* de taille limitée. À chaque itération de GRASP, l'algorithme de path relinking est appliqué entre la solution obtenue après la recherche locale et une solution sélectionnée aléatoirement parmi celles mémorisées dans l'ensemble *Pool*. La meilleure de ces deux solutions est prise comme solution initiale (*initialSolution*), et l'autre comme solution cible (*targetSolution*). À chaque itération du path relinking, une solution admissible (*currentSolution*) est générée à partir de la solution du chemin (*Solution*). Celle-ci est donc réparée si cela est nécessaire. La fonction de réparation considérée utilise l'algorithme 4.5, avec une valeur de 1 pour le paramètre α (c-à-d une procédure gloutonne pure). Elle est ensuite saturée à l'aide d'une fonction *saturation* qui utilise l'algorithme 4.6 (là aussi avec une valeur de 1 pour le paramètre α). Les solutions ainsi générées peuvent aussi être inclues dans l'ensemble *Pool*.

La complexité théorique de l'algorithme *repairing* étant en $O(n^2m)$, la complexité théorique de cet algorithme est en $O(n^3m)$ dans le pire des cas.

4.2.3 Extension de GRASP au cas biobjectif

Nous disposons ainsi d'une procédure GRASP complète pour le problème de set packing mono-objectif. De plus, nous avons utilisé cette procédure pour obtenir une approximation de la frontière efficace pour les problèmes de set packing biobjectifs. Pour cela, nous l'avons encapsulée de manière à l'appliquer successivement et indépendamment sur 20 directions de recherche différentes. Ces directions de recherche correspondent aux combinaisons convexes des objectifs. Cela revient donc à résoudre les 20 problèmes de set packing mono-objectifs définis par l'équation 4.6. La figure 4.5 illustre le principe de résolution utilisé. La méthode décrite ici correspond à celle proposée par Delorme et al. [DGD03].

$$\left[\begin{array}{lll} Max\ z_\lambda(X) = & \lambda z^1(X) + (1-\lambda) z^2(X) & ,\forall \lambda \in \{0, \frac{1}{19}, \ldots, \frac{18}{19}, 1\} \\ sc & X \in \mathcal{D} & \end{array}\right] \tag{4.6}$$

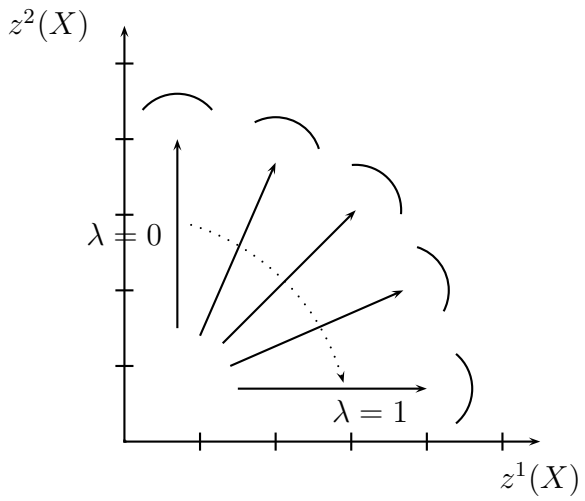


FIG. 4.5 – Schéma de l'utilisation de GRASP dans le cas biobjectif

Nous avons vu au chapitre 1 (section 1.4.2.2, page 24) que les solutions obtenues avec une telle technique et une méthode de résolution exacte appartenaient toutes à l'ensemble SE des solutions supportées. Dans notre cas, la méthode de résolution appliquée étant une méthode approchée, les solutions obtenues, lorsqu'elles sont efficaces, peuvent aussi appartenir à l'ensemble NE des solutions non supportées. Durant ce processus, l'ensemble des solutions potentiellement efficaces générées (à chaque itération de GRASP et même du path relinking) sont conservées.

4.3 Conclusion

Dans ce chapitre, nous avons décrit les différents algorithmes que nous proposons pour résoudre, de manière exacte ou approchée, le problème de set packing. Ceux-ci peuvent ainsi être utilisés pour résoudre le problème d'évaluation de la capacité d'infrastructures ferroviaires tel que nous l'avons modélisé au chapitre 3.

La méthode de résolution exacte que nous avons considérée s'appuie sur des algorithmes de pré-traitements et sur le logiciel Cplex. Nos pré-traitements mettent notamment à profit la structure du problème de set packing et les spécifités des instances ferroviaires. La méthode de résolution approchée, quant à elle, est une adaptation de la métaheuristique GRASP. Plusieurs programmes, inspirés de cette métaheuristique et de certaines de ses extensions les plus connues, ont ainsi été développés et présentés. En outre, afin de prendre en compte l'aspect multiobjectif de notre modélisation, des travaux sur des algorithmes biobjectifs ont aussi été engagés. Ceux-ci reposent sur une extension des méthodes de résolution mono-objectifs proposées.

Dans la suite, nous allons nous intéresser à l'évaluation des performances de ces différents algorithmes. Pour cela, nous allons présenter les résultats que nous avons obtenus sur un ensemble d'instances tests. Ces résultats sont ainsi commentés et analysés.

Chapitre 5

Résultats des expérimentations

Dans le chapitre précédent, plusieurs algorithmes de résolution du problème de set packing ont été présentés. Afin d'évaluer leur intérêt, nous avons testé ces algorithmes sur différentes instances de ce problème. L'ensemble des expérimentations numériques effectuées sont ainsi décrites dans ce chapitre. Dans un premier temps, les conditions dans lesquelles nous avons mené les expérimentations sont indiquées ; les caractéristiques des instances de problèmes utilisées sont notamment précisées. Ensuite, les résultats obtenus avec les méthodes de résolution, exacte puis approchée, sont rapportés et analysés. L'intérêt des pré-traitements pour la résolution exacte et des différents composants de notre algorithme basé sur GRASP est notamment mis en évidence.

5.1 Conditions d'expérimentation

Tous les programmes que nous avons développés à partir des algorithmes décrits au chapitre 4 (les pré-traitements et GRASP) ont été codés en langage Ada. Les autres programmes que nous avons utilisés, comme notamment celui basé sur une méthode dichotomique pour la résolution exacte dans le cas biobjectif, sont codés en langage C. Toutes les expérimentations rapportées dans ce chapitre ont été réalisées sur une même machine avec les caractéristiques suivantes :

- CPU : Pentium III cadencé à 800 MHz
- Mémoire vive : 640 Mo
- Système d'exploitation : Linux 2.4.20
- Compilateur Ada : Gnat 3.14
- Compilateur C : gcc 3.0.4

En raison de l'absence (en tout cas à notre connaissance) d'instances tests disponibles pour le problème de set packing mono-objectif, nous avons basé notre évaluation sur des

instances que nous avons générées. Ces instances peuvent être classées en deux catégories :

- un ensemble d'instances générées aléatoirement afin d'observer le comportement de nos algorithmes sur une gamme de problèmes aux caractéristiques variées,
- et un ensemble d'instances correspondant à l'application de notre modèle ferroviaire d'évaluation de la capacité sur une infrastructure réelle. Ce deuxième ensemble peut ainsi permettre d'identifier des difficultés spécifiques à la structure de ces instances. De plus, il permet de mesurer l'impact de la reformulation du problème ferroviaire (voir section 4.1.1.1, page 71).

En outre, un ensemble d'instances aléatoires de plus petite taille a aussi été généré afin d'effectuer une validation dans le cas biobjectif.

5.1.1 Instances mono-objectifs générées aléatoirement

Pour obtenir la diversité souhaitée, les instances mono-objectifs aléatoires ont été générées avec les caractéristiques suivantes :

- le nombre de variables est égal à 1000 ou à 2000,
- le nombre de contraintes est égal à 100% ou à 500% du nombre de variables,
- le nombre maximum d'éléments non nuls par contrainte est égal à 1% ou 5% du nombre de variables ($\sum_{i \in I} t_{i,j} \leq 0,01 * n$(resp. $0,05 * n$), $\forall j \in J$, c-à-d que le nombre d'éléments non nuls d'une contrainte est choisi aléatoirement entre deux et ce pourcentage ; en effet, une contrainte comprenant moins de deux éléments n'aurait aucun intérêt pour le problème),
- les valeurs des variables (c_i) sont uniformément distribuées dans l'intervalle [1-20] (instances à coûts pondérés) ou toutes fixées à 1 (instances à coûts unitaires).

Toutes les combinaisons possibles de ces caractéristiques ont été considérées. Seize instances, portant le label Rnd, ont ainsi été générées. Leurs caractéristiques sont présentées dans le tableau 5.1. La colonne densité indique le pourcentage d'éléments non nuls dans la matrice des contraintes[1]. Les instances à coûts pondérés sont aussi signalées. Enfin, les valeurs de la fonction objectif de la relaxation linéaire (LP) et de la meilleure solution entière connue (avec un astérisque lorsque cette solution a été démontrée comme optimale par Cplex) sont mentionnées. Notons l'écart très important entre ces deux valeurs.

Toutes ces instances sont disponibles sur un site web présentant une collection d'instances tests du problème de set packing mono-objectif [Del].

[1]Même si ce pourcentage est influencé par le nombre maximum d'éléments non nuls par contrainte, il ne doit pas être confondu avec ce dernier.

5.1.2 Instances ferroviaires issues du nœud de Pierrefitte-Gonesse

Le second ensemble d'instances mono-objectifs du problème de set packing considéré provient de l'application de notre modèle ferroviaire d'évaluation de la capacité sur le nœud de Pierrefitte-Gonesse. Celui-ci est situé à une dizaine de kilomètres de la gare de Paris Nord et s'étend sur environ seize kilomètres. Il représente une sorte de carrefour entre quatre grandes directions (hors lignes RER) et ses parcours se croisent en de nombreux endroits (voir Figure 5.1). On peut principalement distinguer trois grands types de voies dans le nœud de Pierrefitte-Gonesse :

- les voies grande ligne reliant Paris Nord et Chantilly
- les voies grande vitesse reliant Paris Nord à Lille
- les voies reliant Chantilly à la Grande Ceinture

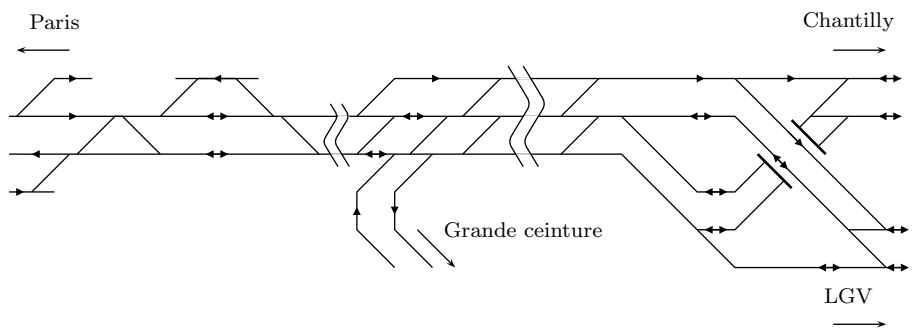


Fig. 5.1 – Plan de voies du nœud de Pierrefitte-Gonesse

De plus, ce nœud accueille un ensemble hétérogène de circulations et des matériels roulants aux caractéristiques très différentes. Les trois types de trains suivants circulent de manière quotidienne à travers ce nœud :

- Des trains grande vitesse comme le TGV Nord (à destination du nord de la France), l'Eurostar (à destination de l'Angleterre) et Thalys (à destination de la Belgique) circulant dans les deux sens. Ceux-ci empruntent habituellement les lignes grande vitesse mais peuvent occasionnellement être détournés vers les voies grande ligne. Ce type de trafic est important et devrait encore augmenter dans les années à venir.
- Des trains voyageurs «classiques» entre Paris Nord et Chantilly dans les deux sens. Là encore le trafic de ce type de train est important. Il ne devrait cependant pas trop évoluer durant les prochaines années.
- Des trains marchandises entre Chantilly et la Grande Ceinture. Ce type de trafic est plus limité que les deux précédents mais ces trains sont aussi beaucoup plus lents.

Cet important trafic peut occasionner de nombreux conflits, et notamment les trains de marchandises dont les parcours coupent l'ensemble des parcours des autres trains à une vitesse très inférieure. De plus, le grand nombre de voies autorisant la circulation dans les

deux sens augmente le nombre de conflits possibles. La détermination de la capacité d'un tel nœud n'est donc pas une chose aisée.

Les instances numériques présentées ici correspondent à des instances du problème de saturation sur des configurations de grille horaire différentes. Ainsi, elles contiennent des contraintes issues des équations 3.5 et 3.13. Tous les trains considérés sont des trains saturants, l'ensemble des trains de la grille horaire initiale est donc vide ($T_{Fais} = \emptyset, T = T_{Sat}$). La fonction objectif de ces instances provient donc de l'équation 3.3 ; ces instances sont à coûts unitaires ($c_i = 1, \forall i \in I$).

Quinze instances, portant le label Gon, ont ainsi été générées. Leurs caractéristiques sont présentées dans le tableau 5.2. Là encore, les valeurs de la fonction objectif de la relaxation linéaire (LP) et de la meilleure solution entière connue (avec un astérisque lorsque cette solution a été démontrée comme optimale) sont mentionnées.

5.1.3 Instances biobjectifs

Dans le cas biobjectif, nous avons travaillé sur une gamme variée d'instances numériques présentant les caractéristiques numériques suivantes :

- 100 ou 200 variables,
- de 300 à 1000 contraintes,
- le nombre maximum d'éléments non nuls par contrainte est égal à 1%, 2% ou 4% du nombre de variables.

De plus, six familles différentes de fonctions objectifs ont été utilisées :

A : objectifs aléatoires
$c_i^1 = rnd[1..100], c_i^2 = rnd[1..100], \forall i = 1, \ldots, n$;

B : objectifs symétriques
$c_i^1 = rnd[1..100], \forall i = 1, \ldots, n$; $c_{n-i+1}^2 = c_i^1, \forall i = 1, \ldots, n$;

C : objectifs aléatoires avec motifs
$l_1 = rnd[1..\frac{n}{10}], l_2 = rnd[1..\frac{n}{10}], \ldots$;
$c_1^1 = c_2^1 = \ldots = c_{l_1}^1 = rnd[1..100]$; $c_{l_1+1}^1 = c_{l_1+2}^1 = \ldots = c_{l_1+l_2}^1 = rnd[1..100]$; ...
(idem pour $c_i^2, \forall i = 1, \ldots, n$)

D : objectifs symétriques avec motifs
$l_1 = rnd[1..\frac{n}{10}], l_2 = rnd[1..\frac{n}{10}], \ldots$;
$c_1^1 = c_2^1 = \ldots = c_{l_1}^1 = rnd[1..100]$; $c_{l_1+1}^1 = c_{l_1+2}^1 = \ldots = c_{l_1+l_2}^1 = rnd[1..100]$; ...
$c_{n-i+1}^2 = c_i^1, \forall i = 1, \ldots, n$;

E : un objectif unitaire et un aléatoire
$c_i^1 = 1, c_i^2 = rnd[1..100], \forall i = 1, \ldots, n$;

F : un objectif unitaire et un avec motifs
$c_i^1 = 1, \forall i = 1, \ldots, n$;

N°	n	m	Densité	c_i	LP	Meilleure solution connue
Rnd1	1 000	5 000	2.60%	[1-20]	330.26	67*
Rnd2	1 000	5 000	2.59%	[1-1]	22.81	4*
Rnd3	1 000	5 000	0.60%	[1-20]	1 365.64	661
Rnd4	1 000	5 000	0.60%	[1-1]	111.77	48
Rnd5	1 000	1 000	2.60%	[1-20]	434.71	222
Rnd6	1 000	1 000	2.65%	[1-1]	29.91	15
Rnd7	1 000	1 000	0.58%	[1-20]	2 349.19	2 260*
Rnd8	1 000	1 000	0.60%	[1-1]	184.32	175
Rnd9	2 000	10 000	2.54%	[1-20]	339.62	40*
Rnd10	2 000	10 000	2.55%	[1-1]	22.59	2*
Rnd11	2 000	10 000	0.55%	[1-20]	1 500.10	478
Rnd12	2 000	10 000	0.55%	[1-1]	114.01	32
Rnd13	2 000	2 000	2.55%	[1-20]	423.98	140
Rnd14	2 000	2 000	2.56%	[1-1]	28.70	9
Rnd15	2 000	2 000	0.56%	[1-20]	2 209.57	1 784
Rnd16	2 000	2 000	0.60%	[1-1]	168.82	132

TAB. 5.1 – Caractéristiques des instances mono-objectifs générées aléatoirement

N°	$\|T\|$	n	m	Densité	LP	Meilleure solution connue
Gon1	90	465	8 661	0.44%	90,00	45*
Gon2	120	620	11 676	0.33%	120,00	60*
Gon3	240	1 240	23 736	0.16%	240,00	120*
Gon4	240	1 240	50 705	0.16%	240,00	94*
Gon5	360	1 860	55 938	0.11%	360,00	151*
Gon6	360	1 860	119 009	0.11%	360,00	76
Gon7	380	1 620	79 5144	0.12%	380,00	88
Gon8	720	3 720	155 985	0.05%	720,00	280*
Gon9	720	3 720	482 887	0.05%	720,00	93
Gon10	150	2 400	186 861	0,08%	150,00	85
Gon11	125	2 683	228 972	0,07%	125,00	87
Gon12	200	2 880	311 228	0,07%	200,00	89*
Gon13	157	3 210	380 292	0,06%	157,00	90*
Gon14	150	2 160	140 632	0,09%	150,00	88
Gon15	130	2 503	185 523	0,08%	130,00	104

TAB. 5.2 – Caractéristiques des instances ferroviaires

$l_1 = rnd[1..\frac{n}{10}], l_2 = rnd[1..\frac{n}{10}], \ldots;$
$c_1^2 = c_2^2 = \ldots = c_{l_1}^2 = rnd[1..100]\,;\ c_{l_1+1}^2 = c_{l_1+2}^2 = \ldots = c_{l_1+l_2}^2 = rnd[1..100]\,;\ \ldots$

Cela représente un total de 120 instances numériques (20 par famille de fonctions objectifs). Du fait de leur nombre, leurs caractéristiques détaillées ne sont pas présentées dans ce mémoire. Toutes ces instances sont disponibles dans la collection d'instances de problèmes MOCO hébergée sur le site web de la MCDM society [MCD].

5.2 Résolution exacte et borne supérieure

Nous allons maintenant nous intéresser à la résolution exacte de ces instances. Dans ce but, nous commencerons par observer l'impact de nos différents algorithmes de pré-traitements (décrits dans la section 4.1.1), puis nous indiquerons la qualité des solutions et des bornes supérieures obtenues. Les résultats présentés dans cette section reprennent et complètent ceux présentés par Delorme et al. [DGR03a, DGR03b].

5.2.1 Impact des pré-traitements

Afin d'évaluer l'impact de nos pré-traitements, nous nous intéressons à l'évolution de trois éléments :

- les caractéristiques (nombre de variables, de contraintes, densité) des instances qui peuvent influer sur le temps et la mémoire nécessaires pour les résoudre,
- la valeur de leur relaxation linéaire qui est utilisée comme borne supérieure lors de la résolution,
- et l'écart entre la meilleure solution et la borne supérieure (appelé gap) trouvées par Cplex.

Les différentes figures représentées dans cette section considèrent ainsi l'état des instances à trois niveaux pour les instances aléatoires et quatre pour les instances ferroviaires de la séquence d'étapes définie dans la section 4.1.2 pour l'application des pré-traitements :

- le problème initial (avant l'étape 1),
- le problème reformulé à l'aide des cliques propres au problème ferroviaire (après l'étape 2, uniquement les instances ferroviaires),
- le problème réduit par la suppression des variables dominées (après l'étape 4),
- le problème final avec les contraintes complétées pour en faire des cliques maximales (après l'étape 6).

5.2.1.1 Caractéristiques des instances

Les tableaux 5.3 et 5.4 rapportent respectivement l'évolution du nombre de variables des instances aléatoires et ferroviaires (entièrement due au test de dominance entre variables, les calculs de cliques ne pouvant pas supprimer de variable). Les Figures 5.2 et 5.3, quant à elles, rapportent respectivement l'évolution du nombre de contraintes et de la densité de la matrice des contraintes. Dans un souci de clarté et en raison des écarts très importants entre les différentes instances étudiées dans le nombre de contraintes, celui-ci est indiqué en proportion du nombre de contraintes du problème initial et non pas en valeur (c-à-d que la valeur 100% sur l'axe des ordonnées correspond au nombre de contraintes du problème avant l'utilisation des pré-traitements).

Rnd01	Rnd02	Rnd03	Rnd04	Rnd05	Rnd06	Rnd07	Rnd08	Rnd09
0,30%	0,10%	0,00%	0,00%	0,00%	0,00%	5,10%	8,00%	74,35%

Rnd10	Rnd11	Rnd12	Rnd13	Rnd14	Rnd15	Rnd16	**Moyenne**
76,20%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	**10,25%**

TAB. 5.3 – Pourcentage de variables supprimées pour les instances aléatoires

Gon01	Gon02	Gon03	Gon04	Gon05	Gon06	Gon07	Gon08
34,62%	34,03%	33,15%	11,37%	20,32%	14,41%	16,30%	10,24%

Gon09	Gon10	Gon11	Gon12	Gon13	Gon14	Gon15	**Moyenne**
11,10%	7,88%	10,73%	9,97%	7,23%	15,69%	9,83%	**16,46%**

TAB. 5.4 – Pourcentage de variables supprimées pour les instances ferroviaires

L'impact sur le nombre de variables est en général assez limité sur les instances aléatoires. Hormis pour les instances Rnd09 et Rnd10, seulement 1% en moyenne, et au plus 8%, des variables des autres instances sont ainsi supprimées.

Sur les instances ferroviaires, l'impact s'avère un peu plus important (16% de variables supprimées) et surtout mesurable sur toutes les instances (au moins 7% de variables supprimées). Il faut cependant noter que le pourcentage semble plus faible sur les instances de plus grande taille : il est d'environ 10% pour les instances Gon08 à Gon15 (celles dont le nombre de variables est supérieur à 2000). En effet, si certains choix de parcours peuvent ne présenter aucun intérêt lorsque le trafic est faible, ce n'est souvent plus le cas quand celui-ci devient plus dense. De plus, ces valeurs restent relativement basses par comparaison avec celles rapportées lors du projet DONS (voir section 2.4.1.2). Ceci peut s'expliquer par la présence de nombreuses voies banalisées et par les différences de temps entre les choix de

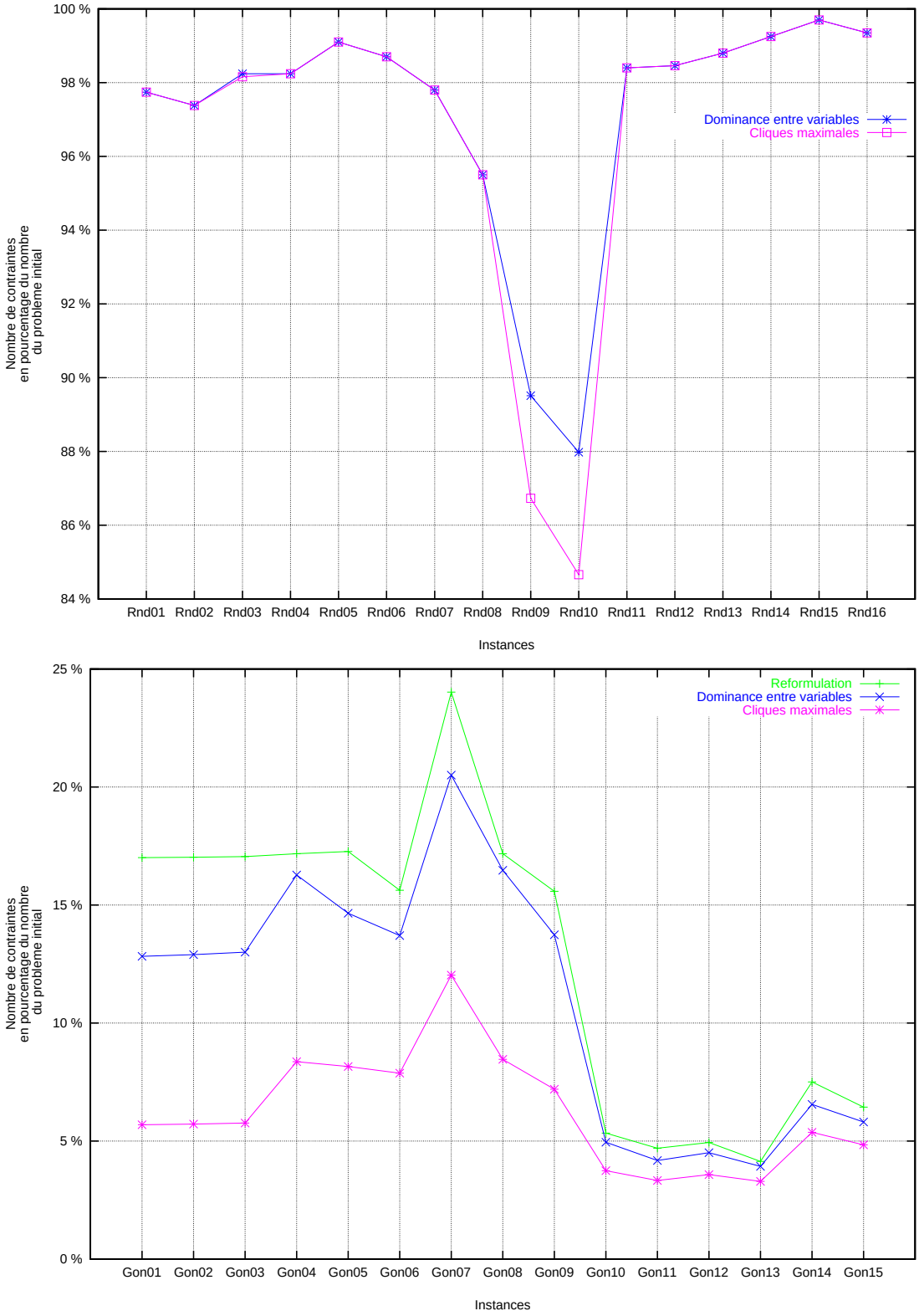
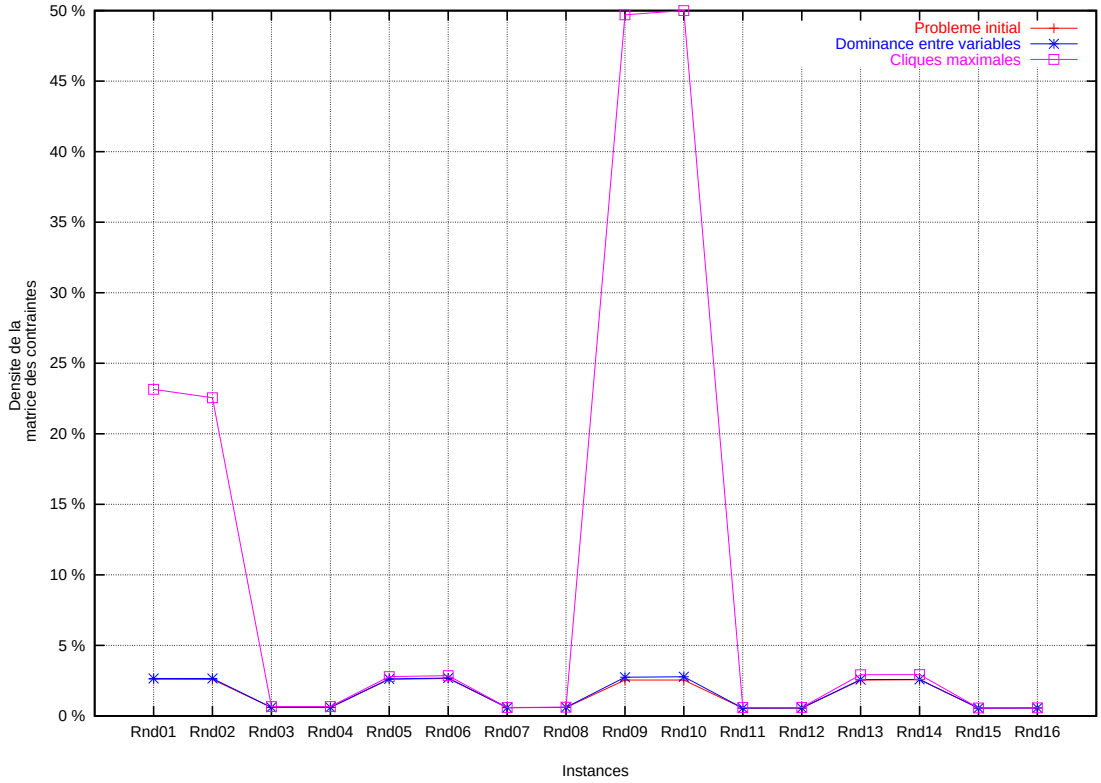


FIG. 5.2 – Impact des pré-traitements sur le nombre de contraintes



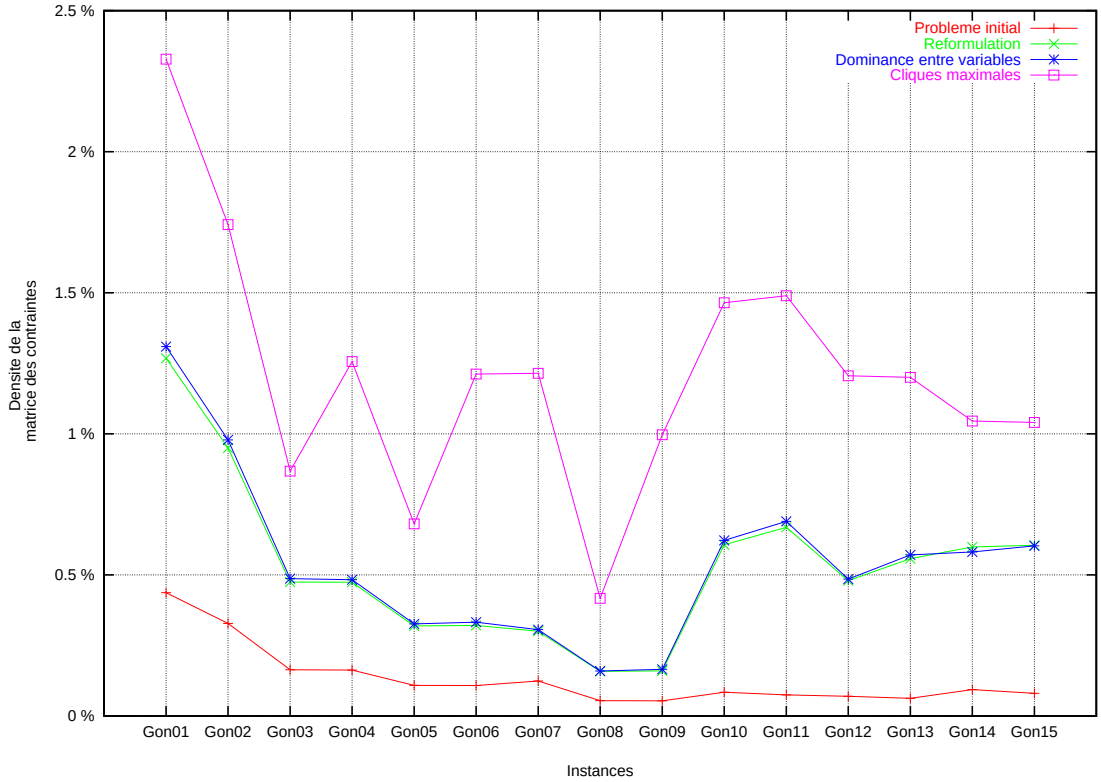


FIG. 5.3 – Impact des pré-traitements sur la densité des contraintes

parcours.

L'impact sur le nombre de contraintes des instances aléatoires est en général très limité. Environ 2% de contraintes sont supprimées sur la plupart des instances grâce au test de dominance entre paires de variables, quasiment rien avec le calcul de cliques maximales. Seules deux instances (Rnd09 et Rnd10) laissent apparaître un effet plus significatif (plus de 10% avec le test de dominance et environ 3% supplémentaires avec les cliques maximales). De même, l'impact sur la densité est négligeable sur la plupart des instances. Quatre instances (Rnd01, Rnd02 et là encore Rnd09 et Rnd10) font cependant exception puisque le calcul de cliques maximales augmente la densité de manière très importante (environ 1000% pour les deux premières et 2000% pour les deux autres). La densité de ces instances après les prétraitements est ainsi beaucoup plus élevée que celle des autres instances. Il est intéressant de noter que ce sont les instances aléatoires les plus contraintes (grand nombre de contraintes par rapport au nombre de variables et densité importante).

Par contre, dans le cas des instances ferroviaires, l'impact est beaucoup plus fort. Si le test de dominance entre paires de variables n'a, là encore, qu'un effet limité sur le nombre de contraintes et quasiment nul sur la densité, ce n'est pas le cas des calculs de cliques. En effet, la reformulation du problème permet de diviser par 5 ou plus le nombre de contraintes, tout en doublant leur densité. De même, le calcul de cliques maximales divise encore par 2 le nombre de contraintes et double à nouveau leur densité. La présence de nombreuses contraintes n'incluant que deux variables dans la formulation initiale du modèle explique sûrement cette marge importante de progrès.

5.2.1.2 Valeur de la relaxation linéaire

La Figure 5.4 rapporte l'évolution de la valeur de la relaxation linéaire. Dans un souci de clarté et en raison des écarts très importants entre les différentes instances étudiées dans la valeur de la relaxation linéaire, celle-ci est indiquée en proportion de la valeur de la meilleure solution entière connue, et non pas en valeur (c-à-d que la valeur 100% sur l'axe des ordonnées correspond à la valeur de la meilleure solution entière connue pour ce problème).

Comme pour la densité de la matrice des contraintes, la valeur de la relaxation linéaire des instances aléatoires n'évolue quasiment pas pour la plupart d'entre elles. Là aussi les instances Rnd01, Rnd02, Rnd09 et Rnd10 représentent l'exception en présentant une évolution marquée. Malgré tout, l'écart entre cette valeur et la meilleure solution connue reste très élevé : il dépasse 200 % pour 12 instances sur 16 et va jusqu'à plus de 700%.

Les résultats observés sur les instances ferroviaires, eux aussi, confirment la tendance observée avec la densité. L'écart entre la valeur de la relaxation linéaire et la meilleure solution connue diminue ainsi de manière importante pour l'ensemble des instances grâce aux calculs de cliques (l'impact du test de dominance entre variables est là aussi quasiment

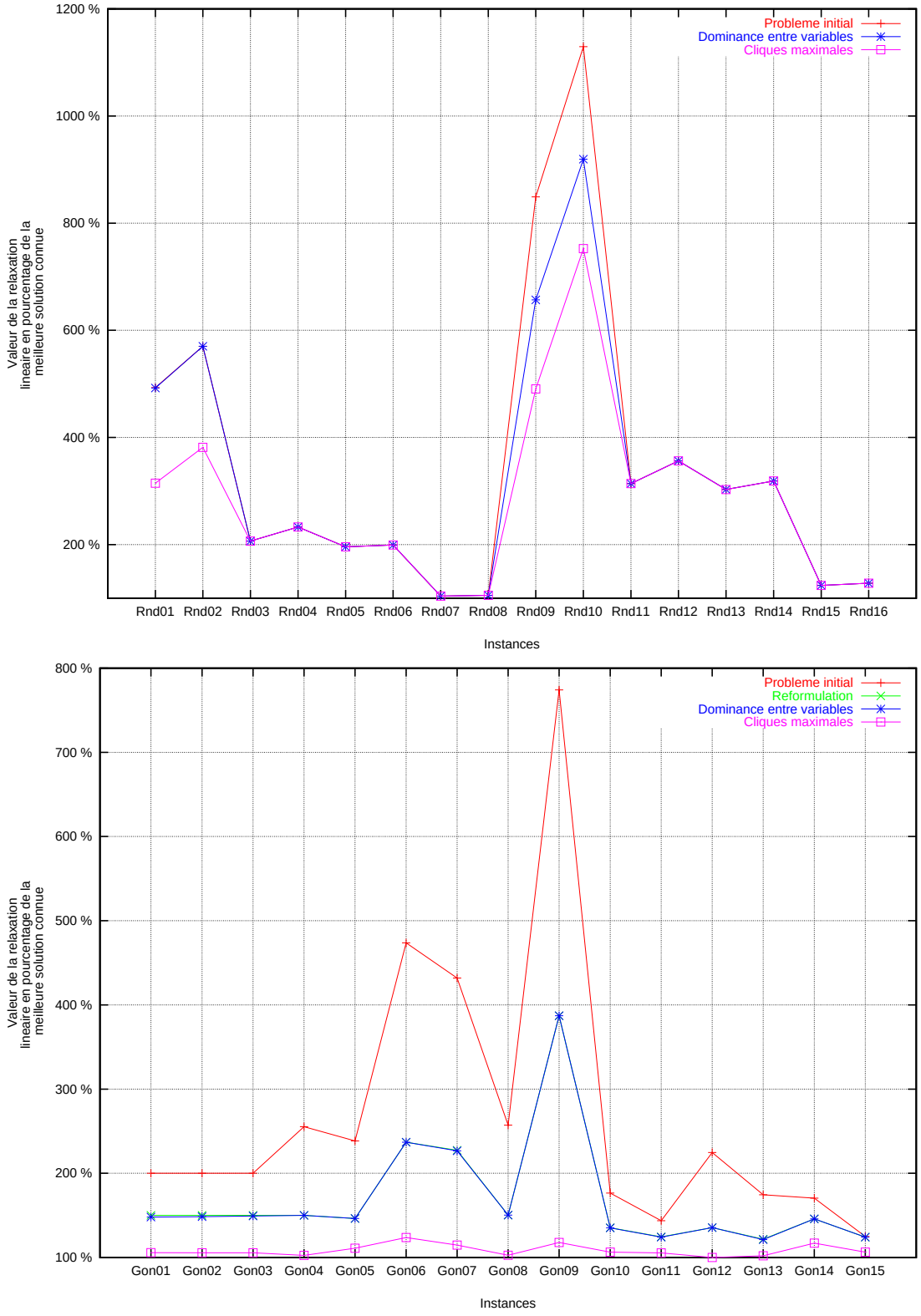


FIG. 5.4 – Impact des pré-traitements sur la valeur de la relaxation linéaire

nul). Suite aux pré-traitements, l'écart se retrouve à un niveau beaucoup plus bas que pour les instances aléatoires (au plus 25%).

5.2.1.3 Résolution en nombres entiers

Pour finir, la Figure 5.5 présente l'évolution des solutions obtenues avec le logiciel Cplex 8.0 [ILO02]. Nous avons utilisé pour cela les réglages par défaut du logiciel. Des expérimentations préliminaires ont en effet montré que ces réglages étaient les plus performants. La valeur indiquée correspond au gap entre la meilleure solution entière et la borne supérieure trouvées par Cplex (exprimé en pourcentage). Il est donc nul lorsque la solution optimale est trouvée. Dans les autres cas, le gap indiqué est celui obtenu après 50 000 secondes de calcul.

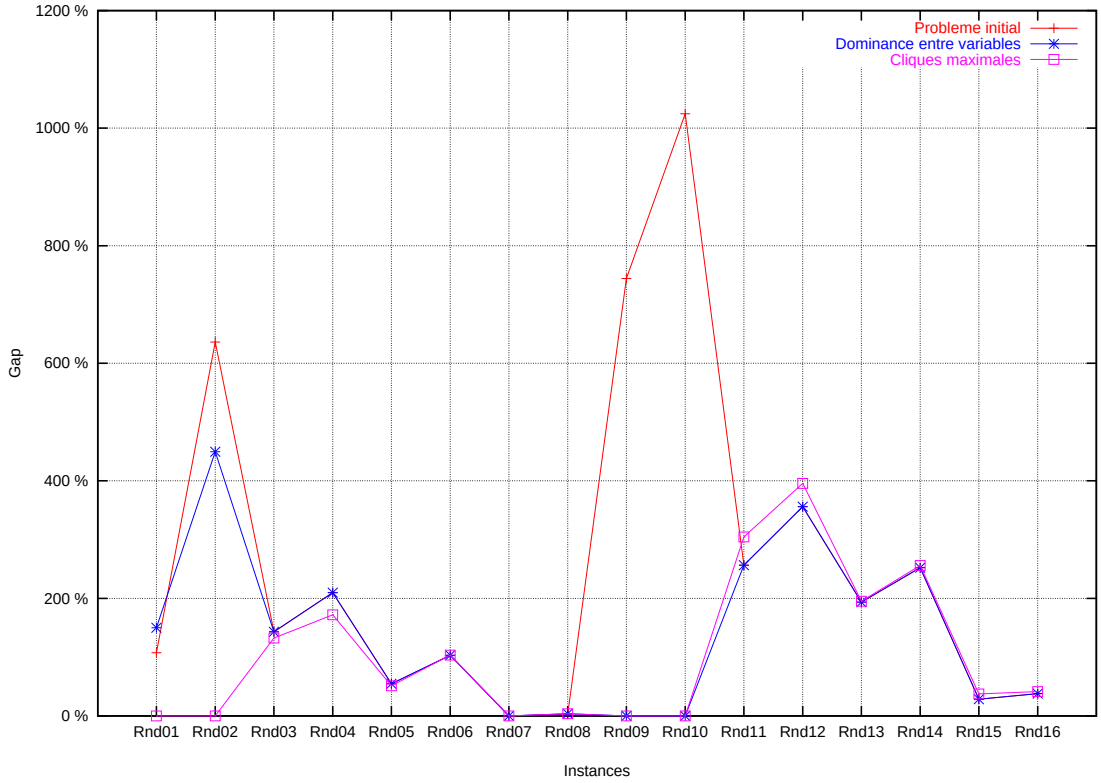
Ces résultats montrent l'amélioration globale observée. Il est cependant intéressant de noter que l'utilisation des pré-traitements ne provoque pas automatiquement une amélioration du gap. Ainsi pour certaines instances (Rnd01, Rnd11, Rnd12, Rnd15, Gon07 et Gon14), le gap obtenu avec Cplex peut être légèrement moins bon après l'utilisation de l'un d'eux. Cela peut notamment être dû à la modification de la structure du problème qui intervient notamment sur le fonctionnement interne de Cplex (ordre de branchement sur les variables, cliques détectées, solutions heuristiques générées, ...). Notons que cette dégradation intervient le plus souvent (instances Rnd11, Rnd12 et Rnd15) pour des instances sur lesquelles l'impact des pré-traitements sur les caractéristiques et la valeur de la relaxation linéaire était très faible. L'impact global est malgré cela positif, avec en particulier six instances dont la solution optimale a été trouvée grâce à l'application préliminaire des pré-traitements (Rnd01, Rnd02, Rnd09, Rnd10, Gon12 et Gon13).

5.2.2 Résultats obtenus dans le cas mono-objectif

Après avoir montré l'intérêt des pré-traitements utilisés, nous allons maintenant présenter les résultats obtenus avec Cplex sur les instances issues de ces pré-traitements. La Figure 5.6 rapporte ainsi la meilleure solution entière et la borne supérieure trouvées par Cplex après au plus 50 000 secondes. Lorsque la solution optimale du problème est trouvée, ces deux valeurs sont égales. La valeur de la relaxation linéaire du problème, une fois les pré-traitements terminés, est aussi indiquée. Dans un souci de clarté et en raison des écarts très importants entre les valeurs des solutions des différentes instances étudiées, celles-ci sont, là encore, indiquées en proportion de la valeur de la meilleure solution entière connue, et non pas en valeur (c-à-d que la valeur 100% sur l'axe des ordonnées correspond à la valeur de la meilleure solution entière connue pour ce problème).

Dans ces résultats, nous pouvons distinguer quatre cas de figure :

- pour les instances de petite taille (Gon 1 à 4) :
  la solution optimale peut être trouvée avec Cplex même sans utiliser les pré-traitements.



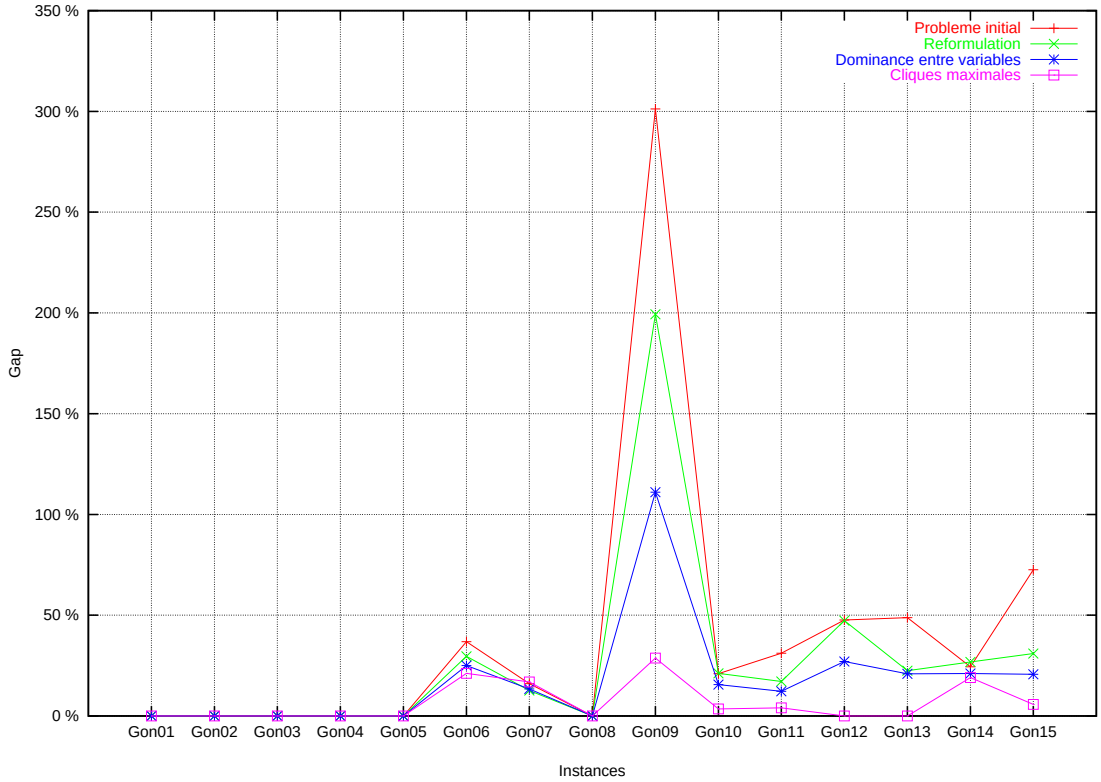


FIG. 5.5 – Impact des pré-traitements sur le gap obtenu avec Cplex

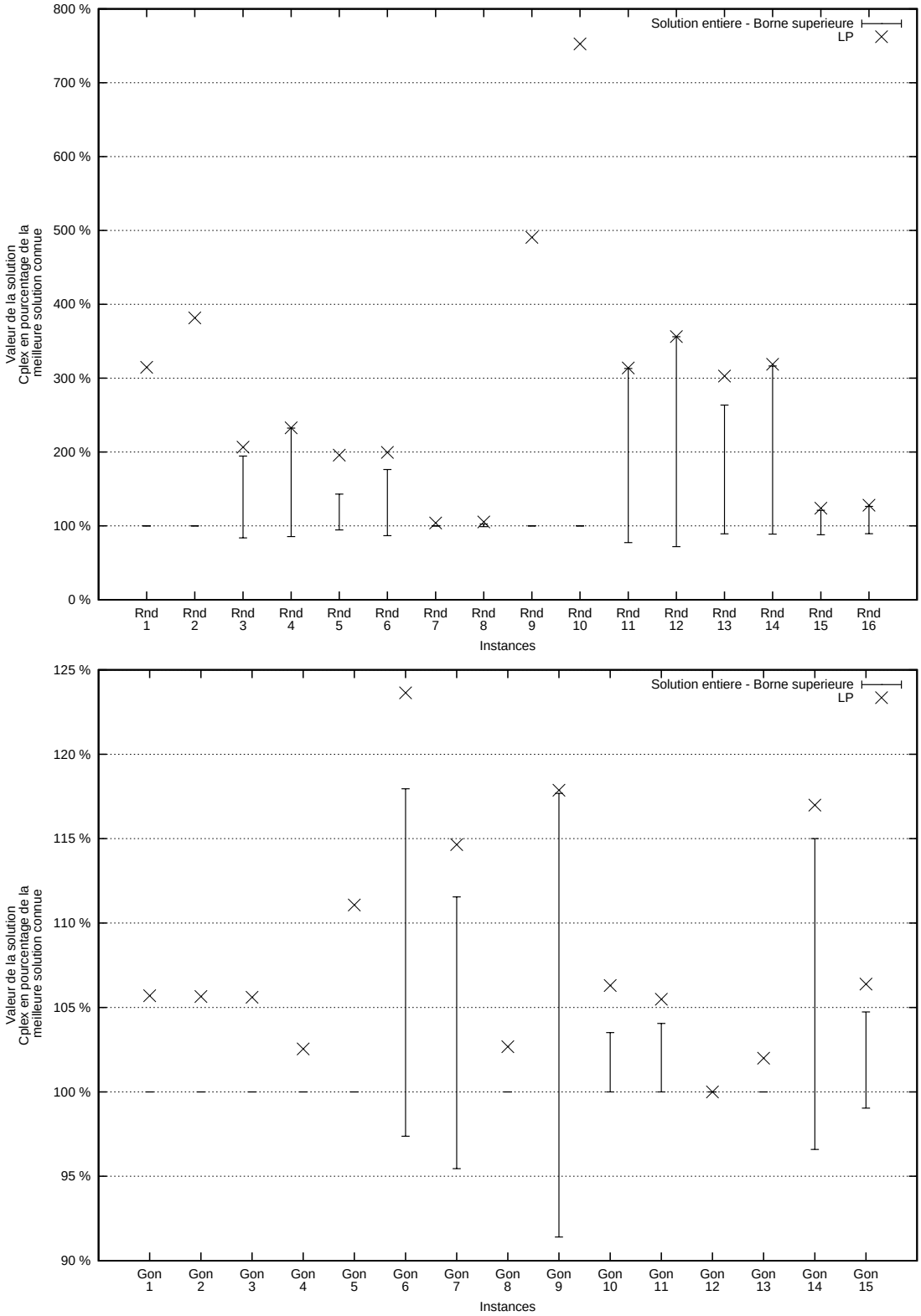


FIG. 5.6 – Résolution avec Cplex et borne supérieure

Ceux-ci permettent tout de même de l'obtenir plus rapidement.

- pour les instances fortement contraintes, c-à-d avec une densité de la matrice des contraintes élevée et un grand nombre de contraintes par rapport au nombre de variables (Rnd 1, 2, 9 et 10) :
  la solution optimale peut être trouvée en utilisant les pré-traitements et Cplex (pour certaines, les pré-traitements ne sont pas indispensables mais permettent de la trouver plus rapidement). Ceci est sûrement dû au faible nombre de variables constituant une solution admissible et donc à la faible profondeur de l'arbre de recherche.
- pour les instances faiblement contraintes, c-à-d avec une faible densité de la matrice des contraintes et un petit nombre de contraintes par rapport au nombre de variables (Rnd 7, 8, 15 et 16, Gon 5 et 8) :
  la solution optimale, ou au moins une solution avec un gap faible, peut être trouvée en utilisant Cplex. Les pré-traitements ont un impact très limité en terme de qualité de solution et de gap. Ceci est certainement dû à la bonne qualité de la relaxation linéaire du problème initial sur ces instances.
- pour les instances intermédiaires (Rnd 3 à 6 et 11 à 14, Gon 6, 7 et 9 à 15) :
  la solution optimale n'est jamais trouvée avec Cplex seul et les gap sont souvent importants. Les pré-traitements permettent d'améliorer de manière significative les résultats observés (les solutions optimales des instances Gon 12 et 13 sont obtenues ainsi), mais le gap reste très important pour de nombreuses instances (jusqu'à un gap proche de 500% pour Rnd12).

5.2.3 Détermination de la frontière efficace dans le cas biobjectif

Notre travail ne portait pas sur la résolution exacte du problème de set packing biobjectif. Il est cependant intéressant de noter que l'utilisation de l'algorithme simple fondé sur un schéma de recherche dichotomique nous a permis d'obtenir l'ensemble minimum complet des solutions non-dominées (ce dernier décrivant la frontière efficace) de toutes les instances considérées, mais dans des temps CPU parfois exhorbitants (c'est-à-dire pouvant monter jusqu'à plus de 360 000 secondes sur la machine que nous avons utilisée dans cette étude) malgré la taille assez petite de ces instances, et le nombre de solutions efficaces assez faible par rapport à d'autres problèmes combinatoires biobjectifs classiques (voir les résultats rapportés par Degoutin et Gandibleux [Deg02, DG02]).

De plus, les bornes immédiates étudiées par Ehrgott et Gandibleux [EG01] sur un ensemble de problèmes MOCO semblent de très mauvaise qualité pour le biSPP. Ainsi, une borne inférieure, obtenue avec un algorithme glouton sur différentes directions de recherche, ne donne pas une bonne description de l'ensemble des solutions efficaces, et en est éloignée. La borne supérieure obtenue par la relaxation linéaire du problème fournit encore moins d'information. Ce résultat est contraire à celui observé sur d'autres problèmes comme par

exemple celui du sac-à-dos biobjectif (voir Gandibleux [Gan01]). La Figure 5.7 donne une illustration de ces écarts en montrant la frontière efficace exacte (01) et les bornes obtenues, par relaxation linéaire (RL) et par un algorithme glouton, sur une des instances considérées.

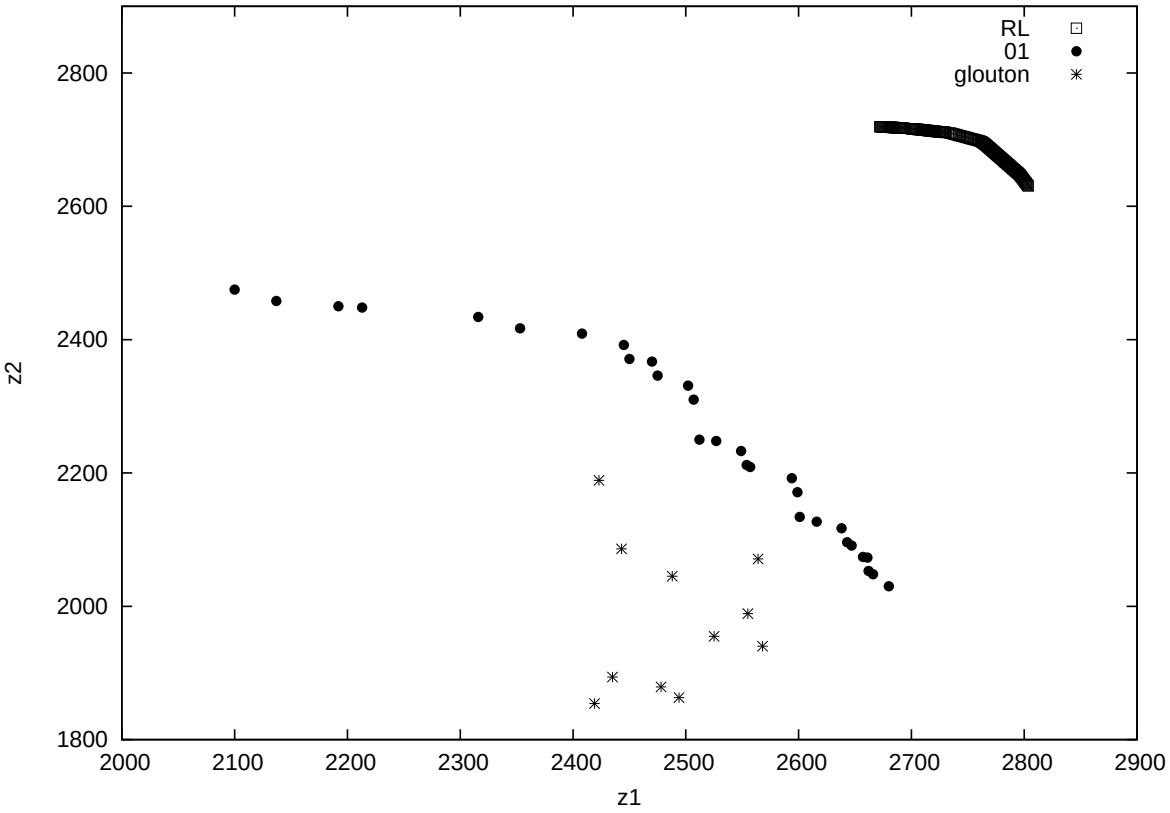


FIG. 5.7 – Exemple de frontière efficace et de bornes

5.3 Résolution approchée

Les résultats présentés dans cette section concernent les algorithmes, de résolution approchée, basés sur la métaheuristique GRASP et présentés dans la section 4.2. Ils reprennent et complètent ceux présentés par Delorme et al. [DGR04, DGD03]. Tous ces résultats ont été obtenus avec des algorithmes utilisant les composants suivants :

- les algorithmes gloutons greedy2 et greedy3 décrits dans la section 4.2.2.1. En effet, des expérimentations préliminaires nous ont montré que l'algorithme glouton greedy1 demandait beaucoup trop de temps pour être utilisé à ce niveau de notre étude.
- les deux techniques de choix du paramètre α proposées dans la section 4.2.2.2 avec
  - $alphaSet = \{0.0, 0.50, 0.75, 0.90, 1.0\}$
  - $|Pool_\alpha| = 5, \forall \alpha \in alphaSet$ (uniquement pour reactive GRASP)

  - condition *probaUpdate* : les probabilités sont mises à jour toutes les 20 itérations de GRASP (uniquement pour reactive GRASP)
  - $\delta = 5$ (uniquement pour reactive GRASP)
- l'algorithme de recherche locale décrit dans la section 4.2.2.3
- la phase d'intensification décrite dans la section 4.2.2.4 avec
  - $|Pool| = 5$
  - Un *Pool* différent pour chaque valeur du paramètre α considérée. En effet, des expérimentations préliminaires avec un *Pool* unique pour toutes les valeurs ont mis en évidence des résultats de qualité inférieure. Cette observation concorde avec les commentaires faits par Glover et Laguna [GL97] sur l'utilisation du path relinking en tant que phase d'intensification.
- le critère d'arrêt : 200 itérations de GRASP.

Seize lancements indépendants de GRASP ont ainsi été effectués pour chaque configuration présentée et pour chaque instance considérée. À chaque lancement, nous considérons la meilleure solution obtenue. Tous les résultats rapportés pour GRASP correspondent (sauf indication contraire) à la moyenne de ces meilleures solutions. Dans la suite, nous allons comparer les différentes stratégies que nous avons proposées, puis étudier l'impact de chacune des phases de GRASP. Une synthèse des résultats obtenus est ensuite présentée.

5.3.1 Impact de chaque stratégie considérée pour GRASP

Pour commencer, nous allons évaluer l'impact de l'utilisation des différentes stratégies proposées (algorithme glouton, choix de la valeur du paramètre α). La Figure 5.8 rapporte ainsi les résultats obtenus avec différentes versions de notre algorithme :

1. une version avec l'algorithme glouton greedy3 (incluant le processus d'apprentissage), et la valeur du paramètre α choisie aléatoirement parmi les valeurs de l'ensemble *alphaSet* à chaque itération de GRASP,
2. une version avec l'algorithme glouton greedy2 et reactive GRASP,
3. une version avec l'algorithme glouton greedy3 et reactive GRASP.

Ces résultats sont présentés en comparaison avec ceux obtenus avec une autre version de GRASP basée sur l'algorithme glouton greedy2 et un choix aléatoire de la valeur du paramètre α, c'est-à-dire une version n'utilisant ni le processus d'apprentissage, ni reactive GRASP. Cette version, appelée GRASP-Ref, correspond à la valeur de 100% dans la figure. En dehors des différences de stratégies, toutes ces versions utilisent les phases de recherche locale et d'intensification. De ce fait, les temps de calcul utilisés par chacune d'entre elles sont similaires. Ainsi, la version 1 permet de mesurer l'impact du processus d'apprentissage seul, la version 2 celui de reactive GRASP seul et la version 3 celui de leur combinaison.

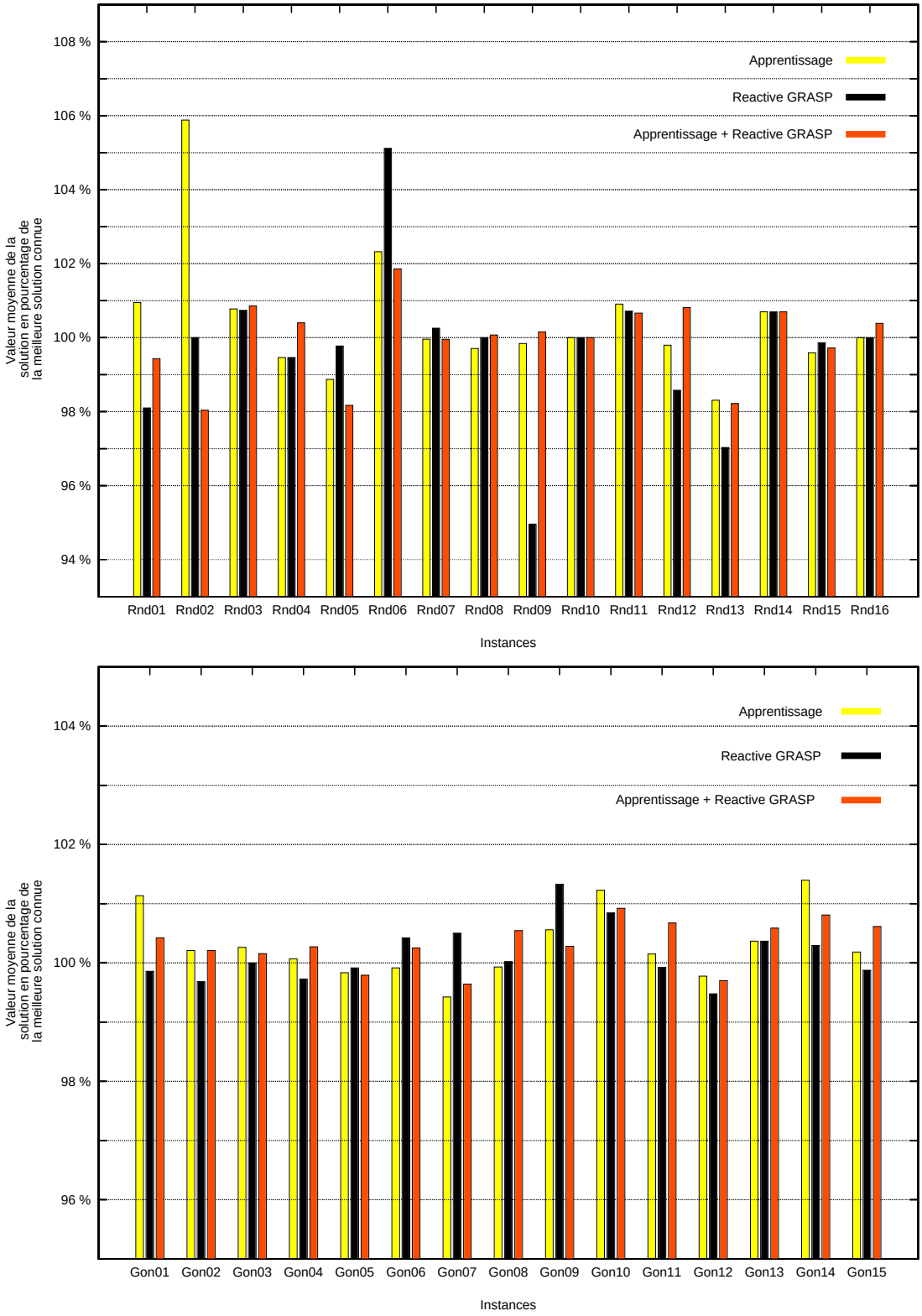


FIG. 5.8 – Impact de chaque stratégie utilisée pour GRASP

En fait, les résultats observés avec chacune des stratégies proposées sont très proches. Chaque version se révèle ainsi la meilleure pour certaines instances, mais aussi la moins bonne pour d'autres : la version 1 est la meilleure pour 10 instances, la version 2 pour 5, la version 3 pour 11 et GRASP-Ref pour 5. De plus, les écarts de performance sont minimes : ils atteignent 8% dans le pire des cas (entre les versions 2 et 4 sur l'instance Rnd2) et sont en général inférieurs à 1%. L'écart de performance moyen sur l'ensemble des instances étudiées est inférieur à 0,5%. Cette proximité de performance ne permet donc pas d'identifier une stratégie surclassant les autres.

Remarquons tout de même que l'utilisation de reactive GRASP seul amène les écarts négatifs les plus importants par rapport à GRASP-Ref (-5% pour Rnd9 et -3% pour Rnd13). Son écart moyen sur l'ensemble des instances est d'ailleurs quasiment nul. Au contraire, le processus d'apprentissage seul présente un impact moyen positif (+0,37% par rapport à GRASP-Ref en moyenne sur l'ensemble des instances). Si la combinaison peut générer des résultats décevants sur certaines instances (particulièrement Rnd2), elle présente elle aussi un écart moyen positif. Étant donné que cette version est celle qui permet d'obtenir la meilleure solution moyenne sur le plus grand nombre d'instances (14 sur 31), nous l'avons retenue pour les expérimentations suivantes.

5.3.2 Impact de chaque phase de GRASP

Nous allons maintenant nous intéresser à l'impact des différentes phases de notre algorithme GRASP (contruction gloutonne aléatoire, recherche locale, intensification avec le path relinking). Comme indiqué précédemment, les résultats rapportés ici ont été obtenus avec la version 3 de notre algorithme (c-à-d celle utilisant l'algorithme greedy3 et reactive GRASP). La Figure 5.9 présente ainsi la valeur moyenne des solutions générées après chaque phase en comparaison avec la meilleure solution connue (qui est égale à 100% dans cette figure). Celle-ci permet donc de mesurer l'importance de chacune des phases dans la qualité globale des solutions. En outre, la Figure 5.10 permet d'observer la répartition du temps de calcul entre ces mêmes phases pour chaque instance.

Les solutions obtenues après la phase de construction gloutonne aléatoire peuvent être très bonnes (et même optimales pour l'instance Rnd10), mais sont la plupart du temps éloignées des meilleures solutions connues (10,7% en moyenne et jusqu'à plus de 20% pour les instances Rnd2 et Gon08). La recherche locale, elle, permet d'améliorer ces résultats de façon significative (+ 5.8% en moyenne et jusqu'à + 13,8% pour l'instance Gon8). Elle a d'ailleurs un impact pour 29 instances sur 31. Cette amélioration est souvent d'autant plus importante lorsque les solutions issues de la phase gloutonne sont de mauvaise qualité (à l'exception des instances Rnd2 et Rnd6). Enfin, la phase de path relinking permet une amélioration supplémentaire des solutions (+ 1.8% en moyenne et jusqu'à + 5,5% pour l'instance Gon8). Le path relinking a d'ailleurs un impact pour 27 instances sur 31.

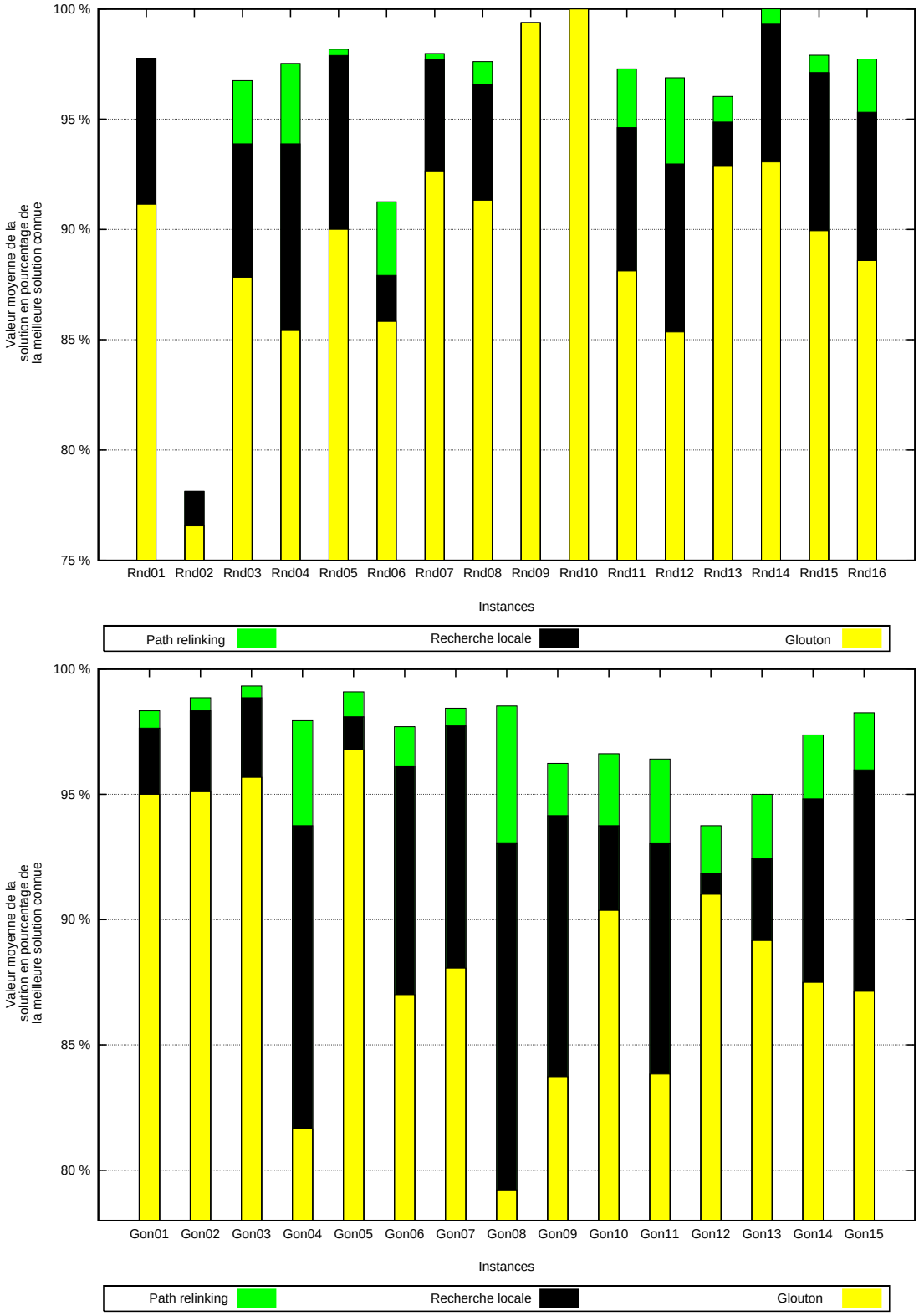


FIG. 5.9 – Impact de chaque phase de notre procédure GRASP

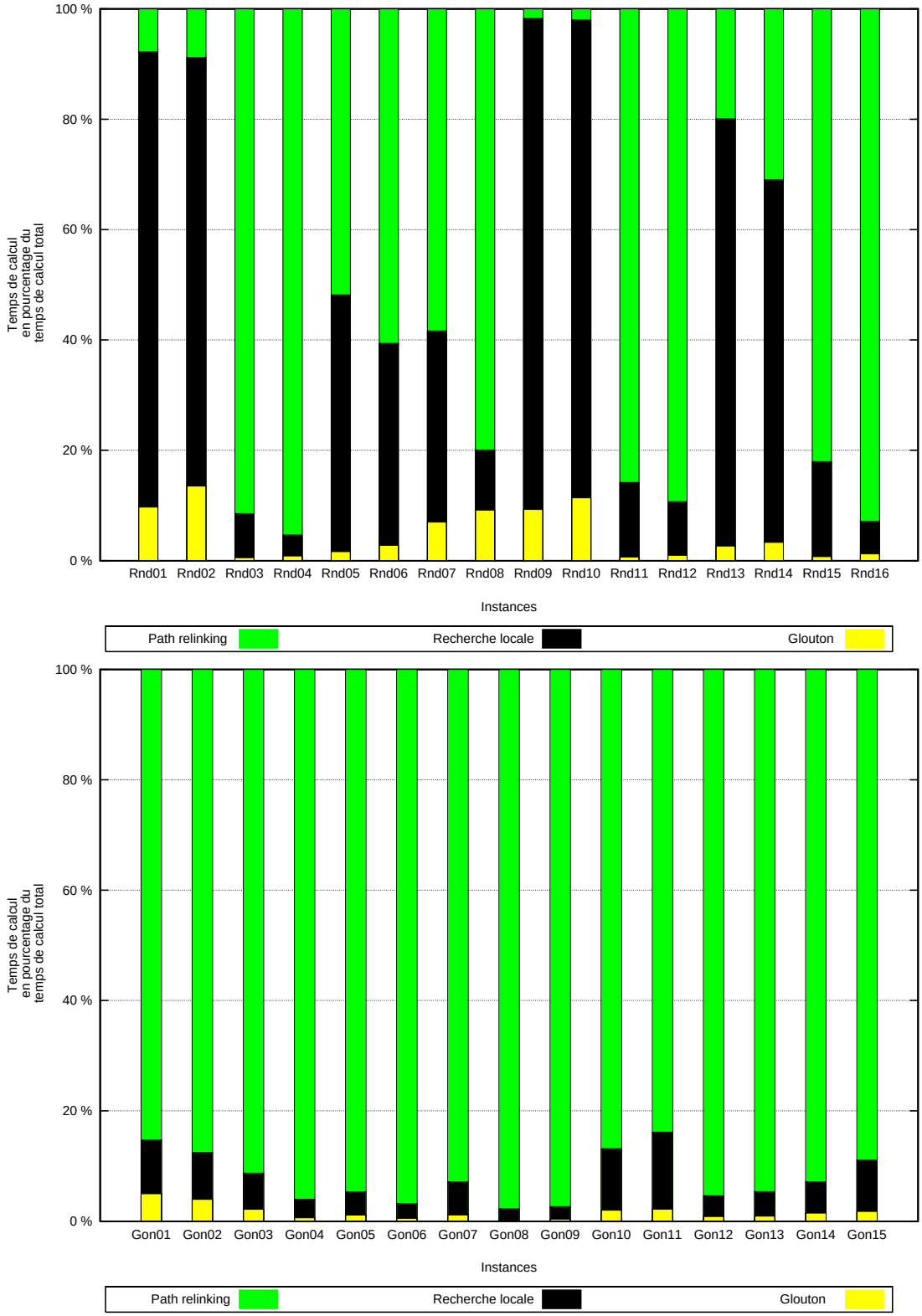


FIG. 5.10 – Répartition par phase du temps utilisé par notre procédure GRASP

Ces améliorations ont cependant un coût important en terme de temps de calcul : les phases de recherche locale et de path relinking utilisent l'essentiel du temps de calcul (respectivement 24,5% et 72,3%). Notons aussi que la phase gloutonne consomme plus de temps pour les instances fortement contraintes (instances Rnd 1, 2, 9 et 10). Ceci est dû au temps nécessaire pour l'évaluation des variables, et ne permet pas pour autant d'obtenir une meilleure qualité des résultats. De même, la recherche locale peut utiliser une grande partie du temps de calcul sans avoir un impact sur la qualité de solution (instances Rnd 9 et 10). Au contraire, le path relinking ne consomme qu'une faible partie du temps de calcul pour les instances où il n'a pas d'impact (instances Rnd 1, 2, 9 et 10). Il est intéressant de noter que ces quatre instances sont les plus contraintes. Les solutions admissibles de ces problèmes contiennent donc peu de variables fixées à 1. Ainsi, l'algorithme de path relinking n'effectue en fait qu'un nombre limité d'itérations sur ces instances.

5.3.3 Solutions obtenues avec notre procédure GRASP complète

Pour finir, la Figure 5.11 résume pour chacune des instances l'intervalle des valeurs minimum et maximum, et la valeur moyenne, observées sur l'ensemble des lancements. Étant donné qu'il n'existe pas d'autre métaheuristique pour le problème de set packing, ces résultats sont indiqués en comparaison avec la meilleure solution connue pour chacune des instances.

Les résultats de notre procédure GRASP sont de bonne qualité pour l'ensemble des instances. En moyenne, ils sont seulement inférieurs de 3,5% par rapport aux solutions optimales (lorsqu'elles sont connues) et de 3,2% par rapport aux meilleures solutions connues. À l'exception de l'instance Rnd2, ils sont au pire 13,3% inférieurs aux meilleures solutions connues ; pour 28 instances sur 31, ils sont au pire 7,9% inférieurs aux meilleures solutions connues. Ces valeurs indiquent une bonne régularité en terme de qualité de solution. Cela s'avère très important dans l'optique d'une utilisation pratique de l'algorithme au sein d'un logiciel d'aide à l'étude de la capacité d'infrastructures ferroviaires.

5.3.4 Approximation de la frontière efficace dans le cas biobjectif

Maintenant que nous avons observé le comportement de notre algorithme GRASP dans le cas mono-objectif, nous allons nous intéresser à son utilisation comme solveur paramétré pour le cas biobjectif. Celle-ci a ainsi été testée sur l'ensemble des instances biobjectifs aléatoires présentées dans la section 5.1.3. Notre évaluation s'est basée sur deux axes :

- l'utilisation d'indicateurs de performance permettant de mesurer la qualité de l'approximation obtenue,
- et la comparaison avec les résultats obtenus dans les mêmes conditions expérimentales avec un autre algorithme.

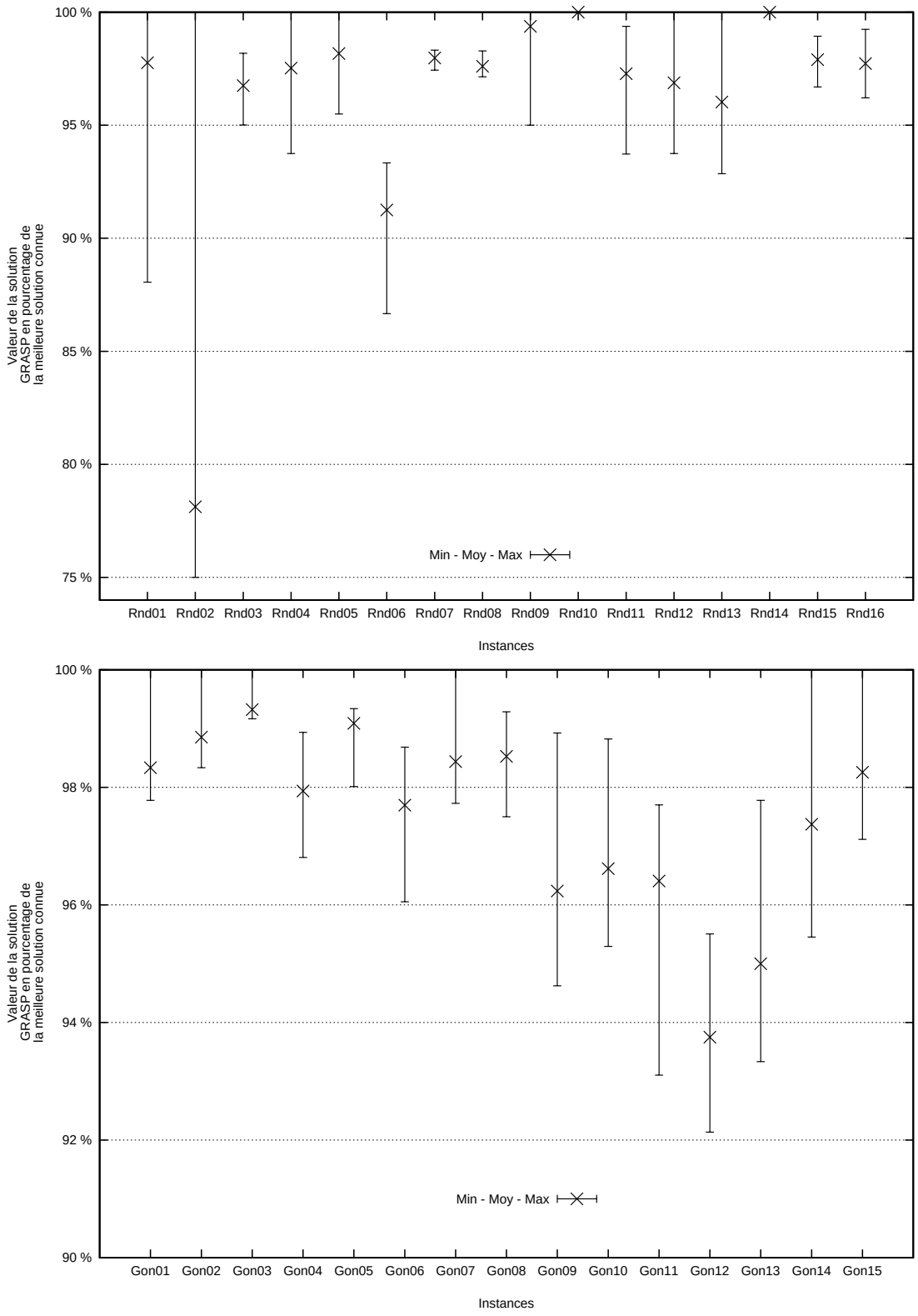


FIG. 5.11 – Synthèse des résultats obtenus avec notre procédure GRASP complète

En effet, nous avons vu dans la section 5.2.3 que les bornes obtenues sur ce problème étaient de mauvaise qualité. Elles ne peuvent donc pas être utilisées pour évaluer nos approximations. De même, aucune relation de dominance n'a pu être observée dans le cas général entre elles.

5.3.4.1 Indicateurs de performance

L'utilisation d'un indicateur de performance unique peut introduire un biais (voir Jaszkiewicz [Jas04]). Ainsi, pour mesurer la qualité des approximations obtenues, nous avons utilisé les trois indicateurs suivants :

1. le pourcentage de solutions efficaces trouvées (M1 proposé par Ulungu et al. [UTF95]),
2. la distance euclidienne moyenne à la frontière efficace,
3. l'hypervolume (S-metric proposé par Zitzler et Thiele [ZT99]) correspondant pour le biSPP à la surface définie dans l'espace des critères par l'ensemble des solutions de l'approximation.

De plus, afin de faciliter la comparaison sur l'ensemble ou sur des familles d'instances, les valeurs de l'indicateur hypervolume ont été exprimées en pourcentage par rapport aux valeurs de cet indicateur pour la frontière efficace (une valeur de 100% correspond donc à une couverture complète de la frontière efficace).

5.3.4.2 SPEA pour le set packing biobjectif

À notre connaissance, la seule méthode proposée pour obtenir une approximation de la frontière efficace du problème de set packing biobjectif est celle présentée par Gandibleux et al. [GDD02]. Celle-ci adapte au biSPP la métaheuristique multiobjectif SPEA (Strength Pareto Evolutionary Algorithm) proposée par Zitzler [Zit99]. SPEA est une métaheuristique évolutionaire multiobjectif avec une évaluation basée sur le concept de dominance Pareto.

Les choix suivants ont été retenus dans cette adaptation. La population initiale est composée de 50 individus, chaque individu correspondant à une solution réalisable. Ces solutions sont toutes différentes vis-à-vis du vecteur de décision X. Elles sont obtenues avec un algorithme glouton appliqué sur 50 directions de recherche différentes (résultant des combinaisons convexes des objectifs $z = \lambda z^1 + (1-\lambda)z^2, \forall \lambda \in \{0, \frac{1}{49}, \ldots, \frac{48}{49}, 1\}$). Lorsque cela ne permet pas d'avoir 50 solutions différentes, la population initiale est complétée avec des solutions aléatoires saturées (c'est-à-dire telles que plus aucune variable ne peut être mise à 1 sans violer au moins une contrainte). L'opérateur de croisement utilisé est un opérateur à un point sélectionné de manière aléatoire sur le gène. À chaque génération, les parents sont remplacés par leurs enfants dans la population dans 80% des cas, et ils sont conservés dans les 20% restants. L'opérateur de mutation est ensuite appliqué à toutes les solutions résultantes. Il autorise la mutation de toutes les variables avec une probabilité d'occurence de 4%. Les solutions obtenues peuvent ainsi violer certaines contraintes, mais une procédure de réparation

(algorithme glouton fixant des variables à 0 tant que toutes les contraintes ne sont pas respectées) permet de les ramener dans le domaine des solutions réalisables. Une procédure de saturation (algorithme glouton fixant des variables à 1 tant que cela est possible en conservant une solution réalisable) permet d'améliorer toutes les solutions générées. Enfin, une sélection par tournois est effectuée afin de limiter la taille de la population.

De plus, afin d'améliorer les performances de l'algorithme sur le biSPP, plusieurs éléments ont été ajoutés. Ainsi, toutes les solutions potentiellement efficaces trouvées sont conservées dans la population (ce qui n'était pas le cas avec la méthode originale). De même, afin d'améliorer les performances de l'algorithme dans les régions de l'ensemble des solutions efficaces proches des solutions extrémales[2], la saturation est effectuée dans trois directions de recherche différentes pour chaque solution (z^1, z^2 et une combinaison des deux objectifs). Enfin, pour diminuer les «trous[3]» observés dans les approximations au cours de premières expérimentations, une recherche locale basée sur des échanges 1-1 (une variable fixée à 0, une autre à 1) a été ajoutée entre chaque génération. Cette procédure est appliquée à toutes les solutions de la population.

5.3.4.3 Résultats observés

Le critère d'arrêt utilisé pour les deux algorithmes est le temps de résolution, fixé à 20 secondes pour les instances à 100 variables et à 80 secondes pour celles à 200 variables. Les résultats indiqués correspondent aux résultats moyens observés sur 16 lancements indépendants de chaque algorithme. Tous ces résultats sont synthétisés pour chaque indicateur, respectivement dans les tableaux 5.5, 5.6 et 5.7.

		A	**B**	**C**	**D**	**E**	**F**	**Total**
SPEA	100 variables	81%	88%	81%	86%	95%	87%	**86%**
	200 variables	71%	68%	71%	80%	72%	81%	**74%**
	Total	**75%**	**76%**	**75%**	**82%**	**82%**	**83%**	**79%**
GRASP	100 variables	91%	92%	85%	94%	84%	88%	**89%**
	200 variables	58%	54%	64%	67%	57%	62%	**60%**
	Total	**72%**	**70%**	**73%**	**79%**	**68%**	**73%**	**72%**

Tab. 5.5 – Pourcentages moyens de solutions efficaces trouvées

Au regard de ces indicateurs, les approximations obtenues avec les deux méthodes sont de bonne qualité. Les résultats sont cependant moins bons sur les instances de plus grande taille (200 variables). À l'examen des indicateurs 2 et 3 (distance moyenne et hypervolume), les résultats de GRASP sur les instances des familles E et F semblent un peu moins satisfaisants.

[2]Une solution s est appelée extrémale si $s \in SE$ et $\exists q \in Q, z^q(s) = \max_{x \in SE} z^q(x)$.

[3]Un trou est une zone de la frontière potentiellement efficace vide de solution.

		A	B	C	D	E	F	Total
SPEA	100 variables	4,00	1,43	1,98	0,60	0,25	0,60	**1,48**
	200 variables	5,08	6,73	6,68	3,44	3,20	1,72	**4,47**
	Total	**4,62**	**4,49**	**4,70**	**2,24**	**1,96**	**1,25**	**3,21**
GRASP	100 variables	0,44	0,71	1,21	0,54	3,12	4,77	**1,80**
	200 variables	8,52	8,53	5,42	6,05	13,61	20,30	**10,40**
	Total	**5,12**	**5,24**	**3,65**	**3,73**	**9,19**	**13,76**	**6,78**

TAB. 5.6 – Distances moyennes à la frontière efficace

		A	B	C	D	E	F	Total
SPEA	100 variables	99,5%	99,4%	99,0%	99,6%	99,7%	99,3%	**99,4%**
	200 variables	98,5%	98,3%	98,9%	99,0%	98,6%	98,7%	**98,7%**
	Total	**98,9%**	**98,8%**	**99,0%**	**99,3%**	**99,1%**	**99,0%**	**99,0%**
GRASP	100 variables	100,0%	100,0%	100,0%	100,0%	99,7%	99,9%	**99,9%**
	200 variables	99,8%	99,7%	99,8%	99,8%	98,9%	99,0%	**99,5%**
	Total	**99,9%**	**99,8%**	**99,9%**	**99,9%**	**99,2%**	**99,4%**	**99,7%**

TAB. 5.7 – Pourcentages moyens de l'hypervolume de la frontière efficace

En fait, ceci est probablement lié à la présence d'un objectif unitaire. En effet, le nombre de solutions efficaces de ces instances est plus faible et les écarts mesurés en pourcentage sont donc plus importants.

Par contre, si SPEA apparaît comme meilleur vis-à-vis des deux premiers indicateurs, GRASP produit de meilleurs résultats vis-à-vis de l'indicateur d'hypervolume. En fait, SPEA génère une population très proche de la frontière et une densité de solutions intéressante, mais éprouve des difficultés à trouver les solutions extrémales. Au contraire, l'approximation produite par GRASP représente une bonne répartition le long de la frontière efficace, mais la densité de solutions obtenues est plus faible (sans doute à cause de l'utilisation de directions de recherche). La Figure 5.12 rapporte les approximations obtenues pour une instance sur une exécution de chacune des heuristiques et illustre bien le comportement caractéristique des deux dernières.

5.4 Conclusion

Dans ce chapitre, nous avons testé les différents algorithmes que nous avons présentés au chapitre 4 et analysé leurs performances. Nous nous sommes appuyés pour cela sur une gamme d'instances mono et biobjectifs. Certaines d'entre elles correspondent au problème ferroviaire étudié dans ce mémoire, mais d'autres instances, générées aléatoirement, ont aussi été utilisées.

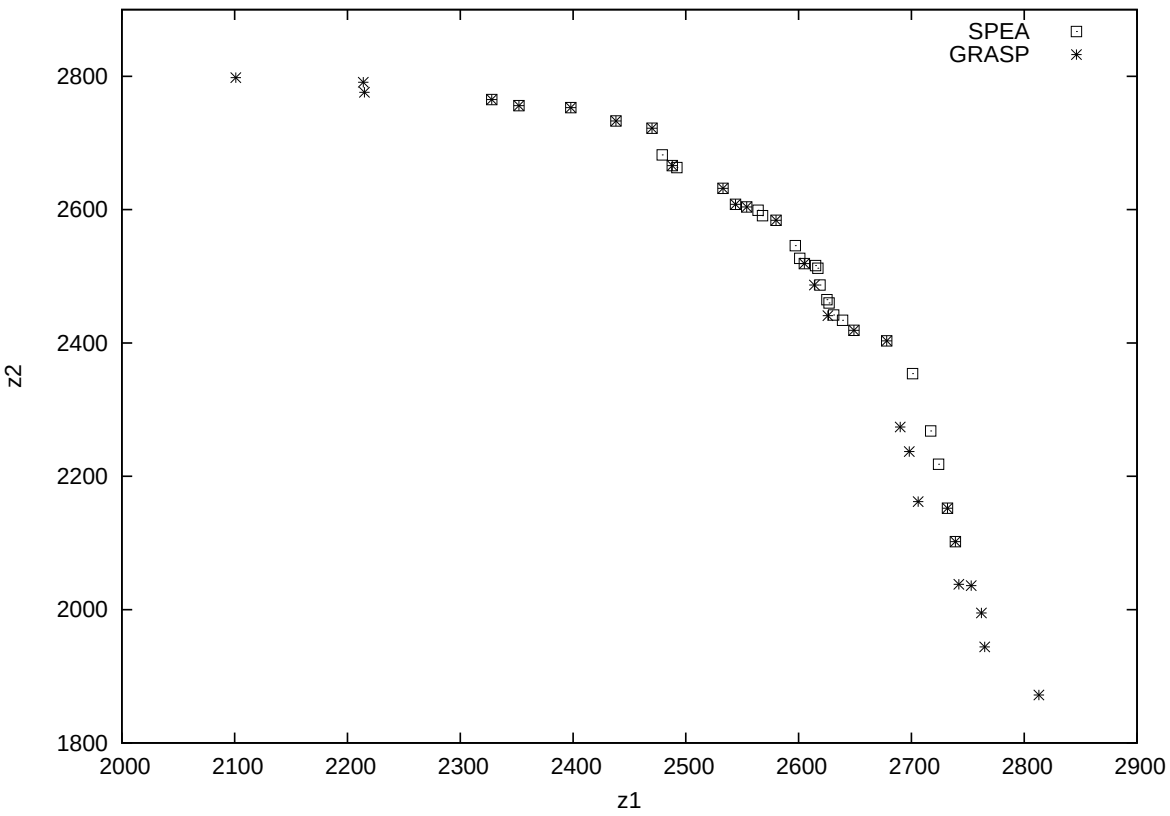


FIG. 5.12 – Exemple d'approximations obtenues

Ces expérimentations numériques ont permis de mettre en évidence l'intérêt des algorithmes de pré-traitements et notamment des recherches de cliques. En effet, ils se sont avérés très efficaces pour réduire la taille des instances considérées et améliorer la qualité de la borne supérieure obtenue. Ils ont ainsi permis d'améliorer de manière significative les résultats obtenus avec Cplex, et de résoudre exactement plusieurs instances. Malgré tout, la résolution exacte de plusieurs des instances considérées est encore impossible, le gap pouvant rester extrêmement important pour certaines d'entre elles.

Au contraire, les résultats enregistrés avec notre algorithme GRASP indiquent qu'il se montre performant sur l'ensemble des instances. Il est ainsi capable de générer des solutions de bonne qualité, même pour des instances de grande taille. Si l'implémentation de reactive GRASP ne donne pas encore entièrement satisfaction, le processus d'apprentissage a un impact positif significatif. De plus, l'apport de chacun des composants utilisés a pu être mesuré. Enfin, l'utilisation de GRASP dans le cas biobjectif, et sa comparaison avec la métaheuristique SPEA, a révélé des résultats très encourageants.

Dans la suite, nous allons voir comment tous les travaux présentés dans cette thèse peuvent être utilisés dans le cadre d'un logiciel d'aide à la décision pour l'étude de la capacité

d'infrastructures ferroviaires.

Chapitre 6

Description de la plate-forme expérimentale

Dans les chapitres précédents, nous avons vu que le problème de capacité d'infrastructures ferroviaires que nous étudions peut se modéliser sous la forme d'un problème de set packing, et que nous disposons d'algorithmes performants pour résoudre ce problème. Nous allons maintenant voir que l'ensemble de ces travaux s'inscrivent dans le cadre d'un projet régional regroupant trois partenaires :

- l'INRETS avec l'Unité de Recherche ESTAS,
- l'UVHC avec l'équipe Recherche Opérationnelle et Informatique du LAMIH,
- et la SNCF avec l'unité Études prospectives Infrastructure (Pôle exploitation) de la Délégation Régionale Infrastructure de Lille.

Ce projet est encore en cours, et certaines des tâches prévues ne sont pas terminées. Après avoir indiqué les objectifs visés, nous détaillons ses principales étapes et présentons l'architecture logicielle considérée pour la plateforme d'expérimentation.

6.1 Le projet RECIFE

Ce projet, baptisé RECIFE [1] (projet GRRT[2]-SPI 2001-2003, voir Rodriguez [Rod01]), vise à développer des outils d'aide à la décision pour l'étude de la capacité d'infrastructures ferroviaires à l'échelle d'une gare ou d'un nœud. Si les principaux problèmes étudiés sont les problèmes de faisabilité et de saturation (voir respectivement les définitions 2.4, page 35, et 2.3, page 35), cette recherche envisage aussi d'autres questions importantes relevant de la problématique liée à la capacité, comme l'optimisation des ressources ou l'appréciation de solutions au regard de différents critères.

[1] REcherche sur la Capacité des Infrastructures FErroviaires
[2] Groupement Régional pour la Recherche dans les Transports

Le chapitre 2 a montré l'importance que revet l'étude de la capacité des infrastructures ferroviaires. Il existe pourtant peu d'études scientifiques s'adressant au problème de l'analyse d'une infrastructure ferroviaire. Ce constat conduit à une quasi-absence d'outils commercialisés, hormis CAPRES (voir section 2.4.2, page 42) pour faire des analyses de capacité. Au niveau national, la Direction de la Recherche et de la Technologie de la SNCF a initié le projet DÉMIURGE (voir section 2.4.3, page 43) qui vise à modéliser la capacité d'un réseau ferroviaire.

Le projet RECIFE se distingue de ces études par une plus grande finesse de modélisation. Ce choix se justifie par l'étendue géographique des types d'infrastructures considérés. Lorsque l'on s'intéresse à une gare ou un noeud ferroviaire, le trafic est très dense et les distances parcourues sont courtes. Des approximations sur la durée des conflits risquent de cacher ou de générer de fausses incompatibilités de circulations. Dans un cas comme dans l'autre, cela change radicalement l'intérêt de la grille horaire obtenue. À cet égard, le projet RECIFE se positionne en complément d'outils d'analyse de réseau comme CAPRES ou DÉMIURGE qui considèrent un niveau de modélisation plus macroscopique.

La région Nord Pas-De-Calais est un contexte de recherche et d'expérimentation riche pour le type de problème traité. Celle-ci est en effet placée en position de carrefour au niveau européen. Elle offre ainsi plusieurs situations réelles propices à valider les techniques proposées comme par exemple le nœud Lillois, la liaison Douai-Ostricourt ou la bifurcation d'Aulnoye. Le noeud Lillois (voir Figure 6.1) a été retenu pour valider le projet RECIFE par un futur utilisateur de la SNCF.

C'est un carrefour ferroviaire dont la complexité révèle son importance dans le réseau national et international. Malgré les multiples transformations qu'il a subies ces dernières années pour faire face aux évolutions de la demande, d'autres transformations seront encore nécessaires. Le PRCI[3] de Lille est aujourd'hui l'un des plus grand de France avec plus de 1100 itinéraires. Plus précisemment, l'infrastructure étudiée dans le cadre du projet RECIFE est la gare de Lille-Flandres.

6.2 Déroulement du projet

En dehors des aspects liés à la modélisation et à la résolution des problèmes de faisabilité et de saturation (qui ont été présentés respectivement aux chapitres 3 et 4), trois autres tâches sont préalables à l'aboutissement du projet :

- le recueil des données du noeud lillois,
- le développement de modules d'analyse et de simulation,

[3]Poste à Relais à Commande Informatique

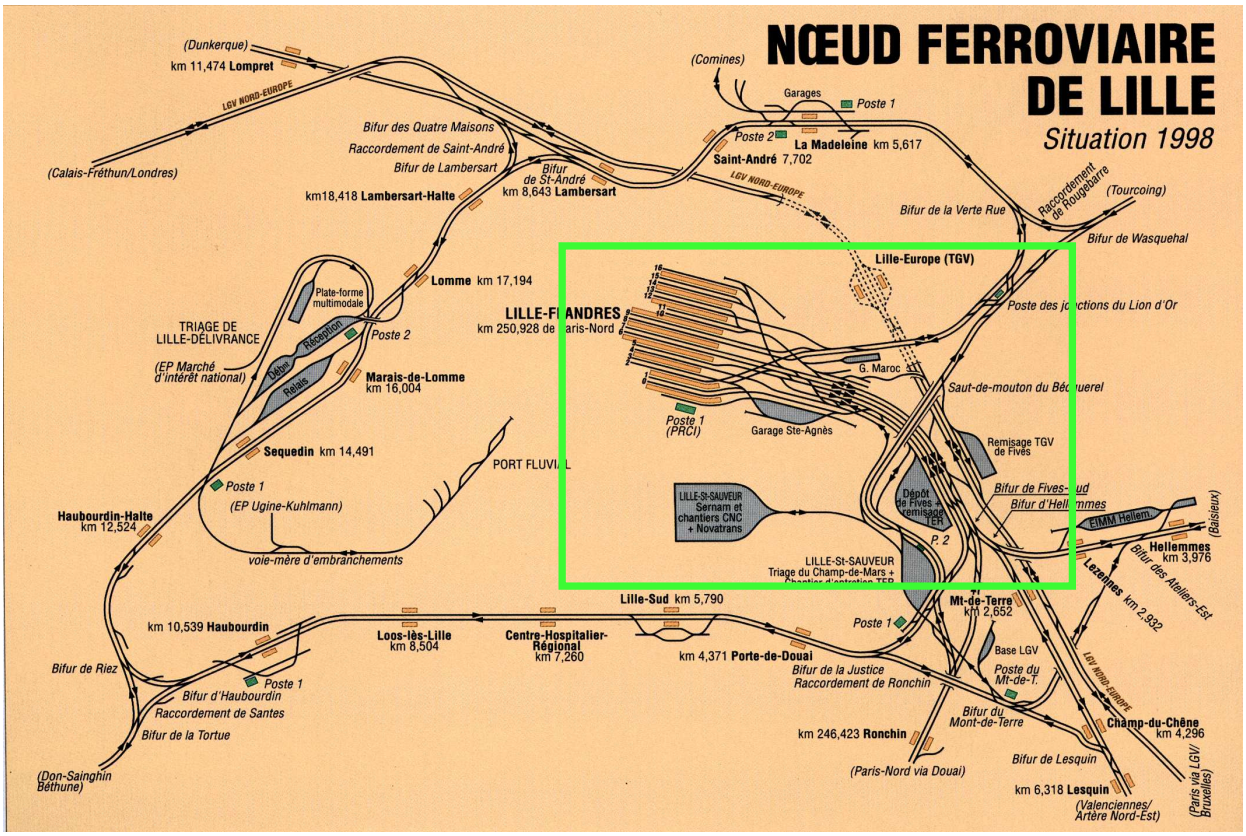


FIG. 6.1 – Schéma de la gare de Lille-Flandres

– et la validation sur le noeud lillois.

En parallèle de ces trois tâches, le développement d'une interface graphique (voir Marlière [Mar01]) est nécessaire pour permettre à l'utilisateur de mener des études de capacité. Cette interface doit aussi permettre de mettre en œuvre une méthode interactive d'aide à la décision prenant en compte différents critères.

6.2.1 Recueil des données du noeud Lillois

Le recueil des données est une tâche complexe et fastidieuse car le volume d'information recherché est très important. En outre, en l'absence d'un système d'information collectant toutes les données, elle nécessite de faire appel à plusieurs sources d'informations. Elle est souvent citée comme un frein à l'utilisation d'outils d'analyse de la capacité. Les trois types de données nécessaires à une analyse de capacité sont :

– les données d'infrastructure,
– les données de l'offre,
– le niveau de qualité souhaité.

Ces données d'infrastructure sont extraites de fichiers issus d'applications SNCF, ou le cas échéant saisies manuellement à partir de documents techniques comme les schémas de signalisation, la *consigne rose* ou le *KVB*. La figure 6.2 représente l'interface graphique permettant de saisir une infrastructure ou de la modifier.

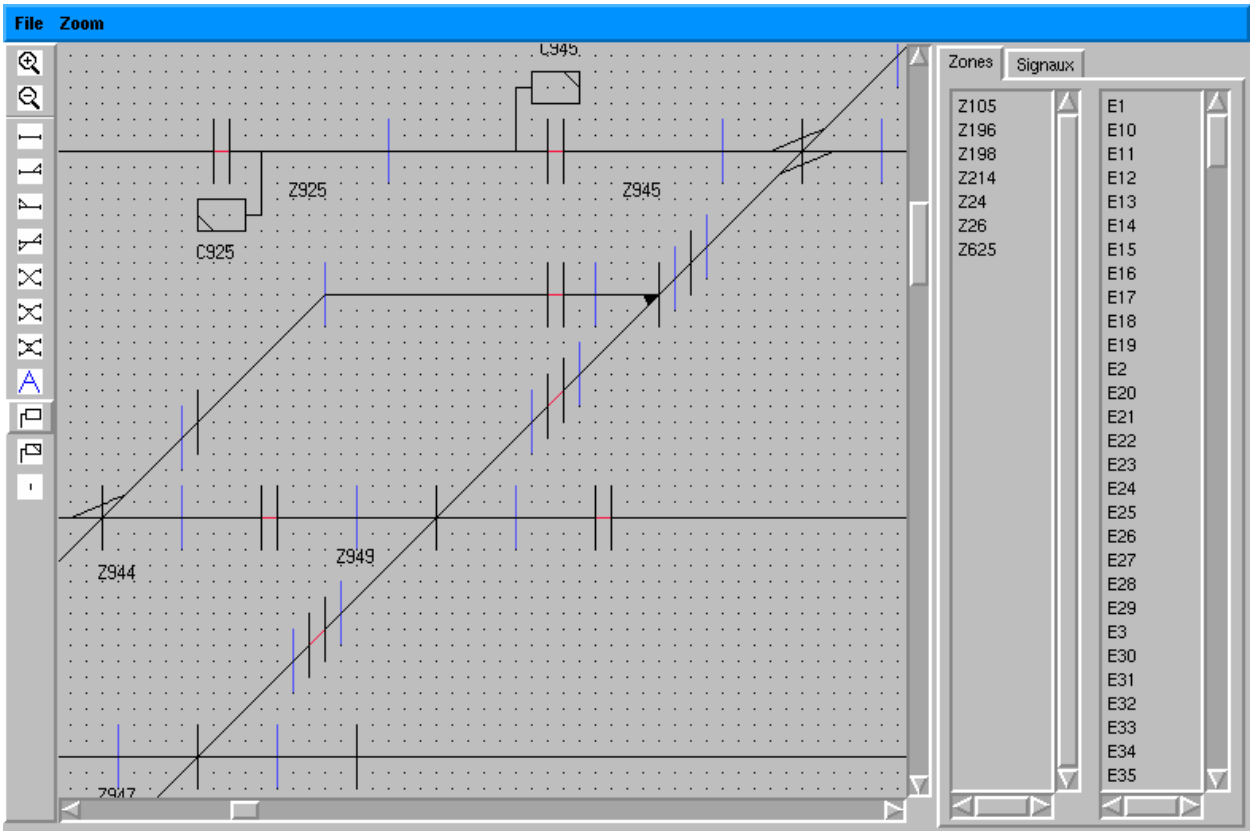


FIG. 6.2 – Édition de l'infrastructure

À l'issue de cette tâche, quatre fichiers sont produits. Le premier fichier décrit l'infrastructure et sa signalisation et notamment les limitations de vitesse. Le second fichier décrit les points d'arrêt et leur durée. Le troisième fichier définit les circulations programmées, y compris les mouvements *haut le pied*. Enfin, pour compléter la description de l'offre, un quatrième fichier donne les contraintes d'*équilibre*.

Cette étape est maintenant pratiquement terminée, et ne demande plus qu'une validation, par un expert de la SNCF, des données saisies.

6.2.2 Modules d'analyse et de simulation

Les modules d'optimisation fournissent des résultats sous la forme de données «brutes» qui rendent difficile le travail d'interprétation. Par exemple, l'énumération des parcours choisis pour chaque circulation n'est pas une information directement exploitable et nécessite un

post-traitement pour rendre accessible ces résultats sous une forme plus ergonomique pour le décideur. Une information plus pertinente est par exemple celle qui permet de se focaliser sur les ressources critiques ou les périodes où l'exploitation est la plus dense.

Lorsque toutes les circulations d'un problème de faisabilité ont été programmées, il est important aussi d'apprécier la marge entre les trains pour faire face aux aléas de l'exploitation (stabilité de la grille horaire, voir section 3.3, page 64). Si au contraire certaines circulations n'ont pu être programmées, il faut pouvoir identifier les circulations en conflit ainsi que l'élément du parcours sur lequel se produit le conflit. Les résultats issus des modules d'optimisation nécessitent donc des post-traitements qui facilitent l'analyse des solutions produites.

À côté de ces modules d'analyse, un second type de modules est utile à l'interprétation des résultats : ce sont des modules de simulation. Ils permettent de déterminer le mouvement complet, et les horaires de réservation des zones, de tous les trains dont le parcours et les horaires de références ont été calculés par le module de résolution. Ces informations peuvent ensuite être utilisées pour visualiser une grille horaire à l'aide de représentations graphiques comme les diagrammes de Gantt d'occupation des zones isolées (voir Figure 6.3) et les diagrammes espace-temps. De plus, les parcours des différents trains peuvent être observés sur le plan de voies de l'infrastructure (voir Figure 6.4). Enfin, un mode simulation permet de visualiser au cours du temps l'évolution de la position de chacun des trains (voir Figure 6.5).

Si plusieurs de ces modules sont terminées, d'autres sont encore en cours de développement.

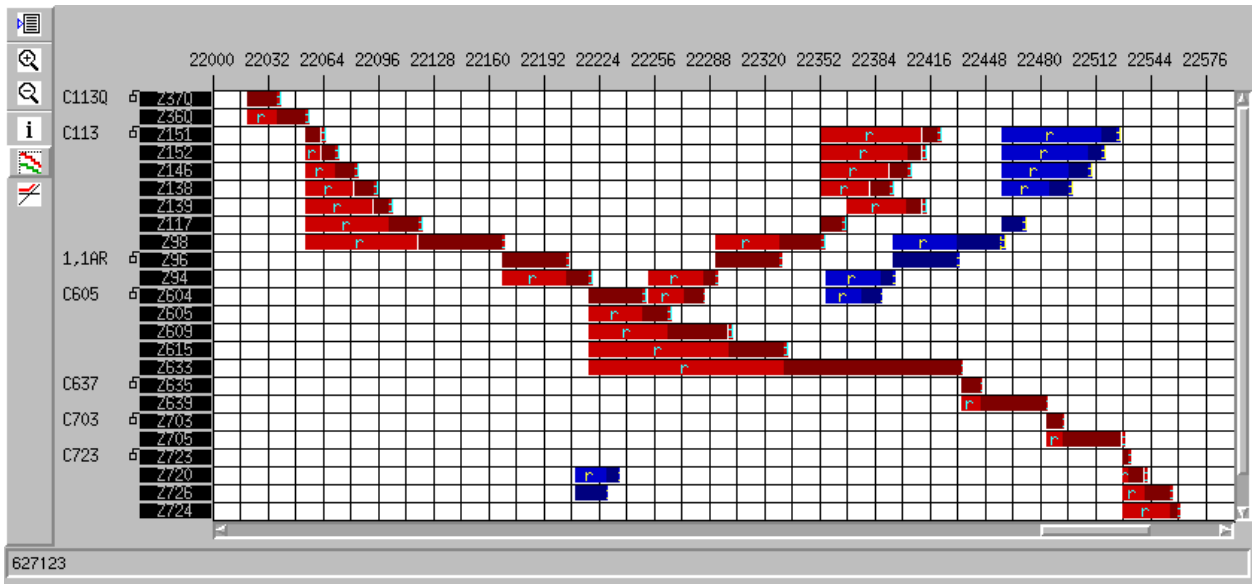


Fig. 6.3 – Visualisation sous la forme d'un diagramme de Gantt

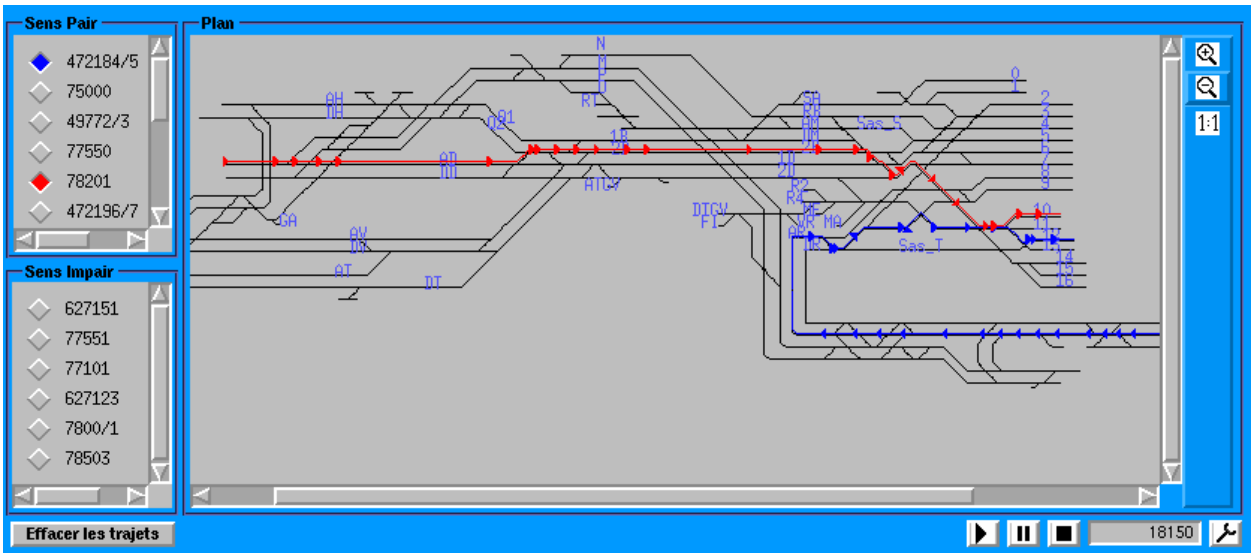


FIG. 6.4 – Visualisation des parcours sur le plan de voies

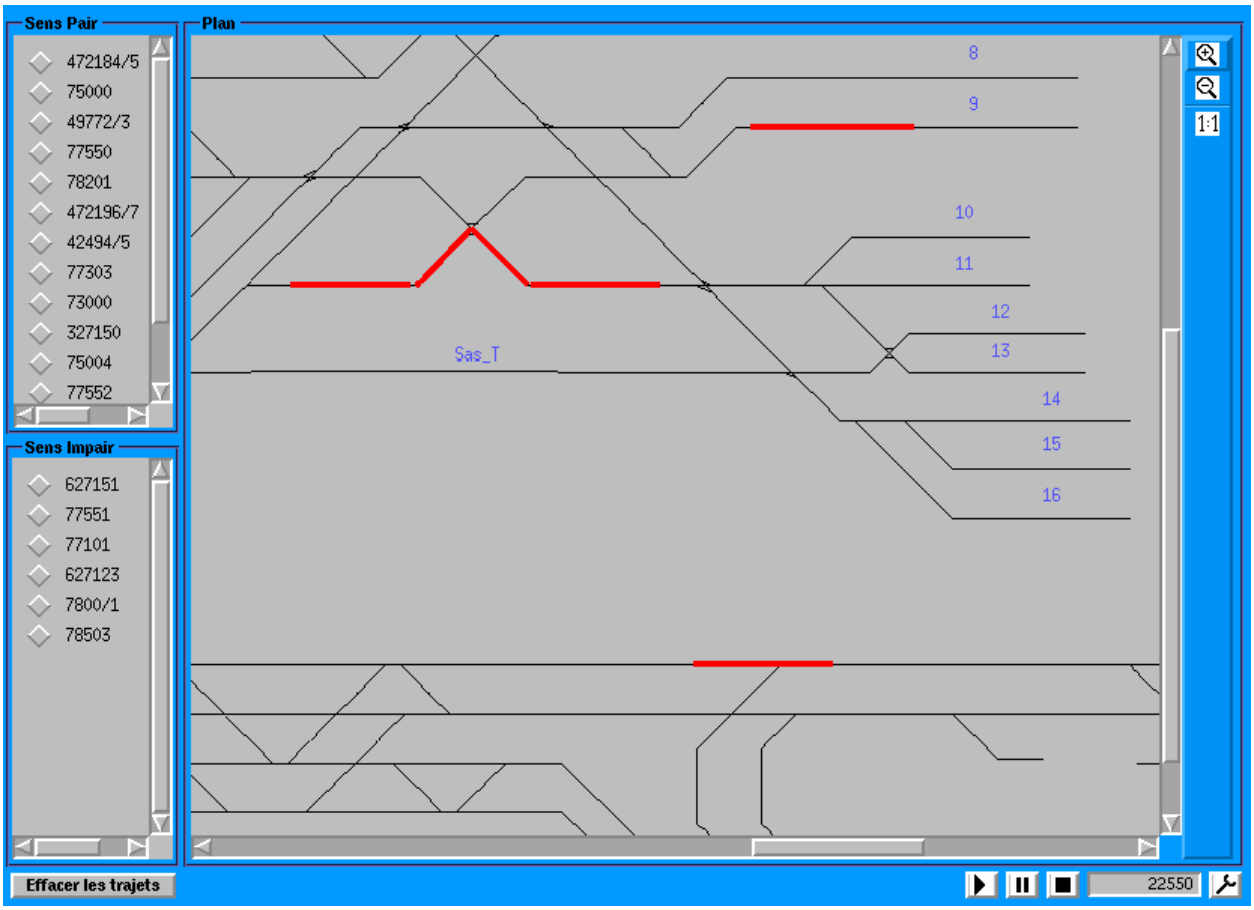


FIG. 6.5 – Visualisation de la position des trains lors d'une simulation

6.2.3 Validation sur le noeud Lillois

Pour finir, tous les modules du projet RECIFE doivent être validés par un futur utilisateur de la SNCF, dans le cadre d'une application sur une situation réelle : la gare de Lille-Flandres. Cette tâche a pour objet de confronter les techniques d'optimisation développées à la réalité d'une étude de capacité sur le noeud lillois. Les dimensions principalement retenues dans cette confrontation sont la flexibilité sur l'utilisation d'hypothèses de modélisation, la traçabilité des hypothèses pour l'interprétation des résultats, et enfin la facilité de mise en œuvre.

Cette étape ne pourra être réalisée que lorsque tous les modules seront terminés.

6.3 Architecture logicielle

La plateforme d'expérimentation du projet RECIFE s'appuie sur des développements et des données provenant d'autres projets réalisés à l'INRETS-ESTAS ayant pour thème l'exploitation ferroviaire (voir figure 6.6). Dans le projet PREDIT[4], baptisé SAFIR [5], de fluidification des circulations ferroviaires, ESTAS avait en charge le développement d'un modèle à base de programmation par contraintes des circulations ferroviaires[Rod98]. Ce travail s'est poursuivi dans un projet prospectif GRRT[Rod00a], [Rod00b] et a abouti sur le développement d'algorithmes basés sur un modèle en programmation par contraintes des circulations ferroviaires. Par conséquent, la plateforme expérimentale s'articule autour de modules d'origines diverses.

Fig. 6.6 – Projets INRETS-ESTAS sur le thème de l'exploitation ferroviaire

L'architecture logicielle résultante pour le projet RECIFE est présentée dans la figure 6.7. Afin d'obtenir le modèle que nous avons présenté au chapitre 3, les horaires de réservation de chacune des zones de l'infrastructure pour chacun des parcours possibles sont nécessaires. Ces données peuvent être calculées une seule fois pour une infrastructure. Elles sont obtenues

[4]Programme National de Recherche et d'Innovation dans les Transports Terrestres
[5]Système d'Aide à la Fluidification des cIRculations

à partir des marches de trains déterminées par le logiciel SISYFE (voir Fontaine et Gauyacq [FG01]). Des modules SAFIR permettent ensuite d'intégrer dans ces horaires les temps de réservation des zones liés à la signalisation (et donc aux cantons). En effet, pendant ces temps, même si le train n'occupe pas encore physiquement une zone qu'il a réservé, aucun autre train ne peut l'utiliser. La conversion des données du format utilisé pour SAFIR au format RECIFE est réalisée par un module développé par Suray [Sur01]. Tous ces modules ont été codés en C++ à l'aide des bibliothèques Ilog Views, Solver et Scheduler [ILO01b, ILO01a].

Pour chaque scénario de données d'exploitation (grille horaire initiale, listes saturantes, préférences, correspondances, ...) étudié, le modèle en variables binaires correspondant peut ainsi être généré. Ce travail est réalisé par une API (codée en C++), développée par Cliqué [Cli02] et Wiart [Wia03]. En outre, celle-ci sert d'interface avec les modules de résolution (codés en Ada) présentés au chapitre 4, et permet de récupérer les grilles horaires correspondant aux solutions générées.

Ces solutions peuvent ensuite être analysées par différents modules (voir section 6.2.2). De plus, les modules de simulation issus de SAFIR permettent de calculer le mouvement de tous les trains de cette grille horaire. Ces informations peuvent ensuite être visualisées grâce à une application regroupant plusieurs interfaces graphiques (codée en C++). Cette application est une évolution de celle développée par Marlière [Mar01] pour le projet SAFIR. Elle permet ainsi de piloter l'ensemble du processus en mettant à jour les données d'infrastructure et d'exploitation des différents scénarios à étudier, et en réglant les paramètres (temps alloué, stratégie retenue, composants utilisés, ...) de la (ou des) méthode(s) de résolution choisie(s).

6.4 Conclusion

Dans ce chapitre, nous avons vu que les travaux présentés dans ce mémoire s'inscrivaient dans le cadre d'un projet régional. Celui-ci, baptisé RECIFE, vise à développer des outils d'aide à la décision pour l'étude de la capacité d'infrastructures ferroviaires à l'échelle d'une gare ou d'un nœud. S'il n'est pas encore terminé, les différents modules développés dans le cadre de ce projet, et son architecture logicielle globale, ont été décrits.

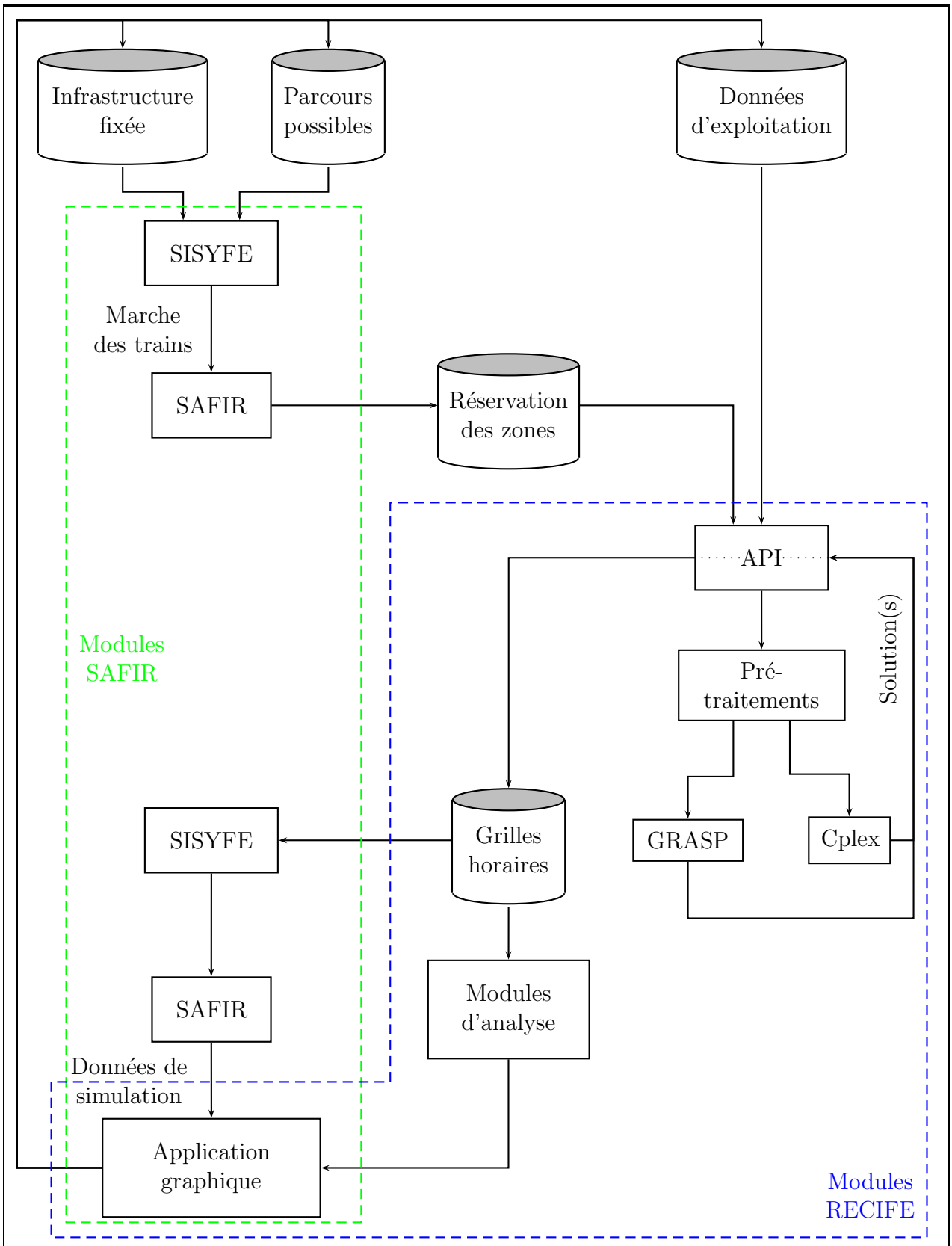


FIG. 6.7 – Architecture de la plate-forme du projet RECIFE

Conclusion générale et perspectives

Les travaux que nous avons présentés dans ce mémoire s'inscrivent tous dans une optique d'évaluation de la capacité d'infrastructures ferroviaires. Ils peuvent cependant être ventilés selon trois axes :

1. la modélisation du problème que nous proposons,
2. les algorithmes de résolution du problème de set packing que nous avons développés,
3. l'intégration de ces travaux dans un logiciel d'aide à la décision pour l'étude de la capacité et leur validation sur des applications réelles.

Dans la suite, nous allons récapituler pour chacun de ces axes les possibilités et les limites du travail accompli. En outre, nous indiquerons ses évolutions possibles et les perspectives de recherche ou d'application qui en découlent.

Modélisation de l'évaluation de la capacité d'infrastructures ferroviaires

Pour commencer, notre travail a permis d'obtenir une **modélisation multiobjectif complète** du problème de l'évaluation et de l'analyse de la capacité d'infrastructures ferroviaires complexes **à l'échelle d'un nœud ou d'une gare**. Elle présente notamment la particularité de considérer un niveau de détail suffisamment fin pour pouvoir obtenir des grilles horaires réalisables dans la pratique sur cette catégorie d'infrastructure. Parmi les nombreuses questions qui se posent dans ce type d'étude, notre modèle permet de considérer les problèmes suivants :

- la vérification de la **faisabilité** du routage d'un ensemble de trains selon une grille horaire établie,
- la détermination du nombre maximum de trains pouvant circuler, pour une configuration d'horaire donnée, par **saturation** d'une grille horaire initiale à l'aide de listes saturantes non ordonnées permettant de considérer séparément différentes catégories de trains,
- l'optimisation d'une grille horaire en fonction de **préférences** émises par le décideur,
- l'**évaluation de la stabilité** des grilles horaires générées.

Ce modèle est entièrement basé sur l'utilisation de variables de décision binaires. Il présente l'avantage d'être linéaire. De plus, sa structure principale est celle de deux problèmes classiques de recherche opérationnelle : l'un, facile, appelé plus court chemin, et l'autre, difficile, appelé set packing.

La modélisation que nous proposons s'est voulue la plus générale possible, de manière à pouvoir traiter un grand nombre de situations. La principale contrepartie de ce choix provient de la complexité de la mise en équations du modèle qui en résulte. En outre, comme pour tout modèle, il a malgré tout été nécessaire de poser certaines hypothèses simplificatrices. Celles-ci sont pour l'essentiel liées à la discrétisation du problème. Nous pensons cependant que ces dernières ne sont pas trop réductrices dans le cadre d'une gestion prévisionnelle de l'utilisation d'infrastructures ferroviaires. Il est par contre fort probable qu'elles se révèleraient beaucoup trop fortes pour un problème lié à l'exploitation en temps réel de ces mêmes infrastructures.

Enfin, même si nous avons essayé de considérer le problème de la manière la plus complète possible, notre modèle permet uniquement d'évaluer la stabilité d'une solution, et non pas de déterminer une solution optimale en terme de stabilité. Une modification du modèle intégrant la stabilité de l'horaire comme l'un des objectifs du problème pourrait s'avérer utile, lors de la programmation de l'exploitation, pour choisir un plan de transport adéquat. De plus, cela permettrait de déterminer de manière précise l'impact de l'ajout d'un train supplémentaire sur une grille horaire.

Résolution du problème de set packing

Le deuxième axe de notre travail concerne la résolution du problème de set packing. En effet, au contraire du problème de plus court chemin, il est connu comme NP-difficile. Il n'existe donc pas d'algorithme permettant de le résoudre de manière exacte en temps polynomial dans le cas général. De plus, il n'existait, jusqu'à maintenant, pas de travaux concernant l'application d'une méthode de résolution approchée basée sur une métaheuristique pour ce type de problème.

Nous avons considéré deux types de méthodes. La première, exacte, est basée sur une méthode de **Branch & Cut** (Cplex) et l'utilisation de **pré-traitements** mettant à profit la structure du problème et les particularités des instances ferroviaires. Nous avons vu au chapitre 5 que les pré-traitements que nous avons développés s'avèraient très efficaces pour réduire la taille des problèmes et améliorer la qualité de la borne supérieure obtenue, notamment sur les instances ferroviaires. Plusieurs instances ont d'ailleurs pu être résolues de

manière exacte grâce à ces pré-traitements. Cependant, en raison de la taille des problèmes considérés dans notre cas, la résolution exacte de ces instances s'est souvent révélée impossible dans un temps raisonnable.

L'implémentation d'une méthode de résolution exacte dédiée pourrait certainement permettre d'améliorer encore la qualité de la borne supérieure. Cette méthode, basée sur une architecture de Branch & Cut, utiliserait, tout au long du processus, les pré-traitements que nous avons développés. L'utilisation des techniques de recherche de cliques, notamment, semble prometteuse. De plus, l'intégration dans cette méthode d'une heuristique performante, pourrait permettre d'améliorer la qualité des solutions entières générées. Si cela ne devrait pas permettre de résoudre toutes les instances que nous avons considérées, la solution optimale de certaines d'entre elles pourrait ainsi être obtenue.

La seconde méthode, approchée, repose sur une adaptation au problème de set packing de la **métaheuristique GRASP**. Plusieurs variantes algorithmiques, inspirées de cette métaheuristique et de certaines de ses extensions les plus connues, ont ainsi été développées et testées. Les résultats observés montrent que notre heuristique est efficace sur une large gamme d'instances du problème. Elle présente une qualité de solution intéressante et relativement stable sur l'ensemble des instances que nous avons traitées. Pour des instances de grande taille, des solutions de bonne qualité ont ainsi été obtenues dans des temps de résolution réduits.

En dépit de ces résultats intéressants, plusieurs pistes restent ouvertes pour améliorer les performances de notre heuristique. Ainsi, il semble que notre procédure reactive GRASP puisse encore être améliorée, notamment pour l'utiliser plus efficacement avec notre processus d'apprentissage. De même, le fait de ne plus considérer le chemin complet entre deux solutions pour le path relinking, mais seulement en partie comme indiqué par Resende et Ribeiro [RR02], pourrait permettre de réduire le temps utilisé sans pour autant compromettre la qualité des solutions.

Le développement d'heuristiques, pour le problème de set packing inspirées par d'autres métaheuristiques, ouvre des perspectives de recherche très nombreuses. Le fait que nous ayons rendu disponibles sur deux sites web [Del, MCD] l'ensemble des instances aléatoires avec lesquelles nous avons travaillé, permet ainsi de proposer une série d'instances de référence sur lesquelles de nouvelles heuristiques pourront être testées et comparées. Nous espérons que cela encouragera le développement de ce genre de travaux.

En outre, nous avons développé une première extension de cette heuristique au cas biobjectif. Celle-ci présente déjà des résultats intéressants, notamment en comparaison avec la métaheuristique multiobjectif classique SPEA. Ainsi, les solutions obtenues s'avèrent très proches de la frontière efficace tout en présentant une répartition régulière le long de celle-ci.

Il semble cependant que des progrès peuvent encore être réalisés. Ainsi, une amélioration possible pourrait venir d'une adaptation plus fine de notre algorithme GRASP au cas biobjectif. Certaines phases comme le path relinking pourraient certainement être plus efficaces en étant utilisées sur des solutions générées pour des directions de recherche différentes. Une autre possibilité serait de développer une phase de post-optimisation permettant de densifier l'approximation de la frontière efficace obtenue. Dans ce cadre, une hybridation avec une méthode évolutionnaire semble être une piste intéressante. Enfin, cette méthode pourrait être étendue pour pouvoir traiter des problèmes avec plus de deux objectifs.

Intégration dans un outil d'aide et validation sur des applications réelles

Le dernier axe correspond au **projet RECIFE** actuellement réalisé en collaboration avec la SNCF. Dans le cadre de ce projet, tous les modèles et les algorithmes de résolutions réalisés sont intégrés dans un logiciel complet. Celui-ci comprend notamment les modules permettant de générer les instances de problèmes et d'interpréter les solutions obtenues avec les méthodes de résolutions développées. Il a ainsi été testé avec succès sur le nœud de Pierrefitte-Gonesse sur lequel de nombreuses expérimentations ont pu être menées.

Dès que les modules d'analyse seront achevés, ce logiciel permettra de réaliser une étude de capacité complète de la gare de Lille-Flandres. Celle-ci servira ainsi à valider le logiciel sur une infrastructure telle qu'une gare. Cette validation pratique achèvera les travaux liés au projet RECIFE. Il serait cependant intéressant de confronter ce logiciel à d'autres infrastructures, en France comme dans d'autres pays européens. À terme, le couplage de ce logiciel avec celui développé actuellement dans le cadre du projet DÉMIURGE, à l'image des projets CADANS et STATIONS au sein du projet DONS, permettrait de disposer d'un outil d'aide à la décision utilisable pour l'étude de la capacité de l'ensemble des sous-parties du réseau ferroviaire français.

Perspective transversale

Pour finir, une perspective transversale aux deux premiers axes est actuellement explorée. En effet, lors d'expérimentations préliminaires (voir Delorme et al. [DRG01] et Rodriguez et al. [RDG02]), nous avons montré la complémentarité de notre approche avec une autre, basée sur un modèle et des techniques de résolution issus de la **programmation par contraintes**. Ainsi, par exemple, si notre approche s'avérait plus performante dans le choix des parcours à attribuer aux trains, celle basée sur la progammation par contraintes permettait de considérer une discrétisation plus fine des horaires de références.

Une amélioration du processus de résolution pourrait ainsi résulter d'une hybridation de

ces deux modèles, notamment par l'utilisation de techniques de décomposition du problème et par la définition de contraintes globales. Cette hybridation pourrait, par exemple, servir à améliorer les horaires de références d'une grille horaire dont les parcours seraient fixés par notre méthode. En outre, l'utilisation en parallèle des deux modèles pourrait servir à mettre à profit les contraintes supplémentaires trouvées sur les deux modèles (comme par exemple les cliques du modèle linéaire) pour restreindre le domaine de recherche. Ces différentes possibilités sont ainsi étudiées dans le cadre d'une thèse en cours.

Glossaire

Blanc-travaux : période pendant laquelle aucune circulation n'est prévue sur une voie donnée, de façon à permettre l'exécution de la maintenance courante de l'infrastructure. À la SNCF, cette période est généralement d'au moins 1h50 par voie et par jour.

Cadence : durée du cycle de l'horaire. Un horaire cadencé est conçu de façon à pouvoir se répéter à la fin de chaque cycle : il ne possède ni début ni fin. Dans la pratique, il va de soi que l'horaire possède un début et une fin, ce qui rompt bien entendu sa cadence.

Canton : section de voie où figure en amont de chaque point d'entrée un signal. Le signal précise si le canton est libre ou occupé par un train. Les états possibles d'un signal sont appelés aspects de signalisation ; chaque aspect est interprété par le mécanicien dans la façon de de conduire le train.

Consigne rose : document technique décrivant les conditions d'établissement et de libération des itinéraires.

Équilibre : contrainte sur les lieux de départ et de destination permettant de former un train à partir du matériel roulant d'un ou de deux autres trains.

Grille horaire : terme caractérisant en général le graphique d'un tronçon de ligne et représentant les heures de passage prévues des trains sur chacune des voies de la zone considérée.

Haut le pied : train roulant «à vide» entre le dépôt et le lieu de départ d'une «circulation commerciale».

KVB : système de contrôle des vitesses par balises.

Ligne : suite ordonnée de tronçons et de nœuds formant un chemin ; l'origine de chaque tronçon doit coïncider avec la destination du tronçon précédent. Généralement, le parcours

d'un train coïncide partiellement ou totalement avec une ligne ferroviaire bien définie.

Marche d'un train : circulation réelle d'un train caractérisée par son passage à un point donné, à une vitesse donnée et à une date donnée.

Mécanicien : nom donné au conducteur d'un train.

Nœud : point critique du réseau, tels une gare, une bifurcation, une jonction, un point de croisement, y compris tous les appareils de voie et voies qui y sont rattachés.

Plan de transport : ensemble des dispositions prévoyant l'horaire des circulations et les prestations assurées par les matériels et les personnes concernées.

Qualité de service : notion regroupant plusieurs critères qualitatifs et quantitatifs. Du point de vue de la clientèle du rail (voyageurs ou fret), la qualité de service s'appuie sur les axes suivants :

- des temps de parcours réduits (de point à point, c'est-à-dire y compris les transbordements)
- des fréquences étoffées
- une bonne ponctualité (respect de l'horaire)
- un confort de voyage et d'attente agréable (sièges, bruit, propreté, possibilités offertes...)
- un prix réduit

Par rapport à l'étude de la capacité, la qualité de service est souvent prise en compte par les temps de parcours, les fréquences et la ponctualité (stabilité de l'horaire).

Routage : affectation de chacun des trains arrivant dans une zone à un parcours lui permettant de rejoindre sa destination.

Saturation :
1. état d'un réseau dans lequel il n'est plus possible d'ajouter un train.
2. pratique consistant à introduire dans le réseau une série de trains, dits saturants, jusqu'à ce qu'il ne soit plus possible de le faire. La capacité du réseau est définie par le nombre de trains à l'horaire saturé.

Signalisation : dispositifs techniques et procédures liés aux fonctions de conduite des trains qui visent à garantir la sécurité des circulations.

Sillon : capacité d'infrastructure requise (voie et horaire) pour faire circuler un train

donné d'un point à un autre à un moment donné (directive 95/19/CE du conseil de l'union européenne).

Réseau : ensemble de lignes.

Taux de saturation : $S = \frac{Nombre\ de\ trains}{Capacité\ pratique}$, un niveau de saturation supérieur à 100 % n'étant pas aberrant mais correspondant à une saturation ne respectant pas strictement les normes de qualité recommandées.

Tronçon : partie d'une ligne composée éventuellement de plusieurs voies reliant deux nœuds du réseau.

UIC ou Union Internationale des Chemins de fer : organisme international faisant office de référence sur certains points précis. La majorité des réseaux ferrés du monde sont adhérents à l'UIC.

Voie banalisée : voie autorisant la circulation des trains dans les deux sens.

Voie libre : fait pour le mécanicien de n'apercevoir que des signaux lui permettant d'adopter la vitesse maximum autorisée.

Zone de détection : section de la voie permettant la détection de la présence d'un train ; cette détection se fait le plus souvent à l'aide d'un dispositif électrique appelé «circuit de voie».

Bibliographie

[APR99] James Abello, Panos M. Pardalos, et Mauricio G.C. Resende. On maximum clique problems in very large graphs. Dans J. Abello et J. Vitter, éditeurs, *External memory algorithms and visualization*, volume 50 de *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 119–130. American Mathematical Society, 1999.

[Bel97] H. Bellaiche. Recherche sur la saturation des lignes ferroviaires (rapport d'étape de la phase 1). Rapport technique 2166/ESF/317-97/RA, SYSTRA, France, mai 1997.

[Bor98] Ralf Borndörfer. *Aspects of set packing, partitioning, and covering.* PhD thesis, Fachbereich 3 Mathematik der Technischen Universität Berlin, Berlin, Germany, 1998.

[BWZ97] M.R. Bussieck, T. Winter, et U.T. Zimmermann. Discrete optimization in public rail transport. *Mathematical programming*, 79 :415–444, 1997.

[CL01] Anne Curchod et Luigi Lucchini. CAPRES : Description générale du modèle. Rapport technique 788/5_f, LITEP, juin 2001.

[Cli02] Manuel Cliqué. Développement d'une API C++ pour la génération du modèle SPP appliqué à un problème de faisabilité et de saturation ferroviaire. Rapport technique RE-02-710-FR, INRETS-ESTAS, 2002.

[CTV98] Jean-François Cordeau, Paolo Toth, et Daniele Vigo. A survey of optimization models for train routing and scheduling. *Transportation Science*, 32(4) :380–404, novembre 1998.

[Deg02] Fabien Degoutin. *Problèmes bi-objectifs en optimisation combinatoire et capacité d'infrastructures ferroviaires.* Mémoire de DEA, Université de Valenciennes et du Hainaut Cambrésis, Valenciennes, France, 2002.

[Del] Xavier Delorme. Instances tests pour le problème de set packing. `http://www3.inrets.fr/~delorme/Instances-fr.html`.

[Del00] Xavier Delorme. *Optimisation combinatoire et problèmes de capacité d'infrastructure ferroviaire.* Mémoire de DEA, Université de Valenciennes et du Hainaut Cambrésis, Valenciennes, France, 2000.

[DG02] Fabien Degoutin et Xavier Gandibleux. Un retour d'expériences sur la résolution de problèmes combinatoires bi-objectifs. 5ème journée du groupe de travail Programmation Mathématique MultiObjectif (PM20), Angers, France, mai 2002.

[DGD03] Xavier Delorme, Xavier Gandibleux, et Fabien Degoutin. Résolution approchée du problème de set packing bi-objectifs. Dans *Proceedings de l'École d'Automne de Recherche Opérationnelle de Tours (EARO)*, pages 74–80, 2003.

[DGR03a] Xavier Delorme, Xavier Gandibleux, et Joaquín Rodriguez. Modélisation du routage des trains dans un nœud complexe et étude de capacité. 5ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2003), Avignon, France, 26-28 février 2003.

[DGR03b] Xavier Delorme, Xavier Gandibleux, et Joaquín Rodriguez. Résolution d'un problème d'évaluation de capacité d'infrastructure ferroviaire. Dans Groupement Régional Nord-Pas de-Calais pour la Recherche dans les Transports (GRRT), éditeur, *Proceedings du colloque international sur l'Innovation Technologique pour les Transports Terrestres (TILT)*, pages 647–654, 2003.

[DGR04] Xavier Delorme, Xavier Gandibleux, et Joaquín Rodriguez. GRASP for set packing problems. *European Journal of Operational Research*, 153 (3) :564–580, 2004. (À paraître).

[DRG01] Xavier Delorme, Joaquín Rodriguez, et Xavier Gandibleux. Heuristics for railway infrastructure saturation. Dans L. Baresi, J-J. Lévy, R. Mayr, M. Pezzè, G. Taentzer, et C. Zaroliagis, éditeurs, *ICALP 2001, proceedings of the satellite workshops of the 28th international colloquium on automata, languages, and programming*, volume 50 de *Electronic Notes in Theoretical Computer Science (URL : http ://www.elsevier.nl/locate/entcs/volume50.html)*, pages 41–55. Elsevier Science, 2001.

[EG01] Matthias Ehrgott et Xavier Gandibleux. Bounds and bound sets for biobjective combinatorial optimization problems. Dans M. Koksalan et St. Ziont, éditeurs, *Multiple Criteria Decision Making in the New Millennium, proceedings of the Fifteenth International Conference on Multiple Criteria Decision Making (MCDM)*, volume 507 de *Lecture Notes in Economics and Mathematical Systems*, pages 241–253. Springer, 2001.

[EG02a] Matthias Ehrgott et Xavier Gandibleux. Multiobjective combinatorial optimization. Dans M. Ehrgott et X. Gandibleux, éditeurs, *Multiple Criteria Optimization : State of the Art Annotated Bibliographic Survey, Kluwer's International Series in Operations Research and Management Science*, volume 52, pages 369–444. Kluwer Academic Publishers, Boston, 2002.

[EG02b] Matthias Ehrgott et Xavier Gandibleux, éditeurs. *Multiple Criteria Optimization : State of the Art Annotated Bibliographic Survey, Kluwer's International Series in Operations Research and Management Science.* Kluwer Academic Publishers, Boston, 2002.

[Esö99] Márta Esö. *Parallel branch & cut for set partitioning.* PhD thesis, Faculty of the Graduate School of Cornell University, Ithaca, USA, 1999.

[Eur91] Conseil des Communautés Européennes. Directive 91/440/CEE relative au développement de chemins de fer communautaires. Journal officiel n°L237, 29 juillet 1991.

[Fau91] Robert Faure. *Précis de recherche opérationnelle.* Dunod Décision, Paris, 1991.

[FG01] Michèle Fontaine et Daniel Gauyacq. SISYFE : a toolbox to simulate the railway network functioning for many purposes. some cases of application. Dans *Proceedings of the World Congress on Railway Research (WCRR 2001)*, 2001.

[FR89] Thomas A. Féo et Mauricio G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8 :67–71, 1989.

[FR95] Thomas A. Féo et Mauricio G.C. Resende. Greedy randomized adaptative search procedures. *Journal of Global Optimization*, 6 :109–133, 1995.

[FR02] Paola Festa et Mauricio G.C. Resende. GRASP : an annotated bibliography. Dans C. C. Ribeiro et P. Hansen, éditeurs, *Essays and surveys on metaheuristics*, pages 325–367. Kluwer academic publishers, Boston, 2002.

[FRS94] Thomas A. Féo, Mauricio G.C. Resende, et Stuart H. Smith. A greedy randomized adaptative search procedure for maximum independant set. *Operations Research*, 42 :860–878, 1994.

[Fré03] Arnaud Fréville. The multidimensional 0-1 knapsack problem : an overview. *European Journal of Operational Research*, 2003. (À paraître).

[FS87] K. Fukumori et H. Sano. Fundamental algorithm for train scheduling based on artificial intellingence. *Systems and Computers in Japan*, 18(3) :52–64, 1987.

[Gan01] Xavier Gandibleux. Quick evaluations of the efficient solution set for the biobjective knapsack problem. Dans M. Koksalan et St. Ziont, éditeurs, *Multiple Criteria Decision Making in the New Millennium, proceedings of the Fifteenth International Conference on Multiple Criteria Decision Making (MCDM)*, volume 507 de *Lecture Notes in Economics and Mathematical Systems*, pages 254–264. Springer, 2001.

[GDD02] Xavier Gandibleux, Fabien Degoutin, et Xavier Delorme. A first feedback on set packing problems with two objectives. Workshop on Multiple Objective Metaheuristics (MOMH), Paris, France, 4-5 novembre 2002.

[Gia03] Vincent Giard. *Gestion de la production*. Economica, Paris, 2003.

[GJ79] Michael R. Garey et David S. Johnson. *Computers and intractability : a guide to the theory of NP-Completeness*. V.H. Freeman and Company, San Francisco, 1979.

[GL97] Fred Glover et Manuel Laguna. *Tabu Search*. Kluwer academic publishers, Boston, 1997.

[GM95] Michel Gondran et Michel Minoux. *Graphes et algorithmes*. Eyrolles, Paris, 1995.

[GVT98] Xavier Gandibleux, David Vancoppenolle, et Daniel Tuyttens. A first making use of GRASP for solving MOCO problems. 14th International Conference in Multiple Criteria Decision-Making, Charlottesville, USA, 8-12 juin 1998.

[Hac97] Patrick Hachemane. *Évaluation de la capacité de réseaux ferroviaires*. Thèse 1632, École Polytechnique Fédérale de Lausanne, Lausanne, Suisse, 1997.

[Han79] Pierre Hansen. Bicriterion path problems. Dans Fandel G. et Gal T., éditeurs, *Multiple Criteria Decision Making theory and application*, volume 177 de *Lecture Notes in Economics and Mathematical Systems*, pages 109–127. Springer, 1979.

[HJ98] Michael P. Hansen et Andrzej Jaszkiewicz. Evaluating the quality of approximations to the non dominated set. Rapport technique IMM-REP-1998-7, Institute of Mathematical Modelling Technical University of Denmark, 1998.

[ILO01a] ILOG. Scheduler 5.1 (user's manual), 2001.

[ILO01b] ILOG. Solver 5.1 (user's manual), 2001.

[ILO02] ILOG. CPLEX 8.0 (user's manual), 2002.

[Jas01] Andrzej Jaszkiewicz. *Multiple objective metaheuristic algorithms for combinatorial optimization*. Habilitation thesis, Poznan University of Technology, Poznan, Poland, 2001.

[Jas04] Andrzej Jaszkiewicz. Evaluation of multiple objective metaheuristics. Dans X. Gandibleux, M. Sevaux, K. Sörensen, et V. T'kindt, éditeurs, *Metaheuristics for Multiobjective Optimisation, proceedings of the Workshop on Multiple Objective Metaheuristics (MOMH)*, volume 535 de *Lecture Notes in Economics and Mathematical Systems*, pages 65–89. Springer, 2004. (À paraître).

[KL97] Si-Hwa Kim et Kyung-Keun Lee. An optimization-based decision support system for ship scheduling. *Computers and Industrial Engineering*, 33 :689–692, 1997.

[KRZ97] Léo G. Kroon, H. Edwin Romeijn, et Peter J. Zwaneveld. Routing trains through railway stations : complexity issues. *European Journal of Operational Research*, 98 :485–498, 1997.

[LD01] Véronique Labouisse et Housni Djellab. DEMIURGE : a tool for the optimisation and the capacity assessment for railway infrastructure. Dans *Proceedings of the World Congress on Railway Research (WCRR 2001)*, 2001.

[LD02] Véronique Labouisse et Housni Djellab. DEMIURGE : un outil d'optimisation et d'évaluation de la capacité d'infrastructure ferroviaire. 4ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2002), Paris, France, 20-22 février 2002.

[Lin00] Thomas Lindner. *Train Schedule Optimization in Public Rail Transport*. PhD thesis, Fachbereich für Mathematik und Informatik der Technischen Universität Braunschweig, Braunschweig, Germany, 2000.

[LJE03] Laszlo Ladanyi, David Jensen, et Márta Esö. Solving lexicographic multiobjective mips with branch-cut-price. EURO XIX, Istanbul, Turquie, juillet 2003.

[LM99] Manuel Laguna et Rafael Martí. Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11 :44–52, 1999.

[Mar01] Grégory Marlière. Développement d'interfaces graphiques pour la gestion des circulations ferroviaires. Rapport technique RE-01-708-FR, INRETS, juin 2001.

[MCD] MCDM society. Site web. `http://www.terry.uga.edu/mcdm/`.

[MMRB98] Aristide Mingozzi, Vittorio Maniezzo, Salvatore Ricciardelli, et Lucio Bianco. An exact algorithm for the project scheduling with ressource constraints based on a new mathematical formulation. *Management Science*, 44(5) :714–729, mai 1998.

[NW99] George L. Nemhauser et Laurence A. Wolsey. *Integer and combinatorial optimization*. Willey-Interscience, New York, 1999.

[OL96] I.H. Osman et G. Laporte. Metaheuristics : a bibliography. *Annals of Operations Research*, 63 :513–623, 1996.

[Pad73] Manfred W. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5 :199–215, 1973.

[Pir02] Marc Pirlot. Métaheuristiques pour l'optimisation combinatoire : un aperçu général. Dans J. Teghem et M. Pirlot, éditeurs, *Optimisation approchée en recherche opérationnelle - Recherches locales, réseaux neuronaux et satisfaction de contraintes*, pages 25–55. Hermès Science Publications, Paris, 2002.

[PR00] Marcelo Prais et Celso C. Ribeiro. Reactive GRASP : An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12 :164–176, 2000.

[PR02] Leonidas S. Pitsoulis et Mauricio G.C. Resende. Greedy randomized adaptive search procedures. Dans P.M. Pardalos et M.G.C. Resende, éditeurs, *Handbook of Applied Optimization*, pages 168–183. Oxford University Press, New York, 2002.

[RDG02] Joaquín Rodriguez, Xavier Delorme, et Xavier Gandibleux. Railway infrastructure saturation using constraint programming approach. Dans J. Allan, R.J. Hill, C. A. Brebbia, G. Sciutto, et S. Sone, éditeurs, *Computers in Railway VIII, proceedings of*

the Eighth International Conference on Computer Aided Design, Manufacture and Operation in the Railway and Other Advanced Mass Transit Systems (COMPRAIL 2002), volume 13 de *Advances in Transport*, pages 807–816. Wit Press, 2002.

[Ree95] Colin Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. McGraw-Hill, London, 1995.

[Rod98] Joaquín Rodriguez. Technique d'aide à la fluidification des circulations dans un noeud ferroviaire complexe, résultats obtenus et analyses. Rapport d'avancement 98-53, INRETS-ESTAS, décembre 1998.

[Rod00a] Joaquín Rodriguez. Empirical study of a constraint programming model applied to a railway traffic management problem. Dans J. Allan, R.J. Hill, C. A. Brebbia, G. Sciutto, et S. Sone, éditeurs, *Computers in Railways VII, proceedings of the Seventh International Conference on Computer Aided Design, Manufacture and Operation in the Railway and Other Advanced Mass Transit Systems (COMPRAIL 2000)*, volume 7 de *Advances in Transport*, pages 1017–1026. Wit Press, 2000.

[Rod00b] Joaquín Rodriguez. Fluidification de noeuds ferroviaires. Projet SAFIR/GRRT prospectif - Rapport Final RR-00-724-FR, INRETS, décembre 2000.

[Rod01] Joaquín Rodriguez. Projet RECIFE : «Analyse de la capacité ferroviaire». Rapport technique, INRETS, 2001.

[Rön95] Mikael Rönnqvist. A method for the cutting stock problem with different qualities. *European Journal of Operational Research*, 83 :57–68, 1995.

[RR02] Mauricio G.C. Resende et Celso C. Ribeiro. Greedy randomized adaptive search procedures. Dans F. Glover et G. Kochenberger, éditeurs, *Handbook in Metaheuristics*, pages 219–249. Kluwer academic publishers, Boston, 2002.

[RR03] Mauricio G.C. Resende et Celso C. Ribeiro. A GRASP with path relinking for permanent virtual circuit routing. *Networks*, 41 :104–114, 2003.

[RS01] Fabrizio Rossi et Stefano Smriglio. A set packing model for the ground holding problem in congested networks. *European Journal of Operational Research*, 131 :400–416, 2001.

[Sak84] Michel Sakarovitch. *Optimisation combinatoire : méthodes mathématiques et algorithmiques*. Hermann, Paris, 1984.

[Sch97] F. Schneider. Recherche sur la saturation des lignes ferroviaires (rapport d'étude de la phase 2). Rapport technique 2166/ESF/721-97/RA, SYSTRA, France, décembre 1997.

[Ste86] Ralph E. Steuer. *Multiple Criteria Optimization - Theory, Computation and Application*. Wiley, New York, 1986.

[Sur01] Stéphanie Suray. Convertion des données RECIFE-SAFIR. Rapport technique, INRETS, août 2001.

[Tah92] Christian Tahon. *Système d'aide à la décision pour la conduite des systèmes de production*. Habilitation à diriger des recherches, Université de Valenciennes et du Hainaut-Cambrésis, Valenciennes, France, 1992.

[Tai98] Éric Taillard. *La programmation à mémoire adaptative et les algorithmes pseudo-gloutons : nouvelles perspectives pour les métaheuristiques*. Thèse d'habilitation, Université de Versailles Saint-Quentin-en-Yvelines, Versailles, France, 1998.

[Tai02] Éric Taillard. Principes d'implémentation des métaheuristiques. Dans J. Teghem et M. Pirlot, éditeur, *Optimisation approchée en recherche opérationnelle - Recherches locales, réseaux neuronaux et satisfaction de contraintes*, pages 57–79. Hermès Science Publications, Paris, 2002.

[U.I78] U.I.C. Fiche 405r. Rapport technique, UIC, 1978.

[UTF95] Ekunda Lukata Ulungu, Jacques Teghem, et Philippe Fortemps. Heuristics for multi-objective combinatorial optimisation problem by simulated annealing. Dans J. Gu, G. Chen, Q. Wei, et S. Wang, éditeurs, *Proceedings of the 6th national conference on Multiple Criteria Decision Making (MCDM : Theory and Applications)*, pages 228–238. SCI-TECH Information Services, Windsor, UK, 1995.

[vdBO94] J.H.A. van den Berg et M.A. Odijk. DONS : computer aided design of regular service timetables. Dans T.K.S. Murphy, B. Mellitt, C.A. Brebbia, G. Sciutto, et S. Sone, éditeurs, *Computers in Railway IV, proceedings of the Fourth International Conference on Computer Aided Design, Manufacture and Operation in the Railway and Other Advanced Mass Transit Systems (COMPRAIL 94)*, volume 2, pages 109–115. Computational Mechanics Publications, Southampton Boston, 1994.

[Wia03] Mickael Wiart. Développement d'une API C++ pour la génération du modèle SPP - Résolution du problème de faisabilité sur la gare de Lille-Flandres. Rapport technique, INRETS-ESTAS, 2003.

[Zit99] Eckart Zitzler. *Evolutionnary Algorithms for Multiobjective Optimization : Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, 1999.

[ZKR+96] Peter J. Zwaneveld, Léo G. Kroon, H. Edwin Romeijn, Marc Salomon, Stéphane Dauzère-Pérès, Stan P.M. Van Hoesel, et Harrie W. Ambergen. Routing trains through railway stations : Model formulation and algorithms. *Transportation Science*, 30(3) :181–194, août 1996.

[ZKVH01] Peter J. Zwaneveld, Léo G. Kroon, et Stan P.M. Van Hoesel. Routing trains through railway a station based on a node packing model. *European Journal of Operational Research*, 128 :14–33, 2001.

[ZT99] Eckart Zitzler et Lothar Thiele. Multiobjective Evolutionary Algorithms : A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4) :257–271, novembre 1999.

[Zwa97] Peter J. Zwaneveld. *Railway planning - routing of trains and allocation of passenger lines*. PhD thesis, Rotterdam school of management, TRAIL research school, Rotterdam, Netherlands, 1997.

Index

algorithme glouton, 79–82, 85

clique, 18, 19, 42, 71–73, 75, 76

facette, 9, 19, 71

inégalité valide, 9, 19, 71

logiciel
- Cplex, 3, 44, 71, 87, 90, 94, 100–103
- SISYFE, 124

métaheuristique
- GRASP, 2, 3, 78–87, 89, 104–111, 115, 129, 130
- path relinking, 79, 84–86, 105, 107, 110, 129, 130
- reactive GRASP, 79, 82, 83, 104, 105, 107, 115, 129
- SPEA, 3, 112, 115, 129

méthode de résolution exacte
- Branch & Bound, 39, 44
- Branch & Cut, 10, 14, 15, 19, 42, 44, 128, 129
- programmation par contraintes, 39, 44

problème de capacité ferroviaire
- faisabilité, 3, 35, 38, 44, 45, 47, 48, 52, 53, 64, 68, 117, 118, 127
- fluidification, 36, 44, 45
- optimisation des préférences, 36, 41, 44, 45, 47, 52, 53, 64, 68, 127
- saturation, 3, 35, 43–45, 47, 48, 52–54, 64, 68, 117, 118, 127
- stabilité, 3, 36, 45–47, 64, 66–68, 127, 128

problème linéaire
- en variables binaires (0-1 ILP), 7, 13, 44
  - clique maximum, 15, 18, 19, 73, 78
  - couplage généralisé, voir set packing
  - couverture d'ensembles, voir set covering
  - maximum independant set, voir node packing
  - node packing, 15, 16, 19, 20, 41, 78, 81
  - partitionnement d'ensembles, voir set partitioning
  - plus court chemin dans un graphe, 2, 3, 5, 11–13, 26, 66–68, 71, 128
  - sac-à-dos multi-dimensionnels en variables binaires, 15
  - set covering, 15–17, 78, 80
  - set packing, 2, 3, 5, 13–17, 19, 20, 26, 40, 41, 61, 67, 69, 71, 73, 75, 78, 80, 85–87, 89–91, 103, 110, 112, 117, 127–129
  - set partitioning, 15–17, 19
- en variables continues (LP), 6
- en variables entières (ILP), 7
- en variables mixtes (MILP), 7, 39, 43, 44

projet
- CADANS, 38, 43, 44, 130
- CAPRES, 42–45, 118

DÉMIURGE, 43, 44, 118, 130
DONS, 38, 44, 45, 130
RECIFE, 1, 3, 117, 118, 123–125, 130
SAFIR, 123, 124
STATIONS, 38, 39, 44, 47, 130

recherche dichotomique, 71, 76
recherche locale, 79, 83–85

Résumé : Cette thèse s'intéresse à la planification de l'exploitation d'infrastructures ferroviaires à l'échelle d'un nœud ou d'une gare. Pour déterminer une stratégie d'offre, il est important de disposer d'outils d'évaluation de la capacité des infrastructures. Cela permet de situer les limites d'un réseau, et d'étudier l'impact de modifications. Dans ce cadre, la faisabilité d'une grille horaire, son optimisation vis-à-vis de critères comme le nombre de trains (saturation), et l'évaluation de la stabilité sont considérées.

Nous proposons une modélisation linéaire multiobjectif de ce problème. Le niveau de détail considéré est suffisamment fin pour obtenir des grilles horaires réalisables dans la pratique sur les infrastructures étudiées. En outre, elle présente deux structures correspondant à des problèmes d'optimisation classiques appelés plus court chemin et set packing. Si le premier est facile à résoudre, le second est connu comme NP-difficile. Nous proposons différents algorithmes de pré-traitements, et de résolution approchée (basée sur la métaheuristique GRASP), pour ce problème. Une première extension de cette heuristique au cas biobjectif est présentée. Les expérimentations numériques, menées sur des instances correspondant au nœud de Pierrefitte-Gonesse, ou générées aléatoirement, montrent l'efficacité de ces algorithmes. L'intégration de ces travaux dans un logiciel dédié aux études de capacité d'infrastructures ferroviaires (projet RECIFE, en collaboration avec la SNCF) est décrite.

Title : Railroad infrastructure operation modelling and resolution

---

Abstract : This thesis deals with a planning problem in railroad infrastructure operation at the node or station level. In order to determine an offer strategy, having tools for evaluating infrastructure capacity is important. These tools allow the network limits to be evaluated, and the impact of proposed modifications to be studied. The main questions considered include the feasibility of a given timetable, its optimization according to such criteria as the number of trains (saturation), and its stability evaluation.

A new multiobjective linear model, which is accurate enough to produce practicable timetables for the studied infrastructures, is proposed for this problem. This model is structured primarily as two classic combinatorial optimization problems named shortest path and set packing. If resolving shortest path problems can be considered fairly easy, the set packing problem is known to be NP-hard. Several pre-processing algorithms, and an approximation method based on the GRASP metaheuristic, dedicated to set packing resolution are proposed. A preliminary extension of this method to the bi-objective case is also presented. These algorithms were successfully tested on both randomly generated instances and instances related to the Pierrefitte-Gonesse node. The integration of this research in a decision support system for evaluating railroad infrastructure capacity (RECIFE project, in collaboration with the French National Railroad Company) is described.

Discipline : Informatique

Mots clés : Capacité d'infrastructure ferroviaire, GRASP, Métaheuristique, Modélisation multiobjectif, Optimisation combinatoire, Set packing

---

Laboratoires : INRETS-ESTAS, 20 rue Élisée Reclus, F-59650, Villeneuve d'Ascq, France
LAMIH-UMR CNRS 8530, UVHC, "Le Mont Houy", F-59313, Valenciennes Cedex 9, France

---