Titre: Title:	Logiciel de génération de colonnes
Auteur: Author:	Daniel Villeneuve
Date:	1999
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Villeneuve, D. (1999). Logiciel de génération de colonnes [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8603/

UNIVERSITÉ DE MONTRÉAL

LOGICIEL DE GÉNÉRATION DE COLONNES

DANIEL VILLENEUVE
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR (Ph.D.)
(MATHÉMATIQUES DE L'INGÉNIEUR)
OCTOBRE 1999

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

LOGICIEL DE GÉNÉRATION DE COLONNES

présentée par : VILLENEUVE Daniel
en vue de l'obtention du diplôme de : Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. SAVARD Gilles, Ph.D., président
M. SOUMIS François, Ph.D., membre et directeur de recherche
M. DESROSIERS Jacques, Ph.D., membre et codirecteur de recherche
M. SMITH Benjamin T., Ph.D., membre
M. VAN CANEGHEM Michel, Ph.D., membre

À mes parents

Remerciements

Je remercie d'abord M. François Soumis, mon directeur de recherche, pour m'avoir soutenu avec confiance tout au long de la réalisation de mon doctorat. Ses idées originales et ses commentaires constructifs ont enrichi à la fois le contenu et la présentation de la thèse. Je remercie également M. Jacques Desrosiers, mon codirecteur de recherche, pour m'avoir épaulé infatigablement lors de la rédaction de la thèse. Je remercie aussi mes deux directeurs pour l'intérêt qu'ils ont porté à la progression des travaux et pour l'aide financière qu'ils m'ont accordée généreusement.

Je remercie M. Olivier du Merle, M. Jacques Desrosiers et M. Pierre Hansen pour le climat de collaboration fructueuse qui a prévalu lors de la rédaction de l'article *Stabilized Column Generation.*

Je tiens à remercier M. Yvan Dumas, auteur principal des versions de GENCOL antérieures à cette thèse, pour ses conseils et sa disponibilité, de même que M. François Lacoursière, M. Éric Gélinas, M. Norbert Lingaya et M. Manuel Pires, mes collègues et amis lors du développement des deux dernières versions de GENCOL, pour l'esprit d'équipe et les discussions tous azimuts qui ont animé régulièrement le bureau. Je remercie aussi tous ceux qui, de près ou de loin, ont collaboré à ce projet, ainsi que les secrétaires et administrateurs du système informatique du GERAD pour leur excellent support technique.

Je remercie finalement mes parents pour leurs encouragements tout au long de mes études et ma femme Chantal qui a partagé mes espoirs et mes doutes durant ce long périple qui tire maintenant à sa fin.

Résumé

Cette thèse porte sur la résolution efficace par génération de colonnes de problèmes de tournées de véhicule et d'horaires d'équipage comportant un très grand nombre de contraintes. La résolution de ces problèmes vise à minimiser les coûts d'opération d'une flotte de véhicules ou d'une équipe de travail lors de la réalisation d'un ensemble de tâches prédéterminées. En 1998, Desaulniers, Desrosiers, Ioachim, Solomon, Soumis et le chercheur ont proposé un modèle générique de flot non linéaire multi-commodités, appelé le *modèle unifié*, pour analyser une grande classe de problèmes déterministes de tournées de véhicule et d'horaires d'équipage. Dans ce modèle, les tâches (par exemple : vols et services de vol, segments d'itinéraires d'autobus, activités de formation) sont réparties sur un réseau structuré généralement selon des dimensions d'espace et de temps, sur lequel circulent des commodités (par exemple : véhicules, équipes de travail, individus). Des contraintes peuvent limiter globalement les combinaisons des chemins parcourus par les commodités et limiter localement la composition de ces chemins. Cette thèse se concentre sur les problèmes pour lesquels la difficulté de résolution provient à la fois des contraintes locales sur la composition de chaque chemin et des contraintes globales sur la combinaison de ces chemins. Étant donné la nature non convexe de certaines contraintes, ces problèmes sont résolus au moyen d'un processus de séparation et d'évaluation, où les bornes inférieures sont calculées par génération de colonnes. Les colonnes correspondent aux chemins parcourus par les commodités sur le réseau.

La thèse comporte une partie théorique et une partie expérimentale. D'une part, l'objectif théorique principal consiste à analyser les conditions d'équivalence entre une formulation bloc-angulaire d'un problème non linéaire et une formulation de génération de colonnes accompagnée d'un oracle. La tranformation directe est étudiée

dans le contexte du modèle unifié, qui sert de cadre théorique pour la partie expérimentale et de fil conducteur pour l'ensemble de la thèse. La transformation inverse est étudiée dans le cadre d'un programme linéaire mixte généralisé accompagné d'un oracle et peut être considérée comme une formalisation de l'évolution du modèle unifié. D'autre part, l'objectif expérimental de la thèse consiste à développer un logiciel permettant de résoudre des problèmes de tournées de véhicule et d'horaires d'équipage de très grande taille, correspondant à un sous-ensemble important du modèle unifié. D'abord, la réalisation de cet objectif passe par la mise en œuvre de deux nouvelles versions du logiciel GENCOL qui constituait, jusqu'en 1992, le seul logiciel visant la résolution de cas particuliers du modèle unifié dans un contexte générique. Ensuite, deux approches algorithmiques complémentaires sont développées pour traiter plus efficacement des problèmes comportant un très grand nombre de contraintes globales linéaires. La première approche consiste à définir un processus de stabilisation qui généralise l'algorithme classique de génération de colonnes en permettant de contrôler la taille du domaine dual à chaque itération de l'algorithme. La deuxième approche permet de réduire le nombre de contraintes de partitionnement dans les problèmes de tournées de véhicule et d'horaires d'équipage de très grande taille en agrégeant ces contraintes selon la structure anticipée des solutions primales.

Au plan théorique, la thèse contribue à éclaircir les relations entre les formulations dites *compacte* et *décomposée* d'un même problème en nombres entiers. Nous montrons que, sous des conditions relativement faibles, on peut retrouver une formulation compacte équivalente à la formulation initiale du problème, qui intègre à la fois les contraintes du problème-maître et la structure de l'oracle. La formulation initiale peut alors être considérée comme une formulation décomposée, s'obtenant par l'application d'une extension du principe de décomposition de Dantzig-Wolfe sur la formulation compacte. L'intérêt de cette contribution provient du potentiel qu'offre la formulation compacte pour exploiter la structure de l'oracle lors du développement

d'algorithmes de séparation dans le contexte de la résolution du problème en nombres entiers.

Dans le cas d'un problème non linéaire à structure bloc-angulaire, nous utilisons la formulation compacte du modèle unifié pour présenter une extension du principe de décomposition de Dantzig-Wolfe permettant d'obtenir une formulation décomposée équivalente à la formulation compacte. La contribution consiste ici à clarifier et à formaliser le principe de décomposition de Dantzig-Wolfe en présence de contraintes non linéaires et de contraintes d'intégrité. L'analyse de la décomposition du modèle unifié permet aussi de corriger une omission dans le modèle unifié concernant les fonctions d'extensions non convexes dans la formulation décomposée.

Au plan expérimental, la thèse contribue d'une part à structurer la recherche et le développement sur le modèle unifié par la mise en œuvre de deux nouvelles versions du logiciel GENCOL. Le succès de cette réalisation se mesure aux nombreuses applications commerciales (plus de 50 depuis 1994) et publications de recherche (plus de 30 depuis 1992) utilisant ces versions de GENCOL comme optimiseur.

D'autre part, la thèse présente deux nouvelles approches algorithmiques pour traiter des problèmes de génération de colonnes comportant un très grand nombre de contraintes globales linéaires. Dans un premier temps, nous proposons un processus de stabilisation combinant une méthode de perturbation et une méthode de pénalités exactes. L'originalité de notre approche est de définir le processus de stabilisation tout en restant dans le contexte de la programmation linéaire, et donc d'en retenir la simplicité et l'efficacité à chaque itération. L'intérêt de cet algorithme réside dans le fait que, pour chacune des trois applications de génération de colonnes étudiées, le processus de stabilisation rend possible soit une réduction significative des temps de résolution, soit la résolution de cas considérés de trop grande taille jusqu'à tout récemment.

Dans un deuxième temps, nous développons deux stratégies algorithmiques pour réduire le nombre de contraintes de partitionnement actives en même temps dans la formulation décomposée d'un problème formulé au moyen du modèle unifié. Nous présentons d'abord plusieurs algorithmes d'agrégation statique pour identifier et éliminer lors d'un prétraitement des contraintes de partitionnement redondantes. La contribution réside dans l'analyse de la complexité de chacun de ces algorithmes, permettant de pondérer le niveau de réduction du nombre de contraintes avec le temps de calcul des algorithmes. La mise en œuvre des variantes les plus efficaces (en temps de calcul) montre que les gains sont importants pour certaines applications de très grande taille mais dépendent fortement de la structure des réseaux des problèmes traités. Nous développons ensuite un algorithme d'agrégation dynamique pour contrôler le nombre de contraintes de partitionnement actives à chaque itération du processus de génération de colonnes. Nous visons plus particulièrement les problèmes de très grande taille de tournées de véhicule et d'horaires d'équipage, de même que les problèmes de réoptimisation d'une solution planifiée. Dans ces cas, les solutions qui comportent des changements de véhicule ou des déviations par rapport à l'horaire planifié sont souvent pénalisées dans l'objectif, mettant en évidence un ordonnancement naturel des tâches par véhicule ou équipage. La contribution de notre approche consiste à exploiter systématiquement cette structure sous-jacente des tâches pour agréger les contraintes de partitionnement, résoudre un problème de taille réduite, retrouver efficacement l'information duale sur l'ensemble des contraintes et modifier l'agrégation. La mise en œuvre de l'algorithme, encore inachevée, laisse entrevoir la résolution de cas comportant jusqu'à cinq fois plus de contraintes de partitionnement que ceux traités avec les techniques actuelles.

Abstract

This dissertation is about the efficient solution of very large scale vehicle routing and crew scheduling problems using column generation. These problems consist in assigning a fixed set of tasks to crew members or vehicles at minimal cost. In 1998, Desaulniers, Desrosiers, Ioachim, Solomon, Soumis and the author proposed a generic nonlinear multicommodity network flow model, called the *unified model*, for analyzing a large class of deterministic vehicle routing and crew scheduling problems. In this model, tasks (e.g., flights or flight legs, bus routes, training activities) are distributed over a network, usually involving time and space dimensions, and are performed by commodities (e.g., vehicles, crews, individuals) moving on the network. Global constraints may restrict the combining of paths followed by the commodities while local constraints may restrict the composition of these paths. The dissertation focuses on problems where the difficulty lies in the existence of both global and local constraints. Because of the non-convexity of the solution domain, these problems must be solved within a branch-and-bound framework. The block angular structure of the problems leads to the computation of lower bounds by column generation, where each column corresponds to a path followed by some commodity on the network.

The dissertation is divided into a theoretical part and an experimental part. First, the theoretical objective consists in analyzing equivalence conditions between a block angular formulation of a nonlinear problem and a column generation formulation associated with its oracle. The direct transformation is studied in the context of the unified model, which constitutes an underlying thread for the whole thesis and forms the theoretical framework for the experimental part. The inverse transformation is studied in the context of a generalized mixed integer linear problem associated with

an oracle. This transformation can be viewed as a formalization of the evolution of the unified model since its inception. Next, the experimental objective consists in developing a software library for solving very large scale vehicle routing and crew scheduling problems from a large subset of the unified model. In the first place, the realization of this objective involves the development of two new versions of the GENCOL software library, which was until 1992 the only computer code aimed at solving specialized instances of the unified model in a generic way. In the second place, two complementary algorithmic approaches are developed for dealing more efficiently with problems made up of a very large number of global linear constraints. In the first approach, we define a stabilization process which generalizes the classical column generation algorithm by allowing one to control the "size" of the dual domain at each iteration. In the second approach, we reduce the number of active set partitioning constraints in very large scale vehicle routing and crew scheduling problems by aggregating constraints according to some anticipated structure of primal solutions.

On the theoretical side, one contribution resides in the clarification of the relations between *compact* and *column generation* formulations of a mixed integer problem. We show how to retrieve, under relatively weak conditions, a compact formulation equivalent to the initial column generation one. This compact formulation integrates the master problem constraints with the structure of the oracle. As a consequence of the equivalence between both formulations, the column generation formulation can be obtained from the compact formulation by applying an extension of the Dantzig-Wolfe decomposition principle. The significance of the contribution comes from the possible exploitation of the oracle's structure when developing separation algorithms for the solution of mixed integer problems.

In the case of a block angular nonlinear problem, we use the compact formulation of the unified model to present an extension of the Dantzig-Wolfe decomposition

principle, leading to a column generation formulation equivalent to the compact one. Another contribution consists in clarifying and formalizing the Dantzig-Wolfe decomposition principle in the presence of nonlinear and integer constraints. The analysis of the decomposition process on the unified model also reveals an omission in the unified model concerning the handling of non convex extension functions in the column generation formulation.

On the experimental side, this dissertation contributes to the structuring of research and development activities around the unified model by releasing the two most recent versions of the GENCOL software library. The success of this realization can be measured by the many commercial applications (more than 50 since 1994) and research publications (more than 30 since 1992) using these versions of GENCOL as their optimizer.

Moreover, the dissertation presents two novel algorithmic approaches for coping with a very large number of global linear constraints in column generation problems. In the first place, we propose a stabilization process that combines a perturbation method and an exact penalty method. The originality of our approach is to define the stabilization process while staying in the context of linear programming, thereby retaining its associated efficiency at each iteration. The interest of this algorithm comes from the fact that, for the three column generation applications studied, the stabilization process leads either to a significant reduction in the solution time or to the solution of instances larger than those previously considered in the literature.

Finally, we develop two algorithmic strategies for reducing the number of set partitioning constraints that are active at the same time in the column generation formulation of a problem formulated using the unified model. First, we present several static aggregation algorithms for identifying and eliminating redundant set partitioning constraints in a preprocessing step. The contribution resides in the complexity

analysis of each algorithm, making it possible to balance the level of constraint reduction with the solution time. The implementation of the most efficient variants (in solution time) shows that the benefits are great for some very large scale applications but the reduction in the number of constraints depends heavily on the network structure of the problems. Next, we develop a dynamic aggregation algorithm to control the number of set partitioning constraints active at each iteration of the column generation process. We especially target very large scale vehicle routing and crew scheduling problems, as well as reoptimization problems starting with an already planned solution. In those cases, new solutions that force crews to change vehicles or that deviate from the incumbent solution are often penalized in the objective, revealing a natural task ordering per vehicle or crew. The contribution of our approach consists in systematically exploiting this underlying structure on the set of tasks for aggregating the set partitioning constraints, solving a reduced-size master problem, retrieving efficiently the dual information for the whole set of constraints and modifying the aggregation. The implementation of the algorithm, still in progress, is expected to allow the solution of problems involving up to five times more set partitioning constraints than what can be processed using the current state-of-the-art algorithms.

Table des matières

Liste des tableaux

Liste des figures

Liste des annexes

Liste des algorithmes

Introduction

L'étude des problèmes de tournées de véhicule et d'horaires d'équipage est principalement motivée par le besoin d'accroître la compétitivité de l'industrie dans le domaine du transport de marchandises et de personnes. La résolution de ces problèmes vise ainsi à minimiser les coûts d'opération d'une flotte de véhicules ou d'une équipe de travail lors de la réalisation d'un ensemble de tâches prédéterminées. Depuis une vingtaine d'années, les modèles utilisés pour résoudre ces problèmes intègrent des contraintes sur la qualité des services offerts, comme des intervalles limitant le temps de début de service chez les clients. Plus récemment, les modèles se sont enrichis pour prendre en compte non seulement les besoins des clients, mais aussi les exigences de plus en plus complexes concernant les modalités d'exécution des tâches (entre autres, conventions collectives, normes du travail, normes d'entretien).

Cette thèse porte sur les problèmes de tournées de véhicule et d'horaires d'équipage qui peuvent se formuler au moyen du modèle dit *unifié* de Desaulniers *et al.* (1998). Ce modèle est un modèle générique qui constitue actuellement l'état de l'art parmi les modèles généraux traitant les problèmes déterministes de tournées de véhicule et d'horaires d'équipage. C'est entre autres le seul modèle qui, d'une part, permet d'inclure suffisamment de réalisme dans la description des problèmes pour que les solutions soient utilisables en pratique dans l'industrie et, d'autre part, possède les propriétés mathématiques nécessaires pour obtenir des solutions optimales. Le modèle unifié tire son origine du domaine du transport et s'applique particulièrement bien aux problèmes consistant à couvrir à moindre coût un ensemble de tâches (par exemple : vols et services de vol, segments d'itinéraires d'autobus, activités de formation) réparties sur un réseau, incluant généralement des dimensions d'espace et

de temps, en y faisant circuler des commodités (par exemple : véhicules, équipes de travail, individus). Des contraintes peuvent limiter globalement les combinaisons des chemins parcourus par les commodités (comme imposer des bornes sur le nombre de véhicules utilisés selon leur type ou répartir les équipes entre des bases selon une distribution donnée) et limiter localement la composition des chemins de chaque commodité (comme cueillir avant de livrer la marchandise ou respecter un nombre maximum d'heures consécutives de travail). La caractéristique essentielle du modèle réside dans la forme des contraintes locales sur les chemins. Celles-ci permettent l'expression directe d'un grand nombre de règles propres à chaque application et contribuent à l'atteinte d'un haut degré de réalisme dans les solutions obtenues. Cette caractéristique explique le succès du modèle au plan commercial, ce qui stimule en retour les activités de recherche pour la résolution de problèmes de taille de plus en plus grande.

L'utilité d'un modèle générique pour résoudre les problèmes de tournées de véhicule et d'horaires d'équipage s'est fait sentir vers le début des années 1990, face à la prolifération de modèles et d'algorithmes de résolution spécialisés selon les applications. Les premières versions du modèle unifié de Desaulniers *et al.* (1998) ont été présentées dans l'article de Desrosiers *et al.* (1995) ainsi que dans la thèse de Ioachim (1994). Le modèle unifié regroupe dans une même formulation mathématique tous les problèmes déterministes de tournées de véhicule et d'horaires d'équipage classifiés par Desrochers *et al.* (1990). Il permet en principe de partager les algorithmes et les développements informatiques entre différents domaines d'application. Ce partage suppose que l'analyse des solutions et des processus de résolution repose sur une structure commune à la plupart des problèmes s'inscrivant dans la formulation. Cette thèse se concentre sur les problèmes pour lesquels la difficulté de résolution provient à la fois des contraintes locales sur la composition de chaque chemin et des contraintes globales sur la combinaison de ces chemins. Étant donné la nature non

convexe de certaines contraintes, ces problèmes sont résolus au moyen d'un processus de séparation et d'évaluation, où les bornes inférieures sont calculées par génération de colonnes. Ces colonnes correspondent aux chemins parcourus par les commodités sur le réseau. Nous notons que les problèmes ne comportant aucune commodité ou une seule commodité[1] ne comportent pas de contraintes sur les combinaisons de chemins et, bien qu'inclus dans le modèle unifié, ne sont pas traités explicitement dans cette thèse.

La thèse comporte une partie théorique et une partie expérimentale. L'objectif théorique principal vise à expliciter et à formaliser la transformation d'un problème non linéaire comportant une structure bloc-angulaire en un problème de génération de colonnes couplé à un oracle, et la transformation inverse. La transformation directe est présentée par le biais de la décomposition du modèle unifié au moyen d'une extension du principe de décomposition de Dantzig-Wolfe. La transformation inverse consiste à retrouver sous certaines conditions une formulation bloc-angulaire à partir d'un programme linéaire mixte généralisé. L'objectif expérimental de la thèse consiste à développer un logiciel permettant de résoudre des problèmes de tournées de véhicule et d'horaires d'équipage de très grande taille, correspondant à un sous-ensemble important du modèle unifié. Nous utilisons ici le qualificatif *très grande taille* pour identifier les problèmes qui se situaient hors des limites de la technologie existant au moment d'amorcer la réalisation de la thèse en 1992, soit par le nombre de commodités, la taille des réseaux, le nombre de contraintes locales sur la composition des chemins ou le nombre de contraintes globales sur les combinaisons de chemins. D'une part, la réalisation de cet objectif implique la restructuration de la version précédente du logiciel GENCOL qui constituait, jusqu'en 1992, le seul logiciel ayant l'ambition de traiter une partie importante du modèle unifié. Ces développements ont pour

[1] Les problèmes ne comportant qu'une seule commodité sont des variantes du problème TSP de voyageur de commerce (*Traveling Salesman Problem*), qui constitue un sujet d'études séparé et sur lequel il existe une littérature abondante (*voir* Junger *et al.*, 1995).

but d'élargir la partie traitée du modèle, d'améliorer les performances des différents algorithmes existants et d'atteindre un degré de robustesse acceptable pour une utilisation commerciale. D'autre part, de nouveaux algorithmes exploitant la structure des problèmes ou de leurs solutions sont mis au point pour gérer efficacement un très grand nombre de contraintes globales linéaires au sein du processus de génération de colonnes.

La thèse est ainsi divisée en deux parties correspondant aux deux objectifs principaux. La première partie couvre les trois premiers chapitres et porte essentiellement sur l'analyse théorique des conditions d'équivalence entre une formulation bloc-angulaire d'un problème non linéaire et une formulation de génération de colonnes accompagnée d'un oracle. On y retrouve aussi une présentation détaillée du modèle unifié et une revue de la littérature sur les composantes de ce modèle, qui sert de cadre théorique à l'ensemble de la thèse. La deuxième partie couvre les trois derniers chapitres et porte sur les développements informatiques et algorithmiques effectués en vue d'atteindre l'objectif expérimental. On y retrouve une description des versions du logiciel GENCOL développées dans le cadre de la thèse, ainsi que deux approches algorithmiques complémentaires permettant de résoudre des problèmes comportant un très grand nombre de contraintes globales linéaires sur les combinaisons de chemins.

Le **chapitre 1** joue deux rôles distincts. D'une part, nous y décrivons la formulation *originale* du modèle unifié de Desaulniers *et al.* (1998) afin d'en définir les concepts essentiels et la notation et d'en faire ainsi le cadre théorique de la thèse. Cette description met en évidence le caractère non convexe du modèle, que nous proposons de résoudre au moyen d'un processus de séparation et d'évaluation. D'autre part, nous utilisons cette formulation originale comme exemple de la transformation d'un problème non convexe à structure bloc-angulaire en un problème de génération de colonnes. La non convexité provient entre autres des contraintes d'intégrité sur les arcs formant les chemins des commodités dans le réseau, tandis que la structure bloc-angulaire porte sur les contraintes locales de composition des chemins de

chaque commodité. Nous proposons une extension du principe de décomposition de Dantzig-Wolfe pour obtenir une formulation dite *décomposée* équivalente à la formulation originale. L'équivalence entre les deux formulations permet de choisir indifféremment l'une ou l'autre pour définir le processus de résolution par séparation et évaluation, alors que l'approche traditionnelle d'application de la décomposition de Dantzig-Wolfe restreint l'utilisation de la formulation décomposée au calcul de bornes inférieures.

Le **chapitre 2** est consacré à la revue de la littérature et justifie la pertinence et l'importance du modèle unifié au moyen d'une énumération sélective mais représentative des travaux antérieurs. Nous présentons d'abord le problème de plus court chemin avec contraintes d'intervalles de temps, dénoté SPTW (*Shortest Path with Time Windows*), d'où le modèle unifié tire ses racines. Nous montrons ensuite, par construction, comment le modèle unifié englobe tous les autres modèles utilisés jusqu'à maintenant pour traiter les problèmes d'intérêt pour cette thèse, en retraçant l'évolution du modèle unifié à partir du problème SPTW jusqu'à sa forme actuelle par ajouts successifs de contraintes. Nous passons par la suite à une revue des algorithmes applicables à chaque composante du processus de résolution par séparation et évaluation, où les bornes inférieures sont calculées par génération de colonnes impliquant un oracle correspondant à l'une ou l'autre des variantes du problème SPTW. Nous terminons le chapitre par une liste d'applications de recherche et commerciales réalisées durant la thèse s'inscrivant directement dans le champ du modèle unifié, démontrant ainsi le potentiel réel de synergie existant entre différents domaines d'application.

Le **chapitre 3** complète l'objectif théorique de la thèse en analysant la transformation d'un problème défini initialement sous la forme d'un programme linéaire mixte généralisé et d'un oracle, en un problème équivalent dont la formulation dite *compacte* intègre le domaine de l'oracle dans une structure bloc-angulaire. L'intérêt de cette étude découle des difficultés survenant lors de la définition des procédures de

séparation directement sur la formulation initiale dite *décomposée*. Nous mettons d'abord en évidence, par un exemple simple similaire à celui de Ben Amor (1997), l'impossibilité théorique pour l'oracle de générer l'ensemble de toutes les variables d'un problème, et même le sous-ensemble de variables contenant la solution optimale. Nous décrivons ensuite les approches utilisées pour résoudre le problème malgré cette incompatibilité apparente entre l'oracle et les procédures de séparation. Nous proposons enfin une démarche pour retrouver, sous des conditions relativement faibles, une formulation compacte (non unique) à partir de la formulation décomposée, permettant ainsi d'exploiter la structure mathématique de l'oracle dans les procédures de séparation.

Le **chapitre 4** est consacré à la description du logiciel GENCOL, dont les deux dernières versions (GENCOL 3.0, 3.1, 4.0, 4.1, 4.2, 4.3) constituent les contributions principales de la thèse au plan expérimental. Le chapitre s'ouvre sur un rappel de l'historique du logiciel GENCOL, permettant d'apprécier son évolution et de bien dégager les contributions des dernières versions. Nous présentons ensuite ces versions en identifiant pour chacune le sous-ensemble du modèle unifié visé par les développements et en analysant les particularités de la mise en œuvre des algorithmes qui ont permis de franchir l'écart entre un prototype de recherche et un logiciel de qualité commerciale. La description de chacune de ces versions est complétée par une discussion critique des décisions prises lors de leur développement. En guise de résultats numériques, nous référons le lecteur aux nombreux travaux de recherche et applications commerciales réalisés au moyen de ces deux dernières versions de GENCOL, prouvant la valeur du logiciel tant au plan économique qu'en tant que projet structurant pour la recherche sur le modèle unifié.

Le **chapitre 5** porte sur la première de deux stratégies pour résoudre plus efficacement des problèmes de génération de colonnes comportant un grand nombre de contraintes globales linéaires. L'algorithme de Kelley (1960), couplé à un oracle, est

une approche linéaire pour résoudre des problèmes par génération de colonnes, mais sa convergence est fréquemment lente. Pour remédier à ce comportement, de nombreuses approches non linéaires ont été étudiées. L'originalité de notre approche est de définir un processus de stabilisation qui reste dans le contexte de la programmation linéaire. Nous proposons de stabiliser et d'accélérer le processus de résolution par génération de colonnes en combinant une méthode de perturbation et une méthode de pénalités exactes. L'algorithme décrit dans ce chapitre constitue une généralisation de l'algorithme classique de génération de colonnes, auquel nous greffons des procédures d'ajustement contrôlant les dimensions de l'espace dual à chaque itération. Cet algorithme tire naturellement profit de la connaissance, même approchée, de solutions duales du problème à résoudre. Nous montrons l'efficacité du nouvel algorithme sur trois applications utilisant la génération de colonnes, pour lesquelles nous détaillons les stratégies d'ajustement utilisées et présentons des résultats numériques prometteurs.

Le **chapitre 6** porte sur la deuxième stratégie développée pour résoudre plus efficacement des problèmes de génération de colonnes comportant un grand nombre de contraintes globales linéaires, dans le cas spécifique de problèmes formulés au moyen du modèle unifié où les contraintes linéaires les plus nombreuses sont des contraintes de partitionnement. L'intérêt de cette classe restreinte de problèmes provient de la tendance, dans l'industrie, à planifier les opérations de façon plus intégrée et sur de plus longues périodes, ce qui se reflète principalement, au niveau des applications du modèle unifié, par une augmentation du nombre de tâches à couvrir par les commodités. La stratégie présentée dans ce chapitre est en fait composée de deux approches complémentaires qui exploitent la forte structure imposée par les contraintes de partitionnement. Dans un premier temps, nous développons et analysons une série d'algorithmes d'*agrégation statique* qui permettent d'identifier et d'éliminer dans un prétraitement des contraintes de partitionnement redondantes.

Dans un deuxième temps, nous présentons un algorithme d'*agrégation dynamique* qui constitue une généralisation de l'algorithme classique de génération de colonnes, que nous inscrivons dans un processus itératif de regroupement des contraintes et de sélection des variables. Cet algorithme tire naturellement profit de la structure des solutions primales du problème à résoudre (par exemple, un ordonnancement temporel des tâches). Les deux approches visent une réduction agressive du nombre de contraintes linéaires actives dans le programme linéaire provenant de la formulation décomposée et permettent d'envisager une diminution proportionnelle du temps de calcul consacré à l'optimisation de ce dernier. Nous rapportons les résultats obtenus au moyen d'une version de l'algorithme d'agrégation statique sur des applications de très grande taille en transport aérien et urbain. Enfin, nous discutons des alternatives stratégiques concernant la mise en œuvre de l'algorithme d'agrégation dynamique, balisant clairement les développements ultérieurs qui mèneront à l'intégration de cet algorithme dans la prochain version du logiciel GENCOL.

La conclusion reprend les principales contributions de la thèse tant au plan théorique qu'au plan expérimental. Nous y mentionnons aussi quelques avenues de recherche et de développement permettant respectivement d'approfondir les connaissances sur le modèle unifié et de faire progresser les techniques utilisées pour résoudre des problèmes de très grande taille.

Chapitre 1

Cadre théorique

La plupart des problèmes de tournées de véhicule et d'horaires d'équipage présentement étudiés dans la littérature peuvent être formulés au moyen du modèle unifié présenté par Desaulniers *et al.* (1998). Ce modèle est aussi l'un des principaux à être utilisés dans l'industrie lors de la résolution de problèmes de grande taille (*voir* le chapitre 2 pour une liste partielle d'applications). Le modèle unifié constitue donc un fondement théorique naturel pour développer un logiciel visant à repousser les limites de la technologie actuelle quant à la taille et au réalisme des problèmes traités.

Ce chapitre reprend le modèle unifié de Desaulniers *et al.* (1998) pour en définir les concepts essentiels et la notation, et en faire ainsi le cadre théorique de cette thèse. La première section décrit le modèle unifié sous sa forme *originale* ou *compacte*, comme une généralisation des problèmes de flot multi-commodités. La deuxième section présente le modèle sous sa forme *décomposée*, tirant avantage de la structure particulière du modèle afin d'en faciliter la résolution éventuelle. La dernière section montre comment utiliser le modèle pour décrire des problèmes qui semblent *a priori* incompatibles avec la formulation des sections antérieures.

1.1 Un modèle unifié sous la forme d'un problème non linéaire de flot multi-commodités

Le modèle étudié dans cette thèse est une formulation non linéaire d'un problème de flot multi-commodités en variables entières, problème auquel s'ajoutent des variables de ressource. Ce modèle constitue une extension d'une formulation du problème de voyageurs de commerce multiples avec variables de ressource, dénoté *m*-TSPR (*multiple-Traveling Salesman Problem with Resource variables*). Dans certains cas, ce problème correspond à visiter un ensemble de clients au moyen d'une flotte hétérogène de véhicules. Les chemins parcourus par ces véhicules sont toutefois soumis à des restrictions sur des ressources comme la capacité des véhicules, la durée de chaque itinéraire ou l'heure de service de chaque client.

Le problème général traité dans cette thèse est défini sur un réseau. Il consiste à effectuer un ensemble de tâches associées aux nœuds ou aux arcs de ce réseau au moyen de véhicules ou d'équipages parcourant des chemins à coût minimum. D'une part, l'utilisation de ces véhicules ou équipages, plus généralement appelés *commodités*, est limitée *localement* par la structure du réseau et par diverses variables de ressource. L'utilisation des commodités peut également être limitée par des relations de préséance et de couplage entre les tâches à effectuer. D'autre part, cette utilisation des commodités est limitée *globalement* par des contraintes de coordination entre celles-ci.

La prochaine section présente la notation utilisée et la formulation mathématique du modèle unifié. La suivante décrit la signification générale des contraintes du modèle.

1.1.1 Formulation mathématique

La commodité $k \in K$ circule sur le réseau $G^k = (N^k, A^k)$, où N^k et A^k sont des ensembles de nœuds et d'arcs. L'ensemble N^k contient au moins deux nœuds : un nœud origine o^k et un nœud puits d^k. À chaque commodité k est également associé un ensemble de ressources R^k.

Les variables représentant le flot de la commodité k dans G^k sont dénotées par le vecteur $\boldsymbol{X}^k = \left(X_{ij}^k \mid (i,j) \in A^k\right)$, où X_{ij}^k prend la valeur 1 si l'arc (i,j) est utilisé par la commodité k, et la valeur 0 sinon. Les variables représentant l'état des ressources à chaque nœud de N^k sont dénotées par le vecteur $\boldsymbol{T}^k = \left(T_i^{kr} \mid i \in N^k, r \in R^k\right)$, où T_i^{kr} prend la valeur de la ressource r cumulée depuis le nœud o^k si le nœud i est visité par la commodité k, et la valeur 0 sinon. L'agrégation intermédiaire $\boldsymbol{T}_i^k = \left(T_i^{kr} \mid r \in R^k\right)$ représente le vecteur de toutes les ressources au nœud $i \in N^k$. Finalement, les variables supplémentaires Y_s, pour tout $s \in S$, contribuent à la coordination entre les commodités.

L'objectif z_{IP} est composé de la somme des coûts c_{ij}^k des arcs $(i,j) \in A^k$ utilisés par chaque commodité k. Il comporte aussi une combinaison linéaire en c_i^{kr} des valeurs des ressources T_i^{kr}, $r \in R^k$, à chaque nœud $i \in N^k$ de cette commodité. L'objectif est complété par des coûts c_s sur les variables supplémentaires Y_s.

Cette notation permet d'introduire le modèle unifié sous la forme du programme mathématique présenté au tableau 1.1.

1.1.2 Description de la formulation originale

Les contraintes (1.2)–(1.4) permettent les interactions entre toutes les commodités du modèle : ce sont les contraintes dites globales ou liantes, dénommées *multiple*

Tableau 1.1 – Formulation originale du modèle unifié

$$z_{\mathrm{IP}} = \min \sum_{\substack{k\in K\\(i,j)\in A^k}} c_{ij}^k X_{ij}^k + \sum_{\substack{k\in K\\i\in N^k\\r\in R^k}} c_i^{kr} T_i^{kr} + \sum_{s\in S} c_s Y_s \tag{1.1}$$

sous les contraintes :

$$\sum_{\substack{k\in K\\(i,j)\in A^k}} a_{wij}^k X_{ij}^k + \sum_{s\in S} a_{ws} Y_s = a_w \qquad \forall w \in W \tag{1.2}$$

$$\sum_{\substack{k\in K\\i\in N^k\\r\in R^k}} b_{hi}^{kr} T_i^{kr} + \sum_{\substack{k\in K\\(i,j)\in A^k}} b_{hij}^k X_{ij}^k + \sum_{s\in S} b_{hs} Y_s = b_h \qquad \forall h \in H \tag{1.3}$$

$$l_s \le Y_s \le u_s \qquad \forall s \in S \tag{1.4}$$

$$\sum_{j:(o^k,j)\in A^k} X_{o^k j}^k = 1 = X_{o^k}^k \qquad \forall k \in K \tag{1.5}$$

$$\sum_{i:(i,d^k)\in A^k} X_{id^k}^k = 1 = X_{d^k}^k \qquad \forall k \in K \tag{1.6}$$

$$\sum_{j:(i,j)\in A^k} X_{ij}^k = \sum_{j:(j,i)\in A^k} X_{ji}^k = X_i^k \qquad \forall k \in K, \forall i \in N^k \setminus \{o^k, d^k\} \tag{1.7}$$

$$X_{ij}^k \in \{0,1\} \qquad \forall k \in K, \forall (i,j) \in A^k \tag{1.8}$$

$$X_{ij}^k (f_{ij}^{kr}(\boldsymbol{T}_i^k) - T_j^{kr}) \le 0 \qquad \forall k \in K, \forall (i,j) \in A^k, \forall r \in R^k \tag{1.9}$$

$$\sum_{\substack{i\in N^k\\r\in R^k}} b_{hi}^{kr} T_i^{kr} + \sum_{(i,j)\in A^k} b_{hij}^k X_{ij}^k \le b_h \qquad \forall k \in K, \forall h \in H^k \tag{1.10}$$

$$l_i^{kr} X_i^k \le T_i^{kr} \le u_i^{kr} X_i^k \qquad \forall k \in K, \forall i \in N^k, \forall r \in R^k. \tag{1.11}$$

path linking constraints dans Desaulniers *et al.* (1998). Les contraintes $w \in W$ (1.2) forment un problème de recouvrement généralisé où chaque coefficient a_w du membre de droite prend une valeur entière non négative représentant le nombre de fois qu'une tâche w doit être effectuée. De façon analogue, chaque coefficient a^k_{wij} prend une valeur entière positive représentant le nombre de fois que l'arc $(i,j) \in A^k$ effectue la tâche w. Les contraintes $h \in H$ (1.3) lient globalement plusieurs commodités et sont une extension des contraintes (1.2) : elles utilisent les variables de ressource T^{kr}_i en plus des variables de flot X^k_{ij}. Les valeurs des coefficients b^{kr}_{hi}, b^k_{hij} et b_h dépendent du problème à résoudre et plusieurs exemples sont décrits par Desaulniers *et al.* (1998).

Les variables supplémentaires Y_s, $s \in S$, sont utilisées pour compléter la structure du problème. Leur domaine peut être restreint à un invervalle donné par des bornes inférieure l_s et supérieure u_s (1.4). Les variables Y_s peuvent servir de variables d'écart et de surplus (Barnhart *et al.*, 1998 ; Graves *et al.*, 1993) dont l'utilisation est pénalisée par les coûts c_s dans l'objectif (1.1). Elles peuvent aussi compter le nombre de commodités dans une solution (Desaulniers *et al.*, 1998) ou encore accumuler certaines variables de ressource sur un ensemble de commodités (Ioachim *et al.*, 1998).

Par opposition aux contraintes (1.2)–(1.4), les contraintes (1.5)–(1.11) regroupent les variables par commodité : ce sont les contraintes dites locales, dénommées *single path constraints* dans Desaulniers *et al.* (1998). Les contraintes (1.5)–(1.8) imposent la structure classique d'un chemin unique pour la commodité k dans le réseau G^k, du nœud source o^k au nœud puits d^k. On nomme *itinéraire* un vecteur $\boldsymbol{X}^k$ de variables de flot respectant les contraintes (1.5)–(1.8). Les variables X^k_i, pour tout $i \in N^k$, prennent la valeur 1 si le nœud i est visité par la commodité k, et la valeur 0 sinon. Elles servent à simplifier l'écriture des contraintes (1.11) et peuvent être éliminées par substitution sans affecter la structure du modèle.

Les contraintes (1.9) donnent le mécanisme général utilisant une fonction d'extension $f^{kr}_{ij}(T^k_i)$ quelconque, $(i,j) \in A^k$, pour restreindre les valeurs admissibles de la

variable de ressource T_j^{kr} au nœud j en fonction du vecteur de ressources $\boldsymbol{T}_i^k$ d'un nœud i qui le précède. Une forme linéaire de cette fonction d'extension est apparue pour la première fois dans Desrosiers, Pelletier et Soumis (1983). La formulation non linéaire utilisant la variable binaire X_{ij}^k désactive la relation de préséance entre deux nœuds i et j si l'arc qui les relie n'est pas utilisé dans le chemin emprunté par la commodité k. Les inégalités en (1.9) admettent des solutions où $T_j^{kr} \geq f_{ij}^{kr}(\boldsymbol{T}_i^k)$. Cette opportunité permet, pour une ressource représentant le temps par exemple, d'introduire un certain délai entre le temps d'arrivée au nœud j, représenté par $f_{ij}^{kr}(\boldsymbol{T}_i^k)$, et le temps de début du service à ce même nœud, représenté par T_j^{kr} (*voir* Desrochers, Desrosiers et Solomon, 1992).

Les contraintes $h \in H^k$ (1.10) sont le pendant par commodité des contraintes $h \in H$ (1.3). Elles peuvent limiter localement la construction des chemins en imposant par exemple des contraintes de couplage entre les tâches, de préséance entre les tâches ou d'élimination de cycles (Desaulniers *et al.*, 1998).

Les contraintes (1.11) permettent de restreindre le domaine d'une variable de ressource T_i^{kr} à un intervalle donné par les bornes inférieure l_i^{kr} et supérieure u_i^{kr} si le nœud i est visité par la commodité k, et imposent que cette variable prenne la valeur 0 sinon. Ainsi, les variables de ressource T_i^{kr} associées à un nœud $i \in N^k$ ne contribuent ni à l'objectif (1.1) ni aux contraintes (1.3) et (1.10) si ce nœud i n'est pas visité par la commodité k. On nomme *horaire* un vecteur $\boldsymbol{T}^k$ de variables de ressource associé à un itinéraire $\boldsymbol{X}^k$ et satisfaisant les contraintes (1.9)–(1.11). Les contraintes (1.9)–(1.10) sont alors appelées contraintes de compatibilité itinéraire-horaire.

1.2 Décomposition de la formulation

Les algorithmes de résolution pour les programmes linéaires mixtes peuvent être utilisés sur des problèmes formulés selon (1.1)–(1.11) lorsque les fonctions d'extension f_{ij}^{kr} sont toutes linéaires. Dans ce cas, les contraintes (1.9) peuvent être linéarisées au moyen d'une constante $M \gg 1$ de grande valeur :

$$f_{ij}^{kr}(T_i^k) - T_j^{kr} \leq M(1 - X_{ij}^k) \qquad \forall k \in K, \forall r \in R^k, \forall (i,j) \in A^k. \qquad (1.9')$$

Cependant, les applications qui comportent un haut degré de réalisme nécessitent souvent l'utilisation de fonctions d'extension non linéaires en (1.9). De telles applications sont notamment fréquentes dans le contexte de la construction d'horaires d'équipage ayant des conventions de travail complexes (Desaulniers *et al.*, 1998 ; Desrochers et Soumis, 1989). De plus, les algorithmes de résolution pour les programmes linéaires mixtes tirent difficilement profit de la structure séparable par commodité du modèle et deviennent inefficaces sur des problèmes de grande taille. Une reformulation par décomposition constitue alors une alternative naturelle, permettant d'utiliser des algorithmes de résolution adaptés aux différentes parties du modèle. La prochaine section se penche sur le choix d'une telle décomposition et la section suivante présente une version décomposée du modèle, équivalente à la formulation originale (ou compacte).

1.2.1 Choix de la méthode de décomposition

L'objectif (1.1) et les contraintes (1.5)–(1.11) sont séparables par commodité, formant une structure diagonale en $|K|$ blocs avec les contraintes liantes (1.2)–(1.4). Les méthodes de la relaxation Lagrangienne (Geoffrion, 1974) et de la décomposition

de Dantzig-Wolfe (Dantzig et Wolfe, 1960) sont celles qui tirent le plus facilement profit de ce type de structure. Ces deux méthodes isolent les contraintes séparables par commodité dans des *sous-problèmes* qui peuvent alors être résolus au moyen d'algorithmes spécialisés.

La méthode de la relaxation Lagrangienne introduit des variables duales associées aux contraintes liantes W (1.2) et H (1.3), qui sont relaxées et pénalisées dans l'objectif (1.1). La reformulation du modèle selon cette méthode correspond à convexifier le domaine de chaque sous-problème $k \in K$. Comme cette reformulation n'est pas équivalente à la formulation originale du tableau 1.1, elle doit être utilisée comme composante d'un processus d'énumération défini sur la formulation originale. Dans la littérature, la relaxation Lagrangienne est souvent utilisée en conjonction avec des algorithmes de sous-gradients (Fisher, 1981) ou de faisceaux (Hiriart-Urruty et Lemaréchal, 1993). Ces algorithmes maximisent une borne inférieure $\underline{z} \leq z_{\text{IP}}$ dont la valeur est une fonction des variables duales. Ils ne construisent pas de solution primale de coût équivalent pour le problème convexifié, ce qui complique l'analyse effectuée par les algorithmes de séparation dans le processus d'énumération.

La méthode de décomposition de Dantzig-Wolfe, initialement développée pour des programmes *linéaires* structurés, consiste à reformuler le domaine *convexe* de chaque sous-problème au moyen de combinaisons convexes de leurs points extrêmes. L'application directe de cette méthode à la formulation originale du tableau 1.1 suppose donc la convexification du domaine de chaque sous-problème $k \in K$, relaxant les contraintes d'intégrité (1.8) et les contraintes de compatibilité non linéaires (1.9). Cette approche mène à une reformulation qui est duale, au sens de la programmation linéaire, de celle obtenue par relaxation Lagrangienne (Geoffrion, 1974). Dans la littérature, la décomposition de Dantzig-Wolfe est principalement associée à des algorithmes de génération de colonnes ou de plans coupants (Dantzig et Wolfe, 1960 ; Goffin et Vial, 1990 ; Kelley, 1960 ; Veinott, 1967). Ces algorithmes calculent une

borne inférieure $z_{\text{LP}} \leq z_{\text{IP}}$ et trouvent une solution primale (du problème convexifié) en résolvant un *problème-maître* linéaire, construit à partir de l'objectif (1.1) et des contraintes liantes (1.2)–(1.4).

Comme les deux reformulations mentionnées correspondent à des programmes linéaires duals l'un de l'autre (Geoffrion, 1974), les deux méthodes de décomposition sont équivalentes en ce sens qu'elles fournissent la même borne inférieure $z_{\text{LP}} \leq z_{\text{IP}}$ pour une même définition des sous-problèmes $k \in K$ identifiés par les contraintes (1.5)–(1.11) du modèle unifié. Autrement dit, les relations $\underline{z} \leq z_{\text{LP}} \leq z_{\text{IP}}$ sont vérifiées pour tous les ensembles de variables duales et il existe une combinaison linéaire des contraintes liantes (1.2)–(1.3) telle que la borne $\underline{z}$ obtenue par relaxation Lagrangienne soit égale à z_{LP}.

Cependant, dans le contexte de la résolution de problèmes de grande taille, l'obtention de solutions primales constitue un avantage important. La décomposition de Dantzig-Wolfe semble donc préférable à la relaxation Lagrangienne puisque les algorithmes de génération de colonnes trouvent de telles solutions naturellement. Par contre, la convexification initiale des domaines des sous-problèmes empêche une caractérisation explicite des solutions du modèle unifié après la reformulation, la formulation décomposée n'étant pas équivalente à la formulation originale. La décomposition présentée à la prochaine section permet cette caractérisation en effectuant le changement de variables sous-jacent au principe de décomposition de Dantzig-Wolfe, ce qui ne nécessite aucune hypothèse de convexité *a priori* sur les domaines des sous-problèmes.

1.2.2 Définition et substitution des points extrêmes

À partir d'une structure diagonale avec contraintes liantes, le principe de décomposition de Dantzig-Wolfe consiste à décrire l'ensemble des solutions de chaque sous-

problème au moyen d'une combinaison convexe de ses points extrêmes. Dans la formulation originale du modèle unifié, les contraintes (1.5)–(1.11) définissent, à partir des commodités k, $|K|$ sous-problèmes en variables X_{ij}^k et T_i^{kr} dont les solutions sont des chemins dans G^k. Les points extrêmes utilisés pour décrire ces solutions sont dénotés par l'indice p et regroupés par commodité en ensembles P^k. Ils sont associés à de nouvelles variables de chemin continues, dénotées $\theta_p^k \in [0,1]$. Ces points extrêmes doivent être substitués dans l'objectif (1.1) et dans *toutes* les contraintes (1.2)–(1.11) afin d'obtenir la formulation décomposée du modèle.

Les points extrêmes sont représentés par des vecteurs contenant les valeurs des variables de flot sur les arcs (itinéraire) et de ressource sur les nœuds (horaire) des chemins correspondants :

$$(\boldsymbol{x}_p^k, \boldsymbol{t}_p^k) = (x_{pij}^k, t_{pi}^{kr}), \qquad \forall k \in K, \forall p \in P^k, \forall (i,j) \in A^k, \forall i \in N^k, \forall r \in R^k.$$

Chaque vecteur $(\boldsymbol{x}_p^k, \boldsymbol{t}_p^k)$ est une solution admissible d'un sous-problème k construit à partir des contraintes (1.5)–(1.11). Les variables de la formulation originale s'écrivent en fonction des variables de chemin θ_p^k et des points extrêmes au moyen des relations suivantes :

$$\sum_{p \in P^k} x_{pij}^k \theta_p^k = X_{ij}^k \qquad \forall k \in K, \forall (i,j) \in A^k \tag{1.12}$$

$$\sum_{p \in P^k} t_{pi}^{kr} \theta_p^k = T_i^{kr} \qquad \forall k \in K, \forall i \in N^k, \forall r \in R^k \tag{1.13}$$

$$\sum_{p \in P^k} \theta_p^k = 1 \qquad \forall k \in K \tag{1.14}$$

$$\theta_p^k \geq 0 \qquad \forall k \in K, \forall p \in P^k. \tag{1.15}$$

En substituant les équations (1.12)–(1.15) dans (1.1)–(1.11) et en réarrangeant l'ordre des sommations, le modèle unifié peut être reformulé tel que présenté au tableau 1.2.

Cette formulation décomposée est *équivalente* à la formulation originale. L'objectif (1.16) et les contraintes (1.17)–(1.19) de la formulation décomposée proviennent

Tableau 1.2 – Formulation décomposée du modèle unifié

$$z_{\text{IP}} = \min \sum_{\substack{k \in K \\ p \in P^k}} c_p^k \theta_p^k + \sum_{s \in S} c_s Y_s \tag{1.16}$$

sous les contraintes :

$$\sum_{\substack{k \in K \\ p \in P^k}} a_{wp}^k \theta_p^k + \sum_{s \in S} a_{ws} Y_s = a_w \qquad \forall w \in W \tag{1.17}$$

$$\sum_{\substack{k \in K \\ p \in P^k}} b_{hp}^k \theta_p^k + \sum_{s \in S} b_{hs} Y_s = b_h \qquad \forall h \in H \tag{1.18}$$

$$l_s \leq Y_s \leq u_s \qquad \forall s \in S \tag{1.19}$$

$$\sum_{p \in P^k} \theta_p^k = 1 \qquad \forall k \in K \tag{1.20}$$

$$\theta_p^k \geq 0 \qquad \forall k \in K, \forall p \in P^k \tag{1.21}$$

$$\sum_{p \in P^k} x_{pij}^k \theta_p^k = X_{ij}^k \qquad \forall k \in K, \forall (i,j) \in A^k \tag{1.22}$$

$$X_{ij}^k \in \{0,1\} \qquad \forall k \in K, \forall (i,j) \in A^k \tag{1.23}$$

$$\sum_{p \in P^k} t_{pi}^{kr} \theta_p^k = T_i^{kr} \qquad \forall k \in K, \forall i \in N^k, \forall r \in R^k \tag{1.24}$$

$$X_{ij}^k (f_{ij}^{kr}(T_i^k) - T_j^{kr}) \leq 0 \qquad \forall k \in K, \forall (i,j) \in A^k, \forall r \in R^k \tag{1.25}$$

avec

$$c_p^k = \sum_{(i,j) \in A^k} c_{ij}^k x_{pij}^k + \sum_{r \in R^k} c_i^{kr} t_{pi}^{kr} \qquad \forall k \in K, \forall p \in P^k$$

$$a_{wp}^k = \sum_{(i,j) \in A^k} a_{wij}^k x_{pij}^k \qquad \forall k \in K, \forall p \in P^k, \forall w \in W$$

$$b_{hp}^k = \sum_{\substack{i \in N^k \\ r \in R^k}} b_{hi}^{kr} t_{pi}^{kr} + \sum_{(i,j) \in A^k} b_{hij}^k x_{pij}^k \qquad \forall k \in K, \forall p \in P^k, \forall h \in H.$$

d'une substitution directe des équations (1.12)–(1.13) dans l'objectif (1.1) et les contraintes liantes (1.2)–(1.4) de la formulation originale. Les contraintes (1.22), (1.24), (1.20) et (1.21) constituent des copies des relations (1.12)–(1.15) définissant les combinaisons convexes de points extrêmes. Toutes les contraintes linéaires des sous-problèmes, (1.5)–(1.7) et (1.10)–(1.11), sont absentes de la formulation décomposée puisqu'elles y sont redondantes par rapport aux relations (1.12)–(1.15). Les cas des contraintes (1.8) et (1.9) sont discutés dans les prochains paragraphes.

Dans le cas général, les contraintes d'intégrité (1.8) et les contraintes de compatibilité non linéaires (1.9) ne deviennent pas redondantes suite à un changement de variables par combinaison convexe de points extrêmes. C'est pourquoi les équations (1.22) et (1.24) redéfinissent les variables X_{ij}^k et T_i^{kr} de la formulation originale afin d'inclure dans la formulation décomposée les contraintes (1.23) et (1.25). Cependant, dans deux cas particuliers, les contraintes (1.22)–(1.25) peuvent se simplifier.

Cas 1 : Le premier cas survient lorsque les fonctions d'extension f_{ij}^{kr} sont convexes. Les vecteurs $(\boldsymbol{x}_p^k, \boldsymbol{t}_p^k)$, avec $k \in K$ et $p \in P^k$, satisfont les contraintes (1.5)–(1.11) par définition. Les contraintes (1.8) sur les composantes x_{pij}^k d'un chemin $p \in P^k$, les contraintes (1.23) sur les variables X_{ij}^k et les contraintes de combinaison convexe sur les variables θ_p^k imposent que les chemins $p \in P^k$ d'une commodité $k \in K$ tels que $\theta_p^k > 0$ possèdent tous un même itinéraire, *i.e.*, $\boldsymbol{x}_p^k = \boldsymbol{x}_q^k$ pour tous chemins $p, q \in P^k$ tels que $\theta_p^k > 0$ et $\theta_q^k > 0$. Les contraintes (1.25) correspondant aux arcs $(i,j) \in A^k$ ne faisant pas partie de cet itinéraire sont toutes satisfaites parce que $X_{ij}^k = 0$. Les contraintes (1.25) correspondant aux arcs $(i,j) \in A^k$ faisant partie de l'itinéraire commun deviennent $f_{ij}^{kr}(\boldsymbol{T}_i^k) \leq T_j^{kr}$ puisque $X_{ij}^k = 1$. On peut montrer que ces dernières contraintes sont toutes satisfaites lorsque les fonctions d'extension sont convexes, en utilisant les contraintes (1.9) le long de l'itinéraire commun. En effet, on obtient dans ce cas pour chaque ressource $r \in R^k$:

$$f_{ij}^{kr}(\boldsymbol{T}_i^k) = f_{ij}^{kr}\Big(\sum_{p \in P^k} t_{pi}^k \theta_p^k\Big) \leq \sum_{p \in P^k} f_{ij}^{kr}(t_{pi}^k)\, \theta_p^k \leq \sum_{p \in P^k} t_{pj}^{kr} \theta_p^k = T_j^{kr},$$

avec $t^k_{pi} = (t^{kr}_{pi} \mid r \in R^k)$ dénotant le vecteur de toutes les ressources au nœud $i \in N^k$ d'un chemin $p \in P^k$. La première inégalité découle de la convexité des fonctions f^{kr}_{ij}. La seconde inégalité provient du fait que les chemins $p \in P^k$ avec $\theta^k_p > 0$ ont leur composante $x^k_{pij} = 1$ pour les arcs $(i,j) \in A^k$ faisant partie de l'itinéraire commun et que chaque chemin $p \in P^k$ satisfait les contraintes (1.9) par définition. Les contraintes (1.24)–(1.25) sont donc redondantes et peuvent être éliminées de la formulation décomposée.

Cas 2 : Le second cas survient lorsque, pour toutes les commodités $k \in K$, l'horaire t^k_p d'un chemin $p \in P^k$ est une fonction de l'itinéraire x^k_p de ce chemin. Autrement dit, il n'y a qu'un seul horaire possible pour un itinéraire donné. Alors, les contraintes d'intégrité (1.22)–(1.23) et les contraintes de combinaisons convexes (1.20)–(1.21) imposent que les variables de chemin θ^k_p soient binaires. Les contraintes

$$\theta^k_p \in \{0,1\} \qquad \forall k \in K, \forall p \in P^k \qquad (1.26)$$

peuvent donc être ajoutées à la formulation décomposée du tableau 1.2, permettant d'éliminer toutes les contraintes (1.22)–(1.25) parce qu'elles sont alors toujours satisfaites.

Dans la plupart des applications, il n'existe pas une telle fonction entre itinéraires et horaires, de sorte que les simplifications associées au second cas semblent inutilisables en pratique. En effet, lorsque les bornes des contraintes (1.11) sur les variables de ressource admettent plus d'une valeur ($l^{kr}_i < u^{kr}_i$), il est souvent possible de construire plusieurs horaires pour un même itinéraire. Cependant, lorsque les variables de ressource T^{kr}_i ne contribuent ni à l'objectif (1.1) ni aux contraintes liantes (1.3), le choix d'un horaire parmi les multiples horaires associés à un itinéraire donné n'a aucune influence sur le processus de résolution. Dans ce cas important, les horaires peuvent être partitionnés en classes d'équivalence selon l'itinéraire auquel ils sont associés et on peut ne considérer qu'un horaire représentatif pour chaque classe. Dans ce contexte, et en supposant que l'algorithme de

résolution des sous-problèmes produit toujours le même horaire pour un itinéraire donné, il existe une fonction entre chaque itinéraire et cet horaire représentatif, ce qui permet d'appliquer les conclusions du paragraphe précédent.

On note que c'est le changement de variables sous-jacent au principe de décomposition de Dantzig-Wolfe qui constitue l'élément essentiel de la transformation du modèle menant à une reformulation équivalente. La décomposition de Dantzig-Wolfe, présentée dans le contexte original de programmes linéaires structurés, correspond à une substitution de domaines : les combinaisons convexes de points extrêmes *remplacent* la description des contraintes des sous-problèmes. Dans le cas du modèle unifié, les domaines des sous-problèmes ne sont pas convexes et les combinaisons convexes de points extrêmes ne donnent qu'une approximation extérieure de ces ensembles. Les contraintes des sous-problèmes doivent donc être analysées pour séparer celles qui sont effectivement remplacées de celles qui ne sont qu'approchées par le changement de variables.

1.3 Portée du modèle

La description contrainte par contrainte du modèle unifié effectuée à la section 1.1 ne couvre pas toutes les implications de la formulation utilisée. L'analyse du modèle d'un point de vue global permet d'identifier des contraintes implicites supplémentaires aux niveaux de la structure des solutions admissibles et du potentiel descriptif du modèle. Cette section montre comment éviter la plupart des difficultés et réaffirme la portée très générale du modèle.

1.3.1 Chemins élémentaires

La principale restriction implicite de la formulation originale provient de l'analyse des contraintes (1.5)–(1.11). Elles imposent que le chemin utilisé par le flot d'une commodité $k \in K$ dans G^k soit élémentaire. En effet, il n'y a qu'un seul vecteur $\boldsymbol{T}_i^k$ pour chaque nœud $i \in N^k$, ce qui implique qu'un nœud peut être visité au plus une fois dans un chemin. Or, le problème de trouver des plus courts chemins élémentaires soumis à des contraintes de ressource, dénoté ESPR (*Elementary Shortest Path with Resource constraints*), est un problème NP-difficile au sens fort (Dror, 1994). Des algorithmes pseudo-polynomiaux existent cependant pour résoudre la relaxation du problème ESPR, dénotée SPR (*Shortest Path with Resource constraints*), consistant à admettre des chemins comportant des cycles sous certaines conditions sur les fonctions d'extension f_{ij}^{kr} (*voir* les méthodes de résolution des sous-problèmes à la section 2.3). Si tous les réseaux G^k sont acycliques, les solutions des problèmes SPR définis sur ces réseaux sont élémentaires et satisfont les contraintes (1.5)–(1.11). Sinon, les deux alternatives suivantes permettent de concilier le modèle unifié et les algorithmes pour problèmes SPR qui produisent des solutions avec cycles.

La première alternative consiste à rendre la formulation du modèle compatible avec les solutions produites par les problèmes SPR. Desaulniers *et al.* (1998) proposent pour ce faire de transformer un réseau G^k comportant des cycles en un réseau acyclique. Cette transformation est possible si les fonctions d'extension f_{ij}^{kr} ne permettent que des cycles strictement lexicographiquement positifs sur les variables $\boldsymbol{T}_j^k$ dans R^k sur G^k. Cette condition implique que le réseau G^k est en fait acyclique dans l'espace d'états des variables de ressource. La restructuration du réseau G^k consiste alors en un découpage des intervalles $[l_i^{kr}, u_i^{kr}]$ de chaque nœud $i \in N^k$, entraînant la duplication de nœuds et d'arcs, de façon que le nouveau réseau devienne explicitement acyclique. Dans le cas général, si l'hypothèse sur les cycles strictement lexicographiquement

positifs n'est pas satisfaite, on peut modifier le problème en ajoutant une ressource cumulant le nombre d'arcs le long des chemins, ce nombre étant borné par $|N^k| - 1$ puisqu'une solution ne comporte que des chemins élémentaires. Cette ressource croît de 1 à chaque arc et permet de satisfaire la contrainte de progression lexicographique sur les cycles. Les contraintes (1.9) et (1.11) qui lui sont associées correspondent aux contraintes d'élimination de sous-tours proposées par Miller *et al.* (1960) dans le contexte du problème de voyageur de commerce.

Lorsque le réseau d'une commodité k comporte des cycles, la version acyclique du réseau G^k obtenue au moyen de la transformation précédente n'est pas équivalente au réseau G^k original. Le réseau transformé comporte en effet plusieurs copies d'un nœud $i \in N^k$ ou d'un arc $(i, j) \in A^k$ du réseau G^k original. Afin d'obtenir une formulation équivalente fondée sur la version acyclique du réseau G^k, il faut ajouter à la formulation acyclique des contraintes forçant l'utilisation dans un chemin admissible d'au plus une copie de chaque nœud $i \in N^k$ du réseau G^k original (*voir* Desaulniers *et al.*, 1998). À chaque nœud $i \in N^k$ du réseau G^k original correspond ainsi, dans la formulation acyclique, une contrainte imposant une borne supérieure de 1 sur le flot sortant de l'ensemble de toutes les copies de ce nœud i. Ces contraintes impliquent que, dans une solution admissible de la formulation équivalente utilisant la version acyclique du réseau G^k, au plus une copie d'une variable X_{ij}^k originale possède un flot non nul (valant 1) et au plus une copie d'une variable T_i^{kr} originale possède une valeur de ressource non nulle. Comme une seule copie de chaque variable originale peut prendre une valeur non nulle, on peut définir les valeurs des variables X_{ij}^k et T_i^{kr} de la formulation originale comme la *somme* des valeurs de leurs copies dans la formulation acyclique équivalente. Au moyen de ce changement de variables, le modèle du tableau 1.1 (en particulier l'objectif (1.1) et les contraintes (1.2), (1.3) et (1.10)) peut être complètement reformulé avec des réseaux G^k acycliques et des contraintes d'élimination de cycles explicites.

La deuxième alternative consiste à relaxer les contraintes de chemins élémentaires pour le calcul de bornes inférieures dans le processus de résolution par énumération. Étant donné un chemin obtenu en résolvant un problème SPR défini à partir du sous-problème k, une variable X_{ij}^k compte alors le nombre de fois que l'arc $(i,j) \in A^k$ est utilisé par ce chemin et une variable T_i^{kr} contient la somme, accumulée sur toutes les visites du nœud $i \in N^k$ par ce chemin, des valeurs de la ressource $r \in R^k$ au nœud i. Cette interprétation est conforme au changement de variables présenté dans la première alternative et permet de calculer les coefficients des variables de chemin de la formulation du tableau 1.2. Afin d'améliorer la qualité des bornes inférieures obtenues, l'élimination des 2-*cycles* (cycles de la forme $i \rightarrow j \rightarrow i$ avec $i, j \in N^k$) peut être incorporée aux algorithmes de résolution des problèmes SPR tout en préservant leur complexité pseudo-polynomiale (Desrochers, Desrosiers et Solomon, 1992 ; Kohl et Madsen, 1997 ; Kolen, Rinnooy Kan et Trienekens, 1987). Les autres contraintes d'élimination de cycles sont réintroduites au besoin en cours de résolution dans le processus d'énumération.

Les deux alternatives mènent à un processus de résolution équivalent : dans les deux cas, les contraintes d'élimination de cycles sont relaxées pour le calcul de bornes inférieures puis réintroduites au besoin dans le processus d'énumération. La première alternative consistant à transformer les réseaux G^k originaux en réseaux acycliques permet d'utiliser des algorithmes spécialisés pour réseaux acycliques lors de la résolution des problèmes SPR. Cependant, cette transformation augmente la taille des réseaux et elle rend difficile la dominance entre les états appartenant à des copies d'un même nœud d'un réseau G^k original. Kohl (1995) rapporte que si la dominance n'est pas effectuée entre les états appartenant à des copies d'un même nœud, la transformation augmente en pratique le temps de résolution des problèmes SPR.

1.3.2 Objectif non linéaire

L'objectif (1.1) de la formulation originale est linéaire et séparable par commodité. Cette séparation est nécessaire pour préserver la structure de chaque sous-problème (problème de plus court chemin avec variables de ressource) et rendre possible la résolution efficace du modèle par le biais d'une formulation décomposée. L'utilisation de fonctions convexes c^k séparables par commodité $k \in K$, comme dans l'objectif

$$z_{\text{IP}} = \min \sum_{k \in K} c^k(\boldsymbol{X}^k, \boldsymbol{T}^k) + \sum_{s \in S} c_s Y_s, \tag{1.1'}$$

peut se reformuler de façon équivalente (*voir* du Merle, 1995) comme

$$z_{\text{IP}} = \min \sum_{k \in K} z^k + \sum_{s \in S} c_s Y_s$$

sous les contraintes

$$c^k(\boldsymbol{X}^k, \boldsymbol{T}^k) - z^k \leq 0 \qquad \forall k \in K.$$

Cette transformation met en évidence la généralisation des contraintes (1.10) nécessaire pour supporter un objectif convexe comme (1.1'). Or, l'utilisation conjointe des contraintes (1.9) et (1.10) pose déjà un défi de taille pour le développement de méthodes efficaces de résolution des sous-problèmes. Comme les contraintes (1.9) constituent le facteur principal de succès du modèle unifié, l'intégration explicite d'un objectif convexe général comporte peu d'intérêt dans le cadre de cette thèse. Toutefois, le modèle du tableau 1.1 supporte certaines formes de non linéarité par le biais des fonctions d'extension f_{ij}^{kr} en (1.9). Il est donc possible d'utiliser certaines formes limitées de fonctions non linéaires (pas seulement convexes) dans l'objectif lorsque celui-ci est à la fois séparable par commodité et par arc.

1.3.3 Contraintes d'intégrité supplémentaires

Pour certains problèmes, les solutions admissibles doivent respecter des contraintes d'intégrité supplémentaires sur les variables Y_s et T_i^{kr}, ou un sous-ensemble de

celles-ci. Dans plusieurs cas, ces problèmes possèdent des coefficients entiers et une structure qui rendent les nouvelles contraintes d'intégrité redondantes par rapport aux contraintes (1.8) imposant que les variables de chemin X_{ij}^k soient binaires. Dans les autres cas, ces contraintes d'intégrité ne peuvent pas être prises complètement en considération lors de la résolution du problème-maître, comme exposé à la section 1.2. Ces nouvelles contraintes d'intégrité n'ont alors d'impact que sur le processus d'énumération. Il n'est donc pas nécessaire d'alourdir la notation afin d'inclure ces contraintes d'intégrité supplémentaires explicitement dans le modèle, bien qu'il soit légitime de les utiliser en pratique.

Conclusion

Ce chapitre a servi à présenter et à analyser un modèle unifié destiné, en premier lieu, aux problèmes de tournées de véhicule et d'horaires d'équipage. La formulation porte peu de traces de ces applications d'origine et s'est déjà révélée utile pour modéliser d'autres types d'applications provenant de contextes différents. La décomposition du modèle par substitution des points extrêmes, développée à la section 1.2, constitue une extension du principe de décomposition de Dantzig-Wolfe appliqué à un problème comportant des contraintes non linéaires et des contraintes d'intégrité. L'analyse de la dernière section a traité des difficultés provenant de l'écriture mathématique du modèle et a proposé certaines alternatives et extensions pour les cas qui ne sont toujours pas couverts par le modèle.

Chapitre 2

Revue de la littérature

Les problèmes de tournées de véhicule et d'horaires d'équipage forment une classe importante de problèmes d'optimisation, notamment pour les raisons suivantes. La complexité des méthodes de résolution pose un défi pour la recherche en mathématiques appliquées. Des outils informatiques suffisants existent pour attaquer des problèmes de taille utile en pratique. La modélisation au moyen de ces problèmes met en évidence d'importantes économies potentielles pour l'industrie. Enfin, les modèles développés s'appliquent à d'autres domaines que le transport, comme la productique, le contrôle optimal, la gestion d'équipes de travail en général.

Il existe de nombreux travaux de recherche et de développement sur les problèmes de tournées de véhicule et d'horaires d'équipage. Les deux plus récentes revues de la littérature sur le sujet sont celles de Desrosiers *et al.* (1995) et de Desaulniers *et al.* (1998). Ce chapitre retrace le contexte qui a permis l'élaboration du modèle unifié et le développement du logiciel GENCOL, et le lecteur peut se référer aux deux textes précédents pour une bibliographie plus exhaustive. La première section pose le problème de plus court chemin avec contraintes d'intervalle de temps, duquel est dérivé le modèle unifié. La deuxième présente l'évolution des modèles par ajout de différents types de contraintes, jusqu'à la formulation du modèle unifié. La troisième porte sur les méthodes de résolution utilisées pour traiter le problème-maître, les sous-problèmes et le problème de séparation dans le contexte d'un processus d'énumération appliqué au modèle unifié décomposé. La dernière section décrit plusieurs applications

de recherche et commerciales formulées au moyen du modèle unifié et pouvant être résolues avec le logiciel GENCOL.

2.1 Problème SPTW

L'élément central commun à tous les modèles inclus dans le modèle unifié est le problème de plus court chemin avec contraintes d'intervalle de temps, dénoté SPTW (*Shortest Path with Time Windows*). Ce problème consiste à trouver un chemin sur un graphe $G' = (N', A')$ reliant le nœud origine $o \in N'$ au nœud destination $d \in N'$ au moyen d'une séquence d'arcs $(i, j) \in A'$ de coût c_{ij} et de durée t_{ij}. Ce chemin doit minimiser la somme des coûts des arcs du chemin tout en respectant les contraintes d'intervalle de temps à chaque nœud visité. Afin de garantir l'existence d'une solution dans le cas général où G' comporte des cycles et où les coûts c_{ij} des arcs $(i, j) \in A'$ sont quelconques, on impose que tout cycle C de G' ait une durée totale positive, soit $\sum_{(i,j)\in C} t_{ij} > 0$, et qu'au moins un nœud i tel que $(i, j) \in C$ ait un temps maximal u_i de visite fini, soit $u_i < \infty$. La recherche sur ce type de problème remonte au début des années 1980. Desrosiers, Pelletier et Soumis (1983) sont les premiers à proposer un algorithme pseudo-polynomial pour résoudre le problème SPTW sous l'hypothèse supplémentaire que $t_{ij} > 0$ pour tout arc $(i, j) \in A'$.

La possibilité de multiples passages à certains nœuds empêche la formulation directe du problème SPTW en programme mathématique. La formulation la plus proche est celle du problème de plus court chemin *élémentaire* avec contraintes d'intervalle de temps, dénoté ESPTW (*Elementary Shortest Path with Time Windows*), qui s'écrit mathématiquement au moyen du programme non linéaire suivant :

$$\min \sum_{(i,j)\in A} c_{ij} X_{ij} \tag{2.1}$$

sous les contraintes :

$$\sum_{j:(o,j)\in A} X_{oj} = \sum_{i:(i,d)\in A} X_{id} = 1 \tag{2.2}$$

$$\sum_{j:(i,j)\in A} X_{ij} - \sum_{j:(j,i)\in A} X_{ji} = 0 \qquad \forall i \in N \setminus \{o,d\} \tag{2.3}$$

$$X_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{2.4}$$

$$X_{ij}(T_i + t_{ij} - T_j) \leq 0 \qquad \forall (i,j) \in A \tag{2.5}$$

$$l_i \leq T_i \leq u_i \qquad \forall i \in N. \tag{2.6}$$

La section 1.3.1 donne les conditions générales selon lesquelles le problème SPTW peut être reformulé comme un problème ESPTW (2.1)–(2.6). La transformation consiste à construire un graphe $G = (N, A)$ acyclique équivalent au graphe $G' = (N', A')$ cyclique initial en discrétisant minimalement les intervalles de temps. Les deux graphes G' et G sont équivalents dans le sens qu'à tout chemin de G' correspond un chemin de G de même coût satisfaisant (2.2)–(2.6), et inversement. La formulation SPTW littérale basée sur le graphe G' est utilisée au niveau du développement d'algorithmes de résolution, alors que la formulation ESPTW basée sur le graphe G est utilisée au niveau de l'analyse mathématique des modèles. Il est important de noter que la résolution du problème ESPTW sur un graphe G comprenant des cycles est rarement considérée, puisque ce problème est réputé NP-difficile au sens fort (Dror, 1994).

Plusieurs extensions au problème SPTW ont été décrites dans la littérature afin de mieux modéliser certains aspects d'applications pratiques. Parallèlement à ces efforts pour augmenter la portée des modèles basés sur le problème SPTW, certains groupes de chercheurs ont tenté d'exprimer au moyen d'un seul modèle l'ensemble des problèmes de tournées de véhicule et d'horaires d'équipage. Bodin et Golden (1981), Carraresi et Gallo (1984), Desrochers *et al.* (1988), Solomon et Desrosiers

(1988) ainsi que Desrosiers *et al.* (1995) ont contribué à la définition de ce modèle général. La présente thèse est basée directement sur le modèle dit *unifié* décrit par Desaulniers *et al.* (1998) et présenté au chapitre 1. Ce modèle permet d'analyser une grande classe de problèmes à partir d'un cadre théorique unique et de développer des algorithmes génériques de résolution, ce qui constitue un avantage certain par rapport à l'analyse de problèmes et de modèles spécifiques.

2.2 Du problème SPTW au modèle unifié

Cette section présente l'histoire et la construction du modèle unifié décrivant le problème m-TSPR (*voir* section 1.1) à partir du problème SPTW. D'une part, on ajoute à ce dernier des contraintes liantes (1.2)–(1.4) menant à la définition d'un problème-maître et de sous-problèmes après décomposition selon le principe de Dantzig-Wolfe. D'autre part, on intègre aux sous-problèmes SPTW un espace de ressources à plusieurs dimensions (1.9)–(1.11) et de nouvelles contraintes linéaires (1.10) et non linéaires (1.9).

L'existence du modèle unifié commence avec les travaux de Desrosiers, Pelletier et Soumis (1983) sur le problème SPTW, réputé NP-difficile, pour lequel ils proposent un algorithme pseudo-polynomial suffisamment efficace permettant d'utiliser SPTW comme sous-problème dans le traitement de problèmes plus complexes. Ainsi, Desrosiers, Soumis et Desrochers (1984) étudient le problème m-TSPTW (*multiple-Traveling Salesman Problem with Time Windows*) consistant à couvrir à coût minimum un ensemble de tâches associées aux nœuds d'un graphe au moyen de plusieurs véhicules. Le modèle correspondant s'obtient en dupliquant la formulation (2.1)–(2.6) en autant de copies qu'il y a de véhicules disponibles, puis en adjoignant à ces formulations jusqu'ici indépendantes des contraintes liantes assurant que chaque tâche est

couverte par un et un seul véhicule. La duplication des graphes mènera plus tard à la notion de commodités indépendantes (indices $k \in K$) dans le modèle unifié, mais tous les réseaux sont considérés identiques dans l'article original ; les contraintes liantes de partitionnement deviendront les contraintes (1.2) dans le modèle unifié. Les auteurs résolvent le problème m-TSPTW au moyen d'un processus de séparation et d'évaluation, intégrant des méthodes de branchement et de coupes. Ils utilisent la décomposition de Dantzig-Wolfe pour calculer les bornes inférieures par génération de colonnes, où les sous-problèmes sont des problèmes SPTW et où la coordination entre ces derniers incombe à un problème-maître formulé comme un programme linéaire.

Les contraintes liantes du problème m-TSPTW ne sont pas les plus générales possibles : entre autres, elles forment uniquement un problème de partitionnement et ne dépendent que des variables de flot. Plusieurs chercheurs ont introduit de nouvelles contraintes liantes linéaires afin d'étendre la portée du modèle m-TSPTW à d'autres applications. Ces contraintes ont l'avantage de préserver les méthodes de décomposition applicables au problème m-TSPTW et les algorithmes de résolution du problème-maître correspondant. Le modèle unifié tente donc d'englober tous ces modèles en incluant tous les types possibles de contraintes liantes linéaires. D'éventuelles contraintes liantes non linéaires définissant des ensembles convexes peuvent aussi être prises en charge en les linéarisant *a priori* ou encore en les relaxant du problème-maître initial et en les réintégrant comme coupes. Par ailleurs, d'autres contraintes liantes non linéaires définissant des ensembles non convexes, telles les contraintes d'intégrité sur les variables X_{ij}^k (1.8), doivent être relaxées du problème-maître initial puis réintégrées dans le processus d'énumération.

Plus spécifiquement, Desrosiers, Soumis et Desrochers (1984) généralisent l'espace des variables participant aux contraintes liantes en ajoutant les variables supplémentaires Y_s, $s \in S$ (1.4). Ils utilisent aussi d'autres contraintes que celles de partitionnement, faisant intervenir uniquement les variables de flot, pour obtenir l'équivalent

des contraintes (1.3) sans variables de ressource. On note également que Desrosiers, Soumis et Desrochers (1984) puis Desrochers et Soumis (1989) définissent leur ensemble de tâches W (1.2) indépendamment de celui des nœuds N, ce qui leur permet de modéliser certaines applications en associant les tâches $w \in W$ aux arcs, obtenant ainsi des graphes dont la taille est linéaire en fonction de $|W|$. Finalement, Ioachim *et al.* (1998) étudient un problème où les variables de ressource contribuent aux contraintes liantes (1.3), ce qui mène à des sous-problèmes de type SPTW dont l'objectif dépend à la fois des variables de flot et des variables de ressource.

Contrairement aux extensions précédentes ne changeant pas la nature du problème-maître, celles généralisant la formulation des sous-problèmes impliquent la plupart du temps le développement de nouveaux algorithmes de résolution spécialisés. La première extension du problème SPTW provient de Desrochers (1986) et Desrochers et Soumis (1988a, 1988b), qui introduisent l'idée d'un espace de ressources multidimensionnel en dupliquant les contraintes (2.5)–(2.6), obtenant ainsi les contraintes (1.9) et (1.11) du modèle unifié. Ils nomment ce nouveau problème SPR (*Shortest Path problem with Resource constraints*), pour lequel ils développent un algorithme pseudo-polynomial dont la complexité est plus faible que celle de l'algorithme de Desrosiers, Pelletier et Soumis (1983) lorsqu'appliqué au problème SPTW original. (Afin de limiter le nombre d'acronymes dénotant les multiples variantes et extensions du problème SPTW, l'acronyme SPR désignera dorénavant tous les problèmes de plus court chemin avec contraintes, tels que définis par l'objectif (1.1) et les contraintes (1.5)–(1.11) du modèle unifié.)

Par la suite, Dumas, Desrosiers et Soumis (1991) étudient le cas particulier du problème m-TSPTW où chaque tâche est une requête comprenant à la fois la cueillette et la livraison d'un bien. Ils proposent un algorithme pseudo-polynomial pour résoudre les sous-problèmes résultants, de type SPTW avec contraintes de couplage et de préséance entre deux nœuds du graphe représentant respectivement la cueillette et la

livraison. Ces contraintes sont un cas particulier des contraintes (1.10) du modèle unifié (*voir* Desaulniers *et al.*, 1998) : les contraintes de couplage s'écrivent mathématiquement au moyen de combinaisons linéaires des variables de flot, tandis que les contraintes de préséance portent sur les variables de ressource (celles représentant le temps). Une autre généralisation du problème m-TSPTW qui mène à des contraintes de type (1.10) portant uniquement sur les variables de flot concerne l'élimination des cycles de longueur deux, nommés 2-*cycles*, sur les tâches. Ce problème provient de la possibilité de cycles sur les tâches lors de la résolution des sous-problèmes SPTW, alors que ces cycles ne peuvent être présents dans une solution entière du problème m-TSPTW à cause des contraintes de partitionnement sur les tâches. Houck *et al.* (1980) proposent un algorithme pseudo-polynomial pour résoudre ce problème, et Kolen, Rinnooy Kan et Trienekens (1987), Desrochers, Desrosiers et Solomon (1992) et Kohl et Madsen (1997) l'utilisent dans le contexte de la résolution du problème VRPTW (*Vehicle Routing Problem with Time Windows*).

Les algorithmes développés pour traiter le problème SPTW avec contraintes de couplage et de préséance ou avec contraintes d'élimination des 2-cycles n'exploitent pas la structure linéaire, très générale, des contraintes (1.10) correspondantes. Ce sont plutôt des algorithmes spécialisés qui tirent profit de la structure particulière des problèmes à résoudre. En théorie, le cas restreint de contraintes (1.10) ne contenant que des variables de flot peut être traité en définissant des variables de ressource supplémentaires (une ressource par contrainte) mais en pratique, cette dernière transformation rend le processus de résolution rapidement inefficace. Ainsi, on peut affirmer qu'il n'existe pas d'algorithme efficace pouvant résoudre des sous-problèmes formulés au moyen de toutes les contraintes (1.5)–(1.11) du modèle unifié. Il faut donc prévoir le développement de nouveaux algorithmes spécialisés pour traiter d'autres formes particulières des contraintes (1.10) ou encore les inclure dans les contraintes (1.3) du problème-maître.

Le problème SPR du modèle unifié comporte aussi deux autres extensions qui ont reçu moins d'attention dans la littérature. D'une part, les fonctions d'extension f_{ij}^{kr} utilisées dans les contraintes (1.9) remplacent les translations des contraintes (2.5) comme mécanisme général de transformation des ressources le long des chemins. Ces fonctions constituent le facteur principal de succès du modèle unifié dans les applications commerciales en permettant d'atteindre un haut degré de réalisme au niveau des contraintes sur la confection des chemins (*voir*, par exemple, Desrochers et Soumis, 1989). On note toutefois que ce type de fonctions d'extension apparaît déjà dans les premiers travaux de Desrosiers, Pelletier et Soumis (1983), mais que les implications reliées à cette formulation n'ont été bien comprises qu'environ 10 ans plus tard. D'autre part, Desaulniers *et al.* (1998) posent l'objectif sous une forme générale semblable à celle utilisée en (1.1) et énoncent une condition suffisante sur l'objectif et les fonctions d'extension des sous-problèmes pour étendre la portée de l'algorithme de Desrochers et Soumis (1988a). Cette condition impose que les coefficients des variables de ressource dans l'objectif du sous-problème soient non négatifs et que les fonctions d'extension soient non décroissantes en fonction des variables de ressource. Finalement, Ioachim *et al.* (1998) considèrent quant à eux un problème dont les coefficients des variables de ressource dans l'objectif des sous-problèmes peuvent prendre des valeurs quelconques et développent un nouvel algorithme pseudo-polynomial pour le résoudre.

La figure 2.1 présente les relations entre les différents problèmes découlant du problème SPTW, jusqu'au problème m-TSPR correspondant au modèle unifié. Les flèches indiquent un lien de complexité croissante entre deux problèmes. Les problèmes de la deuxième ligne sont similaires à ceux de la troisième sauf que les solutions ne comportent qu'un seul chemin, d'où l'appellation commençant par TSP (*Traveling Salesman Problem*). Dans le cas du problème TSPTW, chaque nœud est identifié à une tâche à couvrir et est sujet à des contraintes d'intervalle de temps. À la suite

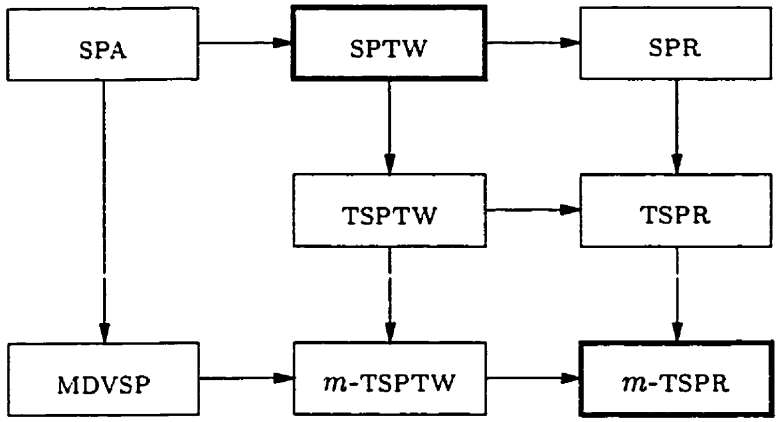


Figure 2.1 – Relations de complexité entre différents problèmes reliés à SPTW

des travaux de Psaraftis (1980, 1983), Desrosiers, Dumas et Soumis (1986) proposent un algorithme de programmation dynamique pour résoudre un cas particulier du problème TSPTW avec contraintes de cueillette et de livraison. Dumas *et al.* (1995) spécialisent ensuite cet algorithme pour résoudre le problème TSPTW classique. Ces algorithmes, de complexité exponentielle, sont efficaces à condition que les intervalles de temps soient suffisamment étroits. La figure 2.1 inclut aussi, dans la colonne de gauche, les problèmes SPA et MDVSP. Le problème SPA (*Shortest Path on a Acyclic graph*) de plus court chemin sur un graphe acyclique (Fulkerson, 1972) constitue le prédécesseur naturel du problème SPTW, puisque l'algorithme de Desrochers et Soumis (1988a) exploite essentiellement la structure acyclique du problème SPTW dans l'espace des ressources. Le problème MDVSP (*Multiple Depot Vehicle Scheduling Problem*) (Bertossi, Carraresi et Gallo, 1987; Carpaneto *et al.*, 1989; Ribeiro et Soumis, 1994) est équivalent au problème m-TSPTW sans variables de ressource, où les sous-problèmes sont des problèmes SPA.

Cette section illustre donc l'affirmation initiale du chapitre 1 selon laquelle le modèle unifié contient la plupart des problèmes de tournées de véhicule et d'horaires d'équipage actuellement étudiés dans la littérature. Le modèle unifié, de par ses multiples

extensions, précède le développement d'algorithmes de résolution capables de le traiter dans son intégrité. Cette situation a l'avantage de fournir un cadre théorique stable commun à tous les chercheurs développant de nouvelles méthodes de résolution. De plus, le modèle unifié isole l'écart entre les algorithmes et la formulation mathématique dans les sous-problèmes de type SPR et met en évidence le caractère générique des algorithmes de séparation et de résolution du problème-maître.

2.3 Méthodes de résolution du modèle unifié décomposé

Le chapitre 1 présente le modèle unifié de Desaulniers *et al.* (1998) sous sa forme originale (1.1)–(1.11) ainsi que sous une forme décomposée (1.16)–(1.25) obtenue au moyen du principe de décomposition de Dantzig-Wolfe. La résolution de ce modèle passe par un processus d'énumération puisque les contraintes d'intégrité (1.23) et les contraintes de compatibilité non linéaires (1.25) ne peuvent être prises directement en compte par le problème-maître linéaire (1.16)–(1.21). Le processus d'énumération consiste à explorer un arbre dont les nœuds correspondent à des problèmes *résiduels* obtenus par division (et préférablement par partition) du domaine de leur problème père (Nemhauser et Wolsey, 1988). Chacun de ces problèmes résiduels possède par hypothèse la même structure que le problème original à la racine de l'arbre. Afin de simplifier la notation, la résolution du problème-maître et des sous-problèmes est considérée dans cette section par rapport à un problème résiduel donné, et non par rapport au problème complet (non divisé) à la racine de l'arbre d'énumération. Le reste de cette section donne un aperçu de la littérature sur la résolution du problème-maître et des sous-problèmes ainsi que sur le processus de séparation dans le contexte de la décomposition de Dantzig-Wolfe du modèle unifié.

Problème-maître : Le problème-maître (1.16)–(1.21) permet d'obtenir une borne inférieure z_{LP} sur la solution optimale z_{IP} d'un problème résiduel. Une borne inférieure $\underline{z} \leq z_{\text{LP}}$ peut aussi être obtenue à partir de la relaxation Lagrangienne de la formulation du tableau 1.1 par rapport aux contraintes (1.2)–(1.3). Cette borne inférieure est une fonction des variables duales associées aux contraintes relaxées. Elle peut être maximisée dans l'espace dual pour atteindre la valeur z_{LP} (Geoffrion, 1974). Nous exprimons cette borne inférieure à partir de la formulation décomposée du tableau 1.2. En associant aux ensembles de contraintes (1.17), (1.18) et (1.20) les vecteurs de variables duales α, β et γ respectivement, la borne inférieure $\underline{z}(\alpha, \beta, \gamma)$ peut être calculée au moyen de l'équation suivante :

$$\begin{aligned} \underline{z}(\alpha, \beta, \gamma) = & \sum_{w \in W} a_w \alpha_w + \sum_{h \in H} b_h \beta_h + \sum_{k \in K} \gamma^k \\ & + \sum_{k \in K} g^k(\alpha, \beta, \gamma) \\ & + \sum_{s \in S : \bar{c}_s < 0} \bar{c}_s(\alpha, \beta)\, u_s + \sum_{s \in S : \bar{c}_s \geq 0} \bar{c}_s(\alpha, \beta)\, l_s, \end{aligned} \tag{2.7}$$

où $g^k(\alpha, \beta, \gamma) = \min_{p \in P^k} \bar{c}_p^k(\alpha, \beta, \gamma)$, avec $\bar{c}_p^k$ et $\bar{c}_s$ dénotant les fonctions qui évaluent respectivement les coûts réduits des variables θ_p^k et Y_s. Notons que les variables duales γ sont superflues en (2.7) puisqu'elles s'annulent de par la définition du coût réduit $\bar{c}_p^k$ des variables θ_p^k impliquant un terme $-\gamma^k$. Elles sont utilisées uniquement pour permettre une définition du coût réduit des variables θ_p^k conforme à la formulation décomposée du tableau 1.2.

Les fonctions g^k sont concaves et linéaires par morceaux, étant définies comme le minimum d'un nombre fini de formes affines. Les termes impliquant les coûts réduits $\bar{c}_s$ des variables Y_s en (2.7) peuvent être remplacés par des contraintes linéaires dans l'espace dual, impliquant possiblement des variables duales supplémentaires pour tenir compte des bornes (1.19). Le problème de maximiser $\underline{z}(\alpha, \beta, \gamma)$ est donc analogue au problème de minimiser une fonction convexe non partout différentiable

sur un domaine convexe, problème pour lequel il existe une abondante littérature (*voir* Hirriart-Urruty et Lemaréchal, 1993; Goffin et Vial, 1999). Nous dénotons par D l'ensemble convexe des variables duales (α, β, γ) satisfaisant les contraintes $g^k(\alpha, \beta, \gamma) \geq 0$ pour tout $k \in K$ et par E l'ensemble convexe des variables duales satisfaisant les contraintes provenant des variables supplémentaires Y_s, $s \in S$. Le problème dual du problème-maître (1.16)–(1.21) consiste alors à maximiser l'objectif dual linéaire suivant :

$$\sum_{w \in W} a_w \alpha_w + \sum_{h \in H} b_h \beta_h + \sum_{k \in K} \gamma^k, \tag{2.8}$$

sur le domaine convexe $D \cap E$. Les prochains paragraphes présentent les principales approches utilisées pour résoudre ce problème.

La formulation décomposée du tableau 1.2 mène naturellement à la définition de problèmes-maîtres restreints à des sous-ensembles de P^k, $k \in K$. L'algorithme de Dantzig-Wolfe (1960) constitue une application de la méthode du simplexe révisé où les sous-problèmes $k \in K$ sont évalués au point dual correspondant à la solution optimale du problème-maître restreint courant. Si tous les chemins $p \in P^k$ de toutes les commodités $k \in K$ ont un coût réduit non négatif, l'algorithme se termine avec une solution optimale du problème-maître (1.16)–(1.21). Sinon, un chemin $p \in P^k$ d'une commodité $k \in K$ est de coût réduit négatif et la variable θ_p^k correspondante est ajoutée au problème-maître restreint qui doit alors être réoptimisé.

L'algorithme de Kelley (1960), dual de celui de Dantzig-Wolfe (1960), consiste à résoudre le problème dual en enrichissant itérativement une approximation linéaire extérieure $\hat{D}$ de l'ensemble D. Les sous-problèmes $k \in K$ sont évalués au point dual maximisant (2.8) sur l'ensemble $\hat{D} \cap E$ courant. Si le point dual est dans l'ensemble D, la solution est optimale. Sinon, il existe au moins un sous-problème $k \in K$ pour lequel $g^k(\alpha, \beta, \gamma) < 0$. Un sous-gradient de la fonction g^k au point (α, β, γ), correspondant au vecteur de coefficients d'une variable θ_p^k associée à un chemin $p \in P^k$

définissant g^k au point (α, β, γ), permet alors d'ajouter une contrainte supplémentaire enrichissant l'approximation $\hat{D}$ de D. La convergence de l'algorithme est assurée si les sous-gradients des fonctions g^k sont de norme finie (Kelley, 1960). Cette condition est satisfaite dans le cas de la formulation du tableau 1.2 puisque les coefficients des variables θ_p^k sont calculés comme des combinaisons linéaires des valeurs des variables X_{ij}^k binaires (1.8) et des variables T_i^{kr} bornées (1.11).

Le taux de convergence des algorithmes de Dantzig-Wolfe (1960) et de Kelley (1960) est très mauvais en théorie et en pratique (*voir* Goffin et Vial, 1999). Cependant, la génération de plusieurs colonnes (sous-gradients) à chaque évaluation des sous-problèmes (*voir* du Merle, 1995 ; Kohl, 1995) ainsi que l'utilisation de stratégies de stabilisation dans l'espace dual (*voir* le chapitre 5) pallient en partie cet inconvénient. Nous notons que les deux algorithmes sont peu sensibles à la résolution heuristique des sous-problèmes parce qu'ils progressent dès qu'un sous-problème permet de séparer le point dual courant de l'ensemble D, ce qui correspond à produire un chemin $p \in P^k$ de coût réduit négatif. Cependant, sans garantie qu'un tel chemin est de coût réduit minimum, la résolution heuristique des sous-problèmes ne permet pas de calculer la valeur d'une borne inférieure $\underline{z}$ au moyen de (2.7).

Veinott (1967) propose une variante de l'algorithme de Kelley (1960) pour générer des contraintes appuyant sur l'ensemble $D \cap E$. Cette variante suppose qu'un point dual initial inclus dans $D \cap E$ est connu. Cette approche permet intuitivement de couper des portions plus importantes de l'ensemble $\hat{D}$ à chaque itération. L'algorithme consiste d'abord à calculer un point dual maximisant (2.8) sur $\hat{D}$ comme avec l'algorithme de Kelley et à trouver ensuite, sur la droite reliant ce point dual maximal et le point dual initial, un point dual légèrement à l'extérieur de l'ensemble $D \cap E$ (les points à la frontière de $D \cap E$ posent des difficultés si certains sous-gradients des fonctions g^k y sont nuls). Une façon de calculer un tel point consiste à utiliser les sous-problèmes dans un processus de bissection entre le point dual maximal et le

point dual initial. Les nouvelles contraintes sont définies à partir des sous-gradients des fonctions g^k évaluées au point ainsi obtenu. Malheureusement, cet algorithme ne possède pas de meilleures propriétés théoriques de convergence que l'algorithme de Kelley. En pratique, parce qu'il nécessite plus d'évaluations des sous-problèmes, l'algorithme de Veinott converge souvent en plus de temps (sinon d'itérations) que l'algorithme de Kelley.

Elzinga et Moore (1975) proposent une modification de l'algorithme de Kelley (1960) qui consiste à évaluer les sous-problèmes au point dual situé au centre de la plus grande hypersphère inscrite dans $\hat{D} \cap E$. Ce point peut être obtenu comme la solution d'un programme linéaire auxiliaire (Nemhauser et Widhelm, 1971) comportant une dimension supplémentaire dans l'espace dual. La variable duale associée correspond à la longueur (non négative) du rayon de l'hypersphère, qui doit être maximisée. L'objectif dual original (2.8) est récupéré dans le programme linéaire auxiliaire sous forme de contrainte associée à une borne inférieure $\underline{z}$. Si le point dual (α, β, γ) est un élément de $D \cap E$, l'évaluation de l'objectif dual (2.8) permet de mettre à jour la borne inférieure $\underline{z}$. Sinon, il existe un sous-problème $k \in K$ pour lequel $g^k(\alpha, \beta, \gamma) < 0$ et une contrainte définie à partir du sous-gradient de g^k au point dual courant est ajoutée à la description de $\hat{D}$. Les auteurs prouvent la convergence de leur algorithme, qui s'arrête lorsque la longueur du rayon s'annule. Ils montrent aussi que les valeurs de l'objectif associées à la sous-séquence des points duals générés réalisables (*i.e.*, qui sont éléments de $D \cap E$) converge linéairement vers la valeur optimale de l'objectif (2.8). L'algorithme est moins robuste que l'algorithme de Kelley (1960) face à la résolution heuristique des sous-problèmes puisqu'il requiert le calcul exact des fonctions g^k afin de déterminer correctement si le point dual courant est un élément de D ou non. Comme l'algorithme de Veinott (1967), il est rarement mentionné dans la littérature lors de comparaisons avec d'autres méthodes d'optimisation.

D'autres algorithmes ont aussi été développés selon cette idée de calculer des points duals au centre de l'ensemble $\hat{D}$ courant, pour diverses définitions de *centre* (*voir*

Goffin et Vial, 1999). Parmi les plus récents travaux, Goffin et Vial (1990) proposent l'algorithme ACCPM (*Analytic Center Cutting Plane Method*) où le centre analytique est défini comme le point dual maximisant le logarithme du produit des variables d'écart associées aux contraintes de l'ensemble $\hat{D}$. L'algorithme s'inspire de la méthode projective de points intérieurs en programmation linéaire et nécessite la minimisation d'une fonction potentiel convexe sous contraintes linéaires (*voir* du Merle, 1995; Goffin et Vial, 1999). Dans le contexte de la résolution du problème de réalisabilité convexe (trouver un point dual dans $D \cap E$, sans objectif à optimiser), Goffin, Luo et Ye (1996) démontrent que l'algorithme ACCPM est un schéma d'approximation pleinement polynomial (Cormen *et al.*, 1990), où la complexité est mesurée en nombre d'évaluations des sous-problèmes. Les auteurs montrent aussi que le nombre de pas de Newton nécessaires pour recalculer un nouveau centre analytique après l'ajout de coupes à l'ensemble $\hat{D}$ est borné par une constante. Comme l'algorithme d'Elzinga et Moore (1975), l'algorithme ACCPM suppose la résolution exacte des sous-problèmes.

D'autres chercheurs (*voir* Hirriart-Urruty et Lemaréchal, 1993; Kiwiel, 1989) ont tenté de pallier l'instabilité de l'algorithme de Kelley (1960) en ajoutant un terme quadratique à l'objectif (2.8) dans le but de forcer le prochain point dual à demeurer dans le voisinage d'un *bon* point dual trouvé précédemment. Ces méthodes nécessitent la résolution d'un programme quadratique sous contraintes linéaires pour calculer le prochaine point dual où sont évalués les sous-problèmes. Lemaréchal *et al.* (1991) prouvent que certaines variantes de méthodes de faisceaux sont des schémas d'approximation pseudo-polynomials, où la complexité est mesurée en nombre d'évaluations des sous-problèmes.

Les méthodes décrites jusqu'à maintenant accumulent l'information provenant des sous-problèmes afin d'enrichir une approximation extérieure $\hat{D}$ de l'ensemble D. L'algorithme des sous-gradients, qui s'apparente à la méthode de la plus forte pente

utilisée pour l'optimisation de fonctions différentiables, est basé sur une stratégie différente ne requérant pas de conserver l'historique, même partiel, des itérations précédentes. Le prochain point dual est obtenu en avançant simplement d'un pas non nul dans la direction d'un sous-gradient d'une fonction g^k évaluée au point dual courant. Plusieurs règles ont été développées pour choisir des pas assurant la convergence de l'algorithme (*voir* Shor, 1985). Ermoliev (1966) et Polyak (1967) montrent que l'algorithme converge à condition que, lorsque le nombre d'itérations tend vers l'infini, les pas tendent individuellement vers 0 et leur somme tende vers l'infini. Goffin (1977) étudie les conditions permettant d'obtenir un taux de convergence géométrique. Kohl (1995) propose une variante de l'algorithme tenant compte des sous-gradients générés aux itérations antérieures. Rochon (1997) tire profit de l'algorithme des sous-gradients pour calculer plus rapidement de nouveaux points duals entre deux itérations de l'algorithme de Kelley (1960).

Dans le contexte de la résolution du problème VRPTW (un cas particulier du problème m-TSPR), Kohl (1995) effectue une comparaison des algorithmes associés respectivement aux méthodes de décomposition de Dantzig-Wolfe et de la relaxation Lagrangienne. Il conclut à ce sujet que l'algorithme de Dantzig-Wolfe est numériquement plus stable et se programme plus facilement que tous les algorithmes associés à la relaxation Lagrangienne qu'il a considérés (méthode traditionnelle de sous-gradients et variantes, méthodes de faisceaux avec programme quadratique). Pour résoudre les mêmes problèmes, Kohl et Madsen (1997) utilisent un algorithme hybride de sous-gradients et de faisceaux. Ils montrent que le temps de résolution des sous-problèmes augmente lorsque les valeurs des variables duales sont grandes, ce qui défavorise l'algorithme de Kelley (1960). De leur côté, Goffin et Vial (1999) mentionnent plusieurs applications, dont des problèmes de flot multi-commodités linéaires et non linéaires, pour lesquels ils concluent que l'algorithme ACCPM, sans être toujours plus rapide que l'algorithme de Kelley (1960), se comporte de façon plus prévisible dans tous les cas, ce qui est en accord avec la théorie des méthodes de points intérieurs sous-jacente à leurs travaux.

Sous-problèmes : Les sous-problèmes correspondent aux contraintes (1.5)–(1.11) du modèle unifié auxquelles on adjoint un objectif linéaire permettant d'identifier le chemin $p \in P^k$ de coût marginal minimum pour chaque commodité $k \in K$. L'objectif des sous-problèmes découle, d'une part, de l'objectif (1.16) en fonction des variables de flot et de ressource et, d'autre part, des variables duales associées aux contraintes (1.17)–(1.18). Le problème SPR obtenu constitue un problème de plus court chemin avec contraintes de ressource pour lequel il n'existe actuellement aucun algorithme général efficace de résolution. La recherche porte donc sur le développement d'algorithmes spécialisés résolvant des formulations plus spécifiques du problème SPR ayant comme prédécesseur commun le problème SPTW.

Desrosiers, Pelletier et Soumis (1983) développent un algorithme pseudo-polynomial de type *label-correcting* pour résoudre le cas particulier du problème SPTW avec durées t_{ij} positives, formulé mathématiquement en (2.1)–(2.4). On peut définir la taille τ d'un problème donné comme $\sum_{i \in N}(u_i - l_i + 1)$ et leur algorithme est alors de complexité $O(\tau^3)$. Desrochers et Soumis (1988a) exploitent plus à fond l'hypothèse que $t_{ij} > 0$ et obtiennent un algorithme de type *label-setting* de complexité $O(\tau^2)$. Desrochers et Soumis (1988b) présentent ensuite un algorithme de réoptimisation du problème SPTW basé sur une approche primale-duale. Lingaya (1996) relaxe la condition sur les durées t_{ij} positives, en conservant tout de même la contrainte que $\sum_{(i,j) \in C} t_{ij} > 0$ pour tout cycle C.

Desrochers (1986) développe une première généralisation du problème SPTW en traitant le cas avec plusieurs variables de ressource, *i.e.*, $|R^k| > 1$ pour une commodité $k \in K$ donnée (*voir* aussi Desrosiers *et al.*, 1995). Il utilise la structure de données des SPLAY (Sleator et Tarjan, 1985) et les algorithmes de Kung, Luccio et Preparata (1975) pour identifier efficacement les états non dominés, pour toutes les valeurs de $|R^k|$. Desaulniers *et al.* (1998) montrent que l'algorithme de Desrochers (1986) permet de résoudre des problèmes avec fonctions d'extension f_{ij}^{kr} (1.9) non décroissantes (1.9) et coefficients c_i^{kr} non négatifs (1.1).

Dans le cadre de la résolution du problème VRPTW comportant deux ressources, Kolen, Rinnooy Kan et Trienekens (1987), Desrochers, Desrosiers et Solomon (1992) et Kohl et Madsen (1997) incorporent des contraintes d'élimination des 2-cycles au problème SPTW comme stratégie d'accélération globale. L'élimination des 2-cycles implique une augmentation du nombre d'états (au pire cas par un facteur 2) qui est compensée par une réduction sensible du gap d'intégrité et du nombre de problèmes résiduels à résoudre dans le processus d'énumération. Enfin, deux algorithmes de type différent ont été développés pour traiter d'autres extensions du problème SPTW. D'une part, Ioachim *et al.* (1998) traitent le problème SPTW avec un objectif incluant à la fois les variables de flot X_{ij}^k sur les arcs et les variables de ressource T_i^{kr} sur les nœuds. D'autre part, Dumas, Desrosiers et Soumis (1991) proposent également un algorithme de résolution pseudo-polynomial pour résoudre un cas particulier du problème SPTW avec cueillette et livraison.

Finalement, Vovor (1997) étudie une variante du problème SPR pour réseaux acycliques et valeurs de ressource discrètes. Des bornes inférieures et supérieures sont associées aux arcs et des fonctions de mise à jour, pas nécessairement non décroissantes, sont associées aux nœuds. Vovor (1997) montre que cette formulation est plus générale que la formulation de Desrosiers *et al.* (1995) lorsque cette dernière est limitée au cas de réseaux acycliques. Vovor (1997) propose entre autres une approche en deux phases, où il construit d'abord un réseau discrétisé ne contenant que les états pouvant faire partie de chemins réalisables, et calcule ensuite le plus court chemin sur ce nouveau réseau sans égard aux valeurs des ressources associées aux états. La deuxième phase de l'algorithme, ne nécessitant plus l'extension des valeurs des ressources, est particulièrement bien adaptée pour la réoptimisation après des modifications aux coûts des arcs, comme dans les applications de génération de colonnes. L'analyse de la complexité de l'algorithme en deux phases permet à Vovor (1997) de trouver pour le cas acyclique une borne de complexité inférieure à celle proposée par Desrochers et Soumis (1988a) pour le cas général.

Le tableau 2.1 résume les développements sur les extensions du problème SPTW et les présente dans l'ordre dans lequel les travaux de recherche ont été effectués, afin de mieux mettre en évidence l'influence des premiers travaux sur les suivants.

Processus de séparation : Dans le contexte général de l'optimisation d'un programme mathématique non convexe, on peut définir un processus de résolution en imposant la division du domaine du problème en sous-ensembles de plus en plus petits, jusqu'au point où la structure de ces sous-ensembles permette l'application d'un algorithme efficace. Un tel processus aveugle est cependant presque toujours inutilisable étant donné le nombre gigantesque de sous-ensembles à diviser et à analyser. Les processus de résolution par énumération incorporent donc en pratique des procédures supplémentaires pour mieux séparer le domaine et pour éliminer le plus tôt possible les problèmes résiduels ne contenant pas de solutions optimales. Parmi les stratégies d'accélération les plus souvent utilisées, on retrouve le calcul de bornes, la détection rapide de problèmes résiduels non réalisables, la construction heuristique de solutions admissibles, la génération de coupes et la partition (par opposition à la simple division) du domaine du problème.

En programmation linéaire en nombres entiers, dont la formulation décomposée du modèle unifié est un cas particulier, un nombre important de travaux (*voir* Nemhauser et Wolsey, 1988) ont permis de développer des algorithmes de séparation efficaces qui exploitent à fond les propriétés du problème à résoudre. Entre autres, pour les problèmes de minimisation, une borne inférieure intéressante peut être calculée en résolvant la relaxation linéaire du problème en nombres entiers, ce qui procure du même coup de l'information sur la faisabilité du problème relaxé et une solution approchée.

Dans le contexte du modèle unifié, la résolution du problème-maître (1.16)–(1.21) par les méthodes décrites précédemment permet d'obtenir une borne inférieure $\underline{z} \leq z_{\text{LP}}$

Tableau 2.1 – Chronologie des développements sur l'évolution du problème SPTW

Recherche	Parution	Auteurs	Description
1981	1983	Desrosiers, Pelletier et Soumis	algorithme pseudo-polynomial pour résoudre le problème SPTW avec $t_{ij} > 0$, de type *label-correcting* et de complexité $O(\tau^3)$
1984	1991	Dumas, Desrosiers et Soumis	algorithme pseudo-polynomial pour résoudre un cas particulier du problème SPTW avec cueillette et livraison
1984	1987	Kolen, Rinnooy Kan et Trienekens	algorithme pseudo-polynomial pour résoudre le problème SPTW avec contraintes d'élimination des 2-cycles
1986	1988a	Desrochers et Soumis	algorithme pseudo-polynomial pour résoudre le problème SPTW, de type *label-setting* et de complexité $O(\tau^2)$
1986	1988b	Desrochers et Soumis	algorithme primal-dual de réoptimisation du problème SPTW après modification des coûts sur les arcs
1986	thèse	Desrochers	algorithme pseudo-polynomial pour résoudre le problème SPTW avec plusieurs ressources (SPR), de type *label-setting* et de complexité $O(\tau^2)$
1994	1998	Ioachim *et al.*	algorithme pseudo-polynomial pour résoudre le problème SPTW dont l'objectif comporte aussi des contributions quelconques des variables de ressource
1996	—	Lingaya	extension de l'algorithme de Desrochers et Soumis (1988a) pour traiter le problème SPTW avec des valeurs de t_{ij} quelconques (sujet aux conditions d'existence d'une solution données à la section 2.1)
1996	1999	Jaumard *et al.*	algorithme pseudo-polynomial pour résoudre une variante acyclique du problème SPR par discrétisation « minimale » de l'espace des ressources

sur la valeur z_{IP} d'une solution admissible du problème résiduel courant. Cette borne inférieure peut être utilisée pour couper des parties non prometteuses de l'arbre d'énumération. Toutes les méthodes de résolution du problème-maître retournent en même temps que $\underline{z}$ une solution primale approchée (pas nécessairement entière) en variables θ_p^k et Y_s ou permettent d'accumuler suffisamment d'information pour en calculer une après coup. Cette solution primale de la formulation décomposée et sa transposition dans la formulation originale au moyen des équations (1.12)–(1.15) peuvent alors servir à concevoir des critères de séparation efficaces. Les prochains paragraphes présentent les principales approches décrites dans la littérature pour réintégrer les contraintes d'intégrité (1.23) sous une forme compatible avec la définition du problème-maître et des sous-problèmes. Les contraintes (1.25) n'ont pas encore été étudiées sous l'angle du processus de séparation car, pour toutes les applications décrites dans la littérature, elles sont redondantes par rapport aux contraintes (1.23) (*voir* la discussion de la section 1.2.2).

Les seules variables de la formulation originale devant obligatoirement être entières sont les variables de flot X_{ij}^k (1.8), avec chacune desquelles on peut créer deux nouveaux problèmes résiduels en imposant qu'une variable X_{ij}^k donnée soit égale à 1 dans une branche et égale à 0 dans l'autre branche. Desaulniers *et al.* (1998) notent que toute somme de ces variables doit aussi être entière et dérivent des critères agrégés sur des sous-ensembles de nœuds et de commodités, permettant de mieux équilibrer l'impact des décisions entre les deux problèmes résiduels. Desrosiers, Soumis et Desrochers (1984) utilisaient déjà ce type de branchement compatible avec la génération de colonnes dans le contexte de la résolution du problème m-TSPTW. Par ailleurs, lors de la modélisation d'applications au moyen du modèle unifié, il est souvent possible de déduire que certaines variables supplémentaires Y_s ou certaines variables de ressource T_i^{kr} prennent nécessairement des valeurs entières dans des solutions admissibles entières. Dans ces conditions, on peut séparer un problème sur les bornes

des variables Y_s (1.4) du problème-maître, comme par exemple lorsque ces variables cumulent un nombre de commodités, un nombre de véhicules, un nombre d'heures créditées ou toute autre quantité discrète. Au niveau des variables de ressource, Gélinas *et al.* (1995) conçoivent, dans le contexte de la résolution du problème VRPTW, un branchement sur les intervalles des variables de temps (contraintes (1.11)) et réduisent de 66% le nombre de problèmes résiduels à résoudre par rapport à un branchement sur les variables de flot.

La formulation originale se prête aussi à l'utilisation de coupes développées pour résoudre d'autres problèmes, comme le problème VRP (Laporte, 1992) et le problème TSP (Junger *et al.*, 1995). Ces coupes s'appliquent uniquement aux variables de flot et s'expriment comme les contraintes (1.3) sans participation des variables de ressource. Le transfert et l'adaptation de coupes connues vers le modèle unifié, de même que la conception de nouvelles coupes, constituent actuellement des avenues de recherche prometteuses, étant donné l'espérance d'une réduction importante du nombre de problèmes résiduels à résoudre. Ainsi, Kohl (1995) utilise une coupe sur le nombre de véhicules ainsi que les contraintes d'élimination de sous-tours et de peignes provenant du problème TSP, et développe les 2-*paths inequalities* inspirées du problème VRP. Dans le contexte de problèmes MDVSP, Jean (1996) développe une méthode de coupes provenant du problème de couplage, basée sur l'identification de cycles impairs. Les résultats obtenus jusqu'à maintenant tendent à favoriser l'utilisation mixte des méthodes de coupes et de branchement, avec des coupes dont les algorithmes d'identification sont simples et dont l'impact négatif sur le temps de résolution du problème-maître et des sous-problèmes est faible.

La formulation décomposée ajoute aux variables de la formulation originale les variables de chemin θ_p^k, sur lesquelles les contraintes d'intégrité s'expriment par le biais des contraintes (1.22)–(1.23). Comme ces variables sont prises en compte implicitement, les décisions les impliquant doivent correspondre à des contraintes imposées

sur les sous-problèmes. Il peut alors exister un problème de compatibilité entre le processus de séparation et la structure des sous-problèmes (*voir* la section 3.3 pour une revue de la littérature spécifique sur ce sujet). Dans le contexte du problème de partitionnement, Ryan et Foster (1981) développent une méthode de branchement fondée sur l'identification d'une paire de contraintes $w_1, w_2 \in W$ telle que la somme des variables couvrant simultanément ces deux contraintes soit fractionnaire. Une telle paire de contraintes existe toujours lorsque la solution du problème de partitionnement est fractionnaire (Barnhart *et al.*, 1998). Les deux problèmes résiduels correspondent à imposer que la solution couvre les deux contraintes avec une variable unique ou deux variables distinctes, respectivement. Desrochers et Soumis (1989) spécialisent cette règle pour la rendre compatible avec les sous-problèmes de type SPR dans un contexte de génération de colonnes. L'ensemble des variables couvrant simultanément les deux contraintes w_1 et w_2 est remplacé par l'ensemble des chemins effectuant la tâche w_2 immédiatement après la tâche w_1. La structure de leur réseau leur permet de créer les problèmes résiduels simplement en éliminant les arcs violant les contraintes sur les séquences de tâches. Haase *et al.* (1998) utilisent les mêmes décisions de branchement mais doivent adapter l'algorithme de Desrochers (1986) pour tenir compte de ces contraintes à cause de la structure plus générale de leurs réseaux. Dumas, Desrosiers et Soumis (1991) présentent une variante du branchement sur les séquences de tâches, portant sur l'ordre des cueillettes, pour résoudre le problème VRPTW avec contraintes de couplage et contraintes de préséance sur les tâches. On note que dans les applications où chaque nœud du graphe (à l'exception des nœuds origine et destination) représente une tâche à effectuer, le branchement sur une séquence de tâches (i, j) correspond à un branchement binaire sur la combinaison de variables de flot $X_{ij} = \sum_{k \in K} X_{ij}^k$.

Dans de nombreuses applications, les contraintes (1.22)–(1.23) sont équivalentes à imposer que les variables de chemin soient binaires (par exemple, lorsque les variables

de ressource ne participent pas aux contraintes (1.18)). Cependant, la méthode de branchement intuitive sur une variable θ_p^k donnée (égale à 1 dans une branche et égale à 0 dans l'autre branche), comporte une difficulté algorithmique cachée. Ainsi, la décision imposant que $\theta_p^k = 1$ suppose de modifier la borne inférieure de cette variable dans le problème-maître, mais celle imposant que $\theta_p^k = 0$ requiert que le chemin $p \in P^k$ associé ne soit plus généré par le sous-problème $k \in K$. Des travaux en cours visent à développer une variante efficace de l'algorithme de Desrochers et Soumis (1988a) pour traiter le problème SPR avec une liste de chemins interdits. L'alternative qui consiste à créer $n + 1$ nœuds de branchement, avec n représentant le nombre de paires de tâches formant le chemin p (Desrosiers, Soumis et Desrochers, 1984), réduit l'efficacité du processus de séparation.

Toutes les méthodes de séparation considérées jusqu'ici peuvent être utilisées avec plus ou moins d'efficacité pour trouver une solution optimale d'un problème m-TSPR donné. Le processus d'énumération peut aussi être arrêté selon d'autres critères que celui de prouver l'optimalité. Il existe par ailleurs d'autres méthodes de séparation qui ne permettent tout simplement pas de prouver l'optimalité d'une solution, mais qui sont efficaces en pratique pour trouver des solutions réalisables. Les méthodes les plus populaires consistent à sélectionner selon divers critères heuristiques une ou plusieurs variables θ_p^k parmi celles déjà générées et à imposer leur utilisation dans la solution en modifiant leur borne inférieure dans le problème-maître. Ces méthodes sont particulièrement importantes dans le contexte de la résolution de problèmes de très grande taille, où l'exploration exhaustive de l'arbre d'énumération n'est pas envisageable.

2.4 Applications basées sur le modèle unifié

L'utilisation du modèle unifié décrit jusqu'à maintenant, comparée à celle de modèles spécifiques, comporte deux inconvénients majeurs : l'analyse des applications est plus

complexe et le temps de développement de méthodes de résolution devient significativement plus long. La pertinence du modèle unifié n'est donc justifiée que s'il existe de nombreuses applications pouvant bénéficier d'une analyse et d'un processus de résolution similaires et si le caractère générique de ce processus de résolution n'entraîne pas une diminution marquée de la performance. Cette section s'attaque à la première de ces deux conditions en présentant une recension limitée mais néanmoins représentative d'applications de recherche et commerciales s'inscrivant directement dans le champ du modèle unifié. Une fois l'existence d'une masse critique d'applications établie, la deuxième condition sur la performance du processus de résolution devient un problème de qualité logicielle auquel le chapitre 4 sur la mise en œuvre de GENCOL tente d'apporter les solutions appropriées.

Comme mentionné au chapitre 1, le modèle unifié est une généralisation du problème de flot multi-commodités, dont l'apparition dans la littérature coïncide avec l'émergence des méthodes de décomposition des programmes linéaires. On peut donc dire que la première application du modèle unifié provient de Ford et Fulkerson (1958) qui proposent une méthode d'énumération implicite, au moyen d'un algorithme de plus court chemin, des colonnes de la formulation arc-chaîne du problème de flot multi-commodités et sont, de ce fait, les précurseurs de la décomposition de Dantzig et Wolfe (1960). Par la suite, Gilmore et Gomory (1961) proposent une méthode de génération de colonnes pour le problème de découpe et posent le sous-problème comme un problème de sac à dos (*knapsack*), cas particulier du problème SPTW. Applegren (1969), quant à lui, utilise directement une formulation avec des sous-problèmes de type SPTW, qu'il résout par discrétisation complète de l'espace des ressources, afin de modéliser un problème de confection d'horaires de cargo avec fenêtres de temps. On pourrait continuer ainsi en énumérant toutes les applications subséquentes se rapportant soit aux variantes du problème de flot multi-commodités, soit aux problèmes résolus par décomposition et dont les sous-problèmes sont des variantes du

problème SPTW. Toutefois, le reste de cette section se limite à une liste thématique des plus récentes applications pouvant être analysées au moyen du modèle unifié et le lecteur intéressé peut se référer aux livres de Lasdon (1970) et Dirickx et Jennergen (1979) pour une analyse plus complète des applications et des méthodes de résolution développées au cours des années 1960–1970.

Transport scolaire :

- Desrosiers, Soumis et Desrochers (1984) : problème d'affectation des autobus aux parcours scolaires modélisé comme un problème m-TSPTW. Ces travaux ont donné lieu à la réalisation du logiciel BUS-OPT distribué par GIRO.

Transport urbain :

- Desrochers et Soumis (1989) : problèmes de confection des horaires de chauffeur d'autobus modélisés comme des problèmes m-TSPR comportant respectivement trois et cinq ressources, pour deux applications différentes. La construction du réseau est analysée plus à fond dans Desrochers *et al.* (1992). L'utilisation de plus d'une ressource est une innovation au niveau de la modélisation et est l'un des principaux résultats de la thèse de doctorat de Desrochers (1986). Ces travaux ont mené à la réalisation du logiciel CREW-OPT intégré au système HASTUS distribué par GIRO.
- Ribeiro et Soumis (1994) : problème d'affectation des autobus aux parcours urbains modélisé comme un problème MDVSP.

Transport de personnes et de marchandises :

- Desrosiers, Dumas et Soumis (1988) : problème de construction des itinéraires de véhicule et des horaires de chauffeur pour effectuer la planification du transport

des personnes handicapées. Il s'agit d'un problème comportant à chaque jour des milliers de requêtes, chacune étant constituée d'un lieu de prise et d'un lieu de dépôt des personnes. Le processus de résolution agrège d'abord l'ensemble des requêtes en *mini-clusters* de façon que les contraintes de couplage et de préséance soient satisfaites *a priori*. Le modèle devient alors un problème m-TSPTW dont les tâches sont les *mini-clusters*. De nombreux développements algorithmiques supplémentaires se retrouvent dans la thèse de doctorat de Dumas (1989), travaux qui ont permis la réalisation du logiciel DARSY distribué par GIRO.

- Ioachim *et al.* (1995) : problème de construction des *mini-clusters* au moyen de l'algorithme de Dumas, Desrosiers et Soumis (1991), afin de résoudre les problèmes TSPTW avec prises et dépôts.
- Desrochers, Desrosiers et Solomon (1992) : expériences numériques sur le problème VRPTW classique résolu par génération de colonnes, où les solutions entières sont obtenues en séparant sur les variables de flot. Gélinas *et al.* (1995) ont par la suite proposé un algorithme de séparation plus efficace sur les intervalles de temps.

Transport aérien :

- Lavoie, Minoux et Odier (1988) : problème de confection des rotations d'équipage pour les flottes moyen-courrier de la compagnie Air France, mis en œuvre au sein du système ICARE en 1984. Le réseau est constitué d'une énumération *a priori* de toutes les journées de travail et de tous les statuts possibles, de façon à n'utiliser aucune ressource. Barnhart *et al.* (1998) utilisent la même approche pour des flottes long-courrier. Desaulniers *et al.* (1997a) décrivent un modèle de type m-TSPR qui est au cœur du système ALTITUDE distribué par AD OPT.
- Desaulniers *et al.* (1994) : problème quotidien de construction d'itinéraires et d'horaires d'avion, modélisé comme un problème m-TSPTW. Un problème hebdomadaire avec synchronisation des horaires de vol sur plusieurs jours est traité par

Ioachim (1994). Il s'agit d'un problème nécessitant des contributions des variables de ressource dans les contraintes liantes et, par conséquent, dans l'objectif des sous-problèmes : ces travaux ont donné lieu au développement d'un nouvel algorithme de plus court chemin (Ioachim *et al.*, 1998).

- Gamache et Soumis (1993) : problème de confection d'horaires mensuels de type *bidline* pour les pilotes en transport aérien, modélisé comme un problème m-TSPR. Ces travaux ont été poursuivis pour la fabrication d'horaires de type *rostering* pour les agents de bord (Gamache *et al.*, 1994) : les problèmes comportent des centaines de sous-problèmes correspondant à un réseau différent pour chaque employé. Enfin, Gamache *et al.* (1998) décrivent un algorithme pour résoudre le problème de type *preferential bidding* pour les horaires avec contrainte de séniorité stricte : ces travaux ont donné lieu à la réalisation du système PBS distribué par AD OPT. Ces développements sont le fruit de la thèse de doctorat de Gamache (1995).
- Desrosiers *et al.* (1999) : cet article décrit les trois composantes du système ALTITUDE résolvant respectivement des problèmes de construction d'itinéraires d'avion, de rotations d'équipage et d'horaires mensuels.

Transport ferroviaire :

- Ziarati *et al.* (1997) : problème d'affectation des locomotives aux trains modélisé comme un problème m-TSPTW. Ces travaux sont à la base du développement d'une des composantes d'un système en transport ferroviaire développé pour AD OPT.
- Cordeau *et al.* (1999) : problème d'assemblage des convois ferroviaires modélisé comme un problème m-TSPTW. Les auteurs gèrent efficacement de grands ensembles de contraintes (1.3) en les relaxant *a priori* en même temps que les contraintes d'intégrité, et en les réintégrant au besoin dans le processus d'énumération. L'application qui résulte de ces travaux constitue une des composantes du système RAIL-WAYS pour le transport de passagers, distribué par AD OPT.

Conclusion

Le modèle unifié a été développé à partir d'applications en transport, d'où la grande concentration d'applications dans ce domaine. Il a par ailleurs été utilisé dans d'autres domaines. Mentionnons le problème de la découpe (Ben Amor, 1997) où les rouleaux sont des véhicules et les items sont des clients à visiter, ainsi qu'une application de contrôle optimal (Laurent *et al.*, 1995) où les courbes de température tracent des itinéraires dans un graphe d'états. Nous rappelons en terminant que la diversité des applications du modèle unifié découle en grande partie de la forme des sous-problèmes, qui permet à la fois de les résoudre « efficacement » et d'atteindre un haut niveau de réalisme lors de la modélisation.

Chapitre 3

Génération de colonnes et programmation en nombres entiers

Certains problèmes de programmation linéaire mixte sont présentés au moyen de deux formulations : une formulation compacte qui dépend explicitement de la structure de sous-problèmes, et une formulation décomposée en programme linéaire généralisé mixte qui résulte de l'énumération d'un sous-ensemble des solutions des sous-problèmes. Les méthodes de résolution proposées dans la littérature passent toutes par un processus de séparation et d'évaluation. D'une part, l'algorithme de séparation analyse les solutions dans le contexte de la formulation compacte, d'autre part, l'algorithme d'évaluation calcule la borne inférieure de chaque problème résiduel (chaque nœud de l'arbre d'énumération) par génération de colonnes dans le contexte de la formulation décomposée.

Or, il arrive que les problèmes de programmation linéaire généralisée mixte soient présentés uniquement au moyen d'une formulation décomposée et d'un oracle. Cette situation mène à des difficultés particulières lors de la résolution, à cause de l'absence d'une formulation compacte explicite sur laquelle analyser les solutions de problèmes relaxés et effectuer la phase de séparation. Dans ce chapitre, on étudie un tel problème P dans le but de recenser, classifier et étendre l'ensemble des processus de résolution applicables dans le cas particulier où seuls le problème décomposé et l'oracle sont donnés.

Après avoir défini le problème P à la section 3.1, on discute à la section 3.2 du calcul

d'une borne inférieure dans le contexte d'un processus de résolution par séparation et évaluation. On s'intéresse ici aux conditions générales sur l'oracle qui permettent d'obtenir cette borne inférieure, et non aux algorithmes de résolution comme tels (relaxation Lagrangienne et algorithmes de sous-gradients, algorithme du simplexe révisé et variantes utilisant des algorithmes de points intérieurs, algorithmes des centres analytiques, ...). Ce chapitre met d'abord en évidence l'impossibilité théorique pour l'oracle de générer l'ensemble de toutes les variables du problème P. L'exemple utilisé, analogue à celui proposé par Ben Amor (1997), confirme l'existence de problèmes de compatibilité entre l'oracle et les algorithmes de séparation. La revue de la littérature à la section 3.3 décrit et classifie les différentes approches utilisées pour résoudre le problème P malgré cette interrelation nécessaire entre les algorithmes de génération de colonnes et les algorithmes de séparation. On note à la fin de cette section que la plupart des difficultés disparaissent lorsqu'on dispose d'une formulation compacte complétant la formulation décomposée du problème P. La contribution de ce chapitre consiste à retrouver, à la section 3.4, une telle formulation compacte sous des conditions relativement faibles (à savoir, que le problème P soit réalisable et borné et qu'on en connaisse une solution admissible). Cette formulation compacte est ensuite utilisée à la section 3.5 pour définir un processus de résolution du problème P fondé sur l'exploitation des caractéristiques de l'oracle vu comme un problème mathématique structuré, et non seulement comme une technique de compression de données. Cette alternative pour la résolution de P ouvre la porte au développement de nouveaux algorithmes de séparation adaptés à la structure de l'oracle, en particulier des méthodes de séparation par coupes qui n'ont pu être intégrées aux processus de résolution jusqu'à maintenant.

3.1 Un problème linéaire P en nombres entiers

Soit le problème de minimisation P suivant, linéaire en nombres entiers :

$$(P) \qquad \min \sum_{j \in J} c_j y_j$$

sujet à :

$$\sum_{j \in J} a_{ij} y_j = b_i \qquad \forall i \in I = \{1, \ldots, m\}$$

$$y_j \in \mathbb{N} \qquad \forall j \in J,$$

où I et J sont des ensembles d'indices finis. On suppose que l'ensemble J inclut l'indice d'une variable qui ne contribue aucunement au problème : il existe un indice $\phi \in J$ pour lequel on a l'égalité vectorielle $(c_\phi, (a_{i\phi})_{i \in I}) = (0, (0)_{i \in I})$. Cette variable servira à la section 3.4 dans le contexte d'une reformulation multi-commodités du problème P. On suppose aussi que toutes les variables d'indice $j \in J$ diffèrent l'une de l'autre par au moins un de leurs coefficients : pour toute paire d'indices $j, k \in J$ où $j \neq k$, l'inégalité vectorielle $(c_j, (a_{ij})_{i \in I}) \neq (c_k, (a_{ik})_{i \in I})$ est satisfaite. On dénote alors par $A = \{(c_j, (a_{ij})_{i \in I})\}_{j \in J}$ l'ensemble de tous les vecteurs de coefficients des variables de P, la dernière condition garantissant l'existence d'une bijection entre J et A.

On s'intéresse à la résolution du problème P dans le cas où $|J|$ est très grand et empêche la formulation explicite de toutes les variables. Les coefficients des variables doivent alors être accédés par le biais d'un *oracle*, dont les principales caractéristiques sont un ensemble X arbitraire couplé à une surjection $\boldsymbol{f} : X \mapsto A$ reliant ce domaine X à l'ensemble A des vecteurs de coefficients du problème P. L'avantage de l'oracle réside dans les hypothèses suivantes :

1. la description de l'ensemble X est beaucoup plus compacte que celle de l'ensemble A, ce qui permet une représentation explicite de X (et implicite de A) ;
2. il existe un algorithme efficace de résolution pour les problèmes de la forme

$$\min_{x \in X} f_0(x) - \sum_{i \in I} \beta_i f_i(x),$$

où $\beta \in \mathbb{R}^m$ est un vecteur de multiplicateurs et $\left(f_0, (f_i)_{i \in I}\right)$ dénote la surjection $\boldsymbol{f}$ sous forme vectorielle.

On restreint aussi l'analyse au cas où la linéarisation des contraintes d'intégrité (par description de l'enveloppe convexe des solutions de P au moyen de contraintes linéaires) mènerait à un trop grand nombre de contraintes linéaires. Cette condition rend le domaine du problème P non convexe de façon irréductible, impliquant un processus de résolution par séparation et évaluation. La phase d'évaluation de ce processus est analysée à la section suivante, alors que la section 3.3 rapporte certaines stratégies de séparation décrites dans la littérature et que la section 3.4 propose une stratégie de séparation complémentaire.

3.2 La phase d'évaluation

Nous dénotons la valeur réelle de la solution du problème de minimisation P par $v(P)$, avec les conventions usuelles que $v(P) = -\infty$ si le problème P est réalisable et non borné et que $v(P) = \infty$ si le problème P est non réalisable.

Une borne inférieure sur $v(P)$ peut être obtenue en résolvant la relaxation linéaire P_R du problème P :

$$(P_R) \qquad \min \sum_{j \in J} c_j y_j$$

sujet à :

$$\sum_{j \in J} a_{ij} y_j = b_i \qquad \forall i \in I \qquad [\beta_i \in \mathbb{R}]$$
$$y_j \geq 0 \qquad \forall j \in J.$$

Nous dénotons par $\bar{c}_j(\beta) = c_j - \sum_{i \in I} \beta_i a_{ij}$ le coût réduit de la variable d'indice $j \in J$ en fonction d'un vecteur de variables duales $\beta \in \mathbb{R}^m$. Puisque l'accès aux coefficients des variables passe par l'appel d'un oracle, le calcul de $v(P_R)$ doit être effectué au moyen d'un *algorithme de génération de colonnes* (ou encore d'un *algorithme de plans coupants* si l'algorithme résout le dual de P_R). Afin d'assurer la convergence du calcul de $v(P_R)$ en un nombre fini d'opérations, ce type d'algorithme suppose que l'oracle choisit une variable de coût réduit minimum, étant donné un vecteur de variables duales $\beta \in \mathbb{R}^m$. Plus formellement, l'oracle est un algorithme qui choisit un vecteur $\boldsymbol{a} \in A$ correspondant à un indice $j \in \arg\min_{j \in J} \bar{c}_j(\beta)$.

La définition de l'oracle ne garantit cependant pas que tous les vecteurs $\boldsymbol{a} \in A$ peuvent être générés. L'exemple qui suit montre en effet que la variable y_2 du problème linéaire Ω_R suivant ne peut être choisie par un oracle défini selon le critère du coût réduit minimum :

$$(\Omega_R) \qquad \min y_1 + \alpha y_2 + y_3$$

sujet à :

$$y_1 + 2y_2 + 3y_3 = 2$$
$$y_1, y_2, y_3 \geq 0$$

avec $\alpha > 1$. Afin que la variable y_2 soit de coût réduit minimum, la variable duale $\beta_1 \in \mathbb{R}$ doit satisfaire les inégalités suivantes comparant $\bar{c}_2(\beta_1)$ avec $\bar{c}_1(\beta_1)$ et $\bar{c}_3(\beta_1)$:

$$\alpha - 2\beta_1 \leq 1 - \beta_1$$
$$\alpha - 2\beta_1 \leq 1 - 3\beta_1,$$

ce qui conduit à $\alpha \leq 1 - |\beta_1|$, en contradiction avec $\alpha > 1$. Cet exemple exploite le fait que le coût réduit $\bar{c}_j(\beta)$ de chaque variable $j \in J$ d'un problème P_R définit un hyperplan dans l'espace dual et que certains de ces hyperplans ne supportent la fonction $\bar{c}(\beta) = \min_{j \in J} \bar{c}_j(\beta)$ (concave et linéaire par morceaux) en aucun point β de l'espace dual. Un autre exemple fourni à l'annexe A illustre selon le même principe que, dans le cas de la décomposition de Dantzig-Wolfe, un algorithme de génération de colonnes n'est pas toujours capable de générer tous les points extrêmes qui décrivent le domaine du sous-problème.

Dans le contexte de la résolution du problème P (en nombres entiers), cette restriction provenant de l'oracle peut mener à une restriction du problème P telle que $v(P) < v(P^*)$, en notant par P^* le problème P restreint aux seules variables dont le vecteur de coefficients peut être généré par l'oracle. Le problème Ω, obtenu en imposant l'intégrité sur les variables du problème Ω_R, en est un exemple : en choisissant α dans l'intervalle ouvert $(1, 2)$, on a comme solution (entière) unique de Ω le vecteur $(y_1, y_2, y_3) = (0, 1, 0)$ et $v(\Omega) = \alpha < 2 = v(\Omega^*)$.

Ben Amor (1997) obtient un résultat analogue au moyen d'un problème de découpe formulé comme un problème de génération de colonnes en nombres entiers (Gilmore et Gomory, 1961), où chaque colonne représente un patron de coupe réalisable. Le problème proposé par Ben Amor, dénoté ici Λ, consiste à minimiser le nombre de rouleaux de taille $L = 15$ nécessaires pour produire une pièce de longueur $l_1 = 5$ et une pièce de longueur $l_2 = 6$. Les patrons extrémaux $(0, 0)$, $(3, 0)$ et $(0, 2)$, de coût 0, 1 et 1 respectivement, forment un triangle qui contient comme point strictement intérieur le patron optimal $(1, 1)$ de coût 1. Les variables duales associées aux contraintes de demande des pièces de longueur l_1 et l_2 sont $\beta_1 \geq 0$ et $\beta_2 \geq 0$ respectivement. Afin de montrer que le patron optimal $(1, 1)$ ne peut être généré par l'oracle, nous comparons les coûts réduits des variables correspondant aux patrons comme dans l'exemple du problème Ω. Les variables duales β_1 et β_2 doivent être nulles

pour que le coût réduit du patron optimal $(1,1)$ soit minimal par rapport aux coûts réduits des patrons extrémaux non vides $(3,0)$ et $(0,2)$. Ces valeurs des variables duales impliquent que le coût réduit du patron optimal $(1,1)$ vaut 1 alors que le coût réduit du patron vide $(0,0)$ vaut 0. Ceci prouve que le patron optimal $(1,1)$ ne peut être généré au moyen d'aucunes valeurs des variables duales associées aux contraintes de demande. La valeur $v(\Lambda)$ de la solution optimale du problème Λ vaut donc 1 alors que le problème Λ^*, restreint aux seules variables pouvant être générées par l'oracle, est non réalisable.

En conclusion, l'oracle ne permet pas toujours de générer tous les vecteurs de coefficients des variables du problème P_R, et donc du problème P. Le sous-ensemble de variables générées peut exclure la solution optimale (entière) de P comme le montre l'exemple du problème Ω, et le problème en nombres entiers restreint à ce sous-ensemble de variables peut même devenir non réalisable comme le montre l'exemple du problème Λ de Ben Amor (1997).

Ainsi, en général, la génération de colonnes ne doit pas être appliquée uniquement sur la relaxation linéaire du problème P. mais aussi sur tous les *problèmes résiduels* issus de la phase de séparation du processus de résolution. On obtient ainsi une confirmation de l'interrelation nécessaire entre les algorithmes de génération de colonnes et les algorithmes de séparation. La prochaine section décrit les approches rapportées dans la littérature permettant de tenir compte de cette interrelation et des problèmes de compatibilité qui en découlent.

3.3 La phase de séparation : revue de la littérature

Dans un processus de séparation et d'évaluation, la phase de séparation est responsable de l'énumération de toutes les solutions du problème par l'ajout progressif de

contraintes. Lorsque l'efficacité de la phase d'évaluation est fondée sur l'exploitation d'une certaine structure, on tente de choisir parmi toutes les contraintes possibles celles qui préservent cette structure. Dans le cas du problème P, le calcul des bornes inférieures utilise la structure de programme linéaire du problème P_R et on cherche donc à séparer une solution fractionnaire de P_R de l'ensemble des solutions de P au moyen de nouvelles contraintes linéaires. Cependant, la conclusion de la section précédente ajoute à ce critère sur le choix des contraintes celui de préserver aussi la structure de l'oracle, puisque ce dernier doit être utilisé lors du calcul des bornes inférieures des problèmes résiduels issus de la phase de séparation.

La caractérisation de l'oracle par le biais d'un ensemble arbitraire X et d'une surjection f ne fournit pas suffisamment d'information pour discuter de sa compatibilité avec les algorithmes de séparation. L'analyse de ces algorithmes passe ainsi par une classification des oracles selon certaines propriétés qui déterminent des ensembles d'algorithmes de séparation compatibles avec la génération de colonnes.

Lorsque seuls le problème P et l'oracle sont donnés, deux classes principales d'oracles sont décrites dans la littérature : les oracles ayant la propriété de retourner des solutions k-optimales (où k est un entier donné) et les oracles permettant de résoudre des problèmes linéaires en nombres entiers. On dresse au début de cette section un portrait critique des processus de séparation fondés sur ces deux classes d'oracles. Par la suite, on analyse le processus naturel de résolution du problème P lorsque celui-ci découle de la décomposition d'une formulation dite compacte. La motivation des développements des sections suivantes provient de la comparaison entre les approches de résolution pour le problème P seul et pour le problème P découlant d'une telle formulation.

L'utilisation d'oracles retournant des solutions k-optimales est proposée par Hansen *et al.* (1991). L'avantage majeur de ce type d'oracles est qu'il est compatible avec

n'importe quel critère de restriction de l'ensemble J des variables du problème P. Les auteurs démontrent la validité de leur processus de résolution au moyen d'un branchement sur les bornes d'une variable y_j fractionnaire, menant à l'exclusion de cette variable lorsqu'elle devient bornée supérieurement. D'autres critères de séparation sont aussi compatibles avec ce type d'oracles, dont l'ajout (implicite) d'une contrainte linéaire au problème P_R de la forme $\sum_{j \in J_0} y_j = 0$, où J_0 est un ensemble de variables exclues. Le principe d'interaction entre l'algorithme de séparation et l'oracle est le suivant. Étant donné une solution fractionnaire du problème P_R, l'algorithme de séparation modifie les bornes inférieures et supérieures de certaines variables, et identifie pour chaque problème résiduel un ensemble J_0 de variables à exclure. Lors du calcul de la borne inférieure d'un problème résiduel, l'oracle calcule itérativement la solution optimale, puis la seconde solution optimale, ..., du problème de trouver une variable de coût réduit minimum en fonction d'un vecteur de variables duales donné jusqu'à ce qu'une solution soit dans l'ensemble $J \setminus J_0$. L'inconvénient principal réside dans l'accroissement de la complexité de l'oracle, peut-être marginal mais toujours présent, que suppose le calcul de la solution k-optimale étant donné les solutions $(k-1)$-optimales déjà trouvées. De plus, cet accroissement est proportionnel à la taille de l'union de tous les ensembles J_0 définissant le problème résiduel courant, ce qui peut être beaucoup plus grand que la profondeur de ce problème dans l'arbre d'énumération. Ainsi, un algorithme de séparation qui exclurait la moitié des variables risquerait d'entraîner une diminution marquée de l'efficacité de l'oracle.

Vanderbeck (1994) utilise une décomposition pour programmes linéaires en nombres entiers par discrétisation du domaine de l'oracle (Nemhauser et Wolsey, 1988), qui est aussi formulé comme un programme linéaire en nombres entiers. Afin de définir des règles de branchement s'appliquant à la formulation décomposée en variables entières ainsi obtenue, il s'inspire de la règle de branchement de Ryan et Foster (1981), développée dans le contexte du problème de partitionnement. Cette règle

consiste à décider si deux contraintes doivent être satisfaites par une variable unique ou par deux variables distinctes. La justification intuitive d'une telle règle est double : d'une part, elle tend à équilibrer la profondeur de l'arbre d'énumération, d'autre part, elle favorise l'émergence de solutions entières en induisant une structure de matrice balancée dans les problèmes résiduels issus de la séparation (Ryan et Foster, 1981). Vanderbeck (1994) puis Vanderbeck et Wolsey (1996) généralisent cette règle pour résoudre des problèmes de génération de colonnes dont les coefficients sont des entiers positifs quelconques. Leur règle de branchement consiste à construire un vecteur de coefficients servant de borne inférieure (vectorielle) pour identifier un sous-ensemble de variables dont la somme est fractionnaire. Ils montrent qu'un tel vecteur existe toujours lorsque la solution de la relaxation linéaire P_R est fractionnaire et ajoutent au problème P_R des contraintes de borne inférieure ou supérieure sur la somme des variables identifiées. La structure de l'oracle doit être modifiée pour tenir compte des nouvelles variables duales correspondant à ces contraintes. Dans le cas le plus simple, seules des contraintes linéaires sont ajoutées au problème de l'oracle. Dans les cas plus complexes, les modifications impliquent l'ajout de nouvelles contraintes linéaires dans le problème P_R ainsi que l'ajout de nouvelles variables binaires et de nouvelles contraintes linéaires dans le problème de l'oracle. L'inconvénient principal réside de nouveau dans l'accroissement de la complexité de l'oracle, qui est proportionnel à la profondeur du problème résiduel dans l'arbre d'énumération.

D'autres tentatives pour définir un processus de séparation compatible avec la génération de colonnes ont été effectuées pour des oracles particuliers. Pour un problème de fabrication d'horaires de chauffeurs d'autobus, Desrochers et Soumis (1989) utilisent une spécialisation de la règle de branchement de Ryan et Foster (1981) consistant à décider si une tâche doit être suivie immédiatement ou non par une autre tâche dans les chemins produits par l'oracle, défini comme un problème de plus court chemin avec contraintes de ressource. Ils imposent les décisions au niveau du réseau en

éliminant les arcs incompatibles avec les contraintes de séquences de tâches. L'impact positif de ces contraintes sur l'oracle découle de la structure particulière de leur réseau, où chaque extrémité d'un arc peut être identifiée à une tâche unique ou au dépôt. Pour un problème de fabrication simultanée de tournées d'autobus et d'horaires de chauffeurs, Haase *et al.* (1998) utilisent aussi des contraintes de séquences de tâches avec un oracle similaire, mais la structure de leur réseau implique qu'ils doivent traiter ces décisions en modifiant l'algorithme de Desrochers (1986) pour conserver plusieurs listes d'étiquettes en chaque nœud. Ces listes regroupent les étiquettes selon la dernière tâche couverte par les chemins correspondants et la dominance entre les étiquettes doit être effectuée liste par liste. Dans ce cas, la complexité de l'oracle croît en fonction de la profondeur du problème résiduel dans l'arbre d'énumération. Pour un problème de fabrication d'horaires de personnel soignant, Vovor (1997) utilise une règle de branchement imposant ou interdisant l'affectation d'une tâche (quart de travail en un jour donné) à une infirmière. Ces décisions correspondent à imposer ou à interdire le passage à certains nœuds des réseaux, qui servent à énumérer implicitement les horaires des infirmières. Les décisions n'impliquent pas de modifier l'algorithme utilisé pour résoudre l'oracle, qui est une variante du problème de plus court chemin avec contraintes de ressource. Elles peuvent même avoir un impact positif sur le temps de résolution (dans le cas des interdictions).

La discussion précédente met en évidence l'aspect contraignant de l'oracle sur le processus de séparation, à moins que les décisions exploitent la structure de l'oracle. Notons que lorsque le problème P_R provient d'un processus de décomposition (par exemple, de Dantzig-Wolfe (1960) ou de Benders (1962)) appliqué à une formulation compacte en nombres entiers, la phase de séparation est naturellement définie sur la relaxation de cette formulation compacte, et non sur le problème P_R lui-même. Ce dernier sert alors uniquement à calculer la valeur d'une borne inférieure et à permettre l'identification d'une solution dans la formulation compacte relaxée. C'est cette solution, exprimée en termes des variables de la formulation compacte, qui est par la suite

utilisée lors de la phase de séparation pour créer les problèmes résiduels par branchements ou coupes. L'avantage de ce processus de résolution sur deux niveaux réside dans le fait que, pour la plupart des applications de ce type, toute contrainte linéaire ajoutée à la formulation compacte est alors compatible avec la génération de colonnes, puisqu'une contrainte ne pouvant être prise en compte directement par l'oracle est alors ajoutée aux contraintes liantes du *problème-maître* et est reliée à l'oracle par l'ajout d'une composante supplémentaire à la surjection f. Cette approche de résolution est utilisée avec succès dans les applications de fabrication d'horaires de véhicule et d'horaires de personnel, notamment par Desrosiers *et al.* (1984) sur le problème m-TSPTW et par Desrochers *et al.* (1992) sur le problème VRPTW, ainsi que par Ribeiro et Soumis (1994) sur un problème de flot multi-commodités (MCFP). Toutefois, il existe aussi des applications pour lesquelles l'ajout d'une composante à la surjection f modifie significativement la structure de l'oracle (*voir* Ioachim, 1994), imposant ici encore des limites sur le choix des contraintes linéaires comme dans le cas des autres approches décrites précédemment.

En résumé, il existe des classes d'oracles qui sont compatibles avec certains algorithmes de séparation et qui permettent de compléter le processus de résolution du problème P. Il est aussi possible de considérer la séparation du domaine de l'oracle lui-même, comme dans le cas d'un problème P découlant d'une formulation compacte par l'application d'un processus de décomposition. Ces approches étant complémentaires, on analyse à la section suivante les conditions permettant de retrouver une formulation compacte quand seuls le problème P et son oracle sont donnés. La section 3.5 présente ensuite un processus de résolution fondé sur l'existence de cette formulation compacte et le compare aux processus de résolution décrits dans la présente section.

3.4 Une reformulation multi-commodités Q de P

L'obtention d'une formulation dite compacte, dont le problème P découle par l'application d'un processus de décomposition, permet de tenir compte dans une même formulation des contraintes linéaires $i \in I$ et du domaine X de l'oracle. Ceci ouvre de nouvelles possibilités pour le choix de contraintes (branchements ou coupes) lors de la phase de séparation du processus de résolution.

Dans cette section et la suivante, on s'intéresse à la classe d'oracles dont le domaine X est un sous-ensemble fermé et borné de $\mathbb{R}^n$, pour une valeur de $n \in \mathbb{N}$ donnée. Il existe donc un hyperrectangle (ou *boîte*) $B \subset \mathbb{R}^n$ tel que $X \subset B$. On limite également l'analyse au cas, courant en pratique, où le problème P est réalisable et borné et pour lequel on connaît une solution $\boldsymbol{y}^*$ dont on pose la somme $\sum_{j \in J} y_j^* = \kappa \in \mathbb{N}$ (en fait, $\sum_{j \in J \setminus \{\phi\}} y_j^* \leq \kappa$ et $y_\phi = \kappa - \sum_{j \in J \setminus \{\phi\}} y_j^*$). On propose comme principe général de séparation la réduction de la boîte B, incluant le domaine de l'oracle, selon chaque dimension jusqu'à la détection d'un problème résiduel non réalisable ou d'une solution (entière) pour le problème P.

La difficulté de cette approche réside dans l'identification des nouvelles contraintes de bornes sur le domaine X à partir de la solution de la phase d'évaluation exprimée sur un sous-ensemble des variables du problème P_R. La démarche présentée dans cette section consiste à formuler un problème multi-commodités Q équivalent au problème P. La section suivante porte sur le processus de résolution par séparation et évaluation pour le problème Q, où la phase d'évaluation est similaire à celle décrite pour le problème P à la section 3.1 et où la phase de séparation procède par restrictions de boîtes sur le domaine de chaque commodité du problème Q.

La particularité principale du problème Q par rapport au problème P est que le

problème Q est structuré explicitement selon les caractéristiques de l'oracle :

$$(Q) \qquad \min \sum_{k \in K} f_0(x^k)$$

sujet à :

$$\sum_{k \in K} f_i(x^k) = b_i \qquad \forall i \in I$$
$$x^k \in X^k \qquad \forall k \in K,$$

où $K = \{1, \ldots, \kappa\}$ est l'ensemble des commodités, une commodité $k \in K$ ayant pour domaine le sous-ensemble $X^k \subseteq X$. Comme un ensemble X^k, $k \in K$, peut être une restriction du domaine X original de l'oracle, on lui associe par f un sous-ensemble $J^k \subseteq J$ d'indices correspondants. Le lien entre les problèmes Q et P s'établit par l'intermédiaire du problème P' suivant, dont les variables y_j^k proviennent d'une répartition des variables y_j du problème P en κ parties :

$$(P') \qquad \min \sum_{k \in K} \sum_{j \in J^k} c_j y_j^k$$

sujet à :

$$\sum_{k \in K} \sum_{j \in J^k} a_{ij} y_j^k = b_i \qquad \forall i \in I$$
$$\sum_{j \in J^k} y_j^k = 1 \qquad \forall k \in K$$
$$y_j^k \in \{0, 1\} \qquad \forall k \in K, \quad \forall j \in J^k.$$

Tout d'abord, la proposition 3.1 utilise une extension du principe de décomposition de Dantzig-Wolfe pour transformer le problème Q en problème P'. Ensuite, la proposition 3.2 montre que, dans le cas où toutes les commodités $k \in K$ ont un domaine X^k identique au domaine X de l'oracle, les problèmes P' et P sont équivalents.

Proposition 3.1. *Les problèmes Q et P' sont équivalents.*

Démonstration. Puisque $\boldsymbol{f}$ est une surjection de X sur A, que $|A| = |J|$ est fini et que les ensembles X^k sont des sous-ensembles de X, on peut diviser ces ensembles X^k, $k \in K$, en un nombre fini de classes d'équivalence où deux points x_1^k et x_2^k sont équivalents si et seulement si l'égalité vectorielle $\boldsymbol{f}(x_1^k) = \boldsymbol{f}(x_2^k)$ est satisfaite. On peut ainsi reformuler Q au moyen de sous-ensembles finis $\bar{X}^k \subseteq X^k$, $k \in K$, formés d'un seul représentant de X^k par classe d'équivalence. On dénote par $x_j^k \in \bar{X}^k$ un représentant dans X^k de la classe d'équivalence $j \in J^k \subseteq J$. Pour chaque indice $k \in K$ dans le changement de variables suivant sur x^k, on utilise une variable binaire y_j^k pour choisir un seul de ces $|J^k|$ représentants :

$$\begin{aligned}
\sum_{j \in J^k} y_j^k x_j^k &= x^k && \forall k \in K \\
\sum_{j \in J^k} y_j^k &= 1 && \forall k \in K \\
y_j^k &\in \{0,1\} && \forall k \in K, \quad \forall j \in J^k.
\end{aligned}$$

Étant donné $k \in K$, une seule des variables y_j^k peut prendre la valeur 1, ce qui correspond à choisir un point $x_j^k \in \bar{X}^k$ unique pour cette valeur de k. On effectue ce changement de variables dans le problème Q en utilisant dans l'objectif et les contraintes cette propriété de combinaison linéaire binaire :

$$\min \sum_{k \in K} \sum_{j \in J^k} y_j^k f_0(x_j^k)$$

sujet à :

$$\begin{aligned}
\sum_{k \in K} \sum_{j \in J^k} y_j^k f_i(x_j^k) &= b_i && \forall i \in I \\
\sum_{j \in J^k} y_j^k &= 1 && \forall k \in K \\
y_j^k &\in \{0,1\} && \forall k \in K, \quad \forall j \in J^k \\
\sum_{j \in J^k} y_j^k x_j^k &= x^k \in X^k && \forall k \in K.
\end{aligned}$$

Dans ce problème en nombres entiers, on remarque que les contraintes $\sum_{j\in J} y_j^k x_j^k = x^k \in X^k$ ne sont pas nécessaires puisque le changement de variables utilisé (combinaison linéaire binaire) choisit toujours un point $x_j^k \in X^k$ et ne permet pas les combinaisons entre les points de X^k. Avec $f_0(x_j^k) = c_j$ et $f_i(x_j^k) = a_{ij}$, la formulation précédente devient identique à celle du problème P'. □

Proposition 3.2. *Lorsque les κ commodités du problème Q ont un domaine X^k identique au domaine X de l'oracle, les problèmes P' et P sont équivalents.*

Démonstration. En utilisant le fait que toutes les commodités sont identiques et en sommant sur $k \in K$ les équations $\sum_{j\in J} y_j^k = 1$, on peut mettre en évidence les sommes $\sum_{k\in K} y_j^k$ dans la formulation du problème P' :

$$\min \sum_{j\in J}\sum_{k\in K} c_j y_j^k$$

sujet à :

$$\begin{aligned}
\sum_{j\in J}\sum_{k\in K} a_{ij} y_j^k &= b_i && \forall i \in I \\
\sum_{j\in J}\sum_{k\in K} y_j^k &= \kappa \\
y_j^k &\in \{0,1\} && \forall k \in K, \quad \forall j \in J.
\end{aligned}$$

L'introduction des variables $y_j = \sum_{k\in K} y_j^k$, avec $y_j \in \mathbb{N}$, complète la transformation. La contrainte supplémentaire $\sum_{j\in J} y_j = \kappa$ est redondante de par la définition de κ au début de la présente section.

On obtient ainsi que le problème P est une relaxation (à cause de la sommation sur $k \in K$) du problème P'. Mais toute solution du problème P en variables y_j peut être décomposée en κ parties entières (en affectant autant de parties qu'il faut à la variable y_ϕ) qu'on peut affecter aux variables correspondantes du problème P'. Le problème P' est donc une relaxation du problème P, ce qui prouve l'équivalence entre les problèmes P' et P dans le contexte de la proposition. □

3.5 Résolution de P par l'intermédiaire de Q

Dans la foulée de la section 3.4, on élabore un processus de résolution du problème P en définissant un problème Q initial au moyen de κ commodités identiques de domaine $X^k = X$, $k \in K$. Le problème Q est nécessairement, de par les propriétés du problème P énumérées à la section 3.1, un problème non convexe à cause du domaine X ou de la surjection $\boldsymbol{f}$. On utilise donc un processus de résolution par séparation et évaluation sur le problème Q pour trouver une solution du problème P.

Phase d'évaluation : Le calcul d'une borne inférieure sur le problème Q s'effectue en résolvant la relaxation linéaire P'_R du problème P' de façon analogue à la phase d'évaluation présentée à la section 3.1. Cette relaxation linéaire P'_R correspond à la relaxation Q_R du problème Q par convexification de chaque ensemble X^k, $k \in K$. La preuve de l'équivalence entre les problèmes Q_R et P'_R est similaire à celle de la proposition 3.1, en utilisant comme changement de variables des combinaisons linéaires convexes au lieu de combinaisons linéaires binaires. En prévision de la phase de séparation, il est souvent avantageux de conserver, pour chaque vecteur de coefficients d'indice $j \in J^k$ produit par l'oracle, le point $x_j^k \in X^k$ correspondant.

Phase de séparation : Plusieurs cas sont possibles lors de l'analyse de la solution de P'_R issue de la phase d'évaluation. La solution de P'_R peut être entière, *i.e.*, elle peut être une solution de P' et donc de Q et de P. Dans ce cas, cette solution remplace la meilleure solution déjà obtenue si elle est de coût inférieur et le processus de résolution continue avec les problèmes résiduels restants.

Lorsque la solution de P'_R est fractionnaire, on utilise l'équation suivante, extraite de

la preuve de la proposition 3.1,

$$\sum_{j \in J} y_j^k x_j^k = x^k \qquad \forall k \in K,$$

pour retrouver une solution en termes des variables x^k, $k \in K$ du problème Q_R. Plusieurs cas sont encore possibles. S'il existe une commodité $k \in K$ pour laquelle $x^k \notin X^k$, deux problèmes résiduels sont créés en séparant en deux parties (le plus égales possibles) la boîte B^k contrôlant le domaine X^k de cette commodité.

Cependant, l'existence d'une telle commodité n'est pas garantie. En effet, il peut arriver que chaque variable x^k, $k \in K$, satisfasse les contraintes de domaine X^k de la commodité correspondante. Cette situation est entre autres possible lorsque le problème P'_R peut générer par combinaison linéaire des solutions valides sur X^k à partir de solutions déjà produites par l'oracle. Une difficulté survient quand le coût de cette solution « générée » par le problème P'_R ne correspond pas au coût de la même solution évalué dans le problème Q_R. Ceci ne peut se produire que lorsque la fonction f_0 est non linéaire. Ainsi, si $x^k \in X^k$ pour chaque commodité $k \in K$ et $v(P'_R) = v(Q_R)$, la solution en x^k est bel et bien une solution du problème Q qui peut être transformée immédiatement en solution du problème P en appliquant la surjection f sur la solution x^k de chaque commodité $k \in K$. Cette solution est alors traitée comme l'aurait été une solution entière du problème P'_R. Cependant, si $x^k \in X^k$ pour chaque commodité $k \in K$ et $v(P'_R) < v(Q_R)$, le processus de séparation doit créer deux problèmes résiduels en séparant la boîte B^k contrôlant le domaine X^k d'une commodité $k \in K$.

Ce processus de séparation permet, par réductions successives des ensembles X^k, d'énumérer toutes les solutions du problème Q, dont au moins une solution du problème P est l'image par f (conséquences de l'hypothèse d'existence d'une solution de P et de la définition de la constante κ au début de la section 3.4).

Analyse comparative : Le but de l'approche décrite dans cette section pour résoudre le problème P est de mieux exploiter la structure de l'oracle, comme cela est possible lorsque le problème P découle de la décomposition d'une formulation dite compacte. Pour y arriver, on utilise la reformulation multi-commodités Q du problème P développée à la section 3.4. Le processus de résolution est donc limité par les conditions d'existence de ce problème Q équivalent, soit la connaissance *a priori* d'une borne supérieure sur la somme des variables d'une solution du problème P, ce qui implique que le problème P soit réalisable en premier lieu. Ces conditions sont satisfaites dans la majorité des applications parce que les modèles tendent à inclure des variables d'écart tenant compte du coût véritable ($< \infty$) de ne pas satisfaire certaines contraintes.

Un des avantages que comporte le processus de séparation provient du fait que plusieurs oracles deviennent plus faciles à résoudre lorsque leur domaine est restreint. Ceci inclut entre autres les algorithmes de programmation dynamique et les algorithmes comportant une phase d'énumération, dont les programmes linéaires en nombres entiers. Par contre, on peut prévoir que des oracles utilisant une formulation duale puissent être pénalisés par une expansion correspondante de leur domaine de recherche.

Par contre, ce gain d'efficacité au niveau de l'oracle est apparamment compensé, lors de la phase d'évaluation, par une formulation P'_R comportant à la fois plus de contraintes (l'ajout de κ contraintes de convexité sur les variables y_j^k) et beaucoup plus de variables (la multiplication du nombre de variables par κ) que la formulation P_R. De plus, l'estimation de κ provenant d'une solution admissible du problème P peut être très mauvaise et amener un surcroît de travail inutile lors de la résolution du problème P'_R. Pis encore, les commodités sont à l'origine toutes identiques, ce qui nuit au processus de séparation à cause d'une trop grande symétrie du problème P'_R.

Pour toutes ces raisons, il y a un intérêt évident à mieux exploiter la structure du

problème P'_R lors de la mise en œuvre de ce processus de résolution. Une stratégie naturelle consiste à agréger les commodités identiques et à éliminer les contraintes de convexité redondantes, menant pour le problème initial à la résolution d'un problème P'_R identique au problème P_R présenté à la section 3.1. Les commodités sont par la suite différenciées lors de la création des problèmes résiduels seulement lorsqu'il devient nécessaire lors de la phase de séparation de réduire différemment les boîtes B^k de commodités d'un même groupe.

On note finalement que les stratégies de séparation ne sont pas limitées aux seules bissections des boîtes B^k contrôlant le domaine des commodités $k \in K$. Les bissections sont utilisées dans cette section à cause de leur caractère général, faisant appel à des propriétés minimales de l'oracle. L'avantage principal de l'approche de résolution du problème P par le biais d'une formulation compacte Q est qu'elle permet d'exploiter trivialement la structure de l'oracle vue comme composante principale du problème à résoudre, et non simplement comme une forme de compression des données du problème P. Par exemple, la structure de problème de réseau multi-commodités a permis à Kohl *et al.* (1999) de développer des coupes au niveau des contraintes liantes du problème VRPTW. Gamache *et al.* (1998) ont quant à eux exploité la structure de problème multi-commodités à l'origine de leur problème de génération de colonnes pour exprimer certaines coupes au niveau du domaine de l'oracle.

Conclusion

Ce chapitre porte sur l'étude des processus de résolution applicables au problème P défini à la section 3.1, non dans une recherche d'efficacité locale de la phase d'évaluation ou de la phase de séparation, mais dans une perspective de faisabilité du processus global de séparation et d'évaluation. Nous avons mis en évidence à la fin

de la section 3.2 une limite de l'oracle comme mécanisme de génération des variables du problème P. Cette limite, également relevée par Ben Amor (1997), n'a pas d'impact sur la valeur obtenue pour la borne inférieure, mais peut affecter la résolution du problème P en nombres entiers. Ce résultat théorique n'affecte pas la validité des processus de résolution décrits dans la littérature, ceux-ci utilisant déjà la génération de colonnes sur les problèmes résiduels pour des raisons empiriques.

La revue de la littérature à la section 3.3 compare les processus de résolution utilisés par différents auteurs selon les propriétés de l'oracle. La discussion met en relief comment, d'une part, l'oracle est reconnu et exploité comme composante même du problème lorsque le problème P est obtenu par décomposition d'une formulation dite compacte alors que, d'autre part, l'oracle est considéré comme un obstacle à contourner lorsque seuls le problème P et l'oracle sont fournis. La deuxième contribution vient du développement de la section 3.4 permettant de retrouver sous des conditions relativement faibles (un problème P réalisable et borné dont on connaît une solution admissible) un problème multi-commodités Q équivalent au problème P. Le processus de résolution du problème P par le biais du problème Q, décrit à la section 3.5, peut alors être considéré comme une généralisation triviale des processus développés dans la littérature pour résoudre les problèmes décomposables en nombres entiers. L'importance de la contribution se situe dans le changement d'attitude par rapport à l'oracle, ouvrant de nouvelles opportunités pour le développement d'algorithmes de séparation exploitant mieux la structure du problème P, y compris celle de l'oracle.

Les stratégies de résolution du modèle unifié présentées au chapitre 2, fondées sur le calcul de bornes inférieures par génération de colonnes selon la formulation décomposée et sur la séparation du problème selon la formulation compacte, constituent une application des principes mis en évidence dans ce chapitre.

Chapitre 4

Mise en œuvre de GENCOL

Le modèle unifié défini au chapitre 1 est l'aboutissement d'une synthèse effectuée par Desaulniers *et al.* (1998). Il correspond à l'union sélective de plusieurs modèles traitant différents problèmes de tournées de véhicule et d'horaires de personnel (*voir* le chapitre 2). Les avantages d'une telle union au plan théorique proviennent principalement d'une analyse mathématique et algorithmique centralisée, facilitée par la sélection de composants compatibles qui enrichissent le modèle sans trop le complexifier. Au plan pratique, cette situation permet de développer des structures de données et des algorithmes génériques pouvant être amortis sur plusieurs applications. C'est dans cette perspective de gains d'efficacité conjoints aux niveaux mathématique et informatique que le modèle unifié et le logiciel GENCOL ont été développés, chacun étant pour l'autre une source de nouveaux problèmes stimulants.

Ce chapitre expose les points saillants du développement des versions successives du logiciel GENCOL, dont les deux dernières versions majeures, GENCOL-3 et GENCOL-4, constituent les contributions principales de la présente thèse au plan expérimental. Ces versions sont présentées de façon différentielle, mettant en lumière le contexte historique qui a influencé leur développement. Leur description n'explique pas les algorithmes de résolution du modèle unifié (résolution d'un problème de plus court chemin avec contraintes de ressource, résolution d'un problème-maître restreint, gestion d'un processus de séparation et d'évaluation) puisque ceux-ci sont déjà bien documentés dans la littérature (*voir* le chapitre 2). Elle aborde plutôt les particularités de la mise en œuvre de ces algorithmes qui ont permis de franchir l'écart entre

un prototype de recherche comme GENCOL-0 et un logiciel utilisé dans plusieurs applications de pointe comme GENCOL-4.

La section 4.1 rapporte l'historique du logiciel GENCOL depuis la création du premier prototype au début des années 1980 jusqu'au développement de GENCOL-2, qui s'est terminé aux environs de 1992 dans un contexte d'excellentes opportunités commerciales. La section 4.2 couvre le développement de GENCOL-3, principalement influencé par des objectifs d'efficacité et de robustesse en vue de la résolution de problèmes de grande taille provenant de l'industrie du transport aérien. La section 4.3 porte sur la version actuelle du logiciel, GENCOL-4, dont les objectifs visent l'amélioration de la robustesse et l'ajout de mécanismes flexibles permettant de se servir du logiciel comme d'un laboratoire pour des recherches algorithmiques et le traitement des parties les plus difficiles du modèle unifié.

4.1 Historique des versions 0, 1 et 2 de GENCOL

Cet historique résume certaines décisions prises lors du développement des versions antérieures de GENCOL, afin de dégager les contributions décrites dans les sections suivantes. Il permet d'apprécier l'évolution de GENCOL par l'évaluation de la distance séparant le logiciel de l'idéal mathématique (déterminé par les fonctionnalités du modèle unifié) et de l'idéal informatique (déterminé par les critères du génie logiciel).

La mise en œuvre de GENCOL-0 constitue à l'origine la partie expérimentale des travaux de Pelletier, Desrosiers et Soumis (1983) sur le problème m-TSPTW appliqué au transport scolaire. Le développement du logiciel s'étend de 1981 à 1987 et implique principalement Paul Pelletier, Martin Desrochers, Yvan Dumas et Michel Sauvé. L'effort est considérable puisque plusieurs composants constituent des innovations autant mathématiques que logicielles, en particulier l'algorithme de plus court

chemin avec contraintes de ressources et l'intégration d'un processus de génération de colonnes dans un processus de séparation et d'évaluation. Le logiciel est codé en FORTRAN, seul langage de programmation à la fois stable et efficace à l'époque, donnant immédiatement accès à des optimiseurs de programmes linéaires performants (celui de Land et Powell (1973), puis XMP de Marsten (1981)). La structure de GENCOL-0 et de son langage de modélisation imposent plusieurs restrictions par rapport au modèle unifié présenté au tableau 1.1 : les variables supplémentaires Y_s ne sont pas supportées, de même que les contraintes générales de l'ensemble H ; les contraintes W correspondent à un problème de partitionnement assurant la couverture d'un sous-ensemble des nœuds ; les fonctions d'extension sont limitées à la forme $f_{ij}(T_i) = \max(T_i + t_{ij}, l_j)$. Bien entendu, la création de versions dédiées à certaines applications a permis de contourner ces restrictions, au prix des problèmes de maintenance associés. Tout de même, GENCOL-0 constitue un véritable succès en tant que prototype fondé sur un champ de recherche en émergence, comme en témoignent les nombreuses publications y faisant (indirectement) référence dans des contextes variés : problème m-TSPTW (Desrosiers, Soumis et Desrochers, 1984), comparaison avec une méthode de la relaxation Lagrangienne sur le problème m-TSPTW (Desrosiers, Sauvé et Soumis, 1988), problème SPTW avec cueillette et livraison (Dumas, Desrosiers et Soumis, 1991).

En 1988, les dernières recherches sur l'algorithme de résolution du problème SPR (Desrochers, 1986 ; Desrochers et Soumis, 1988a) couplées aux performances croissantes des ordinateurs permettent de résoudre optimalement en quelques heures des problèmes de fabrication d'horaires en transport urbain pour des villes de taille moyenne. La compagnie GIRO, qui traite ces problèmes au moyen du système heuristique HASTUS (Rousseau, Lessard et Blais, 1985), s'intéresse aux développements de GENCOL et fournit des données permettant de comparer les deux systèmes (Desrochers et Soumis, 1989). Les résultats justifient le développement du logiciel CREW-OPT dédié

aux horaires des chauffeurs d'autobus. Au cœur de ce système, on retrouve GENCOL-1, la nouvelle version de l'optimiseur impliquant entre autres les analystes Martin Desrochers, Yvan Dumas, Johanne Gilbert, Margaret Maclure, Brigitte Rioux, Michel Sauvé et Serge Taillefer. Les objectifs de GENCOL-1 sont donc influencés par le contexte d'une utilisation commerciale et visent une certaine stabilité des temps de calcul ainsi que la résolution sous-optimale de problèmes de plus grande taille. Ces objectifs mènent à une spécialisation de la version précédente par l'exploitation de la structure des réseaux, l'incorporation de nouvelles heuristiques et le développement d'une hiérarchie de paramètres de contrôle. Rousseau et Desrosiers (1995) tracent un bilan positif de l'utilisation de GENCOL-1 par GIRO pour traiter les problèmes des villes de Lyon (1990), Toulouse (1992), Vienne (1995) et Tokyo (1995). Entre 1989 et 1990, deux autres systèmes fondés sur GENCOL-1, BUS-OPT et DARSY, sont développés conjointement avec GIRO pour résoudre respectivement des problèmes de fabrication d'horaires de chauffeurs d'autobus scolaires et de confection d'itinéraires pour le transport adapté de personnes à mobilité réduite.

Parallèlement aux opportunités en transport urbain et suite aux travaux de Lavoie, Minoux et Odier (1988) sur le système ICARE d'AIR FRANCE, Desrosiers *et al.* (1991) démontrent la possibilité de modéliser et de résoudre presqu'à l'optimalité des problèmes de fabrication d'horaires d'équipage en transport aérien. La taille et la complexité de ces problèmes nécessitent le développement de GENCOL-2 impliquant Yvan Dumas, Johanne Gilbert, Margaret Maclure, Brigitte Rioux et Serge Taillefer. La principale modification au langage de modélisation consiste à dissocier complètement la notion de tâche de celle de nœud lors de la définition des commodités $k \in K$. Les contraintes W correspondent ainsi à un problème de partitionnement des tâches, celles-ci étant associées explicitement aux nœuds et aux arcs du problème. L'innovation algorithmique majeure de cette version consiste à étendre le domaine de l'algorithme de plus court chemin pour y inclure des contraintes sur des séquences de

tâches, ces séquences provenant de la méthode de séparation du processus d'énumération décrit par Desrochers et Soumis (1989). L'effort de développement est cependant concentré sur l'introduction systématique d'heuristiques paramétrisées afin de contrôler la répartition de la consommation des ressources (temps et mémoire) entre les différents algorithmes tout au long d'un processus de résolution pouvant parfois durer plusieurs jours. Le développement de mécanismes de perturbation du membre de droite du problème-maître, ainsi que l'utilisation de l'optimiseur de programmes linéaires CPLEX (*voir* CPLEX, 1994), contribuent significativement à augmenter la puissance de calcul de GENCOL-2 et permettent de traiter des problèmes comportant quelques centaines de tâches. Le logiciel GENCOL-2 est intégré au système ICARE d'AIR FRANCE en 1991 et constitue la composante d'optimisation d'un système complet de planification développé par la compagnie AD OPT pour AIR TRANSAT en 1993 (Desrosiers *et al.*, 1999). À la même époque, plusieurs bancs d'essais sont réalisés pour le compte d'autres compagnies de transport aérien concernant la fabrication des horaires d'équipage mais aussi l'affectation des avions aux vols et la confection d'horaires personnalisés. En plus de ces activités commerciales, GENCOL-2 est aussi activement utilisé comme base d'expérimentation pour la résolution de variantes du problème m-TSPTW. Desrochers, Desrosiers et Solomon (1992) proposent une formulation m-TSPR pour modéliser le problème VRPTW et utilisent GENCOL-2 pour résoudre optimalement des problèmes de grande taille considérés insolubles jusqu'alors. Gélinas *et al.* (1995) étudient un branchement sur les intervalles des variables de temps (contraintes (1.11)) qui leur permet de réduire significativement le nombre de problèmes résiduels dans le processus d'énumération lorsqu'appliqué aux problèmes VRPTW avec cueillette au retour (VRPTW *with backhauling*). Ioachim *et al.* (1995) développent un algorithme de regroupement des requêtes (*mini-clustering*) dans le contexte de problèmes m-PDPTW de très grande taille modélisant le transport de personnes à mobilité réduite.

Les versions précédentes ont eu un impact important sur l'orientation des développements de la version 3 de GENCOL. Au niveau de la stratégie algorithmique, la notion

d'algorithme optimal glisse vers celle d'algorithme paramétrisé, permettant l'élaboration d'heuristiques contrôlées selon les résultats espérés et la difficulté du problème. Au niveau des interdépendances mathématiques, le couplage entre le branchement par contraintes sur des séquences de tâches et l'algorithme de résolution des sous-problèmes est diminué en enrichissant le domaine de l'algorithme de plus court chemin avec contraintes de ressource. Au niveau des développements informatiques, le soucis de portabilité mène à l'indépendance face à la plateforme de développement et au compilateur utilisés, ainsi qu'à l'encapsulation de l'interface concernant l'optimiseur de programmes linéaires. Finalement, on remarque une oscillation des efforts de développement entre les pôles d'innovation (GENCOL-0 et GENCOL-2) et de robustesse (GENCOL-1) qui sera encore visible dans la suite du cycle de développement.

4.2 Mise en œuvre de GENCOL-3

En 1992, les développements accomplis depuis une décennie ont mis en évidence le potentiel technologique du modèle unifié et du logiciel GENCOL. La résolution de problèmes de grande taille est maintenant possible et un marché important se développe, particulièrement dans le domaine du transport aérien, pour des systèmes d'optimisation produisant des solutions de qualité mesurable. Cependant, le logiciel GENCOL-2 comporte certains défauts structurels freinant la production de telles applications, qui nécessitent plusieurs spécialisations des algorithmes et des heuristiques afin de maintenir les temps de calcul acceptables. Ce handicap incombe en grande partie au langage de programmation, FORTRAN, qui est mal adapté aux manipulations d'ensembles, de listes, d'arbres et de fonctions paramétrisées, mais peut aussi être imputé à l'évolution cumulative du logiciel. Le contexte est donc idéal pour le développement d'une nouvelle version de GENCOL tirant profit de l'évolution des ressources informatiques (entre autres, le langage C possède désormais un standard (ANSI, 1989)

garantissant la portabilité des applications) et d'une certaine distance objective par rapport aux versions précédentes.

Le logiciel GENCOL-3, développé de 1992 à 1995 en collaboration avec François Lacoursière, est ainsi la première contribution expérimentale majeure de cette thèse. Cette version est le résultat d'une analyse algorithmique et informatique critique du modèle unifié. L'objectif principal vise la résolution de problèmes de grande taille en transport aérien (nombre de tâches $|W| \geq 500$, nombre de commodités $|K| \geq 20$, nombre d'arcs $|A^k| \geq 100\,000$, nombre de nœuds $|N^k| \geq 10\,000$, nombre de ressources $|R^k| \geq 6$). La présente section établit tout d'abord les restrictions déterminant le « sous-ensemble » du modèle unifié traité par GENCOL-3, pour ensuite couvrir la structure informatique du logiciel et les aspects influençant son efficacité. Un bilan et une liste d'applications concluent la section.

Les objectifs de GENCOL-3 sont biaisés vers la résolution de problèmes de grande taille en transport aérien et le modèle utilisé pour cette version consiste essentiellement en l'intersection du modèle traité par GENCOL-2 et des caractéristiques nécessaires en vue de l'atteinte de ces objectifs. De la version précédente, on retient l'absence de coefficients quelconques a^k_{wij} et b^{kr}_{hi} lors de la définition des contraintes liantes du modèle. Les contraintes W constituent encore un problème de partitionnement des tâches, qui sont associées aussi bien aux nœuds qu'aux arcs. Contrairement à la version précédente, on limite les réseaux correspondant aux commodités $k \in K$ au cas de graphes acycliques, comme pour l'exposition théorique du modèle unifié. Cette dernière restriction permet d'éviter de coûteux calculs lors de la résolution des problèmes SPR. Afin de tenir compte des particularités des problèmes de transport aérien, certaines extensions sont ajoutées au langage de modélisation, qui détermine les données « naturelles » du logiciel (par opposition aux données fournies par des spécialisations selon l'application). En plus des tâches correspondant aux contraintes W, ce langage supporte partiellement la définition de contraintes quelconques H

en acceptant les informations sur le sens des contraintes et la valeur de leur membre de droit. Le mécanisme pour calculer les coefficients b_{hi}^{kr} à partir des chemins doit cependant provenir d'une spécialisation du logiciel. Ces contraintes H étant identifiées, il est ainsi possible de définir des variables supplémentaires Y_s s'y rattachant et de modéliser, entre autres, des contraintes liantes portant sur la répartition des chemins d'une solution entre différentes commodités.

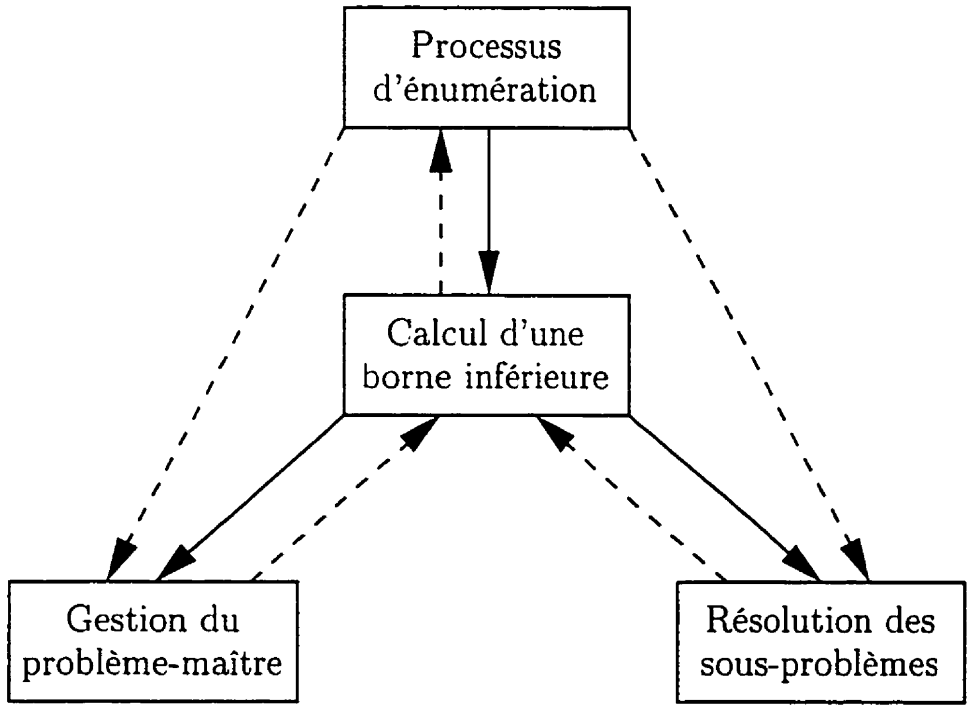


Figure 4.1 – Interactions entre les modules principaux de GENCOL-3.

Un logiciel pour résoudre le modèle unifié par génération de colonnes se sépare naturellement en quatre modules principaux respectivement responsables des aspects suivants : processus d'énumération par séparation et évaluation, calcul d'une borne inférieure, gestion du problème-maître, résolution d'un sous-problème de plus court chemin avec contraintes de ressource. Les relations minimales entre ces modules sont indiquées à la figure 4.1 par des flèches continues, où les modules pointés par une

flèche sont susceptibles d'être utilisés ou modifiés directement par le module à l'origine de la flèche. Bien que les algorithmes sous-jacents soient étudiés isolément du point de vue mathématique, le développement d'heuristiques et de mécanismes de contrôle globaux crée des interdépendances supplémentaires qui sont illustrées à la figure 4.1 par des flèches en pointillé. Un objectif de réutilisation maximale pour chaque module impliquerait normalement la recherche d'un couplage minimum et l'élimination de ces nouveaux liens. Cependant, ces « optimisations » sont justifiées par l'objectif d'efficacité maximale (consommation de mémoire et temps d'exécution) visé par GENCOL-3. Le module de calcul des bornes inférieures soulève peu d'intérêt pour la présente discussion, n'étant essentiellement qu'un gestionnaire de stratégies paramétrisées de résolution. Les trois autres modules sont analysés dans les prochains paragraphes, suivis d'un survol des structures de bas niveau assurant la cohésion de l'ensemble du logiciel.

Processus d'énumération : La caractéristique déterminante du processus d'énumération développé dans GENCOL-3 est que les *problèmes résiduels*, obtenus après séparation, sont représentés au moyen des différences par rapport à leur *problème père*. Ces différences peuvent impliquer l'ajout de contraintes sur des séquences de tâches, des modifications aux bornes des variables supplémentaires, ainsi que le retrait de certaines variables de chemin et les modifications aux contraintes touchées par ces variables (*voir* Lacoursière (1992) pour une description de la méthode d'évaluation SFOG menant à ces dernières modifications). L'alternative consistant à créer des problèmes résiduels indépendants ne peut être envisagée à cause de l'espace mémoire considérable (souvent plus de 20 Moctets) occupé par les données d'un seul problème.

La structure hiérarchique des problèmes résiduels implique entre autres que

- la définition d'un problème doit être conservée jusqu'à ce que tous les problèmes résiduels provenant de la fermeture transitive de ses descendants soient traités ;

- la restauration d'un autre problème résiduel à partir du problème courant nécessite la recherche d'un problème « ancêtre » commun et l'application d'une séquence (possiblement vide) de différences négatives (inverses) suivie d'une séquence (non vide) de différences positives.

Compte tenu de ces contraintes, l'ébauche d'une architecture générique permettant la coexistence de plusieurs méthodes de séparation indépendantes constitue une innovation importante. L'analyse du processus d'énumération permet de dégager l'interface suivante devant être respectée par chacune de ces méthodes :

- évaluation de la solution du problème courant pour déterminer la pertinence, sur une échelle continue de 0 à 1, de séparer le problème père au moyen de la méthode ;
- création d'un ensemble de différences pour chaque problème résiduel (lorsque la méthode est choisie pour séparer le problème père) ;
- application des différences positives pour créer un de ces problèmes résiduels à partir du problème père ;
- application des différences négatives pour recréer le problème père à partir d'un de ces problèmes résiduels.

Un tel formalisme amène naturellement le développement de méthodes de séparation modulaires, facilitant leur développement en parallèle par des équipes différentes. Il permet aussi d'utiliser à la fois des méthodes de séparation heuristiques et des méthodes optimales afin de mieux s'adapter au contexte du processus de résolution (par exemple, en n'utilisant les méthodes heuristiques que lorsque le risque de compromettre la qualité de la solution est faible).

L'interaction du processus d'énumération avec le problème-maître et les sous-problèmes est isolée dans chaque méthode de séparation et un minimum d'information est conservé au sujet du problème père. En particulier, les colonnes qui sont éliminées lors de l'application de différences positives sont conservées dans un ensemble global de colonnes auxiliaires et la structure de la solution du problème père (par

exemple, l'identification des colonnes en base) n'est pas préservée à cause d'interférences avec les stratégies d'élimination de colonnes présentes dans le module de gestion du problème-maître.

Parmi les autres aspects mineurs du développement du processus d'énumération, on note que l'ordre des problèmes résiduels à résoudre est paramétrisé et comprend comme cas extrêmes les stratégies classiques *profondeur d'abord* et *meilleur d'abord*, les problèmes résiduels non encore résolus étant conservés dans un arbre AVL en prévision de cette dernière stratégie.

Gestion du problème-maître : Le module de gestion du problème-maître a comme tâche principale de contrôler l'optimiseur linéaire (versions 2.1 et 3.0 de CPLEX dans ce cas-ci). Lors de la mise en œuvre de GENCOL-3, seule la méthode du simplexe est présente dans CPLEX et le choix entre l'algorithme primal et l'algorithme dual est laissé au module de calcul de la borne inférieure, qui cumule à cet effet des statistiques sur le processus de résolution.

Les optimiseurs linéaires commerciaux, dont CPLEX, conservent les données d'un programme linéaire dans leur propre format, généralement des tableaux de taille fixe, afin de maximiser l'efficacité de leurs algorithmes. Cependant, ces structures de données sont minimales et leur utilisation directe complexifie les algorithmes de gestion qui manipulent les lignes et les colonnes comme des ensembles (par exemple, en forçant ces algorithmes à maintenir explicitement des invariants sur la numérotation des objets). L'approche utilisée dans GENCOL-3 consiste à créer une *structure interne* de programme linéaire fondée sur une représentation par listes et à relayer les modifications à l'optimiseur pour assurer la synchronisation entre les deux représentations. En plus des coefficients du programme linéaire, la structure interne conserve pour chaque colonne générée le chemin correspondant afin de permettre l'analyse des solutions du problème-maître dans le contexte de la formulation originale du modèle unifié.

Comme le processus de génération de colonnes n'admet pas de solutions non réalisables pour le problème-maître, des variables artificielles de grand coût sont ajoutées au programme linéaire et le coût de chacune est multiplié par une constante chaque fois qu'elle est utilisée dans une solution (jusqu'à l'atteinte d'une valeur maximale préétablie). En permettant de choisir un coût initial « raisonnable » pour les variables artificielles, l'approche par multiplications successives évite souvent de grands écarts entre les coefficients du programme linéaire et améliore la stabilité numérique du processus d'optimisation.

Avant d'ajouter les colonnes générées par les sous-problèmes au problème-maître, ces colonnes sont d'abord triées en ordre non décroissant de coût réduit puis réarrangées en sous-séquences de colonnes presque orthogonales. Dans le contexte de problèmes comportant un grand nombre de contraintes de partitionnement, nous négligeons les coefficients associés aux autres contraintes et nous utilisons la couverture d'ensembles disjoints de tâches comme critère approché d'orthogonalité entre deux colonnes. Avec ce réarrangement, nous avons constaté empiriquement une diminution notable (facteur allant de 2 à 3 pour certains problèmes) du temps de résolution du programme linéaire par l'algorithme du simplexe primal. Nous attribuons cette amélioration, par rapport à l'utilisation des colonnes triées en ordre non décroissant de coût réduit, à une plus grande efficacité de la stratégie de *partial pricing* de l'optimiseur linéaire. Cette interprétation est fondée sur l'hypothèse que, la plupart du temps, le coût réduit d'une colonne donnée serait moins affecté par un pivot impliquant une colonne presque orthogonale que par un pivot impliquant une colonne presque colinéaire. Sous cette hypothèse, une stratégie de *partial pricing* qui restreint temporairement la sélection des colonnes entrant en base à une sous-séquence de colonnes consécutives (*voir* Nazareth, 1987) peut potentiellement effectuer plus de pivots avec la même sous-séquence si les colonnes considérées sont orthogonales.

Afin de minimiser la perte de méta-informations du côté de l'optimiseur consécutive à l'élimination de colonnes, cette opération est contrôlée par deux seuils : un nombre

maximum de colonnes admissibles en tout temps dans le programme linéaire et un nombre minimum de colonnes à éliminer lorsque le seuil maximum est atteint. L'algorithme d'élimination trie approximativement (*bucket sort*) les colonnes selon leur coût réduit et ne considère que les colonnes hors base de coût réduit non négatif. Les colonnes éliminées sont conservées dans une liste auxiliaire, elle aussi sujette à élimination (finale cette fois), qui est consultée au début de la résolution d'un nouveau problème résiduel avant de faire appel aux sous-problèmes pour générer de nouvelles colonnes.

Résolution des sous-problèmes : Chaque sous-problème, ou commodité, du modèle unifié correspond à un problème de plus court chemin avec contraintes de ressource. Afin de minimiser le couplage avec le processus d'énumération, et suivant en cela l'approche de GENCOL-2, la formulation des sous-problèmes est étendue pour inclure des contraintes sur des séquences de tâches. Ainsi, chaque nœud et chaque arc du réseau d'une commodité est associé à une séquence (possiblement vide) de tâches, chaque tâche correspondant à une contrainte de partitionnement dans le modèle unifié (une des contraintes $w \in W$).

Le réseau de chaque commodité est représenté au moyen de listes, chaque nœud donnant accès à la liste des arcs entrant (représentation *reverse star*) en prévision du développement de la variante *pulling* de l'algorithme de plus court chemin de Desrochers (1986). L'utilisation de listes, en plus de consommer moins de mémoire pour les réseaux linéaires de type espace-temps, permet des opérations de mise à jour simplifiées qui favorisent le développement d'heuristiques pour réduire la taille des réseaux.

Une heuristique très efficace en pratique pour diminuer le nombre d'itérations de génération de colonnes nécessaire pour résoudre le problème-maître (1.16)–(1.21)

consiste à favoriser la génération de colonnes orthogonales à une itération donnée. Comme pour le traitement du problème-maître, nous utilisons la couverture d'ensembles disjoints de tâches comme critère approché d'orthogonalité entre deux colonnes. Pour chaque sous-problème ayant produit des colonnes de coût réduit négatif, nous trions ces colonnes en ordre non décroissant de coût réduit et nous construisons au moyen d'une heuristique gloutonne un certain nombre de blocs de colonnes orthogonales. Lors de la résolution d'un nouveau sous-problème, nous éliminons des réseaux les tâches couvertes par le premier bloc de colonnes orthogonales, forçant ainsi les prochains sous-problèmes à générer des colonnes orthogonales à celles identifiées dans ce bloc. En plus d'accélérer en pratique la convergence de l'algorithme, cette stratégie favorise la production de solutions admissibles en nombres entiers lors des réoptimisations du problème-maître lorsque les contraintes de partitionnement forment la majorité des contraintes linéaires.

L'efficacité de l'algorithme de plus court chemin lui-même peut être améliorée en factorisant un certain nombre d'opérations dont le résultat reste inchangé pendant une partie importante du processus de résolution. Le fait de ne considérer que des réseaux acycliques permet de précalculer l'ordre de traitement des nœuds d'une commodité au moyen d'un tri topologique lors de l'initialisation des données. Le prétraitement inclut aussi la réduction des intervalles de ressource sur chaque nœud lorsque cette réduction est appropriée. La présence de contraintes sur des séquences de tâches force l'utilisation d'un algorithme plus complexe qui bénéficie aussi d'un prétraitement, effectué au début de la résolution d'un problème résiduel, pour identifier les tâches sans successeurs dans les réseaux de chaque commodité.

L'algorithme de Desrochers (1986) pour résoudre les problèmes de plus court chemin avec contraintes de ressource constitue la pierre angulaire de toutes les versions de GENCOL et sa mise en œuvre efficace revêt une importance majeure pour la plupart des applications fondées sur le modèle unifié. Dans GENCOL-3, les hypothèses

Tableau 4.1 – Complexité du calcul des minima d'un ensemble de n vecteurs

Dimension d	Complexité
1	$O(n)$
2	$O(n \log n)$
3	$O(n \log n)$
≥ 4	$O(n \log^{d-2} n)$

suivantes permettent de simplifier significativement les structures de données et l'algorithme :

- le réseau de chaque commodité est acyclique ;
- les fonctions d'extension f_{ij}^{kr} (y compris l'objectif) sont non décroissantes ;
- les intervalles de ressource sont définis par des entiers ;
- les fonctions d'extension préservent l'intégralité de leurs arguments.

Sur un réseau acyclique, l'algorithme se réduit à construire itérativement, dans l'ordre topologique, la fonction de coût minimum $c_j : \mathbb{N}^{|R^k|} \mapsto (\mathbb{R} \cup \{\infty\})$ en chaque nœud $j \in N^k$ d'une commodité $k \in K$ à partir des fonctions de coût c_i des nœuds précédents, avec $(i, j) \in A^k$. Les fonctions d'extension étant non décroissantes, la fonction $c_j^{p_j}$ associée à un chemin p_j donné reliant les nœuds o^k et j est complètement déterminée d'une part, par les valeurs de consommation minimale des ressources au nœud j en fonction de p_j et d'autre part, par le coût associé à ce point. Ainsi, une fonction $c_j^{p_j}$ est représentée comme un vecteur de $\mathbb{N}^{|R^k|} \times \mathbb{R}$ nommé *étiquette multi-dimensionnelle.* La forme des fonctions $c_j^{p_j}$ permet de compresser la représentation de $c_j = \min_{p_j} c_j^{p_j}$, sans perte d'information, en ne conservant que les vecteurs correspondant à des minima. La complexité de cette opération dépend du nombre n de vecteurs et de leur dimension $d = |R^k| + 1$, comme mentionné au tableau 4.1. Dans GENCOL-3, seuls les algorithmes pour $d = 1$ et $d = 2$ sont de complexité minimale, alors que l'algorithme pour les autres valeurs de d utilise l'approche naïve de complexité $O(n^2)$ (Bentley, 1980 ; Bentley *et al.* 1993 ; Kung *et al.*, 1975).

Une part importante de l'efficacité de GENCOL-3 provient de l'attention rigoureuse

accordée à la gestion des ressources informatiques et, en particulier, à la gestion de la mémoire. En effet, l'objectif principal de cette version consiste à résoudre des problèmes de grande taille représentés par plusieurs mégaoctets de mémoire et l'algorithme de résolution des sous-problèmes génère plusieurs millions d'étiquettes à chaque appel. À titre d'exemple, un prototype de l'algorithme de Desrochers (1986) basé sur l'interface `malloc/free` du langage C pour gérer les étiquettes passait plus de 60% du temps dans ces fonctions. Le développement d'un système spécialisé de gestion de mémoire fondé sur l'utilisation généralisée de listes (simplement et doublement chaînées), l'allocation d'objets par blocs et leur recyclage dans des listes dédiées a permis de réduire la proportion du temps de gestion de la mémoire à moins de 1% du temps total de résolution. Le développement de fonctions optimisées pour la lecture de données, ainsi que le développement d'un gestionnaire de paramètres, ont aussi contribué significativement à la qualité du logiciel.

Les derniers paragraphes ont dressé un bref portrait de GENCOL-3, attirant l'attention sur les décisions ayant le plus influencé sa structure et son efficacité. Malheureusement, cette version de GENCOL ne développe pas tout le potentiel présent dans l'analyse, particulièrement en ce qui a trait au processus d'énumération. Ainsi, le nombre de méthodes de séparation indépendantes est fixé à trois :

- séparation par ajout de contraintes sur des séquences de tâches ;
- séparation sur bornes des variables Y_s ;
- séparation sur bornes des variables générées θ_p^k.

De plus, la première méthode ne tient pas compte de variables Y_s comportant des contributions aux contraintes de partitionnement associées aux tâches, et la dernière méthode de séparation est incomplète, ne permettant pas de calculer les différences négatives pour l'interface générique exposée précédemment et forçant de ce fait une stratégie d'exploration en profondeur seulement. Concernant la résolution du problème-maître, la décision de ne plus perturber le membre de droite du

programme linéaire (en supposant que cette opération était complètement prise en charge par l'optimiseur linéaire) constitue une erreur qui sera corrigée dans la prochaine version (*voir* le chapitre 5 pour une discussion plus approfondie). Finalement, l'absence de certaines primitives de gestion d'ensembles, entre autres une primitive de tri pour listes chaînées, a mené à une version sous-optimale des algorithmes de calcul des minima d'un ensemble de vecteurs, dans le contexte de la résolution des sous-problèmes.

La réalisation de GENCOL-3 atteint, malgré un certain nombre d'imperfections, ses objectifs commerciaux en s'intégrant au système ALTITUDE (Desrosiers *et al.*, 1999) de la compagnie AD OPT et en remplaçant GENCOL-1 dans les produits de la compagnie GIRO. Elle a ainsi contribué à l'obtention ou au renouvellement de contrats importants pour AD OPT (entre autres : UPS, 1994 ; AIR TRANSAT, 1996) et à l'obtention de plusieurs contrats pour GIRO (entre autres : Vienne, Tokyo, Singapour, Helsinki, Barcelone, 1995), sans compter la participation à de nombreux bancs d'essai pour ces deux compagnies. Quelques travaux de recherche, présentés au tableau 4.2, utilisent aussi GENCOL-3 comme moteur d'optimisation dans des applications fondées sur le modèle unifié. Dans le cas de Lacoursière (1992), il s'agit du développement d'un nouvel algorithme dans le cadre de GENCOL-3.

La mise en œuvre de GENCOL-3 est le résultat d'un effort de génie logiciel pour d'une part, obtenir une très grande efficacité en termes de temps de calcul et de consommation de mémoire et d'autre part, contrôler la complexité du logiciel en minimisant le couplage entre les algorithmes. L'objectif d'efficacité semble atteint, le logiciel étant utilisé comme un composant stable (boîte noire) dans plusieurs projets commerciaux et de recherche. Par contre, GENCOL-3 ne présente pas une structure suffisamment flexible pour amorcer de véritables travaux de recherche sur de nouveaux algorithmes, tant pour la phase de séparation du processus d'énumération que pour la résolution des sous-problèmes et la gestion du problème-maître.

Tableau 4.2 – Travaux de recherche utilisant GENCOL-3.

Auteur(s)	Titre
Lacoursière (1992)	Heuristiques pour l'obtention de solutions entières à partir de solutions continues.
Sgaier (1993)	Ordonnancement de la production dans un atelier de type « *job-shop* » avec machines à traitement par lots.
Ribeiro et Soumis (1994)	*A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem.*
Laurent *et al.* (1995)	*A Column Generation Method for Optimal Load Management via Control of Water Heaters.*
Messie (1995)	Problème hebdomadaire d'affectation de locomotives aux trains.
Koty (1996)	Construction d'itinéraires d'une flotte d'avions hétérogène avec contraintes d'entretien.
Desaulniers *et al.* (1997a)	*Crew Pairing at Air France.*
Stojković *et al.* (1998)	*The Operational Airline Crew Scheduling Problem.*

4.3 Mise en œuvre de GENCOL-4

En 1994, le logiciel GENCOL-3 est utilisé dans plusieurs applications commerciales et projets de recherche et les derniers développements sur cette version visent à augmenter sa robustesse dans les cas limites (ensembles vides et hypothèses sur les données, gestion de la mémoire dans le contexte de très longs processus de résolution, ...). Au même moment, il devient évident que cette version ne peut constituer un cadre de recherche acceptable pour l'élaboration de nouveaux algorithmes : structures de données trop minimales ou fermées, couplage encore trop grand entre algorithmes du processus d'énumération et algorithmes de résolution des sous-problèmes. Or, certains projets potentiels impliquent le développement d'algorithmes d'ajout de coupes, de plus court chemin sur graphes avec cycles, de plus court chemin avec coûts linéaires sur les variables de ressource, d'agrégation statique et dynamique des contraintes.

De plus, certaines applications plus complexes nécessitent beaucoup plus de points de contrôles que ceux prévus dans GENCOL-3. Ce contexte mène à une réévaluation des objectifs pour mieux pondérer l'efficacité en fonction des compromis impliqués au niveau de la flexibilité.

Le logiciel GENCOL-4, développé de 1994 à 1998 en collaboration avec Éric Gélinas, Norbert Lingaya et José Manuel Pires, constitue la deuxième contribution expérimentale majeure de cette thèse. Cette version est le résultat d'une analyse plus théorique du modèle unifié en vue d'en déceler les extensions possibles. L'objectif principal est de mieux supporter l'ensemble des fonctionnalités décrites dans le modèle unifié tout en maintenant ou en améliorant les qualités d'efficacité et de robustesse de la version précédente.

Le développement de GENCOL-4 prend explicitement pour cadre théorique le modèle unifié du tableau 1.1. Cependant, certaines parties du modèle nécessitent un traitement suffisamment différent du traitement requis pour la majorité des applications pour justifier leur exclusion de l'interface naturelle définie par le langage de modélisation. Ainsi, les fonctions d'extension f_{ij}^{kr} et la fonction objectif des sous-problèmes sont supposées non décroissantes en prévision de la résolution efficace des sous-problèmes au moyen de l'algorithme SPR de Desrochers (1986). Ces conditions permettent de ne pas supporter l'utilisation de coefficients b_{hi}^{kr} dans les contraintes H. Par contre, l'architecture du logiciel comporte des interfaces de spécialisation pour des recherches algorithmiques sur ces parties du modèle, permettant entre autres le développement par Ioachim *et al.* (1998) d'un nouvel algorithme de résolution des sous-problèmes acceptant une fonction objectif avec coûts linéaires quelconques sur les variables de ressource (*i.e.*, ne possédant pas la propriété de non décroissance). Cette nouvelle version de GENCOL supporte aussi quelques extensions par rapport au modèle théorique : d'une part, les graphes avec cycles sont acceptés, d'autre part, le concept de tâche est conservé (chaque tâche est associée à une contrainte de par-

tionnement de l'ensemble W) et les contraintes sur des séquences de tâches peuvent faire partie des données.

La structure générale de GENCOL-4 découle directement de celle de GENCOL-3. La localisation des principaux algorithmes et les relations entre les modules sont revues afin de diminuer le couplage entre ces derniers. Comme plusieurs aspects du modèle unifié nécessitent un traitement spécialisé (méthodes de séparation, algorithmes de résolution des sous-problèmes, optimiseurs linéaires), des protocoles sont définis afin de permettre la construction d'optimiseurs « sur mesure » par l'assemblage de modules indépendants. Il est aussi possible de spécialiser directement les modules génériques (gestion du processus d'énumération, calcul d'une borne inférieure) en modifiant le comportement de certains sous-algorithmes (détection des solutions réalisables, stratégie d'exploration de l'arbre d'énumération, choix du prochain sous-problème à résoudre, ...). La figure 4.2 illustre les relations entre les modules de GENCOL-4 selon la même convention que celle utilisée pour la figure 4.1. Les lignes pleines entourant un ensemble de composants indiquent que ceux-ci peuvent être considérés comme un seul module dans une analyse macroscopique. Les lignes pointillées représentent des interfaces logicielles locales aux modules qu'elles séparent. Une brève description des fonctions de chaque module est donnée au tableau 4.3. L'assemblage sur mesure d'un optimiseur pour le modèle unifié est évoqué à la figure 4.2 par les points d'interrogation signifiant :

- que les méthodes de séparation (ITFIX, CFIX, ...) peuvent être incluses ou non ; la case ne contenant qu'un point d'interrogation correspond à la possibilité de créer de nouvelles méthodes indépendantes des méthodes existantes (une application construisant un optimiseur pour le modèle unifié en assemblant différents modules comporte en général plusieurs méthodes de séparation mises en compétition) ;
- que n'importe quel optimiseur linéaire (un seul pour un optimiseur donné) respectant l'interface du module MP peut être utilisé pour résoudre le problème-maître

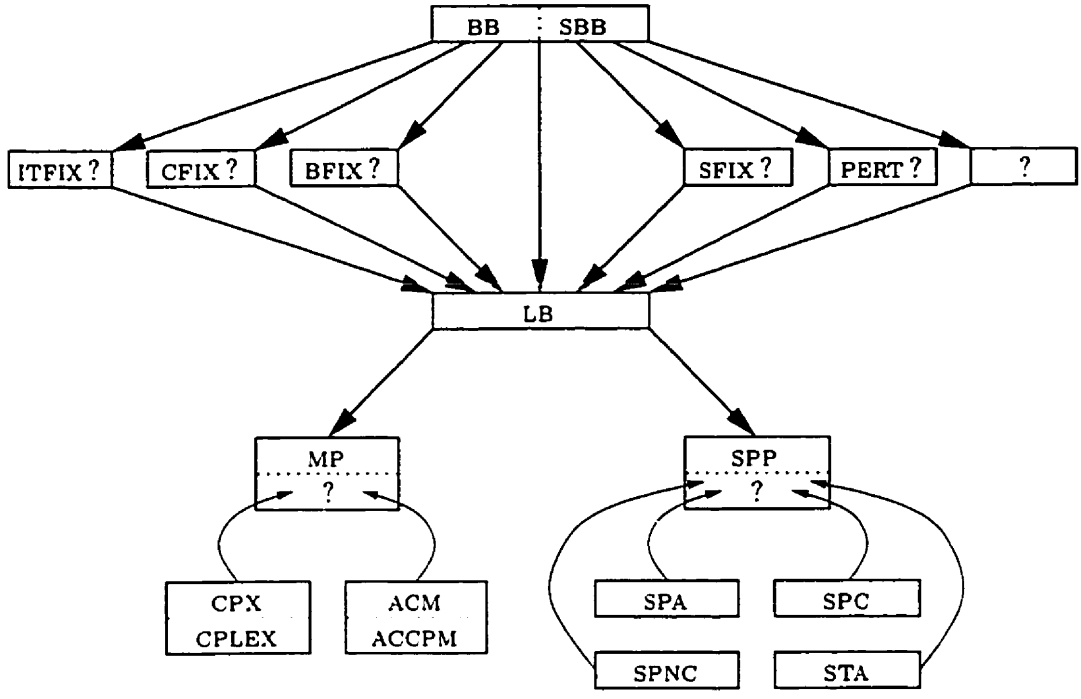


Figure 4.2 – Interactions entre les modules principaux de GENCOL-4.

(ceci implique généralement le développement d'une interface entre le module MP et un optimiseur linéaire existant) ;

- que n'importe quel algorithme de résolution du problème SPR ou d'une de ses variantes (un seul pour un optimiseur donné) respectant l'interface du module SPP peut être utilisé pour résoudre les sous-problèmes.

Les prochains paragraphes décrivent les particularités de chacun des modules principaux de GENCOL-4.

Le processus d'énumération est séparé en trois composants : le module générique BB qui gère les structures de données et la séquence globale des opérations, le module stratégique SBB qui définit les opérations du processus d'énumération propres au modèle unifié, ainsi qu'une famille ouverte de modules indépendants (ITFIX, CFIX, ...) correspondant aux méthodes de séparation. Ces méthodes doivent respecter une

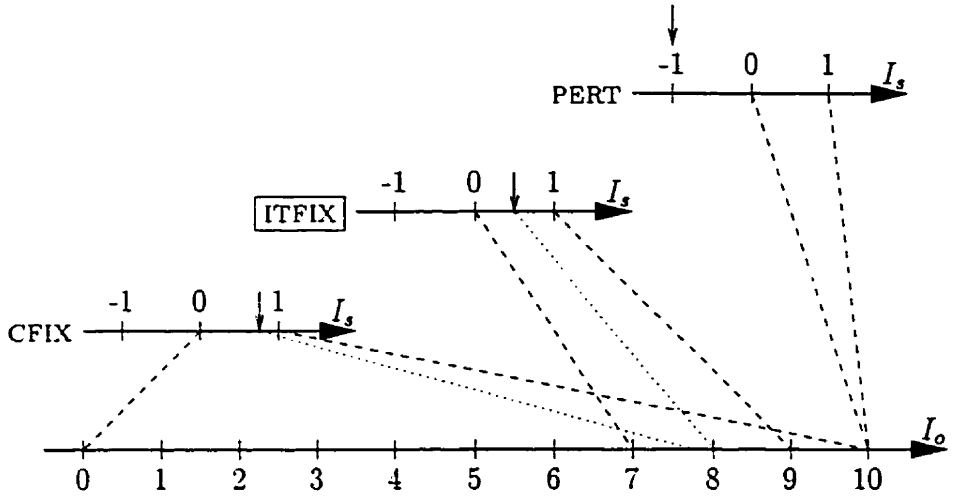
Tableau 4.3 – Signification des acronymes dénotant les modules de GENCOL-4.

Acronyme	Signification
BB	processus d'énumération générique : algorithme principal de résolution du modèle unifié, gestion de l'arbre d'énumération, principe de compétition entre les méthodes de séparation
SBB	stratégies du processus d'énumération : gestion des bornes, détection de solutions réalisables, stratégies d'exploration de l'arbre d'énumération
ITFIX	séparation par ajout de contraintes sur des séquences de tâches
CFIX	séparation par majoration de la borne inférieure des variables θ_p^k
SFIX	séparation par allocation des tâches w aux commodités k
BFIX	séparation par restrictions des intervalles des variables supplémentaires Y_s
PERT	séparation par retrait de perturbations ayant été introduites initialement pour relaxer le domaine du problème (*voir* chapitre 5)
LB	calcul d'une borne inférieure : algorithme principal de génération de colonnes, choix des heuristiques, critères d'arrêt
MP	gestion du problème-maître restreint : représentation des données (lignes, colonnes, coefficients), transformations entre les représentations en phase 1 et en phase 2, définition d'une interface pour les fonctions d'optimisation
CPX	interface entre le module MP et l'optimiseur linéaire CPLEX : synchronisation des données, compilation de statistiques, choix de l'algorithme de résolution, gestion des cas pathologiques
CPLEX	optimiseur linéaire commercial CPLEX 6.0 (CPLEX, 1998)
ACM	interface expérimentale entre le module MP et l'optimiseur linéaire ACCPM
ACCPM	optimiseur linéaire spécialisé pour les problèmes de génération de colonnes utilisant la méthode des centres analytiques de Goffin et Vial (1990)
SPP	gestion des sous-problèmes : représentation des données (nœuds, arcs, réseaux), opérations génériques (tri topologique, certains prétraitements), définition d'une interface pour les fonctions d'optimisation
SPA	algorithme pour résoudre les problèmes SPR sur les graphes acycliques avec contraintes sur des séquences de tâches
SPC	extension du module SPA pour traiter les problèmes avec cycles, supportant l'élimination des 2-cycles sur les séquences de tâches (Lingaya, 1996)
SPNC	extension du module SPA pour traiter les problèmes avec objectif linéaire en fonction des variables de ressource (Ioachim, 1998)
STA	version parallèle du module SPA (Pires, 1997)

interface similaire à celle utilisée dans GENCOL-3, comportant entre autres des services d'analyse de solution, de création de problèmes résiduels et d'application de différences positives et négatives. Le choix de la méthode devant créer de nouveaux problèmes résiduels est établi selon un mécanisme de compétition géré par le module BB. Étant donné un problème résiduel pour lequel une séparation est nécessaire, le module BB analyse itérativement la solution au moyen de chaque méthode et recueille un indicateur de succès « subjectif » $I_s \in [0,1] \cup \{-1\}$ pour chacune d'elles. Un indicateur égal à -1 dénote l'impossibilité d'appliquer la méthode, alors qu'un indicateur entre 0 et 1 représente le degré d'efficacité espérée de la méthode selon ses propres critères (une valeur de 1 correspondant au maximum d'efficacité). Le module BB obtient ensuite un indicateur « objectif » I_o par l'application d'une tranformation linéaire de chaque indicateur non négatif dans $\mathbb{R}^+$. Les paramètres de chaque transformation (extrémités d'un intervalle de $\mathbb{R}$) sont fournis avec les données du problème, afin de contrôler l'exploration de l'arbre d'énumération selon un compromis préétabli entre la qualité de solution et le temps d'exécution. La figure 4.3 illustre ce principe de compétition entre trois méthodes pour un problème résiduel donné. Dans cet exemple, la méthode ITFIX est choisie puisque son indicateur objectif I_o est le plus grand.

Le module LB calcule une borne inférieure d'un problème résiduel par génération de colonnes. Ce traitement accapare souvent plus de 99% du temps de résolution et nécessite l'utilisation de plusieurs heuristiques contrôlées par des paramètres. Ceux-ci permettent de modifier le comportement des modules MP et SPP en spécifiant, entre autres,

- quels chemins transformer en colonnes,
- le nombre de colonnes à conserver dans le problème-maître restreint,
- combien de sous-problèmes résoudre à chaque itération,
- une sélection d'algorithmes pertinents pour l'optimiseur linéaire,



Méthode	Intervalle	Indicateur subjectif I_s	Indicateur objectif I_o
CFIX	[0, 10]	0,75	7,5
ITFIX	[7, 9]	0,50	**8,0**
PERT	[10, 10]	−1,00	non applicable

Figure 4.3 – Exemple de processus de compétition entre trois méthodes de séparation.

- quand arrêter le calcul et retourner au processus d'énumération,
- le type de compression à appliquer à la représentation des fonctions c_i de chaque nœud $i \in N^k$ (*voir* la discussion sur la résolution des sous-problèmes à la section précédente).

L'utilisation d'heuristiques peut réduire significativement le temps de calcul mais peut affecter en même temps la qualité des solutions obtenues. Dans GENCOL-4, l'étape du calcul d'une borne inférieure est donc remplacée par une séquence d'étapes, chacune comportant son propre ensemble de paramètres fournis par l'application. Cette idée était déjà présente depuis GENCOL-2, sans avoir toutefois le rôle de premier plan qu'elle occupe dans GENCOL-4.

Le module MP prend en charge la gestion des données du problème-maître restreint selon la même approche que dans GENCOL-3, mais son développement diffère sur plusieurs points. Premièrement, les communications entre le module MP et l'optimiseur

linéaire (synchronisation des structures de données, optimisation) passent par une interface opaque bien définie qui permet d'expérimenter avec d'autres stratégies d'optimisation (par exemple, en utilisant l'optimiseur ACCPM au lieu de CPLEX). Deuxièmement, les solutions non réalisables sont désormais traitées selon une méthode en deux phases afin d'accroître la stabilité numérique des algorithmes de l'optimiseur linéaire. L'inconvénient relié à la phase I, soit un nombre accru d'itérations découlant de l'utilisation d'un objectif temporaire non corrélé au véritable objectif d'optimisation, est compensé en pratique par un gain notable d'efficacité de l'optimiseur linéaire durant cette phase. Troisièmement, le module MP supporte les algorithmes de points intérieurs, dont les solutions ne sont pas extrémales. L'avantage de cette approche vient de ce que l'effort de calcul requis pour transformer une solution optimale en solution de base (*cross-over*) peut représenter jusqu'à 30% du temps de résolution de l'optimiseur linéaire.

Les données du problème-maître sont représentées de façon à simplifier les algorithmes de résolution du modèle unifié (recyclage des colonnes, analyse des solutions, application de contraintes provenant des méthodes de séparation), sans se soucier de leur utilisation directe par les algorithmes d'un optimiseur linéaire donné. Le module CPX constitue l'interface nécessaire pour utiliser l'optimiseur CPLEX 6.0 dans ce contexte. Ce module incorpore quelques améliorations importantes par rapport au module équivalent de GENCOL-3. D'une part, les cas pathologiques sont traités par un algorithme qui contrôle le processus d'optimisation en ajustant dynamiquement les tolérances, prétraitements et autres paramètres de CPLEX. Les anomalies sont aussi traitées selon l'algorithme de CPLEX utilisé (par exemple, l'algorithme de points intérieurs diagnostique parfois erronément un problème comme non réalisable). D'une part, des statistiques sont cumulées sur les temps de résolution des différents algorithmes (simplexe primal, simplexe dual, points intérieurs) selon la taille du programme linéaire, mesurée comme le nombre de coefficients non nuls.

Ces statistiques sont principalement utilisées pour choisir, parmi l'ensemble d'algorithmes suggérés par le module LB, l'algorithme qui devrait minimiser le temps de résolution. Ce choix est en pratique biaisé en faveur des algorithmes de réoptimisation puisque la nature des modifications apportées au problème-maître (principalement des ajouts de colonnes) est favorable à cette stratégie. Les statistiques sont aussi utilisées pour limiter le temps alloué à la résolution lorsque qu'un algorithme instable est choisi (par exemple, l'algorithme de réoptimisation du simplexe primal présente une grande variance des temps de calculs) et pour identifier un algorithme de rechange.

Le module SPP a suivi une évolution similaire à celle du module MP. Les données sont enrichies par la définition de nouveaux ensembles et par l'utilisation généralisée de listes doublement chaînées, dans le but de simplifier les algorithmes de réduction de graphe et d'analyse de solutions souvent développés dans le contexte d'applications spécifiques. Comme pour le module MP, l'algorithme d'optimisation du module SPP est accédé par le biais d'une interface opaque. L'innovation de ce module consiste à isoler le traitement des contraintes sur les séquences de tâches des différents algorithmes d'optimisation du problème SPR (développés dans les modules SPA, SPC, ...). La présence de contraintes sur les séquences de tâches implique qu'il faut tenir compte de la dernière tâche effectuée sur un chemin partiel entre les nœuds o^k et $j \in N^k$ avant de comparer les vecteurs caractéristiques lors de la compression de la représentation de la fonction c_j au moyen des algorithmes de dominance. L'approche du module SPP consiste

- à calculer au début de la résolution d'un problème résiduel et en chaque nœud $j \in N^k$, l'ensemble des tâches pouvant succéder immédiatement ce nœud j dans toute extension des chemins partiels entre o^k et j ;
- à isoler une fonction auxiliaire des fonctions d'extensions f_{ij}^{kr} permettant de partitionner efficacement les étiquettes selon la dernière tâche effectuée et selon les ensembles précalculés en chaque nœud j.

Le traitement des contraintes sur les séquences de tâches se réduit alors à regrouper les étiquettes selon les sous-ensembles identifiés par la fonction auxiliaire et à appliquer les algorithmes de dominance par sous-ensemble. Cette factorisation diminue ainsi significativement la complexité des algorithmes de résolution du problème SPR en présence de contraintes sur les séquences de tâches.

L'architecture du module SPP permet le développement indépendant de plusieurs algorithmes de résolution du problème SPR, dont les modules SPA, SPNC (Ioachim, 1994), SPC (Lingaya, 1996) et STA (Pires, 1997). La description qui suit s'applique uniquement au module SPA, traitant les graphes acycliques sous l'hypothèse de fonctions d'extension et d'un objectif non décroissants selon les variables de ressource. Les algorithmes étant extrêmement simplifiés par la factorisation du tri topologique et du traitement des contraintes sur les séquences de tâches, l'effort de développement est entièrement dévolu aux algorithmes de dominance qui calculent les minima des ensembles de vecteurs correspondant aux étiquettes en chaque nœud. D'une part, une primitive de tri pour listes chaînées est mise au point, exploitant les sous-séquences déjà triées. Cette primitive est approximativement deux fois plus rapide que la primitive de tri fournie par le langage C. Ceci permet de réviser ou développer les algorithmes de complexité optimale pour le calcul des minima de vecteurs de dimension 1, 2 et 3. L'algorithme traitant les vecteurs de dimension supérieure à 3 demeure de complexité $O(n^2)$ mais constitue tout de même un choix justifié étant donné la taille moyenne des problèmes de minima à traiter (le plus souvent quelques centaines d'étiquettes, rarement plus de mille). D'autre part, l'algorithme de plus court chemin permet de traiter efficacement le cas de deux ressources simulant une contrainte d'égalité au moyen de paires de contraintes (1.9). Une contrainte d'égalité nécessite en effet, dans le modèle unifié, l'utilisation de deux ressources de même grandeur mais de signe opposé, qui restreignent en pratique l'application de l'algorithme de dominance aux etiquettes ayant une valeur égale pour ces ressources. Afin d'éviter les

comparaisons superflues, les étiquettes sont séparées selon la valeur de cette ressource (une seule ressource contenant toute l'information) avant d'appliquer l'algorithme de dominance sur les composantes restantes des vecteurs.

Encore plus que pour GENCOL-3, l'efficacité de GENCOL-4 est tributaire du développement d'une collection de primitives optimisées, entre autres pour la gestion des structures de listes (simplement et doublement chaînées) et d'arbres (AVL, SPLAY), le traitement des tolérances numériques et la lecture rapide de données. Cette bibliothèque constitue non seulement le fondement logiciel de tous les modules de GENCOL-4, mais aussi celui de la plupart des applications qui gravitent autour de GENCOL. L'interface de cette bibliothèque est décrite en détail par Pires et Villeneuve (1999).

Les derniers paragraphes ont attiré l'attention sur les améliorations qui différencient GENCOL-4 de GENCOL-3. Malgré que les problèmes reliés à la version 3 aient été résolus, de nouvelles lacunes sont apparues suite à l'utilisation de la version 4 dans de nombreux projets de recherche. Ces lacunes étaient latentes dans les versions précédentes et sont devenues contraignantes dans le contexte de nouvelles applications de plus grande taille ou exploitant des particularités du modèle unifié. Ce paragraphe présente deux critiques de GENCOL-4 tenant compte de l'expérience acquise ainsi que des besoins anticipés pour certains projets à venir. Premièrement, l'hypothèse que la séquence des problèmes-maîtres restreints correspond à celle définie par l'algorithme de Kelley (1960) rend difficile l'intégration de nouvelles techniques d'optimisation, comme l'algorithme ACCPM de Goffin et Vial (1990). En effet, des tests sur l'intégration d'ACCPM dans GENCOL, effectués en collaboration avec du Merle et Pires en 1998, mettent en évidence des incompatibilités entre ACCPM et plusieurs des heuristiques incorporées à GENCOL, surtout celles impliquant la résolution sous-optimale des sous-problèmes. Une correction possible consisterait à concevoir des heuristiques calculant une borne inférieure sur la solution d'un sous-problème. Ces heuristiques

pourraient alors être utilisées comme méthodes alternatives dans le calcul de la borne inférieure (2.7) d'un problème résiduel dans les cas où les heuristiques habituelles ne produisent pas d'information valable. Deuxièmement, les méthodes de séparation par coupe ne devraient pas être assimilées à des méthodes dégénérées de séparation par branchement. Bien que correcte mathématiquement, cette interprétation empêche l'utilisation globale des coupes (dont les algorithmes générateurs n'utilisent souvent que la structure initiale du problème, rendant les coupes valides globalement) et le développement de méthodes générales pour gérer un grand nombre de contraintes linéaires (qu'elles proviennent de coupes générées au besoin ou qu'elles soient déjà présentes dans la formulation initiale du problème). Le module LB, qui contrôle le processus itératif de résolution des problèmes-maîtres restreints et des sous-problèmes, semble l'endroit le plus approprié pour gérer dynamiquement un grand nombre de contraintes. Cette gestion suppose l'existence d'opérations efficaces pour manipuler des ensembles de contraintes, et les développements convergent ultimement vers l'utilisation de bornes explicites sur les variables duales et l'intégration des techniques de stabilisation présentées au chapitre 5.

Les imperfections mentionnées précédemment n'empêchent toutefois pas GENCOL-4 de constituer en 1994 la nouvelle base de développement du logiciel commercial PBS (*voir* Gamache *et al.*, 1998) développé conjointement avec la compagnie AD OPT et commercialisé chez AIR CANADA en 1995. Ce logiciel est maintenant implanté dans au moins quatre autres compagnies (AIR TRANSAT, DELTA, TWA, 1997 ; CANADIAN REGIONAL, 1998). En 1996, GENCOL-4 est devenu suffisamment stable pour que les compagnies AD OPT et GIRO amorcent la mise à jour de leurs logiciels ALTITUDE et CREW-OPT pour remplacer GENCOL-3 comme moteur d'optimisation. Le logiciel ALTITUDE est par la suite implanté dans au moins quatre compagnies (NORTH WEST, 1997 ; SABENA, FEDEX, CANADIAN REGIONAL, 1998) et le logiciel CREW-OPT, dans une douzaine de villes (Valencienne, Perpignan, Oslo, 1997 ; Montpellier, Val d'Oise,

Tableau 4.4 – Mémoires et thèses utilisant GENCOL-4.

Auteur(s)	Titre
Gentès (1996)	Construction d'itinéraires quotidiens et hebdomadaires d'une flotte d'avions hétérogène.
Jean (1996)	Plans de coupure pour des problèmes de multiflots dans des graphes acycliques.
Lasry (1996)	Rotations d'équipage pour un transporteur aérien régional.
Ben Amor (1997)	Le problème de la découpe binaire.
Rochon (1997)	Ajustement des variables duales dans le contexte d'une méthode de génération des colonnes.
Pires (1997)	Développement de méthodes parallèles pour des problèmes de grande taille.
Stojković (1999)	Gestion journalière d'une flotte d'avions.

Valence, Brunoy, Bruxelles, Figestrans, Mauberge, Stockholm, 1998). Les premiers projets de recherche en transport ferroviaire (Ziarati *et al.*, 1997) débouchent sur une implantation d'un logiciel d'affectation simultanée des locomotives et des wagons chez VIA RAIL en 1997. Aussi, un logiciel de gestion simultanée des hélicoptères et des équipages est réalisé en collaboration avec le ministère de la défense nationale en 1996. Parallèlement, la structure plus ouverte de GENCOL-4 a permis la conception et la réalisation de nombreux projets de recherche. Plusieurs des implantations commerciales mentionnées précédemment ont d'ailleurs vu le jour comme projets de maîtrise ou de doctorat. Les tableaux 4.4 et 4.5 présentent respectivement une liste des mémoires et thèse ainsi que des articles dont les résultats sont dérivés de l'utilisation de GENCOL-4.

La mise en œuvre de GENCOL-4 atteint donc ses objectifs en permettant l'utilisation du modèle unifié dans des applications de plus en plus diversifiées. Plusieurs projets commerciaux et de recherche ont pu être réalisés grâce aux nouvelles structures ouvertes et flexibles permettant aux applications de compléter la description des données ou de modifier le cours par défaut des algorithmes. La robustesse du pro-

Tableau 4.5 – Articles utilisant GENCOL-4.

Auteur(s)	Titre (notes)
Ioachim *et al.* (1994)	*Fleet Assignment and Routing with Schedule Synchronisation Constraints.*
Desaulniers *et al.* (1997b)	*Scheduling-Routing for Air Force Resource Management: Reports on Tasks I, II and III.*
Desaulniers *et al.* (1997c)	*Daily Aircraft Routing and Scheduling.*
Ziarati *et al.* (1997)	*Locomotive Assignment with Heterogeneous Consists at CN North America.*
Desaulniers *et al.* (1998b)	*Multi-Depot Vehicule Scheduling Problems with Time Windows and Waiting Costs.*
Desaulniers *et al.* (1999)	*Crew Pairing for a Regional Carrier.*
Gamache *et al.* (1998)	*The Preferential Bidding System at Air Canada.*
Haase *et al.* (1998)	*Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems.*
Chauny *et al.* (1999)	*A Column Generation Approach to the Aircraft Loading Problem.*
Cordeau *et al.* (1999)	*Simultaneous Assignment of Locomotives and Cars to Passenger Trains.*
Desaulniers et Villeneuve (1999)	*The Shortest Path Problem with Time Windows and Linear Waiting Costs.*
du Merle *et al.* (1999)	*Stabilized Column Generation.*
Gamache *et al.* (1999)	*A Column Generation Approach for Large Scale Aircrew Rostering Problems.*
Ziarati *et al.* (1999)	*A Branch-First, Cut-Second Approach for Locomotive Assignment.*

cessus de résolution s'est surtout améliorée par une meilleure gestion de l'optimiseur linéaire. L'efficacité en temps de calcul n'a pas souffert des généralisations apportées, notamment à cause du développement de meilleures primitives (tris, arbres, gestion de la mémoire) et de factorisations supplémentaires plus faciles à exploiter dans une architecture modulaire.

Conclusion

L'évolution du logiciel GENCOL présentée dans ce chapitre est le reflet des raffinements du modèle unifié mais aussi de la transformation des moyens technologiques, comme l'augmentation de la puissance de calcul et de la taille des mémoires ainsi que l'apparition d'outils pour mieux gérer la complexité au niveau informatique. Ce contexte a permis d'adapter le logiciel en fonction des besoins changeants et toujours croissants des applications provenant de l'industrie et des projets de recherche.

Lors de la conception de GENCOL, le défi principal fut de concilier deux classes d'objectifs contradictoires. D'une part, les objectifs de robustesse impliquent un couplage minimal entre les modules et permettent l'atteinte d'une grande efficacité locale (temps d'exécution et consommation de mémoire). D'autre part, les objectifs de flexibilité algorithmique impliquent une accessibilité maximale aux données afin de permettre l'exploitation optimale de la structure de chaque application par l'adaptation aux biais présents dans les données. Les compromis ont été tranchés à la fois en fonction d'une analyse théorique approfondie du modèle unifié et de tous les algorithmes de résolution développés, mais aussi en tenant compte des résultats expérimentaux obtenus sur les applications-clés en transport aérien, ferroviaire et urbain. La mise en œuvre de GENCOL constitue ainsi une contribution majeure en tant que réalisation opérationnelle sur la base de laquelle toute une équipe de chercheurs et de développeurs conçoivent plusieurs projets aux retombées commerciales et scientifiques positives. À titre d'indicateur du succès du logiciel, on dénombrait en mars 1999 plus de 30 implantations commerciales majeures et environ 30 publications scientifiques dont les résultats proviennent de l'utilisation de GENCOL-3 ou de GENCOL-4.

L'ambition essentielle du modèle unifié et du logiciel GENCOL est de constituer une référence pour un grand nombre de problèmes de tournées de véhicule et d'horaires de

personnel. D'aucuns pourraient soulever qu'au niveau du logiciel, une telle généralité entraîne inévitablement des pertes d'efficacité. Bien que cette affirmation soit incontestable, elle néglige les coûts de développement (surtout pour l'atteinte d'une grande efficacité logicielle) et la synergie entre différents contextes d'utilisation. Les nombreuses applications et publications utilisant GENCOL comme optimiseur confirment la pertinence de l'approche de développement utilisée, visant ultimement l'obtention de l'ensemble des fonctionnalités du modèle dans un logiciel biaisé selon le coût et la fréquence d'utilisation de ces fonctionnalités.

Chapitre 5

Stabilisation pour la génération de colonnes

Considérons le programme linéaire P, réalisable et borné, ainsi que son dual D :

$$\begin{array}{rlll} & \min & c^T x & \\ (P) & \text{s.c.} & Ax = b & \\ & & x \geq 0 & \end{array} \qquad \begin{array}{rlll} & \max & b^T \pi & \\ (D) & \text{s.c.} & A^T \pi \leq c. & \end{array}$$

Lorsque P possède beaucoup plus de variables que D, par exemple dans le cadre de la décomposition de Benders (1962) ou de la décomposition de Dantzig-Wolfe (Dantzig et Wolfe, 1960), le problème P est souvent résolu par génération de colonnes. L'algorithme de Kelley (1960) peut alors être appliqué au problème D mais, fréquemment, sa convergence est lente. Ceci reste vrai même si l'algorithme n'a qu'à prouver l'optimalité d'une solution dégénérée de P. En d'autres termes, de nombreuses itérations sont nécessaires pour obtenir une bonne approximation du polyèdre dual défini par les contraintes du problème D. Un premier moyen d'éviter le problème de dégénérescence consiste à perturber P par l'ajout de variables d'écart et de surplus bornées :

$$(P_\epsilon) \qquad \min_{(x,y_-,y_+)\geq 0} \{c^T x : \; Ax - y_- + y_+ = b, \; y_- \leq \epsilon_-, \; y_+ \leq \epsilon_+\}.$$

Un second moyen est d'utiliser, comme en programmation non linéaire, une méthode de pénalités exactes en norme l_1 (Gill *et al.*, 1989) afin de restreindre le domaine de D. Dans ce cas, chercher les solutions de P revient à résoudre le problème suivant,

avec $\delta \geq 0$:

$$(P_\delta) \quad \min_{x \geq 0} c^T x + \delta \|Ax - b\|_1 = \min_{(x, y_-, y_+) \geq 0} \{c^T x + \delta y_- + \delta y_+ : \; Ax - y_- + y_+ = b\}.$$

Cette formulation implique que les variables duales associées aux contraintes d'égalité du problème P_δ doivent se situer dans la boîte $[-\delta e, \delta e]$, où e est un vecteur composé de 1. Si la boîte est centrée sur un vecteur dual autre que l'origine, le problème P_δ correspond à un des problèmes résolus dans le processus itératif de la méthode BOXSTEP de Marsten *et al.* (1975).

De nombreuses approches non linéaires ont été proposées pour éviter le mauvais comportement associé à l'algorithme de Kelley : méthode du Lagrangien augmenté et méthodes de faisceaux (*voir* Hiriart-Urruty et Lemaréchal, 1993), méthodes de plans coupants centraux (*voir* Ye, 1997) telles que la méthode des ellipsoïdes de Shor, la méthode volumétrique, la méthode des centres analytiques. L'originalité de notre approche est de définir un processus de stabilisation qui reste dans le contexte de la programmation linéaire et d'en démontrer l'efficacité. Nous proposons de stabiliser et d'accélérer le processus de résolution par génération de colonnes en combinant la méthode de perturbation et la méthode de pénalités exactes. La section 5.1 présente un problème $P_{\epsilon\delta}$ qui dérive à la fois des problèmes P_ϵ et P_δ. La section 5.2 décrit, dans le contexte de la génération de colonnes, un algorithme qui met à jour les paramètres (ϵ_-, ϵ_+) et (δ_-, δ_+), définis plus bas, dans le but d'obtenir rapidement une solution exacte du problème P. La section 5.3 conclut avec trois applications où l'utilisation du processus de stabilisation permet de résoudre des cas difficiles beaucoup plus rapidement et rend possible la résolution de cas d'une taille beaucoup plus grande que ceux rapportés jusqu'à maintenant dans la littérature.

5.1 Problème $P_{\epsilon\delta}$

Soit le problème primal $P_{\epsilon\delta}$ et son dual $D_{\epsilon\delta}$:

$$
\begin{array}{rll|rll}
 & \min & c^T\tilde{x} - \delta_-^T y_- + \delta_+^T y_+ & & \max & b^T\tilde{\pi} - \epsilon_-^T w_- - \epsilon_+^T w_+ \\
(P_{\epsilon\delta}) & \text{s.c.} & A\tilde{x} - y_- + y_+ = b & (D_{\epsilon\delta}) & \text{s.c.} & A^T\tilde{\pi} \leq c \\
 & & y_- \leq \epsilon_- & & & -\tilde{\pi} - w_- \leq -\delta_- \\
 & & y_+ \leq \epsilon_+ & & & \tilde{\pi} - w_+ \leq \delta_+ \\
 & & \tilde{x}, y_-, y_+ \geq 0 & & & w_-, w_+ \geq 0.
\end{array}
$$

Dans le problème primal $P_{\epsilon\delta}$, y_- et y_+ sont respectivement des vecteurs de variables de surplus et d'écart, avec bornes supérieures ϵ_- et ϵ_+. Ces variables sont respectivement pénalisées dans la fonction objectif par les vecteurs δ_- et δ_+. Dans le problème dual $D_{\epsilon\delta}$, les deux dernières contraintes peuvent être récrites comme $\delta_- - w_- \leq \tilde{\pi} \leq \delta_+ + w_+$, ce qui revient à pénaliser les variables duales $\tilde{\pi}$ quand elles prennent des valeurs hors de l'intervalle $[\delta_-, \delta_+]$. La figure 5.1 illustre la valeur de la pénalité, en fonction d'une variable $\tilde{\pi}$ donnée, qui s'ajoute à l'objectif du problème D pour composer l'objectif du problème $D_{\epsilon\delta}$. Sur cette figure, les flèches indiquent le sens croissant (en particulier, la variable w_- croît lorsque la variable $\tilde{\pi}$ décroît). Dans la méthode BOXSTEP, $\epsilon_- = \epsilon_+ = \infty$ et les variables duales doivent toujours prendre des valeurs dans l'intervalle $[\delta_-, \delta_+]$. Dans le modèle ci-dessus, même si la solution duale optimale de P est loin de l'intervalle $[\delta_-, \delta_+]$, le fait de choisir les paramètres ϵ_- et ϵ_+ suffisamment petits permet au problème $P_{\epsilon\delta}$ de posséder des solutions primales et duales proches de celles du problème P.

Dénotons par x^*, π^*, $(\tilde{x}^*, y_-^*, y_+^*)$ et $(\tilde{\pi}^*, w_-^*, w_+^*)$ des solutions optimales des problèmes P, D, $P_{\epsilon\delta}$ et $D_{\epsilon\delta}$ respectivement, et par $v(\cdot)$ la valeur d'une solution optimale du problème $(\cdot)$. Alors $P \equiv P_{\epsilon\delta}$ (*i.e.*, $y_-^* = y_+^* = 0$) si l'une des deux conditions suivantes est satisfaite : (i) $\epsilon_- = \epsilon_+ = 0$, (ii) $\delta_- < \tilde{\pi}^* < \delta_+$. De plus, (iii) $v(P_{\epsilon\delta}) \leq b^T\tilde{\pi}^* \leq v(P)$. Les conditions (i) et (ii) peuvent servir de critères d'arrêt

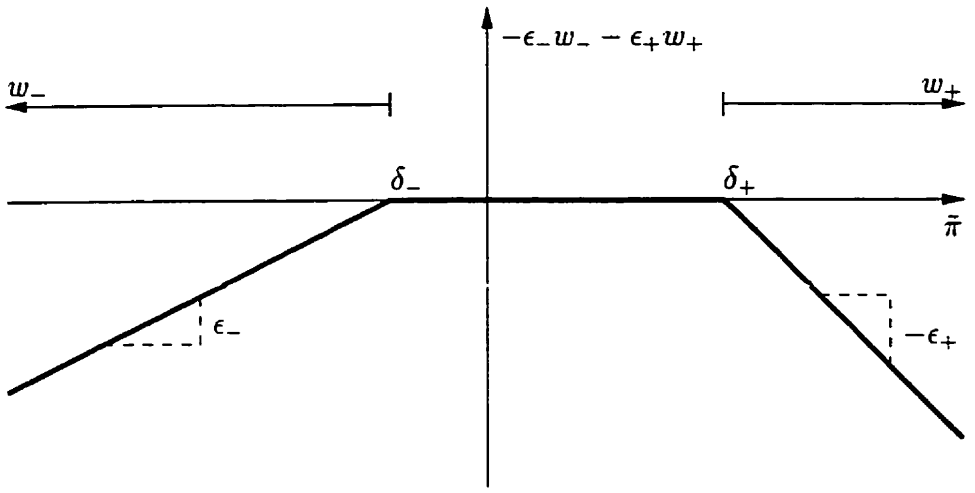


Figure 5.1 – Pénalité dans l'objectif du problème $D_{\epsilon\delta}$ pour une variable duale $\bar{\pi}$ donnée

dans l'algorithme décrit à la section suivante. L'inégalité (iii) montre que lorsque $P_{\epsilon\delta}$ est une relaxation de P, $b^T\bar{\pi}^*$ peut être une meilleure borne inférieure que $v(P_{\epsilon\delta})$ sur $v(P)$. Ce résultat est particulièrement intéressant lorsque l'algorithme de génération de colonnes stabilisé est inscrit dans une procédure de séparation et évaluation pour résoudre des programmes linéaires mixtes en nombres entiers.

5.2 Algorithme stabilisé

Soit le programme linéaire P et son dual D tels que définis dans l'introduction de ce chapitre. Le problème P peut être résolu par la méthode classique de génération de colonnes présentée en partie gauche du tableau 5.1. À l'itération k, un programme linéaire restreint est donné par P^k : $\min\{c^{kT}x^k : A^kx^k = b,\ x^k \geq 0\}$ et son dual par D^k : $\max\{b^T\pi^k : A^{kT}\pi^k \leq c^k\}$. On dispose de deux procédures :

- $(x^k; \pi^k) \leftarrow$ `optimiseur`(P^k), où x^k et π^k sont les solutions optimales de P^k et D^k respectivement ;

- $(\hat{a};\hat{c}) \leftarrow \texttt{oracle}(\pi^k)$, où $\hat{a}$ est une colonne de A et $\hat{c}$ la composante correspondante dans c, telles que le coût réduit $\hat{c} - \hat{a}^T\pi^k$ soit minimum sur toutes les colonnes de A.

Tableau 5.1 - Algorithmes génériques de génération de colonnes

Initialisation : A^0, c^0	Initialisation : $A^0, c^0, \delta^0, \epsilon^0$
Hypothèses : P^0 réalisable et borné	Hypothèses : $P^0_{\epsilon\delta}$ réalisable et borné
$k \leftarrow 0$, fin $\leftarrow$ faux	$k \leftarrow 0$, fin $\leftarrow$ faux
FAIRE	FAIRE
$(x^k;\pi^k) \leftarrow \texttt{optimiseur}(P^k)$	$(\tilde{x}^k, y_-, y_+; \tilde{\pi}^k, w_-, w_+) \leftarrow \texttt{optimiseur}(P^k_{\epsilon\delta})$
$(\hat{a};\hat{c}) \leftarrow \texttt{oracle}(\pi^k)$	$(\hat{a};\hat{c}) \leftarrow \texttt{oracle}(\tilde{\pi}^k)$
SI $\hat{a}^T\pi^k \leq \hat{c}$	SI $\hat{a}^T\tilde{\pi} \leq \hat{c}$ et $y_- = y_+ = 0$
$x^* \leftarrow x^k$, fin $\leftarrow$ vrai	$x^* \leftarrow \tilde{x}^k$, fin $\leftarrow$ vrai
SINON	SINON
$(A^{k+1};c^{k+1}) \leftarrow (A^k,\hat{a};c^k,\hat{c})$	$(A^{k+1};c^{k+1}) \leftarrow (A^k,\hat{a};c^k,\hat{c})$
	$\delta^{k+1} \leftarrow \texttt{mise-à-jour-}\delta(k)$
	$\epsilon^{k+1} \leftarrow \texttt{mise-à-jour-}\epsilon(k)$
$k \leftarrow k+1$	$k \leftarrow k+1$
TANT QUE $\neg$ fin	TANT QUE $\neg$ fin

L'utilisation de la perturbation et des pénalités stabilise le processus de résolution dans l'espace dual et mène la plupart du temps à une réduction du nombre d'itérations nécessaires pour obtenir une solution optimale de P. À l'itération k, nous définissons le programme linéaire restreint $P^k_{\epsilon\delta}$: $\min\{c^{kT}\tilde{x}^k - \delta^{kT}_- y_- + \delta^{kT}_+ y_+ : A^k\tilde{x}^k - y_- + y_+ = b,\ \tilde{x}^k \geq 0,\ 0 \leq y_- \leq \epsilon^k_-,\ 0 \leq y_+ \leq \epsilon^k_+\}$ et son dual $D^k_{\epsilon\delta}$. En plus des procédures `optimiseur` et `oracle` définies plus haut, on dispose de deux procédures supplémentaires :

- $\delta^{k+1} \leftarrow \texttt{mise-à-jour-}\delta(k)$ et
- $\epsilon^{k+1} \leftarrow \texttt{mise-à-jour-}\epsilon(k)$,

où le paramètre k dans $\delta(k)$ et $\epsilon(k)$ réfère au processus complet de résolution jusqu'à l'itération k. Par exemple, les procédures de mise à jour peuvent utiliser les solutions primales et duales des problèmes restreints résolus auparavant. Un algorithme générique est présenté en partie droite du tableau 5.1. Cet algorithme, qui peut être interprété comme une méthode de faisceaux en norme l_1, diffère de l'algorithme classique par sa capacité de stabiliser les variables duales $\bar{\pi}^k$ au moyen de pénalités linéaires $(\epsilon_-^k, \epsilon_+^k)$ qui deviennent actives hors de l'intervalle $[\delta_-^k, \delta_+^k]$. Évidemment, il est capital de développer des stratégies efficaces pour ajuster ces paramètres.

Une stratégie naturelle pour mettre à jour δ_- et δ_+ consiste à utiliser la valeur de la solution duale courante $\delta_-^{k+1} = \delta_+^{k+1} = \bar{\pi}^k$ ou à utiliser la variante $\delta_-^{k+1} + \xi = \delta_+^{k+1} - \xi = \bar{\pi}^k$ avec $\xi > 0$, prenant en compte l'incertitude sur l'approximation de π^* par $\bar{\pi}^k$. Toutefois, s'il est possible d'estimer la qualité de $\bar{\pi}^k$ comme approximation de π^*, la mise à jour n'est effectuée que lorsque $\bar{\pi}^k$ est la meilleure estimation connue de π^*. Par exemple, lorsque le problème P est obtenu par la décomposition de Dantzig-Wolfe, il a la structure suivante :

$$\min\left\{\sum_{i=1}^{p} c_i^T x_i : \sum_{i=1}^{p} A_i x_i = b_0,\ e_i^T x_i = b_i,\ x_i \geq 0,\ i \in \{1, \ldots, p\}\right\},$$

où e_i est un vecteur composé de 1 de même dimension que x_i. Une estimation de la qualité de $\bar{\pi}^k$ peut se faire en calculant la valeur d'une borne inférieure sur P (*voir* Lasdon, 1970) :

$$v(P; \bar{\pi}^k) = b_0^T \bar{\pi}_0^k + \sum_{i=1}^{p} b_i(\hat{c}_i - \hat{a}_i^T \bar{\pi}_0^k),$$

où $\bar{\pi}_0^k$ est un vecteur de composantes de $\bar{\pi}^k$ associées au premier ensemble de contraintes $\sum_{i=1}^{p} A_i x_i = b_0$. Dans le cas où il existe plusieurs contraintes de convexité (*i.e.*, $p > 1$), l'oracle doit retourner une paire (*colonne*, *coût*) pour chaque ensemble i $(1 \leq i \leq p)$ afin de calculer la borne inférieure $v(P; \bar{\pi}^k)$.

Une stratégie naturelle pour mettre à jour ϵ_- et ϵ_+ consiste à diminuer toutes les composantes de ces paramètres si $\tilde{\pi}^k$ est la meilleure estimation connue de π^* ou si la colonne retournée par l'oracle est de coût réduit non négatif, et d'augmenter toutes les composantes dans le cas contraire.

La mise à jour des deux ensembles de paramètres doit assurer la convergence finie de l'algorithme. Les deux stratégies suivantes ont cette propriété :

- Après un nombre donné d'itérations, diminuer ϵ_- et ϵ_+ de façon qu'ils s'annulent en un nombre fini d'itérations, auquel cas la condition (i) de la section 5.1 est satisfaite.
- Après un nombre donné d'itérations, mettre à jour δ_- et δ_+, avec $\xi > 0$, seulement si la colonne retournée par l'oracle est de coût réduit non négatif, comme dans la méthode BOXSTEP. Alors, $\tilde{\pi}^k$ est réalisable pour D et $v(D^k_{\epsilon\delta}) \leq v(D)$. Si $v(D^k_{\epsilon\delta}) = v(D)$, la condition (ii) de la section 5.1 est satisfaite après la mise à jour. Sinon, $\xi > 0 \Rightarrow v(D^k_{\epsilon\delta}) < v(D^{k+t}_{\epsilon\delta})$, avec $t > 0$ dénotant le nombre d'itérations nécessaire pour que l'oracle retourne une colonne de coût réduit non négatif après la mise à jour. La condition (ii) est ainsi satisfaite après un nombre fini d'itérations, puisque le problème D est réalisable et borné par hypothèse.

Finalement, même si les paramètres (δ^0_-, δ^0_+) peuvent prendre des valeurs quelconques (sujet à $\delta^0_- \leq \delta^0_+$), une bonne estimation des variables duales optimales π^* devrait être préférable à des valeurs arbitraires.

5.3 Applications

Le processus de stabilisation a été utilisé sur le problème de confection de rotations d'équipages en transport aérien, sur le problème de localisation de Weber multi-sources et sur le problème de la p-médiane. Chacune de ces applications est décrite

dans les paragraphes suivants, avec les différentes stratégies d'initialisation et de mise à jour qui permettent d'obtenir une efficacité accrue du processus de résolution.

Confection de rotations d'équipage : Dans cette application de programmation linéaire en nombres entiers, les contraintes représentent les segments de vol à affecter aux équipages et les variables représentent les rotations réalisables pour ces équipages. Ce problème peut être formulé comme un problème de partitionnement. Un algorithme de génération de colonnes est utilisé pour calculer des bornes inférieures dans une procédure de séparation et évaluation. L'oracle est un problème de plus court chemin avec contraintes de ressources (Desaulniers *et al.*, 1998). Une estimation précise des variables duales optimales est difficile à obtenir mais une bonne approximation de la valeur optimale de la fonction objectif permet de calculer une valeur moyenne pour chaque variable duale. L'algorithme classique de génération de colonne converge très lentement, la dégénérescence étant présente à deux niveaux : lors de la résolution du programme linéaire courant et aussi durant plusieurs itérations majeures successives où l'ajout de nouvelles colonnes ne suffit pas à modifier la valeur de la fonction objectif. En pratique, l'utilisation de l'algorithme de génération de colonnes stabilisé réduit les temps de résolution d'un facteur variant de 2 à 10.

Tableau 5.2 - Problème de confection de rotations d'équipages—variations sur δ^0

Paramètres (δ^0_-, δ^0_+)	Itérations majeures	Ratio du temps CPU `optimiseur/oracle`	Temps CPU (secondes)	Facteur d'accélération
$(-\infty, \infty)$	433	1,037	1491,0	1,00
$[-50, \infty)$	440	1,444	1985,3	0,75
$[0, 100]$	157	0,521	267,6	5,57
$[50, 100]$	130	0,301	201,2	7,41

Afin d'illustrer le processus de stabilisation, nous avons résolu un cas correspondant à la relaxation linéaire d'un problème de 986 segments de vol d'un transporteur régional (Desaulniers *et al.*, 1999) : le nombre d'itérations de génération de colonnes

est réduit par un facteur 3,33 et le temps CPU, par un facteur 7,41. D'une part, la stratégie pour la procédure `mise-à-jour-`δ est de mettre à jour les paramètres vectoriels (δ_-, δ_+) lorsque l'algorithme de génération de colonnes stagne, en utilisant la séquence prédéfinie suivante d'intervalles emboîtés : $[50, 100]$, $[0, 100]$, $[-50, \infty)$ et $(-\infty, \infty)$. Le dernier intervalle n'impose aucune contrainte sur les variables duales et correspond à la formulation originale du problème de partitionnement ; le premier intervalle provient d'une estimation grossière des variables duales optimales, la solution optimale valant environ 100 000. D'autre part, les paramètres vectoriels ϵ_- et ϵ_+ sont respectivement choisis à partir d'une distribution uniforme dans les intervalles $[9, 11]$ et $[0, 10^{-4}]$, et restent fixes pour toute la durée du processus de résolution. Le premier intervalle, contrôlant le choix de ϵ_-, permet de trop couvrir certains segments de vol et simule une formulation de problème de recouvrement, alors que le second intervalle, contrôlant le choix de ϵ_+, constitue une petite perturbation.

Pour différentes valeurs initiales de (δ_-^0, δ_+^0) prises dans la séquence d'intervalles emboîtés mentionnés plus haut, le tableau 5.2 montre les résultats obtenus sur une station de travail SUN Ultra SPARC 1300 : le nombre d'itérations majeures, le ratio du temps CPU de la procédure `optimiseur` sur le temps CPU de la procédure `oracle`, le temps CPU total et le facteur d'accélération par rapport au temps requis pour résoudre la formulation du problème de partionnement sans stabilisation. Le meilleur temps de résolution est obtenu en utilisant tous les intervalles de la séquence, ce qui confirme la pertinence du processus de stabilisation. L'anomalie observée dans le cas de l'intervalle $[-50, \infty)$ peut être expliquée par sa redondance par rapport à l'intervalle $(-\infty, \infty)$ et au désavantage numérique d'avoir un borne inférieure différente de 0 ou de $-\infty$. Cependant, l'intervalle $[-50, \infty)$ s'est montré efficace en tant que transition entre les intervalles $[0, 100]$ et $(-\infty, \infty)$.

Problème de localisation de Weber multi-sources : Ce problème est un problème classique en théorie de la localisation continue. Il consiste à localiser simul-

tanément p points, ou facilités, dans le plan Euclidien et à allouer n clients dont la position est fixée à la facilité la plus proche de chacun d'eux de façon à minimiser la somme pondérée des distances parcourues, calculées selon la norme Euclidienne $\|\cdot\|_2$. Soit U l'ensemble des n clients, L l'ensemble des p facilités à localiser dans le plan et w_i un poids associé au client $i \in U$. Le problème de Weber multi-sources peut être formulé comme le problème d'optimisation globale suivant (*voir* Krau, 1997) :

$$\min \sum_{i \in U} \sum_{j \in L} w_i x_{ij} \|y_j - a_i\|_2$$

sujet à :

$$\begin{aligned} \sum_{j \in L} x_{ij} &= 1 && \forall i \in U && (5.1) \\ y_j &\in \mathbb{R}^2 && \forall j \in L \\ x_{ij} &\in \{0,1\} && \forall i \in U, \forall j \in L, \end{aligned}$$

où $a_i \in \mathbb{R}^2$ correspond à la position fixe du client $i \in U$, $y_j \in \mathbb{R}^2$ représente la paire de variables donnant la position de la facilité $j \in L$ et x_{ij} est la variable binaire déterminant si le client i est desservi par la facilité j. Les contraintes (5.1) imposent que chaque client i soit associé à une facilité j unique.

Le cas $p = 2$ peut être résolu par des algorithmes d'optimisation pour programmes exprimés comme une différence de fonctions convexes (Chen *et al.*, 1998) mais seuls les petits cas peuvent être résolus optimalement pour $p \geq 3$ (*i.e.*, $n \leq 50$ pour $p = 3$ (Chen *et al.*, 1998), $n \leq 30$ pour $p = 4, 5$ et $n = 25$ pour $p = 6$ (Rosing, 1992)).

Le problème peut aussi s'exprimer comme un problème de partitionnement. Les colonnes sont associées à tous les sous-ensembles possibles de clients, leur coût provenant de la solution d'un problème de Weber à une seule source formulé sur le sous-ensemble de clients correspondant. Un algorithme de génération de colonnes

est utilisé pour calculer des bornes inférieures dans une procédure de séparation et évaluation. L'oracle est un problème de Weber à une seule source avec contraintes de distance (Drezner *et al.*, 1991). Ces contraintes sont dérivées des variables duales des contraintes (5.1) et imposent un coût constant pour desservir un client i lorsque la distance entre la source et le point a_i excède une certaine valeur. Krau (1997) utilise d'abord l'approche de Chen *et al.* (1998) par différence de fonctions convexes pour résoudre l'oracle et traite des problèmes de Weber multi-sources avec $n \leq 150$ et $p \leq 50$. Krau (1997) propose ensuite une variante de l'algorithme BSSS (*Big Square-Small Square*) de Hansen *et al.* (1985) pour résoudre l'oracle. Cet algorithme est une procédure d'énumération partitionnant itérativement le carré de taille minimale contenant tous les points $a_i \in \mathbb{R}^2$ associés aux clients $i \in U$. La procédure originale divise en quatre parties égales un carré donné, calcule pour chaque nouveau carré des bornes inférieure et supérieure sur l'objectif de l'oracle, élimine les carrés dont la borne inférieure excède la valeur de la meilleure solution déjà trouvée et choisit un nouveau carré à analyser jusqu'à l'atteinte d'un degré de précision donné sur la position de la solution. En calculant des bornes plus serrées pour chaque carré et en choisissant les carrés selon une stratégie meilleur d'abord, Krau (1997) réussit à résoudre des problèmes de Weber multi-sources avec $n \leq 287$ et $p \leq 100$.

L'algorithme de génération de colonnes stabilisé permet la résolution de cas avec $n \leq 1060$ et $p \leq 100$ en utilisant les stratégies suivantes. Une solution heuristique, souvent optimale ou très près de la solution optimale, est obtenue dans un premier temps par une heuristique de type VNS (*Variable Neighborhood Search*) (Brimberg *et al.*, 1997 ; Hansen et Mladenović, 1998b ; Mladenović et Hansen, 1997). Pour les cas de grande taille, Brimberg *et al.* (1997) concluent que la structure de voisinage donnant les meilleurs résultats correspond à celle utilisée pour le problème de la p-médiane, décrite plus loin. Étant donné une solution heuristique du problème de Weber multi-sources, dénotons par c_1, c_2, ..., c_p les ensembles d'indices des sous-ensembles de

clients et $f(c_1)$, $f(c_2)$, ..., $f(c_p)$ leur valeur. Alors, pour $l \in c_j$ donné avec $j \in L$, des intervalles initiaux pour les variables duales sont définis par

$$\delta^0_{l-} = f(c_j) - f(c_j \setminus \{l\}) \qquad \text{et} \qquad \delta^0_{l+} = \min_{j' \neq j} \Big(f(c_{j'} \cup \{l\}) - f(c_{j'}) \Big).$$

D'une part, la stratégie pour la procédure `mise-à-jour-`δ se divise en deux parties :

1. fixer les paramètres vectoriels δ^k_- et δ^k_+ autour de $\bar{\pi}^k$ si $\bar{\pi}^k$ est la meilleure solution duale trouvée depuis le début du processus de résolution, tout en préservant les différences individuelles entre les composantes correspondantes de δ^{k-1}_- et δ^{k-1}_+ ;
2. dans le cas contraire ($\bar{\pi}^k$ n'est pas la meilleure solution duale ...) fixer les paramètres vectoriels δ^k_- et δ^k_+ autour de la même solution duale qu'à l'itération $k-1$, et doubler les différences entre les composantes correspondantes de δ^{k-1}_- et δ^{k-1}_+ pour les intervalles qui sont trop serrés[1] (*i.e.*, lorsque les contraintes associées sont actives dans la solution de $D_{\epsilon\delta}$).

D'autre part, les paramètres vectoriels ϵ^0_- et ϵ^0_+ sont initialisés à la valeur du membre de droite de la formulation du problème de partitionnement, *i.e.*, vecteur unité. La stratégie pour la procédure `mise-à-jour-`ϵ est de modifier ϵ_- et ϵ_+ seulement quand la colonne retournée par l'oracle a un coût réduit non négatif, en divisant chaque composante par 2^t où t est le nombre de fois que de telles mises à jour ont été effectuées au cours de tout le processus de résolution. Le tableau 5.3 présente les résultats obtenus sur une station de travail SUN SPARC 10.

Problème de la p-médiane : Ce problème est un problème classique en théorie de la localisation discrète. Il est équivalent au problème de localisation de Weber multi-sources avec la contrainte supplémentaire que les p points à localiser doivent

[1] Cette mise à jour de la largeur des intervalles n'arrive que très rarement puisque la solution heuristique est très proche de la solution optimale.

Tableau 5.3 – Résultats sur des problèmes de Weber multi-sources avec $n = 1060$

Nombre de sources (p)	Valeur optimale	temps CPU (secondes)	Itérations majeures	Nœuds de branchement
10	1 249 564	18 283	824	0
40	529 660	44 470	265	10
50	453 109	11 740	304	0
100	282 536	8 314	161	0

être choisis parmi un ensemble discret de points. Soit U l'ensemble des n clients, L l'ensemble des m points parmi lesquels doivent être choisis exactement p points pour desservir les clients et $D = (d_{ij})$ une matrice de coûts $n \times m$ avec $(i, j) \in U \times L$. Le problème de la p-médiane peut se formuler comme un programme linéaire en variables entières (*voir* Hansen et Mladenović, 1998a) :

$$\min \sum_{i \in U} \sum_{j \in L} d_{ij} x_{ij} \tag{5.2}$$

sujet à :

$$\sum_{j \in L} x_{ij} = 1 \qquad \forall i \in U \tag{5.3}$$

$$x_{ij} \leq y_j \qquad \forall i \in U, \forall j \in L \tag{5.4}$$

$$\sum_{j \in L} y_j = p \tag{5.5}$$

$$x_{ij}, y_j \in \{0, 1\} \qquad \forall i \in U, \forall j \in L, \tag{5.6}$$

où x_{ij} est une variable binaire prenant la valeur 1 si le client i est desservi par le point j et où y_j est une variable binaire prenant la valeur 1 si le point j est un point de service choisi (*ouvert*) parmi les m points de l'ensemble L. Les contraintes (5.3) imposent que chaque client i soit desservi par un point j unique. Les contraintes (5.4) assurent qu'un client i ne peut être desservi que par un point de service j ouvert. La contrainte (5.5) fixe le nombre de points de service à ouvrir.

Christofides et Beasley (1982) proposent une approche de résolution par séparation et évaluation. Les bornes inférieures sont calculées au moyen d'un algorithme de sous-gradients appliqué à la relaxation Lagrangienne du problème (5.2)–(5.6). Ils considèrent la relaxation des contraintes (5.4) agrégées par point $j \in L$ ou des contraintes (5.3), cette dernière approche donnant les meilleurs résultats. Beasley (1985) raffine l'algorithme de sous-gradients et utilise un super-ordinateur pour résoudre optimalement les plus grands cas rapportés dans la littérature, dont un cas avec $n = 500$ et $p = 167$ et un cas avec $n = 900$ et $p = 90$.

Tableau 5.4 – Résultats sur des problèmes de p-médiane avec $n = 3038$

Nombre de sources (p)	Valeur de la relaxation	Meilleure solution entière	Temps CPU (heures)	Gap d'intégrité (%)
10	1 213 082,03	1 213 082,02	15,05	0
40	571 878,43	572 032,42	4,46	0,000 269
50	507 418,80	507 743,64	7,62	0,000 639
100	352 494,07	354 433,99	7,61	0,005 473

Un algorithme de génération de colonnes pour le problème (5.2)–(5.6), avec un oracle très simple dérivé des contraintes (5.4) et (5.6), a été proposé par Garfinkel *et al.* (1974) mais n'a pu être utilisé que pour de petits cas ($n \leq 24$ et $p \leq 16$). Le processus de stabilisation couplé à une heuristique de type VNS (Hansen et Mladenović, 1998a) a permis de reprendre cette formulation pour résoudre la relaxation continue de cas avec $n = 3038$ et $p \leq 100$. Pour le problème de la p-médiane, Hansen et Mladenović (1998a) obtiennent des minima locaux au moyen d'une heuristique efficace pour introduire un nouveau point de service dans la solution courante et identifier le point de service devant être remplacé. L'heuristique VNS est alors utilisée pour explorer des voisinages de plus en plus éloignés de la solution courante, chaque voisinage étant identifié par un entier $k \in [1, p]$ correspondant à l'ensemble de solutions dont le nombre de variables y_j prennent des valeurs différentes de celles de la solution courante. Les auteurs montrent que cette heuristique produit des solutions de meilleur

coût que les autres heuristiques auxquelles ils se comparent et que ces solutions sont souvent optimales (en comparant avec les résultats obtenus par Beasley (1985)). Les stratégies de stabilisation décrites pour le problème de Weber multi-sources sont reprises pour initialiser les paramètres δ_-^0, δ_+^0, ϵ_-^0 et ϵ_+^0 de même que pour définir les procédures `mise-à-jour-`δ et `mise-à-jour-`ϵ. Le tableau 5.4 présente les résultats obtenus sur une station de travail SUN SPARC 20 ; le problème avec $p = 10$ est résolu optimalement alors que, pour les autres valeurs de p, les solutions sont pratiquement optimales (gaps d'intégrité très faibles).

Conclusion

Dans chacune des applications précédentes, le processus de stabilisation a rendu possible soit une réduction significative des temps de résolution, soit la résolution de cas considérés de trop grande taille jusqu'à tout récemment. Étant donné ce succès, il semble que l'algorithme de génération de colonnes stabilisé présenté dans ce chapitre ait un excellent potentiel pour améliorer la résolution de plusieurs problèmes similaires. Cette conclusion vaut spécialement pour les problèmes massivement dégénérés pour lesquels il existe des heuristiques efficaces permettant d'obtenir de bonnes solutions primales et duales.

Chapitre 6

Agrégation de contraintes

L'évolution du modèle unifié est la conséquence de deux axes principaux d'utilisation. D'une part, certaines applications nécessitent plus de flexibilité au niveau de la modélisation. Cette tendance mène à une formulation plus générale, telle qu'illustrée au tableau 1.1, incluant les contraintes liantes H (1.3) et les fonctions d'extension f_{ij}^{kr} (1.9). Les travaux de recherche qui élargissent le domaine du modèle portent souvent sur des problèmes de petite taille, puisque les nouveaux algorithmes sont en général plus complexes. D'autre part, certaines applications visent la résolution de problèmes de très grande taille. Cette tendance provient souvent du désir d'obtenir des solutions plus « globalement » optimales. Elle constitue aussi une alternative, par discrétisation ou énumération, à une formulation plus générale pour laquelle les algorithmes n'existent pas ou sont inefficaces. La difficulté de ces problèmes réside alors dans le nombre d'objets qui les définissent, comme la taille, le nombre de commodités, le nombre de ressources ou le nombre de contraintes liantes.

Plusieurs travaux de recherche portent sur les extensions mathématiques du modèle unifié et sur de nouveaux algorithmes pour les traiter : problème de cueillette et livraison avec temps d'attente (PDPTW, Dumas *et al.*, 1991) ; problème de plus court chemin avec coûts linéaires sur les nœuds (SPNC, Ioachim *et al.*, 1998). Les problèmes reliés à la taille des réseaux (nombre de nœuds et d'arcs) relèvent de la modélisation en fonction d'une application spécifique à cause des interactions avec les fonctions d'extension. Nagih et Soumis (1999) étudient ainsi les problèmes reliés

à l'existence d'un grand nombre de variables de ressource dans le contexte de réseaux de très grande taille. Ce chapitre se concentre toutefois sur la résolution efficace du problème-maître dans les cas comportant un très grand nombre (plus de 10 000) de contraintes liantes W (1.2) de type partitionnement. Ce type de difficulté est courant dans les applications de transport aérien et de transport urbain où plusieurs tâches, correspondant chacune à une contrainte de partitionnement, doivent être affectées à des groupes de travail correspondant aux variables. En transport aérien, Hoffman et Padberg (1993) résolvent des problèmes de rotations d'équipage comportant environ 800 tâches, Desaulniers *et al.* (1997a) considèrent des problèmes hebdomadaires allant jusqu'à 1 100 tâches et Dumas (1999) mentionne des problèmes hebdomadaires de plus de 7 000 tâches pour des compagnies comme NORTH WEST et DELTA, de même que des problèmes mensuels d'environ 12 000 tâches pour AIR CANADA. En transport urbain, Löbel (1997) s'attaque à des problèmes comportant plus de 20 000 tâches.

Les contraintes de partitionnement W imposent une forte structure aux solutions admissibles du modèle unifié. La première section montre comment exploiter cette structure dans le contexte, souvent rencontré en pratique, de l'existence d'un petit nombre de variables Y_s complicantes. On y obtient une technique de reformulation fondée sur une partition des contraintes W, permettant de réduire le nombre de coefficients et souvent le nombre de contraintes dans le problème-maître. La deuxième section propose deux algorithmes *statiques* pour construire, lors d'un prétraitement appliqué à un problème résiduel donné, une partition menant à l'agrégation de contraintes selon l'approche décrite à la première section. On développe finalement à la troisième section un algorithme *dynamique* qui intègre la technique de reformulation et l'algorithme de génération de colonnes dans un processus itératif, afin de réduire très agressivement le nombre de coefficients et de contraintes dans le problème-maître et d'accroître significativement l'efficacité du processus de résolution d'un problème résiduel.

6.1 Redondance, partitions et agrégation

Le principe d'agrégation de contraintes est fondé sur l'intuition que seul un petit sous-ensemble des contraintes est nécessaire pour déterminer le domaine des solutions. Dans le contexte du modèle unifié, les contraintes liantes sont exprimées sous la forme d'égalités et les contraintes inutiles correspondent à des contraintes redondantes (par opposition aux contraintes dominées dans le cas de contraintes d'inégalités). Cette section rappelle les conditions selon lesquelles des contraintes sont redondantes entre elles et analyse les possibilités d'application de ces conditions aux formulations originale et décomposée du modèle unifié. L'approche directe menant à un faible taux d'agrégation, on propose une approche alternative impliquant le découplage des variables par une reformulation des contraintes. Cette approche forme le cadre théorique dans lequel les algorithmes des sections suivantes sont ensuite développés.

Approche directe : Les contraintes liantes (1.2)–(1.4) de la formulation originale du modèle unifié, de même que les contraintes (1.17)–(1.21) de la formulation décomposée, peuvent être récrites schématiquement comme :

$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} = \boldsymbol{b} \tag{6.1}$$

$$\boldsymbol{x} \geq \boldsymbol{0}, \boldsymbol{y} \geq \boldsymbol{0}. \tag{6.2}$$

Dans cette formulation condensée, le terme $\boldsymbol{Ax}$ représente les contributions des variables rattachées aux commodités (variables de flot X_{ij}^k et variables de ressource T_i^{kr} dans la formulation originale ; variables de chemin θ_p^k dans la formulation décomposée) alors que le terme $\boldsymbol{By}$ représente les contributions des variables supplémentaires (variables Y_s dans les deux formulations). Dans un tel système, certaines contraintes

sont redondantes si et seulement si il existe un vecteur $\boldsymbol{\lambda} \neq \mathbf{0}$ tel que (Luenberger, 1989) :

$$\boldsymbol{\lambda}^\top \boldsymbol{A} = \mathbf{0}, \quad \boldsymbol{\lambda}^\top \boldsymbol{B} = \mathbf{0} \quad \text{et} \quad \boldsymbol{\lambda}^\top \boldsymbol{b} = 0. \tag{6.3}$$

La formulation originale du tableau 1.1 se prête immédiatement à la recherche d'un vecteur $\boldsymbol{\lambda}$ satisfaisant (6.3). Les difficultés provenant du très grand nombre de variables X_{ij}^k et T_j^{kr} (quelques centaines de milliers de variables) rendent cette recherche coûteuse mais réalisable (Bixby *et al.* (1992) rapportent que les optimiseurs linéaires CPLEX et OB1 permettent de résoudre des problèmes de plus de dix millions de variables). Cependant, les applications du modèle unifié comportent souvent peu de contributions (moins de 5) associées à chaque variable X_{ij}^k et T_j^{kr}. Dans les problèmes traités en pratique, il existe fréquemment une sous-matrice diagonale, surtout dans la matrice $\boldsymbol{B}$ associée aux variables Y_s qui sont souvent utilisées comme variables d'écart et de surplus pour contrôler individuellement la violation des contraintes. La faible espérance d'agrégation qui en découle, plus que la difficulté du problème, rend peu attrayante la recherche de contraintes redondantes dans la formulation originale.

L'approche directe décrite au paragraphe précédent peut aussi s'appliquer à la formulation décomposée du tableau 1.2, en cherchant un vecteur $\boldsymbol{\lambda}$ satisfaisant (6.3) au moyen d'un algorithme de génération de colonnes fondé sur une méthode du simplexe en deux phases. Les contraintes redondantes sont alors détectées lors de la transition entre les deux phases (Bazaraa *et al.*, 1990). L'espérance de trouver des contraintes redondantes est plus élevée que pour la formulation originale puisque le nombre de contributions associées aux variables de chemin θ_p^k correspond à la somme du nombre de contributions provenant des arcs formant ces chemins et varie généralement de 5 à 15 dans les applications visées (par exemple, rotations d'équipage durant de 2 à 5 jours et couvrant de 2 à 4 tâches par jour). Cependant, les difficultés reliées à la structure de la matrice $\boldsymbol{B}$ restent présentes et minent le potentiel de cette approche.

Contraintes de partitionnement : Nous proposons une approche alternative permettant d'une part, de profiter du nombre de contributions par variable relativement élevé de la formulation décomposée et d'autre part, de minimiser l'impact de la structure diagonale qui apparaît souvent dans la matrice $\boldsymbol{B}$. Nous restreignons l'analyse aux contraintes de partitionnement, qui représentent souvent plus de 90% des contraintes liantes des problèmes de très grande taille. Ces contraintes forment un sous-ensemble bien structuré des contraintes W (1.2) de la formulation originale, correspondant aux contraintes liantes $w \in W$ telles que $a_w = 1$ et que $a^k_{wij} \in \{0,1\}$ pour tout arc $(i,j) \in A^k$ de toute commodité $k \in K$. Nous supposons sans perte de généralité que tout arc $(i,j) \in A^k$, pour toute commodité $k \in K$, ne contribue qu'à au plus une contrainte $w \in W$: un arc couvrant plusieurs tâches $w \in W$ peut être remplacé par un chemin comportant autant d'arcs que de tâches. On remarque que les coefficients a^k_{wp} ne sont pas nécessairement binaires malgré la condition analogue sur les coefficient a^k_{wij}. En effet, plusieurs arcs $(i,j) \in A^k$ peuvent avoir une contribution positive à une contrainte $w \in W$ donnée et appartenir à un chemin valide. Cette situation se produit entre autres lorsque le réseau de la commodité k correspond au découpage d'un réseau avec cycles selon la méthode exposée à la section 1.3.1.

L'approche consiste à reformuler les contraintes (6.1), qui impliquent à la fois les matrices $\boldsymbol{A}$ et $\boldsymbol{B}$, en exploitant l'existence d'ensembles de contraintes dont les lignes provenant de la matrice $\boldsymbol{A}$ forment une sous-matrice de rang 1. En pratique, les variables Y_s correspondant à la matrice $\boldsymbol{B}$ servent à compenser les anomalies potentielles dans la structure de la matrice $\boldsymbol{A}$, comme l'existence d'une contrainte $w \in W$ ne comportant que des coefficients $a^k_{wp} = 0$ ou que des coefficients $a^k_{wp} > 1$. Ces anomalies sont peu fréquentes et les variables Y_s contribuant à une contrainte $w \in W$ avec $a_{ws} \neq 0$ prennent donc rarement une valeur différente de 0. Dans un ensemble de contraintes dont les lignes de $\boldsymbol{A}$ forment une sous-matrice de rang 1, les hypothèses que les coefficients $a^k_{wp} \in \mathbb{N}$ et qu'il existe au moins un chemin $p \in P^k$ tel que $a^k_{wp} = 1$

impliquent que toutes les colonnes de cette sous-matrice sont colinéaires avec un vecteur de 1 de dimension appropriée. Ce raisonnement permet de reformuler les contraintes d'un tel ensemble en deux groupes liés par une variable supplémentaire, l'un impliquant les vecteurs colonnes de la matrice $\boldsymbol{A}$, l'autre ceux de la matrice $\boldsymbol{B}$. La colinéarité avec un vecteur de 1 et l'existence d'un chemin $p \in P^k$ tel que $a^k_{wp} = 1$ impliquent à leur tour que, pour un ensemble donné, toutes les lignes de la matrice $\boldsymbol{A}$ sont identiques et non nulles. On peut donc ne conserver qu'une seule de ces lignes et réduire ainsi le nombre de coefficients non nuls de la matrice des contraintes.

Approche par découplage des variables : Nous pouvons maintenant décrire le processus d'agrégation en trois étapes. Dans un premier temps, nous reformulons les contraintes (6.1)–(6.2) en ensembles d'indice $l \in L$ choisis de façon que chaque matrice $\boldsymbol{A}_l$ soit constituée de lignes identiques :

$$\boldsymbol{A}_l\boldsymbol{x} + \boldsymbol{B}_l\boldsymbol{y} = \mathbf{1}_l \qquad \forall l \in L \tag{6.4}$$

$$\boldsymbol{x} \geq \mathbf{0}, \boldsymbol{y} \geq \mathbf{0}, \tag{6.5}$$

où $\mathbf{1}_l$ est un vecteur de 1 de dimension appropriée. Une telle partition comporte, dans le cas général, des singletons et des *blocs* regroupant plusieurs contraintes. Une partition est dite *valide* si la matrice $\boldsymbol{A}_l$ de chaque bloc l est constituée de lignes identiques. Dans un deuxième temps, nous procédons à la reformulation des contraintes (6.4) d'un bloc l donné lorsque la matrice $\boldsymbol{B}_l$ est non nulle (si la matrice $\boldsymbol{B}_l$ est nulle, on passe à la troisième étape directement). La reformulation consiste à récrire les contraintes (6.4) de ce bloc l comme

$$\boldsymbol{A}_l\boldsymbol{x} + \mathbf{1}_l z_l = \mathbf{1}_l \tag{6.6}$$

$$\boldsymbol{B}_l\boldsymbol{y} - \mathbf{1}_l z_l = \mathbf{0}_l, \tag{6.7}$$

où $\mathbf{0}_l$ est un vecteur de 0 de dimension appropriée et z_l est une variable scalaire supplémentaire non contrainte. Le choix d'associer la matrice $\boldsymbol{A}_l$ avec un membre de droite

égal à $\mathbf{1}_l$ en (6.6) (plutôt que $\mathbf{0}_l$) provient du désir de préserver autant que possible la structure originale des contraintes de partitionnement (par exemple, pour permettre l'enchaînement de plusieurs processus d'agrégation dans le contexte de la résolution du modèle unifié au moyen d'un processus d'énumération). Finalement, dans un troisième temps, nous remplaçons par une seule contrainte les contraintes associées à la matrice $\boldsymbol{A}_l$ de chaque bloc l. Cette transformation est valide puisque toutes les lignes associées à cette matrice $\boldsymbol{A}_l$ sont identiques et non nulles par construction des ensembles $l \in L$. On note que la structure de la matrice $\boldsymbol{B}_l$ n'intervient pas dans cette dernière étape du processus d'agrégation.

La reformulation des contraintes (6.4) d'un bloc $l \in L$ en contraintes (6.6)–(6.7) peut entraîner une augmentation du nombre de contraintes par rapport à la formulation initiale (6.1)–(6.2). Cependant, l'agrégation des lignes d'une matrice $\boldsymbol{A}_l$ implique une diminution significative du nombre de coefficients non nuls, permettant ainsi d'espérer une réduction correspondante du temps de calcul consacré à l'optimisation du problème-maître. De plus, lorsqu'une matrice $\boldsymbol{B}_l$ est entièrement caractérisée par une sous-matrice régulière affectant une partie $\boldsymbol{y}_l$ correspondante des variables $\boldsymbol{y}$ et que ces variables $\boldsymbol{y}_l$ ne contribuent qu'aux contraintes du bloc l, on peut substituer $\boldsymbol{y}_l$ par $\boldsymbol{B}_l^{-1}\mathbf{1}_l z_l$, transférer les contraintes de non négativité des variables $\boldsymbol{y}_l$ en bornes sur la variable scalaire z_l et éliminer les contraintes (6.7) pour ce bloc l. Comme cette situation est fréquente pour les applications ciblées dans ce chapitre, où la matrice $\boldsymbol{B}$ correspond souvent à une matrice identité, le processus d'agrégation entraîne plutôt, en pratique, une diminution du nombre de contraintes.

Les sections suivantes portent respectivement sur les agrégations statique et dynamique des contraintes W (1.17) du modèle unifié. Dans ces sections, on suppose que $|K| > 0$, que N et A sont respectivement les ensembles de nœuds et d'arcs de plus grande cardinalité en comparant les réseaux de chaque commodité, que $|W| > 0$ et que W ne contient que des contraintes de partitionnement, sans considérer les contributions des variables Y_s (les contraintes pouvant être modélisées dans l'ensemble W

mais ne satisfaisant pas cette condition sont relocalisées dans l'ensemble H). On utilise le terme *tâche* pour dénoter chacune des contraintes $w \in W$ et on dénote par $\mathcal{W}$ une collection $\{W_l\}_{l \in L}$ de sous-ensembles formant une partition de W.

6.2 Agrégation statique

Le processus d'agrégation défini à la section 6.1 n'est pas constructif, *i.e.*, ne suggère pas de moyens pour obtenir une partition valide. Les deux algorithmes décrits dans cette section construisent des partitions valides et complètent l'exposé sur l'agrégation *statique*, ainsi nommée parce qu'elle constitue un prétraitement et n'interagit pas avec le processus de résolution d'un problème résiduel du modèle unifié.

6.2.1 Partition par génération de colonnes

Le principe général du présent algorithme consiste à raffiner une partition initiale $\mathcal{W}$ de l'ensemble des tâches W afin qu'elle devienne valide selon la définition donnée à la section 6.1. On dit qu'une partition $\mathcal{W}'$ est un *raffinement* d'une autre partition $\mathcal{W}$ lorsque tout élément de $\mathcal{W}'$ est un sous-ensemble d'un des éléments de $\mathcal{W}$. L'algorithme 6.1 présente une variante du processus de raffinement qui tient compte du coût (temps de calcul) relativement élevé de sélectionner et de traiter une commodité $k \in K$ (une autre variante de complexité équivalente s'obtient en intervertissant les deux premiers niveaux de boucle).

L'algorithme analyse les contraintes de partitionnement W un sous-ensemble P^k, $k \in K$, de variables à la fois, ce qui explique la stratégie par raffinements successifs. La boucle de la ligne 2 consiste ainsi à sélectionner le sous-ensemble des variables correspondant aux chemins $p \in P^k$ d'une commodité $k \in K$. À la ligne 3, l'ensemble $\mathcal{W}$

Algorithme 6.1 Algorithme RAFF de partition par raffinements successifs

Antécédent: partition $\mathcal{W}$ de l'ensemble des tâches $\overline{W}$

Conséquent: partition valide $\mathcal{W}'$ étant un raffinement de $\mathcal{W}$

1: $\mathcal{W}' \leftarrow \mathcal{W}$
2: **pour tout** $k \in K$ **effectuer**
3: $\quad \mathcal{W} \leftarrow \mathcal{W}' ; \mathcal{W}' \leftarrow \emptyset$
4: $\quad$ **tant que** $\mathcal{W} \neq \emptyset$ **effectuer**
5: $\quad\quad$ soit $\overline{W} \in \mathcal{W} ; \mathcal{W} \leftarrow \mathcal{W} \setminus \{\overline{W}\}$
6: $\quad\quad$ **tant que** $\overline{W} \neq \emptyset$ **effectuer**
7: $\quad\quad\quad$ soit $\bar{w} \in \overline{W} ; \overline{W}' \leftarrow \{\bar{w}\}$
8: $\quad\quad\quad$ **pour tout** $w \in \overline{W} \setminus \{\bar{w}\}$ **effectuer**
9: $\quad\quad\quad\quad$ **si** $\forall p \in P^k, a^k_{\bar{w}p} = a^k_{wp}$ **alors**
10: $\quad\quad\quad\quad\quad \overline{W}' \leftarrow \overline{W}' \cup \{w\}$
11: $\quad\quad\quad \mathcal{W}' \leftarrow \mathcal{W}' \cup \{\overline{W}'\} ; \overline{W} \leftarrow \overline{W} \setminus \overline{W}'$

prend la valeur de la partition raffinée par suite du traitement des commodités précédentes et l'ensemble $\mathcal{W}'$ est vide. Après le traitement d'une commodité $k \in K$ aux lignes 3–11, l'ensemble $\mathcal{W}'$ contient une version raffinée de la partition $\mathcal{W}$ initiale, tenant compte de toutes les variables d'indice $p \in P^k$ de toutes les commodités déjà traitées à la boucle de la ligne 2, incluant la commodité k.

La boucle de la ligne 4 analyse séparément chaque bloc $\overline{W}$ de la partition $\mathcal{W}$ courante pour déterminer s'il existe des chemins $p \in P^k$ permettant de couvrir un sous-ensemble strict des tâches $w \in \overline{W}$. Si c'est le cas, il faut partitionner le bloc $\overline{W}$ en fonction de ces chemins. Pour ce faire, les lignes 7–10 construisent à partir d'une tâche $\bar{w} \in \overline{W}$ un ensemble $\overline{W}' \subseteq \overline{W}$ de cardinalité maximale respectant la contrainte de partition valide sur tous les chemins $p \in P^k$. Si l'ensemble $\overline{W}$ choisi à la ligne 5 respecte déjà cette contrainte (*i.e.*, tous les chemins $p \in P^k$ couvrent de façon identique toutes les tâches $\bar{w} \in \overline{W}$), nous obtenons que $\overline{W}' = \overline{W}$ à la sortie de la boucle

des lignes 8–10 et l'ensemble $\overline{W}' = \overline{W}$ est réintégré à la ligne 11 à la partition $\mathcal{W}'$ en construction, ce qui termine la boucle des lignes 6–11. Sinon, $\overline{W}' \subset \overline{W}$ et la boucle 6–11 continue avec l'ensemble des tâches résiduelles de $\overline{W}$, qui peut ainsi être partitionné en plusieurs blocs.

Le bon fonctionnement de l'algorithme 6.1 repose sur le parcours, à la ligne 9, de tous les indices $p \in P^k$ de chaque commodité $k \in K$. Dans le contexte de la formulation décomposée du modèle unifié, le test de la ligne 9 doit donc être converti pour devenir compatible avec un processus d'énumération implicite des variables. En transformant par double négation l'expression de la ligne 9, on obtient l'expression équivalente

$$\neg \left[\exists p \in P^k \mid a^k_{\bar{w}p} \neq a^k_{wp}\right], \tag{6.8}$$

dont le test d'existence peut être vérifié en résolvant une version modifiée du sous-problème de la commodité k, tel que décrit au paragraphe suivant.

L'inégalité $a^k_{\bar{w}p} \neq a^k_{wp}$ peut s'écrire sous la forme des deux contraintes disjonctives

$$a^k_{\bar{w}p} - a^k_{wp} \leq -1 \quad \text{ou} \quad a^k_{wp} - a^k_{\bar{w}p} \leq -1 \tag{6.9}$$

puisque les coefficients a^k_{wp} sont tous entiers par définition des contraintes W. Le test d'existence implique donc la résolution de deux sous-problèmes, un pour chaque alternative de la disjonction (6.9). Le problème associé à la deuxième contrainte étant symétrique au premier, la transformation du sous-problème k est décrite pour ce premier problème seulement. Cette transformation consiste à ajouter une ressource supplémentaire δ au problème de plus court chemin avec contraintes, associée à de nouvelles variables $T^{k\delta}_i$ cumulant la différence entre les coefficients $a^k_{\bar{w}p}$ et a^k_{wp}. Les bornes (1.11) sur les nouvelles variables de ressource sont données par

$$[l^{k\delta}_i, u^{k\delta}_i] = \begin{cases} [0,0] & \text{si } i = o^k \\ [-1,-1] & \text{si } i = d^k \\ [-|A^k|, |A^k|] & \text{sinon} \end{cases}$$

et les fonctions d'extension utilisées en (1.9), par

$$f_{ij}^{k\delta}(T_i^k) = T_i^{k\delta} + a_{\bar{w}ij}^k - a_{wij}^k.$$

Comme ces fonctions d'extension sont non décroissantes, la nouvelle ressource est compatible avec l'algorithme de Desrochers (1986) pour la résolution de sous-problèmes de type SPR. L'objectif à utiliser est arbitraire car seule l'existence d'au moins un chemin satisfaisant à la fois les contraintes de la commodité k et les contraintes associées à la ressource δ doit être vérifiée. On discute plus loin d'une stratégie d'accélération qui tire profit de cette indépendance du test d'existence par rapport à l'objectif.

On peut questionner le recours à une ressource supplémentaire pour modéliser les contraintes disjonctives (6.9), étant donné l'hypothèse que les coefficients a_{wij}^k proviennent tous de l'ensemble $\{0, 1\}$. Rappelons que cette condition n'implique pas que les coefficients a_{wp}^k soient limités à l'ensemble $\{0, 1\}$. Cependant, dans les cas où les coefficients a_{wp}^k proviennent toujours de l'ensemble $\{0, 1\}$, il est possible de simplifier la procédure du paragraphe précédent en récrivant les contraintes disjonctives (6.9) sous la forme

$$a_{\bar{w}p}^k = 0, a_{wp}^k = 1 \quad \text{ou} \quad a_{wp}^k = 0, a_{\bar{w}p}^k = 1. \tag{6.10}$$

La discussion qui suit ne s'applique qu'au premier de ces deux problèmes, le second étant symétrique. La contrainte d'égalité $a_{\bar{w}p}^k = 0$ est prise en compte par l'élimination de tous les arcs ayant une contribution positive à la contrainte $\bar{w}$. Le test d'existence de l'expression (6.8) consiste alors à résoudre un sous-problème dont l'objectif incite à la construction d'un chemin contribuant à la contrainte w. On obtient un tel objectif en associant un coût -1 aux arcs contribuant à la contrainte w et un coût 0 aux autres arcs. Si aucun chemin ne contribue à la contrainte w, la première contrainte disjonctive de (6.10) n'est pas satisfaite et la seconde contrainte doit être considérée.

Cette approche alternative spécialisée est préférable à l'approche générale du paragraphe précédent pour deux raisons. Premièrement, la complexité de la résolution du sous-problème de la commodité k est réduite (quelques arcs éliminés) plutôt qu'augmentée (par la présence de la ressource supplémentaire δ). Deuxièmement, l'approche alternative est compatible avec tous les algorithmes de résolution du sous-problème, y compris ceux qui supposent un nombre prédéterminé de ressources (comme l'algorithme de Ioachim *et al.* (1998) qui ne traite que les problèmes avec une seule ressource).

Complexité : La complexité $C_{\text{RAFF}}(K, N, A, W)$ de l'algorithme 6.1 en fonction des caractéristiques du problème provient essentiellement du nombre d'évaluations de la condition de la ligne 9. On note deux possibilités lors du traitement d'une paire de tâches $(\bar{w}, w)$. D'une part, si la condition est vraie, la tâche w n'est plus considérée dans le reste du traitement de la commodité k courante. En effet, la tâche w est alors incluse dans l'ensemble $\overline{W}'$ à la ligne 10, qui est lui-même retranché de l'ensemble $\overline{W}$ à la ligne 11. Par contre, l'ensemble $\overline{W}'$ devra être analysé de nouveau dans le contexte des commodités subséquentes (boucle de la ligne 2), ce qui implique de reconsidérer le regroupement des tâches $(\bar{w}, w)$. D'autre part, si la condition est fausse, le traitement de la paire de tâches $(\bar{w}, w)$ est terminé et cette paire n'est plus considérée par aucune autre commodité. Par contre, la tâche w, n'étant pas intégrée à l'ensemble $\overline{W}'$ à la ligne 10, reste dans l'ensemble $\overline{W}$ à la ligne 11 et peut être traitée à nouveau lors des itérations subséquentes de la boucle de la ligne 6.

Cette asymétrie dans le traitement mène au pire cas suivant : la partition initiale $\mathcal{W}$ ne contient que l'ensemble $\{W\}$, les $|K| - 1$ premières commodités traitées ne raffinent pas la partition et la dernière commodité raffine la partition en singletons. En approchant par $\tau(K)$ le nombre d'opérations élémentaires nécessaires pour résoudre le plus difficile des sous-problèmes $k \in K$, on note la complexité de l'algorithme 6.1

par

$$C_{\text{RAFF}}(K, N, A, W) = O\big((|K| + |W|)|W|\tau(K) + |W|^2\big).$$

Une mise en œuvre naïve de l'algorithme 6.1 pourrait donc nécessiter la résolution d'un nombre quadratique de sous-problèmes en fonction du nombre de tâches.

Cependant, les chemins générés lorsque la condition est fausse contiennent l'information pour traiter plusieurs autres paires de tâches. En effet, un tel chemin « couvrant » $\rho|W|$ tâches, avec $\rho \in (0, 1)$, permet de déduire simultanément une condition fausse pour $\rho(1-\rho)\binom{|W|}{2}$ paires de tâches, où $\binom{|W|}{2} = |W|(|W|-1)/2$ est le nombre total de paires de tâches dans le problème. Soit $T(x)$ le travail à effectuer, en nombre de paires de sous-problèmes à résoudre, pour déterminer que les conditions associées à x paires de tâches sont fausses sachant que les conditions associées aux autres $\binom{|W|}{2} - x$ paires sont fausses. On suppose que les chemins sont générés l'un après l'autre au moyen d'un algorithme de résolution des sous-problèmes qui choisit arbitrairement une solution parmi l'ensemble de toutes les solutions optimales.[1] On observe aussi qu'à cause de la structure des réseaux dans la plupart des applications du modèle unifié, la longueur du plus long chemin en nombre d'arcs est proportionnelle au nombre de tâches et non au nombre total de nœuds et d'arcs. Ces deux conditions justifient l'hypothèse que chaque nouveau chemin couvre un ensemble de $\rho|W|$ tâches indépendant des ensembles couverts par les chemins déjà générés. Dans ces conditions, le nombre x de paires de tâches est réduit en moyenne d'un facteur $\rho(1-\rho)$ à chaque paire de sous-problèmes résolus. On obtient ainsi les équations de récurrence probabilistes suivantes :

$$\begin{aligned} T(x) &= 0 && \text{si } x < 1, \\ T(x) &= 1 + T(h(x)) && \text{si } x \geq 1 \end{aligned}$$

[1] Il est possible de développer un tel algorithme en ajoutant une perturbation de distribution uniforme et de faible amplitude au coût c_{ij}^k des arcs associés aux tâches.

où $h(x)$ est une variable aléatoire dans $[0, x]$, de moyenne $x(1-\rho(1-\rho))$. Ces équations peuvent être analysées dans le contexte probabiliste au moyen des relations proposées par Karp (1994). Cependant, comme on ne recherche que l'ordre de la complexité du travail $T(x)$ à effectuer, on se limite ici au cas déterministe où la variable aléatoire $h(x)$ est remplacée par sa moyenne. On obtient dans ce cas la solution $u(x)$ suivante (Cormen *et al.*, 1990) :

$$\begin{aligned} u(x) &= 0 && \text{si } x < 1, \\ u(x) &= \left\lfloor \frac{-\log x}{\log(1-\rho(1-\rho))} \right\rfloor + 1 && \text{si } x \geq 1. \end{aligned}$$

Cette formule permet de remplacer le terme quadratique $|W|^2\tau(K)$ par l'expression $u\left(\binom{|W|}{2}\right)\tau(K)$. Dans le cas où le rapport ρ est constant, la complexité de l'algorithme 6.1 se réduit à

$$C_{\text{RAFF}}(K, N, A, W) = O\big((|K||W| + \log|W|)\tau(K) + |W|^2\big).$$

Par contre, comme mentionné à la section 6.1, le nombre de tâches par chemin est en pratique relativement constant pour une application donnée. Ce nombre varie entre 5 et 20 selon les applications. Ceci implique que c'est la quantité $\rho' = \rho|W|$, plutôt que le rapport ρ, qui doit être maintenue constante dans l'analyse de complexité. Dans ce cas, le dénominateur de $u(x)$ devient

$$\lim_{|W|\to\infty} \log\left(1 - \frac{\rho'}{|W|} + \frac{\rho'^2}{|W|^2}\right) = \lim_{|W|\to\infty} \log\left(1 - \frac{\rho'}{|W|}\right) = \lim_{|W|\to\infty} \frac{-\rho'}{|W|}$$

et la complexité de l'algorithme 6.1 peut s'écrire

$$C_{\text{RAFF}}(K, N, A, W) = O\big((|K| + \log|W|)|W|\tau(K) + |W|^2\big).$$

Stratégies d'accélération : Malgré la diminution de complexité provenant de l'exploitation de toute l'information contenue dans les chemins générés, l'algorithme 6.1 demeure coûteux. Ce paragraphe propose deux stratégies d'accélération indépendantes. La première possibilité consiste à générer, dans le cadre de l'algorithme 6.1, des chemins couvrant plus de tâches que le nombre moyen de tâches

espéré dans les chemins d'une solution réalisable. Il est possible de favoriser la génération de tels chemins en associant un coût -1 à tous les arcs qui couvrent une tâche. Il faut alors modifier la variante de l'algorithme sans ressource supplémentaire pour associer aux arcs couvrant la tâche w de la paire $(\bar{w}, w)$ un coût $-|W|$ puisque cette variante utilise déjà l'objectif comme moyen d'évaluer le test d'existence de l'expression (6.8). La deuxième possibilité consiste à relaxer certaines contraintes de ressource des sous-problèmes. L'algorithme 6.1 considère alors, à la ligne 9, un sur-ensemble de l'ensemble P^k des chemins valides d'une commodité k. La résolution de chaque sous-problème s'en trouve accélérée à cause de la diminution de la taille de l'espace des états des problèmes SPR consécutive à la réduction du nombre de ressources. Cependant, chaque chemin supplémentaire qui est introduit par la relaxation des contraintes de ressource ajoute de nouvelles contraintes d'égalité à la ligne 9 de l'algorithme, ce qui mène éventuellement à une augmentation de la taille de la partition finale $\mathcal{W}'$ obtenue.

L'intérêt de l'algorithme 6.1 provient principalement de la possibilité d'obtenir une partition finale $\mathcal{W}'$ de taille minimale, permettant d'éliminer *a priori* le maximum de contraintes lors de l'application du processus d'agrégation de la section 6.1. La résolution de nombreux sous-problèmes peut aussi comporter des avantages, en servant d'heuristique pour la génération de colonnes initiales. Cependant, les problèmes de très grande taille comportent souvent des fonctions d'extension qui ne possèdent pas les propriétés requises pour une résolution optimale au moyen des algorithmes connus. En particulier, plusieurs de ces applications utilisent une formulation avec des fonctions d'extension qui ne sont pas non décroissantes en conjonction avec l'algorithme de Desrochers (1986) pour la résolution de problèmes SPR. Il est alors impossible de garantir que l'algorithme 6.1 considère bien un sur-ensemble de chaque ensemble P^k de chaque commodité $k \in K$. Dans ce cas, la partition finale $\mathcal{W}'$ peut ne pas être suffisamment raffinée, menant ainsi à une modification (reformulation)

erronée du problème. La relaxation des contraintes de ressource (1.9) permet de surmonter certaines de ces difficultés, mais la partition finale $\mathcal{W}'$ ainsi obtenue est alors plus influencée par la structure locale des réseaux (par exemple, par le degré des nœuds) que par la structure de l'ensemble des chemins (où certains enchaînements d'arcs respectant la structure des réseaux ne sont pas considérés à cause des contraintes de ressource). Ces remarques mènent au développement, à la prochaine section, d'un algorithme moins ambitieux par rapport à la taille de la partition finale mais tirant profit de la structure locale des réseaux afin d'obtenir une complexité plus faible que celle de l'algorithme 6.1.

6.2.2 Partition par analyse locale des réseaux

Parce que l'algorithme 6.1 ne considère que les chemins valides $p \in P^k$ pour chaque commodité $k \in K$, il offre la possibilité de construire une partition $\mathcal{W}'$ de taille minimale. Pour atteindre ce degré d'optimalité, l'algorithme 6.1 dépend de l'existence d'algorithmes exacts pour résoudre les sous-problèmes. Or, dans les applications de très grande taille, il arrive souvent que les sous-problèmes doivent être résolus de façon heuristique (par exemple, à cause de l'absence de certaines propriétés de l'objectif ou des fonctions d'extension, ou encore à cause de la présence d'un trop grand nombre de ressources). La relaxation des contraintes (1.9) permet d'éviter les difficultés qui y sont rattachées, au prix d'une augmentation importante du nombre de chemins considérés et d'une réduction correspondante de la probabilité d'agréger deux tâches quelconques « éloignées » dans un réseau (la distance étant mesurée en nombre d'arcs). Cette observation constitue le point de départ des développements présentés dans cette section.

On restreint le nouvel algorithme de partition à n'utiliser que la structure des réseaux de chaque commodité $k \in K$, sans tenir compte des contraintes de ressource (1.9).

Le principe de l'algorithme consiste à n'agréger la paire orientée de tâches $(\bar{w}, w)$ que si chaque chemin $p \in P^k$ couvrant l'une de ces deux tâches couvre *immédiatement* l'autre tâche, en respectant l'orientation de la paire. Cette variante du test de la ligne 9 de l'algorithme 6.1, appliquée sur l'ensemble des paires orientées $(\bar{w}, w) \in W \times W$, avec $\bar{w} \neq w$, ne peut être satisfaite que pour un sous-ensemble de toutes les paires acceptées par l'expression originale, ce qui confirme la validité du nouveau test.

La notion de paires orientées de tâches suggère la définition d'un graphe orienté auxiliaire utilisé comme interface entre l'analyse locale des réseaux et la construction d'une partition valide $\mathcal{W}'$ de l'ensemble des tâches W. Dans ce graphe, les nœuds correspondent aux tâches (avec un nœud supplémentaire dénotant à la fois le début et la fin de tous les chemins) et les arcs correspondent aux paires orientées de tâches provenant des chemins valides sur les réseaux des commodités $k \in K$ (dont les contraintes de ressource (1.9) sont relaxées). On dénote par $\Gamma^+(w)$ et $\Gamma^-(w)$ deux fonctions associant à un nœud (tâche) w d'un graphe orienté l'ensemble des successeurs et des prédécesseurs de ce nœud, respectivement. Formellement, la relation d'équivalence qui définit la partition $\mathcal{W}'$ sur le graphe auxiliaire de tâches est la suivante :

$$w_1 \sim w_2 \Leftarrow \left[\Gamma^+(w_1) = \{w_2\} \quad \text{et} \quad \Gamma^-(w_2) = \{w_1\}\right]. \tag{6.11}$$

L'algorithme 6.2 présente la construction d'une partition valide $\mathcal{W}'$ selon cette relation d'équivalence.

Complexité : La construction des ensembles $W^+(i)$ implique au pire cas, pour chaque arc $(i, j) \in A^k$ de la commodité k, la réunion de deux ensembles de tâches dont la cardinalité peut être proportionnelle à $|W|$. La construction des ensembles $W^-(i)$ est similaire. Puisque les réseaux de chaque commodité sont acycliques,

Algorithme 6.2 Algorithme LOC de partition par analyse locale des réseaux

Conséquent: partition valide $\mathcal{W}'$

1: créer un graphe auxiliaire de tâches comportant un nœud par tâche $w \in W$ et un nœud supplémentaire, sans arcs

2: **pour tout** $k \in K$ **effectuer**

3: construire en chaque nœud $i \in N^k$ l'ensemble $W^+(i) \subseteq W$ des tâches qui pourraient succéder immédiatement à la tâche associée au nœud i

4: construire en chaque nœud $i \in N^k$ l'ensemble $W^-(i) \subseteq W$ des tâches qui pourraient précéder immédiatement à la tâche associée au nœud i

5: ajouter les arcs $W^-(i) \times W^+(i)$, $i \in N^k$, au graphe de tâches

6: créer la partition $\mathcal{W}'$ à partir de la relation d'équivalence (6.11)

il est possible de ne traiter chaque arc qu'une seule fois selon une variante de l'algorithme du tri topologique, pour une complexité de l'ordre de $O(|K||A||W|)$ opérations. L'étape du produit cartésien des ensembles $W^+(i)$ et $W^-(i)$ peut nécessiter jusqu'à $O(|K||N||W|^2)$ opérations, à cause du pire cas concernant la cardinalité des ensembles impliqués. Le graphe auxiliaire de tâches peut être construit implicitement, en ne conservant que les arcs définissant les classes d'équivalence. Cette représentation permet de créer la partition $\mathcal{W}'$ à partir de la relation d'équivalence en $O(|W|)$ opérations. La complexité globale de l'algorithme 6.2 est donc

$$C_{\text{LOC}}(K, N, A, W) = O(|K||A||W| + |K||N||W|^2 + |W|).$$

Cette expression ne dépend pas de la complexité de résoudre les sous-problèmes mais suggère que l'algorithme 6.2 peut parfois être très coûteux en temps de calcul. Le reste de cette section présente deux stratégies d'accélération et les résultats obtenus sur quelques applications du modèle unifié.

Stratégies d'accélération : La première stratégie d'accélération découle d'une modification du réseau de chaque commodité $k \in K$ qui n'affecte pas la validité

de l'algorithme 6.2. Cette modification consiste à ajouter des nœuds et des arcs au réseau d'une commodité, opération qui ne peut qu'augmenter le nombre d'arcs dans le graphe auxiliaire de tâches et donc augmenter le nombre de classes d'équivalence et la taille de la partition W'. Si l'on ajoute au réseau de chaque commodité tous les nœuds et les arcs des autres commodités, il est alors possible de ne traiter qu'une seule de ces commodités augmentées. Cette stratégie comporte un avantage seulement si $|\cup_{k\in K} N^k| \ll \sum_{k\in K} |N^k|$ et $|\cup_{k\in K} A^k| \ll \sum_{k\in K} |A^k|$, ce qui est vrai en pratique parce que les réseaux de chaque commodité sont souvent obtenus en ajoutant à un réseau générique un petit ensemble de nœuds et d'arcs spécialisés. On peut même, dans la plupart des cas, borner par $O(|K|)$ la somme du nombre de nœuds et d'arcs ainsi ajoutés au réseau générique pour spécialiser le réseau chaque commodité. Sous ces hypothèses, la complexité de l'algorithme 6.2 devient ainsi

$$C_{\text{LOC}}(K, N, A, W) = O\left((|K| + |A|)|W| + (|K| + |N|)|W|^2 + |W|\right).$$

Cette stratégie consistant à appliquer l'algorithme 6.2 sur la réunion de tous les réseaux n'est cependant valide que si le réseau résultant est acyclique.

Résultats d'expérimentation : L'algorithme 6.2 a été mis en œuvre dans le logiciel GENCOL-4 selon la stratégie d'accélération précédente et testé sur des problèmes de grande taille ($500 \leq |W| \leq 1000$) de quelques applications :

- horaires mensuels personnalisés (*Preferential Bidding System* ou PBS, Gamache *et al.* (1998)),
- rotations d'équipages (*Pairing Generation* ou PG, Desrosiers *et al.* (1999), *voir* aussi Desaulniers *et al.* (1997a) pour application similaire),
- itinéraires et horaires de véhicules (*Multi-Depot Vehicle Scheduling Problem* ou MDVSP, Ribeiro et Soumis (1994)).

Seuls les problèmes PG ont permis à l'algorithme de construire des partitions menant à l'agrégation de contraintes, pour une réduction moyenne du nombre de contraintes d'environ 30% et une réduction proportionnelle du temps de calcul consacré à

la résolution du problème-maître. Le succès de l'algorithme pour les problèmes PG peut s'expliquer par l'existence de longues chaînes (séquences d'arcs dont les nœuds intermédiaires sont de degré 2) dans les réseaux, conséquence de l'utilisation d'heuristiques lors de la construction des données. L'échec de l'algorithme sur les autres problèmes s'explique par la possibilité de couvrir n'importe quelle combinaison de tâches au moyen de chemins dans les réseaux. En effet, la structure des problèmes PBS et MDVSP ne mène pas à une réduction significative du nombre d'arcs dans le graphe auxiliaire et l'algorithme ne découvre aucune chaîne de tâches dans ce graphe. Le succès de l'algorithme dans le cas des problèmes PG est lui-même mitigé, puisque le temps de calcul de l'algorithme 6.2 est parfois du même ordre de grandeur que celui du processus de résolution du problème original (sans l'application de l'algorithme d'agrégation).

La performance de cette première version de l'algorithme 6.2 est décevante, mais la réduction de 30% du nombre de tâches pour les problèmes PG reste intéressante. Puisque c'est l'existence de longues chaînes d'arcs dans les réseaux qui semble permettre cette réduction, on considère une deuxième stratégie d'accélération menant à une variante encore plus heuristique de l'algorithme 6.2. Le principe consiste à analyser uniquement les chaînes d'arcs des réseaux de chaque commodité. Dans ce cas, nous ne calculons pas l'ensemble des tâches précédant la première tâche couverte par une telle chaîne d'arcs et nous imposons artificiellement, lors de la construction du graphe auxiliaire de tâches, que cette première tâche possède un degré entrant plus grand que 2. L'objectif de cette contrainte sur la première tâche d'une chaîne d'arcs est d'empêcher le regroupement en (6.11) de cette tâche avec une tâche qui la précède. Le même argument s'applique symétriquement à la dernière tâche couverte par une chaîne d'arcs. La complexité de l'algorithme 6.2 devient alors

$$C_{\text{LOC}}(K, N, A, W) = O(|K||A| + |W|),$$

et l'application de la première stratégie d'accélération réduit cette complexité à

$$C_{\text{LOC}}(K, N, A, W) = O(|K| + |A| + |W|).$$

Les tests sur les problèmes PG montrent que la réduction du nombre de tâches passe ainsi à 25% (au lieu de 30%), ce qui constitue un compromis acceptable compte tenu que le temps d'exécution de cette dernière variante de l'algorithme 6.2 est négligeable par rapport au temps de résolution total des problèmes. La faible détérioration de taux de réduction confirme aussi l'explication du succès de l'algorithme avancée au paragraphe précédent.

Les algorithmes de partition décrits dans cette section, couplés à la reformulation de la section 6.1, définissent un processus d'agrégation statique fondé sur l'analyse *a priori* des contraintes de partitionnement d'un problème résiduel du modèle unifié. Le premier algorithme permet d'obtenir une partition de taille minimale menant à l'agrégation du plus grand nombre de contraintes, mais sa complexité et ses hypothèses restrictives le rendent inutilisable dans le contexte d'applications pratiques ou de problèmes de très grande taille. Le second algorithme est plus efficace en temps de calcul, au détriment du taux d'agrégation des contraintes. Parmi les problèmes étudiés, ceux qui bénéficient du prétraitement de ce deuxième algorithme sont ceux dont les réseaux sont faiblement connectés (existence de plusieurs nœuds de degré 2). L'efficacité de l'algorithme par rapport au processus complet de résolution permet de l'utiliser de façon inconditionnelle peu importe l'espérance de réduction du nombre de contraintes. Ainsi, l'approche d'agrégation statique atteint partiellement l'objectif visé, qui est de permettre la résolution de problèmes possédant de l'ordre de 10 000 contraintes en limitant la taille du problème-maître à environ 2000 contraintes. L'approche d'*agrégation dynamique*, développée à la prochaine section, exploite la structure particulière des solutions admissibles du modèle unifié afin d'*intégrer* les techniques de reformulation de la section 6.1 et l'algorithme de génération de colonnes.

6.3 Agrégation dynamique

Le processus de résolution du modèle unifié consiste à explorer un arbre d'énumération dont chaque nœud, ou problème résiduel, permet de calculer des bornes inférieures et des solutions admissibles par génération de colonnes. L'approche d'agrégation statique présentée à la section 6.2 vise la réduction du nombre de contraintes liantes d'un problème résiduel par l'identification et la reformulation de blocs de contraintes comportant des sous-matrices aux lignes identiques. Or, malgré la forte proportion (souvent plus de 90%) de contraintes de partitionnement W (1.17) observée dans les problèmes de grande taille traités avec GENCOL, l'efficacité de cette approche dépend fortement de la structure des réseaux des problèmes traités. On propose dans cette section une approche d'agrégation *dynamique* où l'algorithme de génération de colonnes est modifié pour s'inscrire dans un processus itératif de partition des contraintes et de sélection des variables. Le principe mathématique sous-jacent à l'agrégation dynamique est exposé au prochain paragraphe. On présente ensuite l'algorithme, dont chacune des étapes est analysée dans les paragraphes subséquents. La convergence de l'algorithme en un nombre fini d'itérations est établie au terme de cette analyse dans le cas où le problème résiduel est un programme linéaire non dégénéré. La section se termine par une discussion sur la mise en œuvre de l'algorithme dans le contexte d'applications de très grande taille de fabrication d'horaires d'équipage ou de mise à jour d'itinéraires d'avion et d'équipage.

6.3.1 Principe de l'algorithme

Le principe de l'agrégation dynamique est l'inverse de celui de l'agrégation statique. L'agrégation statique construit une partition $\mathcal{W}$ de l'ensemble W telle que tous les chemins soient compatibles avec la partition. Un chemin est *compatible* avec la partition $\mathcal{W}$ si, pour toute paire de tâches $(\bar{w}, w) \in W$ équivalentes selon la partition $\mathcal{W}$, le

chemin satisfait $a^k_{wp} = a^k_{\bar{w}p}$; ainsi un chemin est compatible avec la partition si deux tâches équivalentes sont couvertes ou non couvertes en même temps. L'agrégation dynamique choisit une partition $\mathcal{W}$ donnée et restreint temporairement l'ensemble des chemins d'un problème résiduel aux chemins compatibles avec la partition $\mathcal{W}$. Un problème résiduel restreint selon la partition $\mathcal{W}$ s'obtient en imposant que les variables correspondant aux chemins incompatibles avec $\mathcal{W}$ soient nulles (ou rejetées). Cette définition de compatibilité implique que la partition $\mathcal{W}$ est valide sur un problème résiduel restreint selon $\mathcal{W}$ et que la reformulation de la section 6.1 peut s'appliquer. On note que les effets des agrégations statique et dynamique sont identiques lorsque tous les chemins sont compatibles avec la partition $\mathcal{W}$.

L'algorithme 6.3 présente un processus de génération de colonnes modifié pour tirer profit d'une partition $\mathcal{W}$ des contraintes W, dont la structure en deux boucles emboîtées rappelle la stratégie de *partial pricing* utilisée pour accélérer l'algorithme du simplexe. Les deux boucles correspondent chacune à une façon d'améliorer la solution. La boucle INTERNE améliore la solution en utilisant seulement les colonnes compatibles avec la partition $\mathcal{W}$ actuelle. La boucle EXTERNE modifie la partition $\mathcal{W}$ pour pouvoir utiliser d'autres colonnes afin de continuer à améliorer la solution. Le prédicat `modifier` contient une stratégie décidant de passer à la boucle EXTERNE plutôt que de continuer dans la boucle INTERNE si l'amélioration espérée y est plus grande.

6.3.2 Analyse de l'algorithme

Les antécédents déterminent le domaine d'un algorithme. La seule contrainte qui apparaît dans ceux de l'algorithme 6.3 implique que le problème résiduel soit réalisable et borné, même lorsqu'il est restreint aux seules variables supplémentaires Y_s, $s \in S$. Cette contrainte permet d'alléger la présentation de l'algorithme sans en atténuer

Algorithme 6.3 Algorithme d'agrégation dynamique

Antécédent: problème résiduel, réalisable (sans variables de chemin) et borné

Antécédent: ensemble initial $P' \subset P = \cup_{k \in K} P^k$ de variables générées

Antécédent: partition initiale $\mathcal{W}$ des contraintes W

Notation: pour un ensemble donné P, on dénote par $\wp(P)$ l'ensemble de tous les sous-ensembles de P

Antécédent: prédicat `modifier` : $\wp(P) \times \wp(P) \times \mathbb{R}^{|W|} \mapsto \{\texttt{vrai}, \texttt{faux}\}$

Notation: pour un ensemble donné P de variables, on dénote par $P_{\mathcal{W}} \subset P$ le sous-ensemble des variables compatibles avec la partition $\mathcal{W}$ et par $\bar{P}_{\mathcal{W}} = P \setminus P_{\mathcal{W}}$ son complément dans P

Conséquent: solution optimale du problème résiduel

1: **boucle** EXTERNE {ajuster la partition}
2: **boucle** INTERNE {optimiser une restriction}
3: résoudre le problème-maître restreint aux variables de l'ensemble $P'_{\mathcal{W}}$, pour obtenir une solution primale ainsi qu'une solution duale *agrégée*, en variables $\hat{\boldsymbol{\alpha}}$
4: construire une solution duale en variables $\boldsymbol{\alpha}$ respectant les valeurs des variables duales agrégées $\hat{\boldsymbol{\alpha}}$, en tirant profit de l'ensemble $\bar{P}'_{\mathcal{W}}$
5: générer un ensemble de variables d'indice $p \in P'' \subset P$ de coût réduit $\bar{c}_p(\boldsymbol{\alpha}) < 0$ en résolvant les sous-problèmes en fonction des variables duales $\boldsymbol{\alpha}$
6: **si** $P'' = \emptyset$ **alors**
7: **sortir de la boucle** EXTERNE {solution optimale}
8: $P' \leftarrow P' \cup P''$
9: **si** $P''_{\mathcal{W}} = \emptyset$ ou $[(\exists p \in \bar{P}'_{\mathcal{W}} \mid \bar{c}_p(\boldsymbol{\alpha}) < 0)$ et $\texttt{modifier}(P''_{\mathcal{W}}, \bar{P}'_{\mathcal{W}}, \boldsymbol{\alpha})]$ **alors**
10: choisir une variable d'indice $\bar{p} \in \bar{P}'_{\mathcal{W}}$ telle que $\bar{c}_{\bar{p}}(\boldsymbol{\alpha}) < 0$
11: **sortir de la boucle** INTERNE {pour ajuster la partition}
12: transformer la solution du problème-maître en solution de base
13: modifier la partition $\mathcal{W}$ de façon que les variables en base et la variable d'indice $\bar{p}$ soient compatibles avec la nouvelle partition

la portée, puisqu'en cas d'incertitude il est toujours possible de résoudre le problème résiduel en deux phases, visant la réalisabilité d'abord et l'optimalité ensuite. La présence d'un ensemble initial de variables générées reflète le contexte général d'utilisation de l'algorithme dans un processus d'énumération impliquant la résolution d'une séquence de problèmes résiduels. L'algorithme laisse beaucoup de latitude quant au choix des partitions $\mathcal{W}$ de l'ensemble des contraintes W, tant pour la partition initiale que pour les partitions subséquentes construites à l'étape 13. Cette marge de manœuvre est essentielle pour permettre l'exploitation de structures diverses sur l'ensemble des tâches au niveau des applications, comme par exemple un ordonnancement temporel des tâches ou un séquencement naturel de celles-ci dans les chemins. Ainsi, la partition initiale $\mathcal{W}$ devrait idéalement être choisie en fonction d'une analyse propre aux données d'une application, en visant une taille $|\mathcal{W}|$ qui maximise l'efficacité de l'optimiseur linéaire utilisé pour résoudre les problèmes-maîtres restreints à l'étape 3. Finalement, l'existence du prédicat stratégique `modifier`, dont la définition ne comporte aucune contrainte, met en évidence une source d'alternatives qui sont étudiées à la fin de la section dans le contexte de la mise en œuvre de l'algorithme.

La résolution d'un problème-maître restreint selon la partition $\mathcal{W}$, à l'étape 3, constitue la raison d'être de l'algorithme d'agrégation dynamique. En formulant le programme linéaire sur le sous-ensemble $P'_{\mathcal{W}}$ des variables générées P' qui sont compatibles avec la partition $\mathcal{W}$, les conditions pour l'application de la transformation de la section 6.1 sont satisfaites. Le problème-maître restreint comporte alors $|L| = |\mathcal{W}|$ contraintes de partitionnement et on suppose que l'optimiseur linéaire permet de le résoudre plus efficacement que si le problème-maître comportait toutes les $|W|$ contraintes de partitionnement. Le problème résiduel étant réalisable même en l'absence de variables de chemin, l'optimiseur trouve toujours une solution même si $P'_{\mathcal{W}} = \emptyset$. Les variables duales *agrégées*, associées aux contraintes de partitionnement agrégées $l \in L$, sont dénotées par le vecteur $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_l)_{l \in L}$.

L'objectif des sous-problèmes est posé en fonction de variables duales $\boldsymbol{\alpha} = (\alpha_w)_{w \in W}$ correspondant à chaque contrainte $w \in W$. Comme la transformation de la section 6.1 permet l'agrégation de $|W|$ contraintes en $|L| = |\mathcal{W}| \leq |W|$ contraintes *équivalentes*, la solution à l'étape 3 du problème-maître restreint agrégé est aussi une solution du problème-maître restreint comportant les $|W|$ contraintes originales. L'étape 4 transforme donc le vecteur $\hat{\boldsymbol{\alpha}} \in \mathbb{R}^{|L|}$ en un vecteur $\boldsymbol{\alpha} \in \mathbb{R}^{|W|}$ qui préserve l'optimalité de la solution obtenue à l'étape 3. Puisque les contraintes $w \in W_l$ d'un bloc l sont toutes identiques sur le sous-ensemble $P'_{\mathcal{W}}$, on peut écrire la condition d'équivalence entre $\hat{\boldsymbol{\alpha}}$ et $\boldsymbol{\alpha}$ au moyen des égalités suivantes :

$$\sum_{w \in W_l} \alpha_w = \hat{\alpha}_l \qquad \forall l \in L. \tag{6.12}$$

Pour une partition $\mathcal{W}$ ne comportant pas que des singletons, les contraintes (6.12) forment un système linéaire sous-déterminé. Une solution admissible consiste à répartir également les valeurs $\hat{\alpha}_l$ entre les variables α_w :

$$\alpha_w = \hat{\alpha}_l / |W_l| \qquad \forall l \in L, \forall w \in W_l. \tag{6.13}$$

Il est aussi possible d'exploiter le système sous-déterminé (6.12) pour obtenir un vecteur dual $\boldsymbol{\alpha}$ qui soit optimal pour un certain nombre de variables de l'ensemble $\bar{P}'_{\mathcal{W}}$ tout en restant une solution duale pour les variables de l'ensemble $P'_{\mathcal{W}}$. La prise en compte des variables de l'ensemble $\bar{P}'_{\mathcal{W}}$, exclues de la formulation du problème-maître restreint à l'étape 3 parce qu'elles sont incompatibles avec la partition $\mathcal{W}$, augmente la probabilité de conclure à l'optimalité de la solution à l'étape 6 ou force les sous-problèmes à générer de *nouvelles* variables dans le cas contraire. Deux méthodes heuristiques permettant d'ajuster les variables duales selon ce principe sont discutées dans les paragraphes sur la mise en œuvre de l'algorithme.

Une fois un vecteur $\boldsymbol{\alpha}$ obtenu, satisfaisant au moins les conditions d'optimalité sur les variables de l'ensemble $P'_{\mathcal{W}}$, l'algorithme résout les sous-problèmes pour confirmer

l'optimalité de la solution ou générer de nouvelles variables de coût réduit négatif. On impose que les sous-problèmes soient résolus sur l'ensemble P de toutes les variables du problème résiduel et pas seulement sur les variables de l'ensemble $P_{\mathcal{W}}$ compatibles avec la partition $\mathcal{W}$. Ceci découle du fait que les algorithmes connus pour résoudre les sous-problèmes dans le contexte du modèle unifié ne supportent pas (ou mal) une restriction de leur domaine selon une partition $\mathcal{W}$ quelconque. De plus, la résolution des sous-problèmes sur l'ensemble P de toutes les variables est surtout nécessaire pour déterminer l'optimalité de la solution à l'étape 6. L'arrêt de l'algorithme à l'étape 7 correspond au critère habituel d'arrêt d'un algorithme du simplexe, la solution primale étant celle calculée lors de la résolution du problème-maître restreint à l'étape 3 et la solution duale provenant de cette même résolution en remplaçant le vecteur $\hat{\boldsymbol{\alpha}}$ par le vecteur $\boldsymbol{\alpha}$. Lorsque la solution n'est pas optimale, les sous-problèmes génèrent au moins une variable $p \in P$ telle que son coût réduit $\bar{c}_p(\boldsymbol{\alpha})$ est négatif. L'ensemble P'' des variables nouvellement générées est ajouté à l'étape 8 à l'ensemble P' cumulant toutes les variables déjà produites par les sous-problèmes. Comme ceux-ci ne tiennent pas compte de la partition $\mathcal{W}$, la suite de l'algorithme doit considérer le cas général où un certain nombre des variables de l'ensemble P'' peuvent être incompatibles avec $\mathcal{W}$.

Le test de l'étape 9 n'est évalué que lorsque l'ensemble $P'' \neq \emptyset$, ce qui implique que la solution duale $\boldsymbol{\alpha}$ n'est pas optimale. Il faut alors s'assurer que le prochain problème-maître restreint résolu à l'étape 3 permette d'améliorer la solution de façon que le processus d'optimisation soit fini (au moins dans le cas d'un programme linéaire non dégénéré). Le principe sous-jacent à l'étape 9 est le suivant :

- on *doit* modifier la partition $\mathcal{W}$ si l'ensemble $P'_{\mathcal{W}}$ n'a pas changé à l'étape 8, parce que nous n'avons plus de variables dans la boucle INTERNE pour améliorer la solution ;
- on *peut* modifier la partition $\mathcal{W}$ seulement si l'on dispose d'au moins une variable $p \in P'$ incompatible avec la partition $\mathcal{W}$ et de coût réduit négatif, afin de pouvoir améliorer la solution lors de la prochain itération de la boucle EXTERNE.

Le prédicat `modifier` constitue une stratégie paramétrisable de l'algorithme, permettant d'assujettir un changement possible de la partition $\mathcal{W}$ à une comparaison du progrès espéré en la conservant (fonction de $P''_{\mathcal{W}}$) et en la changeant (fonction de $\bar{P}'_{\mathcal{W}}$).

Lorsqu'il est décidé à l'étape 9 de modifier la partition $\mathcal{W}$, l'étape 10 identifie une variable $\bar{p}$ de coût réduit négatif incompatible avec la partition. Cette variable existe toujours puisque dans le cas où la première branche de la disjonction est vraie, $P''_{\mathcal{W}} = \emptyset$ implique que $\emptyset \neq \bar{P}''_{\mathcal{W}} \subset \bar{P}'_{\mathcal{W}}$ et dans le cas où la deuxième branche de la disjonction est vraie, la condition est vérifiée explicitement. Nous assurons que la valeur de l'objectif obtenue lors de la prochaine résolution du problème-maître restreint à l'étape 3 ne croît pas en imposant à l'étape 13 que les variables formant la solution primale courante soient compatibles avec la nouvelle partition. Dans le cas d'un programme linéaire non dégénéré, on force la décroissance de l'objectif par l'inclusion de la variable $\bar{p}$ dans la nouvelle partition. Dans le cas habituel d'un programme linéaire dégénéré, la dégénérescence au niveau de la boucle INTERNE peut être prise en charge directement par l'optimiseur linéaire à l'étape 3 et indirectement par l'application des stratégies de perturbation proposées au chapitre 5. Ces mécanismes, bien qu'ils empêchent en pratique la boucle INTERNE d'entrer dans un cycle, n'assurent pas une décroissance stricte de l'objectif entre les itérations de la boucle EXTERNE. Il est donc possible que le processus itératif de la boucle EXTERNE mène à un cycle, si l'étape 13 génère une séquence cyclique de partitions associées à une même valeur de l'objectif. Nous proposons une stratégie pour éviter ce cas pathologique : lorsque la valeur de l'objectif ne décroît pas au cours de la dernière itération de la boucle EXTERNE, la partition $\mathcal{W}$ ne doit être modifiée à l'étape 13 que par des raffinements permettant d'inclure la variable $\bar{p}$. Avec cette stratégie de raffinements obligatoires, une séquence de partitions générées à l'étape 13 débouche soit sur une décroissance stricte de l'objectif, soit sur une partition compatible avec toutes les variables, auquel cas la gestion de la dégénérescence est entièrement prise en compte dans la boucle INTERNE.

6.3.3 Mise en œuvre de l'algorithme

Nous proposons d'analyser la mise en œuvre de l'algorithme d'agrégation dynamique dans le contexte d'applications de très grande taille de fabrication d'horaires d'équipage ou de mise à jour d'itinéraires d'avion ou d'équipage. Dans ces applications, les tâches sont associées à des intervalles de temps et la structure du réseau empêche les horaires ou les itinéraires de couvrir une même tâche plusieurs fois. Nous observons dans les solutions optimales que les tâches correspondant aux équipages ou aux avions s'enchaînent selon un critère de continuité spatiale (en plus de l'ordonnancement temporel) laissant émerger une structure sous-jacente de chemins.

Dans le cas de la fabrication d'horaires d'équipage (par exemple, pour des équipages d'avion ou des chauffeurs d'autobus), nous observons que le personnel « suit » ou fait corps avec le même véhicule la plupart du temps dans la solution optimale. Ceci s'explique en grande partie par les coûts associés aux changements de véhicule. En effet, les connexions longues impliquent une perte de temps qui se traduit directement par une détérioration de l'objectif tandis que les connexions courtes augmentent le risque de rater le changement de véhicule en pratique et sont de ce fait pénalisées dans l'objectif. Ainsi, lors de la fabrication d'horaires pour les chauffeurs d'autobus, la solution optimale affecte la plupart du temps un chauffeur à plusieurs tronçons de ligne consécutifs (pouvant atteindre quelques dizaines de tâches). Lors de la fabrication des horaires d'équipage en transport aérien, ces équipages effectuent la plupart du temps une séquence de vols sans changer d'avion (séquences d'environ 3 ou 4 vols).

Dans le cas de la mise à jour d'itinéraires d'avion ou d'équipage, les nouvelles solutions « suivent » en grande partie les solutions planifiées. Ceci est dû à la présence de pénalités représentant les inconvénients de modifier ces solutions planifiées. Notons

d'abord qu'en général, le salaire du personnel (équipage à bord d'un avion ou personnel d'entretien au sol) est basé sur le montant maximum associé soit aux tâches planifiées, soit aux tâches travaillées. Cette règle a pour but de prendre partiellement en compte les inconvénients d'un changement d'horaire ou d'itinéraire pour le personnel, dont la vie hors les heures de travail est aussi affectée. Ainsi, les changements d'itinéraires des avions perturbent la planification à long terme des entretiens, qui ne peuvent souvent être effectués que dans certains aéroports. Ceci mène à une utilisation sous-optimale des équipements et à des salaires supplémentaires versés aux équipes de soutien au sol. Quant aux modifications des itinéraires des équipages, elles entraînent inévitablement des frais supplémentaires d'hébergement, de transport au sol et de salaires.

La description des deux contextes d'application précédents met en évidence un ordonnancement linéaire des tâches par véhicule ou selon l'horaire planifié. Nous référons par la suite aux séquences maximales de tâches ainsi ordonnées en tant que *chemins sous-jacents* du problème. Ces chemins forment une partition de l'ensemble W de toutes les tâches. On considère dans la discussion qui suit sur la mise en œuvre de l'algorithme d'agrégation dynamique que cette partition est de taille minimale et que toutes les partitions construites ou modifiées par l'algorithme sont issues de raffinements de cette partition. Nous restreignons de plus les raffinements à préserver la contiguïté des tâches dans la partition de taille minimale, chaque ensemble d'une partition constituant alors un *sous-chemin* qui hérite de l'ordonnancement de son chemin sous-jacent. Puisque les solutions optimales dévient peu des chemins sous-jacents, on voit intuitivement qu'un petit nombre de sous-chemins produits par le raffinement d'un chemin sous-jacent peut en pratique être suffisant pour obtenir une solution presque optimale, sans impliquer nécessairement un raffinement en singletons.

Étant donné un logiciel de génération de colonnes, l'algorithme 6.3 d'agrégation dynamique requiert le développement des opérations supplémentaires suivantes :

- gestion du problème-maître restreint concernant l'agrégation des contraintes selon une partition $\mathcal{W}$ à l'étape 3 ;
- construction d'une partition $\mathcal{W}$ initiale ;
- construction d'un vecteur de variables duales α désagrégées à partir d'un vecteur de variables duales agrégées $\hat{\alpha}$ à l'étape 4 ;
  décision du moment opportun pour modifier la partition à l'étape 9 ;
- modification de la partition à l'étape 13.

Voyons maintenant l'état des travaux sur ces opérations supplémentaires.

Gestion du problème-maître restreint : Les développements concernant la gestion du problème-maître restreint sont avancés mais non encore intégrés au logiciel GENCOL-4 au moment d'écrire ces lignes. Ces développements mènent à une refonte du module MP et comportent entre autres les innovations suivantes :

- une généralisation de la représentation d'un programme linéaire, incluant des bornes explicites sur les variables primales *et duales* ;
- un prétraitement pour simplifier la représentation du programme linéaire fourni à l'optimiseur (CPLEX) en fixant les valeurs des variables primales *et duales* dont les bornes sont identiques ;
- un mécanisme de mémorisation des modifications apportées au problème-maître, qui sont transmises à l'optimiseur (CPLEX) en lots le plus tard possible, juste avant l'optimisation.

Ces transformations ont nécessité la récriture du module MP, dont la taille a plus que doublé par rapport à la version précédemment utilisée dans GENCOL-4. La représentation généralisée du programme linéaire et le prétraitement des bornes sur les variables duales permettent de développer à partir de la même interface informatique les stratégies de contrôle de l'espace dual présentées au chapitre 5 et les stratégies de modification des partitions présentées dans cette section. Par ailleurs, le mécanisme de mémorisation des modifications apportées au problème-maître devient essentiel pour assurer l'efficacité du module MP dans le contexte où plusieurs stratégies

agissent à des niveaux différents du processus de résolution (perturbation, agrégation statique, agrégation dynamique, génération de colonnes) afin d'éviter l'enchaînement de plusieurs opérations atomiques inefficaces (comme par exemple le retrait d'un ensemble de lignes ou de colonnes une à la fois).

Les primitives de la nouvelle version du module MP constituent le fondement de la mise en œuvre de l'algorithme d'agrégation dynamique au sein du processus de résolution déjà en place dans le logiciel GENCOL-4. Les quatre autres items présentés dans les paragraphes suivants doivent être pris en charge par le module LB qui gère les différentes stratégies pour obtenir une borne inférieure sur l'objectif du problème résiduel courant.

Construction d'une partition initiale : Le deuxième item concerne la construction d'une partition $\mathcal{W}$ initiale en tirant profit de la structure sous-jacente de chemins provenant du contexte d'application. Nous proposons d'abord une heuristique gloutonne dans le cas des problèmes de fabrication d'horaires d'équipage d'avion et de chauffeurs d'autobus, que nous adaptons ensuite au cas des problèmes de mise à jour d'une solution déjà planifiée. L'heuristique consiste à résoudre séquentiellement des problèmes de plus court chemin sur le réseau constitué par la réunion des réseaux de chaque commodité $k \in K$. Chaque arc $(i,j) \in A^k$ permettant un changement de véhicule est pénalisé par un coût positif représentant une mesure de l'inconvénient associé au changement. Chaque tâche contribue d'une valeur $-M$, $M \gg \max_{ijk} |c_{ij}^k|$, au coût des arcs qui la couvrent. On ajoute au réseau des arcs artificiels reliant un nœud origine commun o à tous les nœuds origines o^k, $k \in K$, ainsi que des arcs artificiels reliant tous les nœuds puits d^k à un nœud puits commun d. La solution du problème de plus court chemin entre les nœuds o et d sur ce réseau donne un chemin optimal qui couvre un certain ensemble de tâches au moyen des arcs. Nous suggérons d'agréger les tâches si elles sont effectuées par un même véhicule et sont consécutives

sur ce chemin. Une fois le chemin analysé et les regroupements déterminés, les arcs couvrant les tâches du chemin optimal sont retirés du réseau et l'heuristique est appliquée de nouveau sur le réseau résiduel jusqu'à ce que toutes les tâches soient traitées. Dans le cas de la mise à jour de solutions déjà planifiées, on pénalise les arcs qui ne font pas partie de la solution planifiée au lieu de ceux impliquant des changements de véhicule. Nous suggérons alors d'agréger les tâches si elles sont consécutives dans la solution planifiée et sur un chemin optimal. Notons qu'une solution du problème obtenue de façon heuristique peut jouer le rôle d'une solution planifiée lorsque cette heuristique produit des solutions proches des solutions optimales.

L'heuristique gloutonne permet ainsi de former une partition $\mathcal{W}$ initiale qui respecte l'ordonnancement spatio-temporel observé dans les solutions optimales des applications étudiées. Elle laisse aussi une certaine marge de manœuvre quant à la taille de la partition $\mathcal{W}$ puisque celle-ci peut être raffinée sans violer les critères qui ont servi à la construire. Lorsque l'heuristique de construction de la partition $\mathcal{W}$ est appliquée telle quelle, la taille $|\mathcal{W}|$ de la partition est minimale compte tenu des critères utilisés. Une telle partition contribue, par l'agrégation d'un plus grand nombre de contraintes au moyen du processus défini à la section 6.1, à une efficacité accrue de la boucle INTERNE en accélérant la résolution du problème-maître restreint à l'étape 3 de l'algorithme. Cependant, une telle partition peut impliquer une réduction du nombre de chemins compatibles et en conséquence une augmentation du nombre d'itérations de la boucle EXTERNE. L'heuristique de construction de la partition $\mathcal{W}$ initiale peut donc être couplée à une phase de raffinements permettant d'« optimiser » le nombre de contraintes actives dans le problème-maître en fonction d'un compromis entre l'efficacité des boucles INTERNE et EXTERNE.

Construction d'un vecteur de variables duales désagrégées : Le troisième item porte sur la construction d'un vecteur de variables duales α désagrégées en

fonction des variables duales agrégées $\hat{\boldsymbol{\alpha}}$. Comme mentionné lors de l'analyse de l'étape 4 de l'algorithme, le système d'équations associé à cette construction est sous-déterminé dans le cas habituel où la partition $\mathcal{W}$ ne contient pas que des singletons. On peut alors tenter de choisir un vecteur $\boldsymbol{\alpha}$ qui satisfait plus de contraintes d'optimalité duale en intégrant à ce système des contraintes provenant des variables d'un sous-ensemble $\bar{P}^{\bullet}_{\mathcal{W}}$ de l'ensemble $\bar{P}'_{\mathcal{W}}$:

$$\sum_{w \in W_l} \alpha_w = \hat{\alpha}_l \qquad \forall l \in L \tag{6.14}$$

$$\sum_{w \in W} a_{wp} \alpha_w \leq c_p \qquad \forall p \in \bar{P}^{\bullet}_{\mathcal{W}} \subseteq \bar{P}'_{\mathcal{W}}, \tag{6.15}$$

où c_p est le coût de la variable $p \in \bar{P}^{\bullet}_{\mathcal{W}}$ modifié pour tenir compte de la contribution des variables duales associées aux contraintes du problème-maître restreint autres que les contraintes W. Ce nouveau système décrit l'intersection d'un hyperplan (6.14) et d'un polyèdre (6.15). Même si on n'a qu'à trouver un vecteur dual réalisable pour les contraintes (6.14)–(6.15), la difficulté de résoudre ce système est analogue à celle de résoudre le problème-maître sans agrégation. Comme la motivation de ce chapitre découle de l'inefficacité des méthodes de résolution sur ce dernier problème, on ne peut donc considérer la résolution complète du système (6.14)–(6.15) par programmation linéaire. Nous proposons plutôt une approche qui est fondée sur l'ordonnancement des tâches dans chaque ensemble formant la partition $\mathcal{W}$ et qui permet d'identifier un sous-ensemble $\bar{P}^{\bullet}_{\mathcal{W}}$ de variables définissant un système (6.15) facile à résoudre.

Notre approche utilise la propriété de la partition $\mathcal{W}$ que chaque ensemble W_l, $l \in L$, correspond à un sous-chemin qui hérite de l'ordre linéaire de son chemin sous-jacent. La technique décrite au deuxième item produit une partition $\mathcal{W}$ initiale possédant cette propriété. Les stratégies décrites au cinquième item pour modifier la partition assurent que la propriété est préservée lors du passage à la prochaine itération de la boucle EXTERNE. L'approche consiste d'abord à effectuer un changement de variables

sur le système (6.14)–(6.15), passant des variables α_w, $w \in W$, aux variables π_i^l, $i \in \{1, \ldots, |W_l|\}$, $l \in L$. Cette transformation (décrite un peu plus loin) permet ensuite d'identifier parmi toutes les contraintes correspondant aux variables de l'ensemble $\bar{P}'_{\mathcal{W}}$ un sous-ensemble $\bar{P}^*_{\mathcal{W}}$ de contraintes de la forme

$$\pi_j^{l_2} - \pi_i^{l_1} \leq \bar{c}_{ij}^{l_1 l_2}.$$

Une solution réalisable pour un tel système peut être obtenue en résolvant un problème de plus courts chemins sur un réseau dont les nœuds correspondent aux variables, les arcs aux contraintes, les coûts des arcs au membre de droite des contraintes (Cormen *et al.*, 1990). Il suffit alors d'effectuer le changement de variables inverse pour obtenir une solution réalisable pour le système (6.14)–(6.15) formé des contraintes associées aux variables de l'ensemble $\bar{P}^*_{\mathcal{W}}$.

Puisque les ensembles W_l, $l \in L$, sont totalement ordonnés, le changement de variables utilise des bijections

$$g_l : W_l \mapsto \{1, \ldots, |W_l|\} \qquad \forall l \in L,$$

pour associer à chaque tâche $w \in W_l, l \in L$, une nouvelle variable $\pi^l_{g_l(w)}$. Les équations suivantes définissent chaque nouvelle variable π_i^l, $i \in \{1, \ldots, |W_l|\}$, $l \in L$, en fonction des variables α_w :

$$\pi^l_{g_l(\bar{w})} = \sum_{w \in W_l : g_l(w) \leq g_l(\bar{w})} \alpha_w \qquad \forall \bar{w} \in W_l, \forall l \in L. \tag{6.16}$$

On effectue ce changement de variables dans le système (6.14)–(6.15). De par leur définition et les équations (6.14), les variables $\pi^l_{|W_l|}$ valent $\hat{\alpha}_l$ et sont en fait des constantes. En substituant ces variables par leur valeur dans toutes les équations du système transformé (6.14)–(6.15), on obtient que les équations (6.14) sont redondantes et peuvent être retirées du problème.

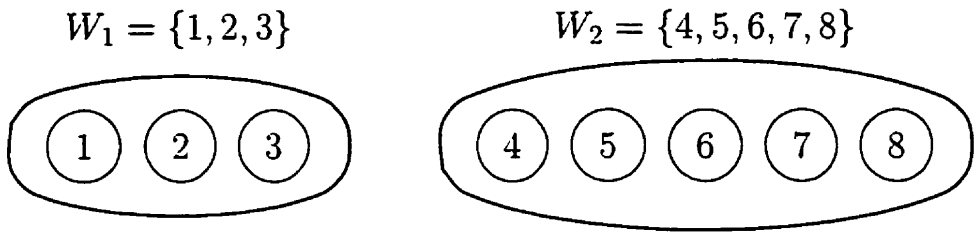


Figure 6.1 – Exemple de partition en deux blocs

La figure 6.1 illustre une partition $\mathcal{W} = \{W_1, W_2\}$ d'un ensemble W de 8 tâches, chacune étant associée à une variable α_w, $w \in W = \{1, \ldots, 8\}$. Le changement de variables (6.16) permet de définir les variables π_i^l au moyen des équations suivantes :

$$\begin{array}{l|l} \pi_1^1 = \alpha_1 & \pi_1^2 = \alpha_4 \\ \pi_2^1 = \alpha_1 + \alpha_2 & \pi_2^2 = \alpha_4 + \alpha_5 \\ \pi_3^1 = \alpha_1 + \alpha_2 + \alpha_3 = \hat{\alpha}_1 & \pi_3^2 = \alpha_4 + \alpha_5 + \alpha_6 \\ & \pi_4^2 = \alpha_4 + \alpha_5 + \alpha_6 + \alpha_7 \\ & \pi_5^2 = \alpha_4 + \alpha_5 + \alpha_6 + \alpha_7 + \alpha_8 = \hat{\alpha}_2. \end{array}$$

Comme mentionné à la fin du paragraphe précédent, les valeurs des variables π_3^1 et π_5^2 sont connues après la résolution du problème-maître restreint à l'étape 3 de l'algorithme et les équations correspondantes du changement de variables peuvent être éliminées. Le système résultant est complètement déterminé et inversible, ce qui permet de trouver les valeurs des variables α_w en fonction des variables π_i^l.

Notre démarche consiste ensuite à définir l'ensemble $\bar{P}_{\mathcal{W}}^*$ par l'intermédiaire des contraintes qui sont formées soit d'une seule variable π_i^l, soit de la différence de deux variables $\pi_j^{l_2}$ et $\pi_i^{l_1}$, après le changement de variables (6.16). Nous rappelons que la notion de chemins sous-jacents introduite au début de cette section découle de l'observation que les solutions optimales dévient peu de ces chemins la plupart du temps. Ainsi, une partition $\mathcal{W}$ correspondant au raffinement des chemins sous-jacents en sous-chemins devrait induire un ensemble $P_{\mathcal{W}}$ de variables compatibles contenant presque toutes les variables $p \in P$ utiles pour construire une solution optimale.

Parmi les variables $p \in \bar{P}'_{\mathcal{W}}$ incompatibles, il semble donc que les plus utiles pour compléter la solution courante soient celles qui ne sont incompatibles que sur un ou deux sous-chemins, puisqu'elles n'impliquent qu'une faible déviation par rapport à la partition courante. Ce raisonnement confirme au niveau de la structure des applications l'intérêt de cette définition de l'ensemble $\bar{P}^*_{\mathcal{W}}$, dont on sait par ailleurs qu'il existe des algorithmes efficaces pour résoudre le système (6.15) associé. Nous donnons ici une méthode constructive pour identifier les variables $p \in \bar{P}^*_{\mathcal{W}}$ par inspection des variables $p \in \bar{P}'_{\mathcal{W}}$ avant d'effectuer le changement de variables sur le système (6.14)–(6.15). Notons que chaque variable $p \in \bar{P}'_{\mathcal{W}}$ est incompatible avec la partition $\mathcal{W}$ en au moins un bloc $l \in L$, ce qui signifie qu'au moins un bloc $l \in L$ n'est couvert que partiellement par chacune de ces variables $p \in \bar{P}'_{\mathcal{W}}$.

Nous nous intéressons d'abord aux variables $p \in \bar{P}'_{\mathcal{W}}$ correspondant aux contraintes (6.15) formées d'une seule variable π_i^l après le changement de variables (6.16). Ces contraintes imposent une borne supérieure ou inférieure sur π_i^l et sont associées à des variables $p \in \bar{P}'_{\mathcal{W}}$ de deux types :

1. celles qui couvrent le début d'un bloc $l_2 \in L$ et le quittent après la j-ième tâche $(j < |W_{l_2}|)$;
2. celles qui entrent dans un bloc $l_1 \in L$ après la i-ième $(i < |W_{l_1}|)$ tâche et couvrent le reste du bloc jusqu'à la fin.

Dans le premier cas, on obtient une contrainte de borne supérieure sur $\pi_j^{l_2}$ de la forme

$$\pi_j^{l_2} \leq c_j^{l_2},$$

où $c_j^{l_2}$ est une constante obtenue en soustrayant de la valeur c_p provenant de la contrainte (6.15) correspondant à la variable p traitée la somme des valeurs $\hat{\alpha}_l$ de tous les blocs $l \in L$ couverts en totalité par la variable p. Dans le second cas, on obtient une contrainte sur $\pi_i^{l_1}$ de la forme

$$\pi_{|W_{l_1}|}^{l_1} - \pi_i^{l_1} \leq \hat{c}_i^{l_1},$$

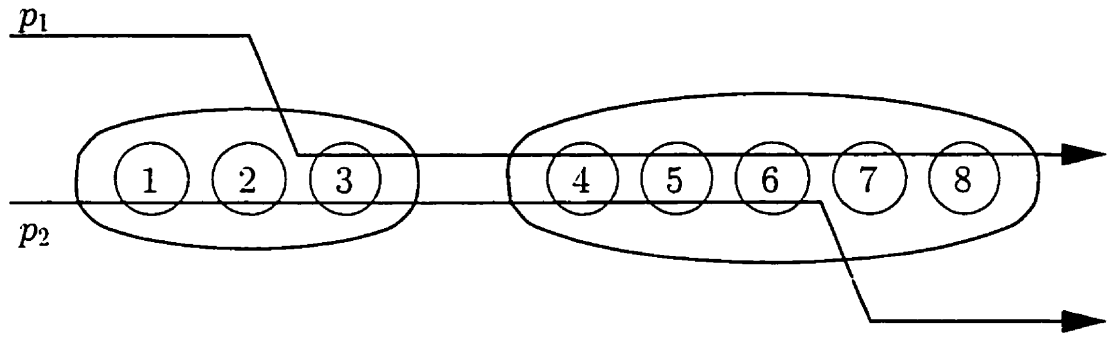


Figure 6.2 – Exemple de bornes inférieure et supérieure

où $\hat{c}_i^{l_1}$ est une constante obtenue d'une façon analogue à $c_j^{l_2}$, en soustrayant de la valeur c_p correspondant au coût de la variable p traitée la somme des valeurs $\hat{\alpha}_l$ de tous les blocs $l \in L$ couverts en totalité par la variable p. Cette contrainte peut ensuite être récrite sous la forme de la contrainte de borne inférieure

$$\pi_i^{l_1} \geq c_i^{l_1} = \hat{\alpha}_{l_1} - \hat{c}_i^{l_1}.$$

La figure 6.2 reprend l'exemple de la figure 6.1 pour illustrer au moyen d'un chemin p_1 une contrainte de borne inférieure sur π_2^1 et au moyen d'un chemin p_2 une contrainte de borne supérieure sur π_3^2. La flèche associée au chemin p_1, de coût c_1, couvre les tâches $\{3, \ldots, 8\}$. Ce chemin p_1 correspond ainsi à une contrainte

$$\alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 + \alpha_7 + \alpha_8 \leq c_1$$

qui peut se récrire

$$\pi_3^1 - \pi_2^1 \leq c_1 - \hat{\alpha}_2,$$

ou encore

$$\pi_2^1 \geq \hat{\alpha}_1 + \hat{\alpha}_2 - c_1.$$

La flèche associée au chemin p_2, de coût c_2, couvre les tâches $\{1, \ldots, 6\}$. Ce chemin p_2 correspond donc à une contrainte

$$\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 \leq c_2$$

qui peut se récrire

$$\pi_3^2 \leq c_2 - \hat{\alpha}_1.$$

On note que la couverture complète d'ensembles de la partition ne change pas la structure des équations obtenues.

Nous incluons aussi dans l'ensemble $\bar{P}^*_{\mathcal{W}}$ les variables $p \in \bar{P}'_{\mathcal{W}}$ qui correspondent aux contraintes (6.15) formées par la différence entre deux variables $\pi_j^{l_2}$ et $\pi_i^{l_1}$ après le changement de variables (6.16). Ces contraintes de différence s'écrivent

$$\pi_j^{l_2} - \pi_i^{l_1} \leq c_{ij}^{l_1 l_2}$$

et sont aussi associées à des variables $p \in \bar{P}'_{\mathcal{W}}$ de deux types.

1. Lorsque $l_1 \neq l_2$, la contrainte de différence correspond à une variable p qui couvre la fin d'un bloc l_1 et le début d'un bloc l_2, en plus d'autres blocs $l \in L$ couverts en totalité.
2. Lorsque $l_1 = l_2 = l$, la contrainte de différence peut représenter
   - soit la couverture partielle d'une séquence de tâches consécutives de $i+1$ à j inclusivement, avec $1 \leq i < j < |W_l|$,
   - soit la couverture du début et de la fin d'un bloc avec un vide au milieu entre les tâches $j+1$ et i inclusivement, avec $1 \leq j < i < |W_l|$.

On remarque que l'avant-dernier cas, conjointement avec les cas pour les contraintes de bornes supérieure et inférieure, permet d'inclure dans l'ensemble $\bar{P}^*_{\mathcal{W}}$ toutes les variables $p \in P$ qui ne couvrent qu'une seule tâche $w \in W$ (singletons).

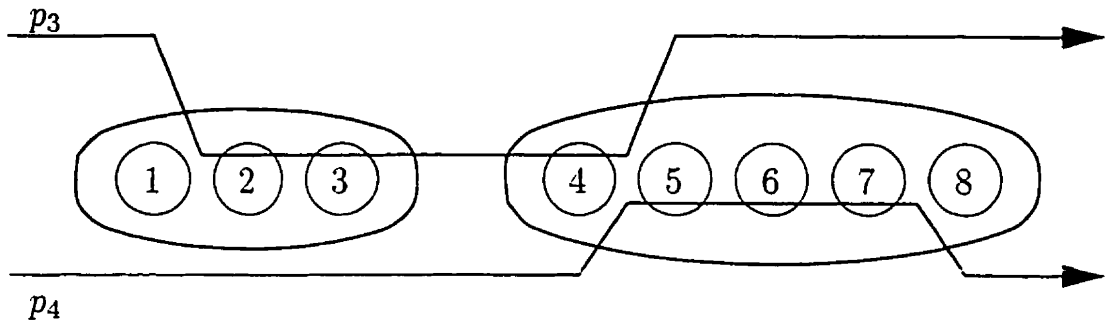


Figure 6.3 – Exemple de contraintes de différence

La figure 6.3 reprend l'exemple de la figure 6.1 pour illustrer au moyen des chemins p_3 et p_4 les deux premiers des trois cas de contraintes de différence mentionnés au paragraphe précédent. La flèche associée au chemin p_3, de coût c_3, couvre les tâches $\{2, 3, 4\}$. Ce chemin p_3 correspond ainsi à une contrainte

$$\alpha_2 + \alpha_3 + \alpha_4 \leq c_3$$

qui peut se récrire

$$(\pi_3^1 - \pi_1^1) + \pi_1^2 \leq c_3,$$

ou encore

$$\pi_1^2 - \pi_1^1 \leq c_3 - \hat{\alpha}_1.$$

La flèche associée au chemin p_4, de coût c_4, couvre les tâches $\{5, 6, 7\}$. Ce chemin p_4 correspond donc à une contrainte

$$\alpha_5 + \alpha_6 + \alpha_7 \leq c_4$$

qui peut se récrire

$$\pi_4^2 - \pi_1^2 \leq c_4.$$

Le troisième cas implique la couverture partielle d'un ensemble de tâches avec un vide au milieu (par exemple, un chemin qui couvrirait les tâches $\{4, 7, 8\}$ de la figure 6.3) et s'analyse de façon analogue aux deux autre cas. Il peut survenir entre autres lors de la fabrication d'horaires pour des chauffeurs d'autobus, où un chauffeur quitte son véhicule pour dîner et y revient ensuite.

Une fois le système de contraintes de différence construit à partir des variables de l'ensemble $\bar{P}^*_{\mathcal{W}}$, on peut trouver une solution réalisable au système (6.14)–(6.15) transformé selon (6.16) en résolvant un problème de plus courts chemins sur un réseau correspondant (Cormen *et al.*, 1990). Le réseau comporte un nœud n^l_i pour chaque variable π^l_i et un arc (i, j) de coût $c^{l_1 l_2}_{ij}$ pour chaque contrainte de différence de la forme $\pi^{l_2}_j - \pi^{l_1}_i \leq c^{l_1 l_2}_{ij}$, où $c^{l_1 l_2}_{ij}$ provient du coût c_p de la formulation originale des inégalités (6.15), modifié pour tenir compte de la substitution des variables $\pi^l_{|W_l|}$ par leur valeur $\hat{\alpha}_l$. On ajoute au réseau un nœud origine o à partir duquel les plus courts chemins sont calculés et on relie ce nœud o à chacun des autres nœuds n^l_i au moyen d'un arc dont l'orientation et le coût sont déterminés comme suit. Si la variable π^l_i est bornée inférieurement, l'arc (n^l_i, o) prend comme coût le négatif de la valeur de la borne inférieure. Si la variable π^l_i est bornée supérieurement, l'arc (o, n^l_i) prend comme coût la valeur de la borne supérieure, sinon il prend comme coût une valeur qui garantit que l'arc ne peut faire partie d'aucun cycle de coût négatif. Par exemple, on obtient une telle valeur en sommant la valeur absolue du coût de tous les arcs de coût négatif du réseau. Dans le cas où il n'existe pas de contraintes de bornes supérieure ou inférieure sur les variables π^l_i du système de contraintes de différence, Cormen *et al.* (1990) montrent qu'il existe une solution au système si et seulement s'il existe une solution au problème de plus courts chemins du nœud o à tous les autres nœuds n^l_i du réseau. Cette preuve se généralise aisément au cas avec bornes inférieures et supérieures. Si le problème de plus courts chemins sur ce réseau possède une solution, la valeur du plus court chemin du nœud o à un nœud n^l_i est une

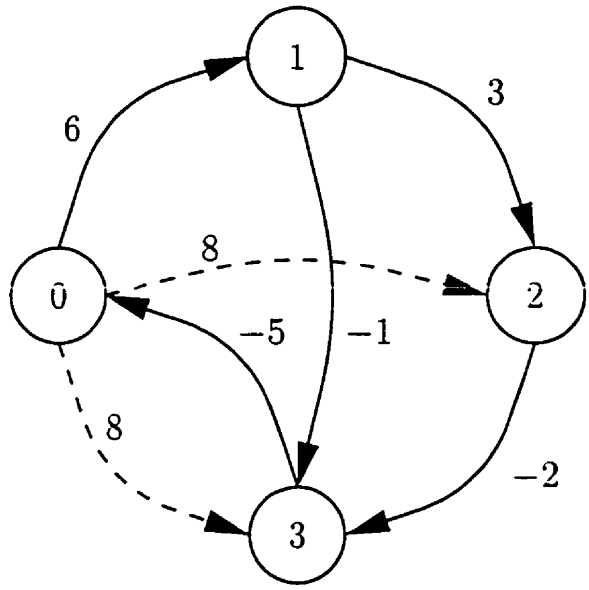


Figure 6.4 - Réseau correspondant au système de contraintes de différence (6.17)

solution pour la variable π_i^l, $i \in \{1, \ldots, |W_l|\}$, $l \in L$. Les valeurs des variables π_i^l peuvent ensuite être transformées en valeurs pour les variables α_w en appliquant le changement de variables inverse de celui donné en (6.16). Cette solution $\boldsymbol{\alpha}$, si elle existe, satisfait alors le système (6.14)–(6.15). L'existence du vecteur dual $\boldsymbol{\alpha}$ signifie que les variables duales $\hat{\boldsymbol{\alpha}}$ trouvées sur le problème-maître restreint à l'étape 3 sont aussi optimales (une fois désagrégées) pour toutes les variables de l'ensemble $\bar{P}^*_{\mathcal{W}}$.

Les équations suivantes constituent un exemple (indépendant des exemples précédents) de système de contraintes de différence avec bornes sur certaines des variables :

$$\begin{aligned} \pi_1 &\leq 6 \\ \pi_2 - \pi_1 &\leq 3 \\ \pi_3 &\geq 5 \\ \pi_3 - \pi_1 &\leq -1 \\ \pi_3 - \pi_2 &\leq -2. \end{aligned} \tag{6.17}$$

Un réseau permettant de résoudre ce système est présenté à la figure 6.4. Le nœud 0 correspond au nœud origine supplémentaire servant de racine pour le problème de plus courts chemins. Les arcs artificiels $(0, i)$ qui relient à la racine les nœuds i correspondant aux variables ne possédant pas de borne supérieure explicite sont présentés en pointillés pour les distinguer des autres arcs provenant directement des contraintes (6.17). Le coût choisi pour les arcs artificiels est $8 = |-5| + |-1| + |-2|$, mais on remarque par inspection que toute valeur plus grande ou égale à 7 serait également valide pour ce système.

Deux difficultés peuvent survenir lors de l'application de l'approche précédente. La première difficulté provient de la possibilité que le problème de plus courts chemins et le système de contraintes de différence n'aient aucune solution. Cette situation correspond à l'existence d'un cycle de coût négatif dans le réseau, ou de façon équivalente à une intersection vide entre l'hyperplan (6.14) et le polyèdre (6.15). Une alternative possible consiste à briser le cycle en retirant du réseau un des arcs du cycle, ce qui correspond à retirer une variable p de l'ensemble $\bar{P}^*_{\mathcal{W}}$. Cette action est répétée jusqu'à ce que le problème devienne réalisable. Plusieurs critères peuvent être élaborés pour choisir l'arc du cycle à retirer du réseau. Globalement, il serait intéressant de minimiser le nombre d'arcs à retirer du réseau, et donc de variables p à retirer de l'ensemble $\bar{P}^*_{\mathcal{W}}$, de façon à tenir compte du plus grand nombre possible de variables incompatibles avec la partition $\mathcal{W}$. Comme ces critères globaux supposent la résolution de problèmes combinatoires, on peut chercher à les approcher au moyen de critères locaux où un arc du cycle de coût négatif est choisi en fonction de la « distance » qu'il permet de franchir entre le réseau actuel et un réseau de coûts non négatifs ou acyclique. Sans entrer dans une analyse détaillée, nous mentionnons que les critères utilisés, par l'impact qu'ils ont sur la solution duale α, influencent les conditions pour rester ou sortir de la boucle INTERNE à l'étape 9 de l'algorithme.

La deuxième difficulté survient lorsque le nombre de variables prises en compte dans l'ensemble $\bar{P}^*_{\mathcal{W}}$ est relativement faible par rapport au nombre de variables de l'en-

semble $\bar{P}'_{\mathcal{W}}$. Dans ce cas, l'heuristique ne tire pas significativement profit de l'information disponible pour calculer le vecteur dual α désagrégé. Une fois un vecteur α réalisable obtenu, il est alors possible d'utiliser un algorithme de gradient projeté pour calculer un nouveau vecteur α' qui satisferait les contraintes d'optimalité duale d'un plus grand nombre de variables $p \in \bar{P}'_{\mathcal{W}}$ incompatibles avec la partition $\mathcal{W}$. À chaque étape de l'algorithme, on choisit une variable $p \in \bar{P}'_{\mathcal{W}}$ correspondant à une contrainte (6.15) violée au point dual α courant. La direction de descente est définie par la négation de la normale de la contrainte violée, projetée sur l'hyperplan (6.14). La longueur du pas de descente doit être limitée en fonction de l'ensemble des contraintes (6.15) déjà satisfaites, de façon que l'algorithme de gradient projeté ne détériore pas la qualité (mesurée en nombre de contraintes satisfaites) de la solution duale α obtenue aux itérations précédentes.

Décision du moment opportun pour modifier la partition : Le quatrième item concerne l'élaboration d'une stratégie robuste pour le prédicat `modifier` à l'étape 9 de l'algorithme. D'une part, la stratégie triviale consistant à toujours retourner `faux` minimise le nombre de changements de partition en empêchant la sortie de la boucle INTERNE dès qu'une variable de coût réduit négatif compatible avec la partition $\mathcal{W}$ est générée ($P''_{\mathcal{W}} \neq \emptyset$). D'autre part, la stratégie triviale consistant à toujours retourner `vrai` constitue l'autre extrême, menant à un changement de partition aussitôt qu'il existe une variable incompatible de coût réduit négatif. Il est cependant souhaitable de concevoir un prédicat qui estime l'impact d'un changement de partition sur l'évolution du processus de résolution.

Nous proposons la définition suivante, qui dépend d'un paramètre $\lambda \in \mathbb{R}_+$:

$$\texttt{modifier}(P_1, P_2, \alpha) = \begin{cases} \texttt{vrai} & \text{si } \frac{\min_{p \in P_1} \bar{c}_p(\alpha)}{\min_{p \in P_2} \bar{c}_p(\alpha)} < \lambda, \\ \texttt{faux} & \text{sinon,} \end{cases} \tag{6.18}$$

où P_1 est l'ensemble des variables générées par le sous-problème à l'étape 5 qui sont compatibles avec la partition $\mathcal{W}$ et P_2 est l'ensemble de toutes les variables générées aux itérations précédentes qui sont incompatibles avec la partition $\mathcal{W}$. Les variables $p \in P_1$ ont toutes un coût réduit strictement négatif et l'ensemble P_2 est modifié pour contenir l'ensemble P_1 à l'étape 8. Le prédicat `modifier` est donc bien défini pour l'usage qui en est fait dans l'algorithme. Ce prédicat repose sur l'intuition que la variable de plus petit coût réduit donne une indication sur l'amélioration de l'objectif si on ajoute cette variable à la formulation du problème-maître restreint. Les deux stratégies précédentes sont des cas particuliers de la dernière, en affectant au paramètre λ la valeur 0 et une très grande valeur respectivement. En pratique, la valeur de λ sera inférieure à 1 car on préfère réaliser une itération de réoptimisation de la restriction qui coûte moins cher que de modifier la partition en plus de réoptimiser.

Modification de la partition : Le cinquième et dernier item porte sur les critères balisant le choix d'une nouvelle partition à l'étape 13 de l'algorithme. Les contraintes sur les variables en base et la variable $\bar{p}$ incompatible de coût réduit négatif, couplées à la stratégie de raffinements obligatoires lors d'itérations dégénérées de la boucle EXTERNE à l'étape 13, assurent la convergence de l'algorithme d'agrégation dynamique. À moins que ces contraintes ne forcent à elles seules l'utilisation d'une partition formée uniquement de singletons, elles laissent une certaine latitude pour exploiter la structure sous-jacente de chemins du problème et contrôler l'efficacité de la suite du processus de résolution. L'heuristique proposée au deuxième item pour construire une partition $\mathcal{W}$ initiale assure que les ensembles W_l, $l \in L$, de cette partition correspondent à des sous-chemins qui héritent de l'ordre linéaire de leur chemin sous-jacent. Nous proposons une heuristique de modification de la partition $\mathcal{W}$ en deux phases qui préserve cette propriété.

La première phase consiste à consolider les ensembles W_l formant la partition $\mathcal{W}$. Pour

ce faire, on analyse la solution primale de base obtenue à l'étape 12 pour identifier des collections d'ensembles $\{W_l\} \subset \mathcal{W}$ telles que

- les ensembles W_l d'une collection soient couverts de façon atomique par toutes les variables de base de valeur strictement positive ;
- les ensembles W_l contiennent des sous-chemins contigüs d'un même chemin sous-jacent.

Il est alors possible de créer une partition consolidée $\mathcal{W}'$ formée d'ensembles $W_{l'}$ correspondant à la réunion des ensembles W_l de chaque collection. Le premier critère assure que la solution primale reste compatible avec la partition consolidée $\mathcal{W}'$, alors que le second critère préserve l'existence d'un ordre linéaire pour chaque ensemble de la partition. Cette opération a comme effet de restreindre l'ensemble des variables compatibles avec la partition.

La seconde phase de l'heuristique consiste à raffiner la partition consolidée $\mathcal{W}'$ pour incorporer dans l'ensemble des variables compatibles la variable $\bar{p}$ identifiée à l'étape 10, de même que le plus grand nombre possible de variables auparavant incompatibles avec la partition $\mathcal{W}$ et de coût réduit négatif en fonction des dernières variables duales obtenues à l'étape 4. Lors de cette seconde phase, la prise en compte de la variable $\bar{p}$ est nécessaire pour assurer la convergence de l'algorithme, mais le nombre de variables supplémentaires à intégrer aux variables compatibles dépend de la taille $|\mathcal{W}|$ visée. En effet, le nombre de contraintes actives dans le problème-maître restreint doit être assez petit pour permettre une résolution efficace à l'étape 3, sans ralentir la convergence globale de l'algorithme en impliquant un trop grand nombre d'itérations de la boucle EXTERNE.

Il n'est pas toujours possible de maintenir la taille $|\mathcal{W}|$ de la partition aux environs d'une valeur cible maximisant l'efficacité de l'optimiseur linéaire, compte tenu des contraintes sur la compatibilité de certaines variables à l'étape 13. Par exemple, si la solution linéaire du dernier problème-maître restreint n'est pas dégénérée, la

taille $|\mathcal{W}|$ de la partition ne peut qu'augmenter. De plus, si la solution linéaire du problème résiduel n'est pas dégénérée, la seule partition qui permette de l'obtenir est la partition en singletons. On remarque aussi l'effet similaire provoqué par l'application de la stratégie proposée pour contrer la dégénérescence de la boucle EXTERNE, qui pourrait mener à une partition compatible avec toutes les variables (possiblement une partition en singletons). Toutefois, dans le contexte de la résolution du modèle unifié au moyen d'un processus d'énumération, il n'est nullement indispensable d'obtenir la solution optimale du problème résiduel. On doit par contre en calculer une borne inférieure, ce qui peut être effectué après l'étape 4 de l'algorithme 6.3 en évaluant l'équation (2.7) en fonction du vecteur de variables duales α désagrégées. Ainsi, lorsque la taille de la partition devient trop grande, ou encore lorsque le temps alloué à l'algorithme est excédé, il est possible de retourner au processus d'énumération avec la meilleure borne inférieure calculée à ce point.

Conclusion

Ce chapitre a proposé diverses stratégies algorithmiques exploitant la présence d'un très grand nombre de contraintes de partitionnement pour réduire l'effort de calcul dans le problème-maître lors de la résolution d'un problème résiduel donné du modèle unifié. L'analyse de ces contraintes, à la section 6.1, présente une technique de reformulation et d'agrégation adaptée à la structure des problèmes de très grande taille qui sont visés par cette étude. Les deux algorithmes d'agrégation *statique* de la section 6.2, ainsi que l'analyse de leur complexité, sont des contributions originales de ce chapitre. La mise en œuvre des variantes les plus efficaces (en temps de calcul) montre que les gains sont importants pour certaines applications de très grande taille mais dépendent fortement de la structure des réseaux des problèmes traités. Leur analyse a aussi permis d'ordonner toute une famille d'algorithmes selon leur complexité et

de déterminer la possibilité de les utiliser en fonction des propriétés des problèmes traités. Finalement, la présentation à la section 6.3 d'un algorithme original d'agrégation *dynamique* pour résoudre un problème résiduel complète les contributions du chapitre. Cet algorithme intègre dans un processus itératif convergent les techniques de reformulation et d'agrégation des contraintes décrites à la première section et un algorithme modifié de génération de colonnes. L'algorithme d'agrégation dynamique est conçu de façon à permettre la résolution efficace du problème-maître au moyen des logiciels disponibles de programmation linéaire en contrôlant le nombre de contraintes de partitionnement dans le problème-maître.

Les développements informatiques correspondant aux différents algorithmes présentés dans ce chapitre sont incomplets. Cependant, deux variantes de l'algorithme heuristique d'agrégation statique ont déjà été développées et testées sur des problèmes de grande taille de différentes applications. Ces tests confirment que l'efficacité de ces variantes en nombre de contraintes éliminées dépend de l'existence de chaînes d'arcs dans les réseaux. Ces chaînes découlent, dans le cas des problèmes de rotations d'équipage, de l'utilisation d'heuristiques lors de la construction des réseaux. Les développements concernant l'algorithme d'agrégation dynamique sont avancés mais pas encore opérationnels. Sa mise en œuvre complète fera l'objet de travaux subséquents à cette thèse, entre autres dans le cadre de problèmes de réoptimisation quotidienne en transport aérien dont la taille est présentement un obstable important pour leur résolution et dont la forte structure de l'ensemble des tâches (enchaînement des vols) peut être exploitée à plusieurs étapes de l'algorithme.

Conclusion

Cette thèse comporte deux objectifs principaux. D'une part, l'objectif théorique consiste à analyser les conditions d'équivalence entre une formulation bloc-angulaire d'un problème non linéaire et une formulation de génération de colonnes accompagnée d'un oracle. La tranformation directe est étudiée dans le contexte du modèle unifié de Desaulniers *et al.* (1998), qui constitue un modèle générique pour les problèmes de tournées de véhicule et d'horaires d'équipage. Ce modèle sert de cadre théorique et de fil conducteur pour l'ensemble de la thèse. La transformation inverse est analysée dans le cadre d'un programme linéaire mixte généralisé accompagné d'un oracle. D'autre part, l'objectif expérimental vise le développement d'un logiciel permettant de résoudre des problèmes de très grande taille appartenant à un sous-ensemble important du modèle unifié. La réalisation de cet objectif passe par la mise en œuvre des deux dernières versions du logiciel GENCOL et suscite le développement de deux approches algorithmiques complémentaires pour traiter des problèmes comportant un très grand nombre de contraintes globales linéaires. Les prochains paragraphes rappellent les contributions majeures de la thèse qui ont été réalisées en vue de l'atteinte de ces objectifs. Nous proposons ensuite quelques avenues de recherche et de développement constituant selon le chercheur un prolongement naturel des travaux rapportés dans cette thèse.

Au plan théorique, la thèse contribue à éclaircir les relations entre les formulations dites *compacte* et *décomposée* d'un même problème en nombres entiers. Nous avons montré que, sous des conditions relativement faibles, on peut retrouver une formulation compacte équivalente à la formulation initiale du problème, qui intègre à la fois les contraintes du problème-maître et la structure de l'oracle. La formulation initiale peut alors être considérée comme une formulation décomposée, s'obtenant par

l'application d'une extension du principe de décomposition de Dantzig-Wolfe sur la formulation compacte. L'intérêt de cette contribution provient du potentiel qu'offre la formulation compacte pour exploiter la structure de l'oracle lors du développement d'algorithmes de séparation dans le contexte de la résolution du problème en nombres entiers.

Dans le cas d'un problème non linéaire à structure bloc-angulaire, nous avons utilisé l'exemple de la formulation compacte du modèle unifié de Desaulniers *et al.* (1998) pour présenter une extension du principe de décomposition de Dantzig-Wolfe permettant d'obtenir une formulation décomposée équivalente à la formulation compacte. La contribution consiste ici à clarifier et à formaliser le principe de décomposition de Dantzig-Wolfe en présence de contraintes non linéaires et de contraintes d'intégrité. L'analyse de la décomposition du modèle unifié a ainsi permis de corriger une omission de l'article de Desaulniers *et al.* (1998) concernant les fonctions d'extensions non convexes dans la formulation décomposée. Mentionnons aussi que la présentation du modèle unifié dans la thèse a été légèrement révisée par rapport à la version originale de Desaulniers *et al.* (1998) pour mieux tenir compte des différentes formes de sous-problèmes, et que cette nouvelle version peut prétendre devenir la référence concernant la formulation générale du modèle unifié.

Au plan expérimental, la thèse contribue d'une part à structurer la recherche et le développement sur le modèle unifié par la mise en œuvre des deux dernières versions du logiciel GENCOL. Le succès de cette réalisation se mesure aux nombreuses applications commerciales (plus de 50 depuis 1994) et publications de recherche (plus de 30 depuis 1992) utilisant ces versions de GENCOL comme optimiseur.

D'autre part, la thèse présente deux nouvelles approches algorithmiques pour traiter des problèmes de génération de colonnes comportant un très grand nombre de contraintes globales linéaires. Dans un premier temps, nous avons proposé un processus

de stabilisation combinant une méthode de perturbation et une méthode de pénalités exactes. L'originalité de notre approche est de définir le processus de stabilisation tout en restant dans le contexte de la programmation linéaire, et donc d'en retenir la simplicité et l'efficacité à chaque itération. L'intérêt de cet algorithme réside dans le fait que, pour chacune des applications étudiées, le processus de stabilisation a rendu possible soit une réduction significative des temps de résolution, soit la résolution de cas considérés de trop grande taille jusqu'à tout récemment.

Dans un deuxième temps, nous avons développé deux stratégies algorithmiques pour réduire le nombre de contraintes de partitionnement actives en même temps dans la formulation décomposée d'un problème formulé au moyen du modèle unifié. Nous avons d'abord présenté plusieurs algorithmes d'agrégation statique pour identifier et éliminer lors d'un prétraitement des contraintes de partitionnement redondantes. La contribution réside dans l'analyse de la complexité de chacun de ces algorithmes, permettant de pondérer le niveau de réduction du nombre de contraintes avec le temps de calcul des algorithmes. La mise en œuvre des variantes les plus efficaces (en temps de calcul) montre que les gains sont importants pour certaines applications de très grande taille mais dépendent fortement de la structure des réseaux des problèmes traités. Nous avons ensuite développé un algorithme d'agrégation dynamique pour contrôler le nombre de contraintes de partitionnement actives à chaque itération du processus de génération de colonnes. La contribution de notre approche consiste à exploiter systématiquement la forte structure des solutions optimales, découlant à la fois des propriétés des applications et des contraintes de partitionnement, pour réduire agressivement le nombre de contraintes actives dans le programme linéaire de la formulation décomposée. La mise en œuvre de l'algorithme, encore inachevée, laisse entrevoir la résolution de cas comportant jusqu'à 5 fois plus de contraintes de partitionnement que ceux traités avec les techniques actuelles. Nous visons plus particulièrement les problèmes de très grande taille de tournées de véhicule et d'horaires d'équipage où les tâches forment l'essentiel des contraintes et possèdent un

ordonnancement naturel pour guider le processus d'agrégation, comme c'est notamment le cas des problèmes de chauffeurs d'autobus et des problèmes de réoptimisation quotidienne d'horaires planifiés à moyen et long terme.

Nous terminons cette thèse en suggérant quelques avenues de recherche et de développement s'inscrivant en continuité avec les travaux déjà réalisés. Nous notons qu'il existe encore quelques écarts entre la présentation théorique du modèle unifié et les algorithmes utilisés pour sa résolution. Du côté du modèle, une meilleure caractérisation des contraintes sur les séquences de tâches, telles que mises en œuvre dans les algorithmes traitant les sous-problèmes, permettrait de formaliser la description des procédures de séparation exploitant cette notion. Du côté des algorithmes, une généralisation de l'algorithme de Ioachim (1998) pour traiter les problèmes comportant plusieurs ressources compléterait le processus de résolution du sous-ensemble principal du modèle unifié correspondant aux problèmes avec fonctions d'extension non décroissantes.

Plusieurs développements devraient aussi être envisagés au niveau du logiciel GENCOL afin de maintenir sa position de chef de file dans le domaine des optimiseurs commerciaux pour les problèmes de tournées de véhicule et d'horaires d'équipage. Nous recommandons évidemment de compléter la mise en œuvre de l'algorithme d'agrégation dynamique afin d'en vérifier l'efficacité espérée. Ces développements impliquent d'ailleurs des primitives communes avec le processus de stabilisation et permettraient de consolider la structure du logiciel dans les modules de gestion du programme linéaire. Nous considérons finalement que l'algorithme des centres analytiques de Goffin et Vial (1990) pourrait améliorer significativement la convergence du processus de génération de colonnes en complétant l'algorithme de Kelley (1960) et les stratégies de stabilisation développées dans le cadre de la thèse.

Bibliographie

ANSI (1989). *X3.159-1989.* American National Standards Institute. Broadway, New York, NY.

APPLEGREN, L.H. (1969). A Column Generation Algorithm for a Ship Scheduling Problem. *Transportation Science* **3**, 53–68.

BARNHART, C., JOHNSON, E.L., ANBIL, R. et HATAY, L. (1994). A Column Generation Technique for the Long-Haul Crew Assignment Problem. *Optimization in Industry 2 : Mathematical Programming and Modeling Techniques in Practice*, T.A. Ciriani et R. Leachman (éd.), John Wiley and Sons, 7–22.

BARNHART, C., JOHNSON, E.L., NEMHAUSER, G.L., SAVELSBERGH, M.W.P. et VANCE, P.H. (1998). Branch and Price : Column Generation for Solving Huge Integer Programs. *Operations Research* **46**, 316–329.

BAZARAA, M.S., JARVIS, J.J. et SHERALI, H.D. (1990). *Linear Programming and Network Flows.* John Wiley and Sons, New York, NY.

BEASLEY, J.E. (1985). A Note on Solving Large p-Median Problems. *European Journal of Operational Research* **21**, 270–273.

BEN AMOR, H. (1997). *Le problème de la découpe binaire.* Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

BENDERS, J.F. (1962). Partitioning Procedures for Solving Mixed-Variables Programming Problems. *Numerische Mathematik* **4**, 238–252.

BENTLEY, J.L. (1980). Multidimensional Divide-and-Conquer. *Communications of the ACM* **23**, 214–229.

BENTLEY, J.L., CLARKSON, K.L. et LEVINE, D.B. (1993). Fast Linear Expected-Time Algorithms for Computing Maxima and Convex Hulls. *Algorithmica* **9**, 168–183.

BERTOSSI, A.A., CARRARESI, P. et GALLO, G. (1987). On Some Matching Problems Arising in Vehicle Scheduling Models. *Networks* **17**, 271–281.

BIXBY, R.E., GREGORY, J.W., LUSTIG, I.J., MARSTEN, R.E. et SHANNO, D.F. (1992). Very Large-Scale Linear Programming : A Case Study in Combining Interior Point and Simplex Methods. *Operations Research* **40**, 885–897.

BODIN, L. et GOLDEN, B. (1981). Classification in Vehicle Routing and Scheduling. *Networks* **11**, 97–108.

BRIMBERG, J., HANSEN, P., MLADENOVIĆ, N. et TAILLARD, E.D. (1997). Improvements and Comparison of Heuristics for Solving the Multisource Weber Problem. Cahiers du GERAD G-97-37, École des Hautes Études Commerciales, Canada.

CARPANETO, D., DELL'AMICO, M., FISCHETTI, M. et TOTH, P. (1989). A Branch and Bound Algorithm for the Multiple Vehicle Scheduling Problem. *Networks* **19**, 531–548.

CARRARESI, P. et GALLO, G. (1984). Network Models for Vehicle and Crew Scheduling. *European Journal of Operations Research* **16**, 139–151.

CHAUNY, F., RATSIRAHONANA, L. et SAVARD, G. (1999). A Column Generation Approach to the Aircraft Loading Problem. Document de travail.

CHEN, P.C., HANSEN, P., JAUMARD, B. et TUY, H. (1998). Solution of the Multi-source Weber and Conditional Weber Problems by D.-C. Programming. *Operations Research* **46**, 548–562.

CHRISTOFIDES, N. et BEASLEY, J.E. (1982). A Tree Search Algorithm for the *p*-median problem. *European Journal of Operations Research* **10**, 196–204.

CORDEAU, J.-F., SOUMIS, F. et DESROSIERS, J. (1999). Simultaneous Assignment of Locomotives and Cars to Passenger Trains. À paraître dans *Transportation Science.*

CORMEN, H., LEISERSON, C. et RIVEST, R. (1990). *Introduction to Algorithms.* McGraw Hill, MIT Press.

CPLEX OPTIMIZATION INC. (1994). *Using the CPLEX Callable Library.* Incline Village, NV.

CPLEX OPTIMIZATION INC. (1998). *Using the CPLEX Callable Library.* Incline Village, NV.

DANTZIG, G.B. et WOLFE, P. (1960). Decomposition Principle for Linear Programs. *Operations Research* **8**, 101–111.

DESAULNIERS, G., DESROSIERS, J., DUMAS, Y., MARC, S., RIOUX, B., SOLOMON, M.M. et SOUMIS, F. (1997a). Crew Pairing at Air France. *European Journal of Operational Research* **97**, 245–259.

DESAULNIERS, G., DESROSIERS, J., IOACHIM, I., SOLOMON, M.M., SOUMIS, F. et VILLENEUVE, D. (1998). A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems. *Fleet Management and Logistics*, T.G. Crainic et G. Laporte (éd.), Kluwer, Norwell, MA, 57–93.

DESAULNIERS, G., DESROSIERS, J., LASRY, A. et SOLOMON, M.M. (1999). Crew Pairing for a Regional Carrier. *Computer-Aided Transit Scheduling*, N.H.M. Wilson (éd.), Lecture Notes in Economics and Mathematical Systems 471, Springer, Berlin, 19–41.

DESAULNIERS, G., DESROSIERS, J., LINGAYA, N.C., RANCOURT, E. et SAVARD, G. (1997b). *Scheduling-Routing for Air Force Resource Management : Reports on Tasks I, II and III.* Préparés pour le CRDV, Ministère de la Défense Nationale, Valcartier, Québec.

DESAULNIERS, G., DESROSIERS, J., DUMAS, Y., SOLOMON, M.M. et SOUMIS, F. (1997c). Daily Aircraft Routing and Scheduling. *Management Science* **43**, 841–855.

DESAULNIERS, G., LAVIGNE, J. et SOUMIS, F. (1998b). Multi-Depot Vehicule Scheduling Problems with Time Windows and Waiting Costs. *European Journal of Operational Research* **111**, 479–494.

DESAULNIERS, G. et VILLENEUVE, D. (1999). *The Shortest Path Problem with Time Windows and Linear Waiting Costs.* Soumis en août 1998 à *Transportation Science.*

DESROCHERS, M. (1986). *La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes.* Thèse de doctorat, Université de Montréal, Canada, Centre de Recherche sur les Transports, Publication 470.

DESROCHERS, M., DESROSIERS, J. et SOLOMON, M.M. (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research* **40**, 342–354.

DESROCHERS, M., LENSTRA, J.K. et SAVELSBERGH, M.W.P. (1990). A Classification Scheme for Vehicle Routing and Scheduling Problems. *European Journal of Operational Research* **46**, 322–332.

DESROCHERS, M., LENSTRA, J.K., SAVELSBERGH, M.W.P. et SOUMIS, F. (1988). Vehicle Routing with Time Windows : Optimization and Approximation. *Vehicle Routing : Methods and Studies*, B. Golden et A.A. Assad (éd.), North-Holland, Amsterdam, 65–84.

DESROCHERS, M. et SOUMIS, F. (1988a). A Generalized Permanent Labeling Algorithm for the Shortest Path Problem with Time Windows. *INFOR* **26**, 191–212.

DESROCHERS, M. et SOUMIS, F. (1988b). A Reoptimization Algorithm for the Shortest Path Problem with Time Windows. *European Journal of Operational Research* **35**, 242–254.

DESROCHERS, M. et SOUMIS, F. (1989). A Column Generation Approach to the Urban Transit Crew Scheduling Problem. *Transportation Science* **23**, 1–13.

DESROSIERS, J., DUMAS, Y., DESROCHERS, M., SOUMIS, F., SANSÒ, B. et TRUDEAU, P. (1991). A Breakthrough in Airline Crew Scheduling. In *Proceedings of the 26th Annual Meeting of the Canadian Transportation Research*, Québec, 464–478.

DESROSIERS, J., DUMAS, Y., SOLOMON, M.M. et SOUMIS, F. (1995). Time Constrained Routing and Scheduling. *Handbooks in Operations Research and Management Science, Volume 8 : Network Routing*, M.O. Ball *et al.* (éd.), North Holland, Amsterdam.

DESROSIERS, J., DUMAS, Y. et SOUMIS, F. (1986). A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-a-Ride Problem with Time Windows. *The American Journal of Mathematical and Management Sciences* **6**, 301–325.

DESROSIERS, J., DUMAS, Y. et SOUMIS, F. (1988). The Multiple Vehicle Dial-a-Ride Problem. *Computer-Aided Transit Scheduling*, J.R. Daduna et A. Wren (éd.), Lecture Notes in Economics and Mathematical Systems 308, Springer, Berlin, 15–27.

DESROSIERS, J., LASRY, A., MCINNIS, D., SOLOMON, M.M. et SOUMIS, F. (1999). ALTITUDE : The Airline Operations Management System at Air Transat. Cahiers du GERAD G-95-23, École des Hautes Études Commerciales, Canada. À paraître dans *Interfaces*.

DESROSIERS, J., PELLETIER, P. et SOUMIS, F. (1983). Plus court chemin avec contraintes d'horaires. *RAIRO* **17**, 357–377.

DESROSIERS, J., SAUVÉ, M. et SOUMIS, F. (1988). Lagrangian Relaxation Methods for Solving the Minimum Fleet Size Multiple Traveling Salesman Problem with Time Windows. *Management Science* **34**, 1005–1022.

DESROSIERS, J., SOUMIS, F. et DESROCHERS, M. (1984). Routing with Time Windows by Column Generation. *Networks* **14**, 545–565.

DIRICKX, Y.M.I. et JENNERGEN, L.P. (1979). *Systems Analysis by Multilevel Methods*. John Wiley and Sons, New York, NY.

DREZNER, Z., MEHREZ, A. et WESOLOWSKY, G.O. (1991). The Facility Location Problem with Limited Distances. *Transportation Science* **25**, 183–187.

DROR, M. (1994). Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW. *Operations Reseach* **42**, 977–978.

DU MERLE, O. (1995). *Points intérieurs et plans coupants : mise en œuvre et développement d'une méthode pour l'optimistion convexe et la programmation linéaire structurée de grande taille.* Thèse de doctorat, Université de Genève, Suisse, Thèse 414.

DU MERLE, O., VILLENEUVE, D., DESROSIERS, J. et HANSEN, P. (1999). Stabilized Column Generation. *Discrete Mathematics* **194**, 229–237.

DUMAS, Y. (1989). *Confection d'itinéraires pour le transport adapté.* Thèse de doctorat, Université de Montréal, Canada, Cahiers du GERAD G-89-44.

DUMAS, Y. (1999). Communication privée. AD OPT Technologies.

DUMAS, Y., DESROSIERS, J., GÉLINAS, É. et SOLOMON, M.M. (1995). An Optimal Algorithm for the Traveling Salesman Problem with Time Windows. *Operations Research* **43**, 367–371.

DUMAS, Y., DESROSIERS, J. et SOUMIS, F. (1991). The Pickup and Delivery Problem with Time Windows. *European Journal of Operational Research* **54**, 7–22.

ELZINGA, J. et MOORE, T.G. (1975). A Central Cutting Plane Algorithm for the Convex Programming Problem. *Mathematical Programming* **8**, 134–145.

ERMOLIEV, Y.M. (1966). Methods of Solution of Nonlinear Extremal Problems. *Cybernetics* **2**(4), 1–14.

FISHER, M.L. (1981). The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science* **27**, 1–18.

FORD, L.R. et FULKERSON, D.R. (1958). Suggested Computation for Maximal Multi-Commodity Network Flows. *Management Science* **5**, 97–101.

FULKERSON, D.R. (1972). Flow Networks and Combinatorial Operations Research. *Perspective on Optimization*, A.M. Geoffrion (éd.), Addison-Wesley, Reading, Ma, 197–220.

GAMACHE, M. (1995). *Fabrication d'horaires mensuels pour les membres d'équipages en transport aérien.* Thèse de doctorat. École Polytechnique de Montréal. Canada.

GAMACHE, M. et SOUMIS, F. (1993). A Method for Optimally Solving the Rostering Problem. Cahiers du GERAD G-93-40, École des Hautes Études Commerciales, Canada.

GAMACHE, M., SOUMIS, F., MARQUIS, G. et DESROSIERS, J. (1999). A Column Generation Approach for Large Scale Aircrew Rostering Problems. *Operations Research* **47**, 247–262.

GAMACHE, M., SOUMIS, F., VILLENEUVE, D., DESROSIERS, J. et GÉLINAS, É. (1998). The Preferential Bidding System at Air Canada. *Transportation Science* **32**, 246–255.

GARFINKEL, R.S., NEEBE, A.W. et RAO, M.R. (1974). An Algorithm for the *M*-Median Plant Location Problem. *Transportation Science* **8**, 217–236.

GÉLINAS, S., DESROCHERS, M., DESROSIERS, J. et SOLOMON, M.M. (1995). A New Branching Strategy for Time Constrained Routing Problems with Application to Backhauling. *Annals of Operations Research* **61**, 91–109.

GENTÈS, I. (1996). *Construction d'itinéraires quotidiens et hebdomadaires d'une flotte d'avions hétérogène.* Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

GEOFFRION, A.M. (1974). Lagrangian Relaxation for Integer Programming. *Mathematical Programming Study* **2**, 82–114.

GILL, E.P., MURRAY, W., SAUNDERS, M.A. et WRIGHT, M.H. (1989). Constrained Nonlinear Programming. *Handbooks in Operations Research and Management Science, Volume 1 : Optimization*, G.L. Nemhauser *et al.* (éd.), North-Holland, Amsterdam.

GILMORE, P.C. et GOMORY, R.E. (1961). A Linear Programming Approach to the Cutting-Stock Problem. *Operations Research* **9**, 849–859.

GOFFIN, J.-L. (1977). On the Convergence Rate of Subgradient Optimization. *Mathematical Programming* **13**, 329–347.

GOFFIN, J.-L., LUO, Z.-Q. et YE, Y. (1996). Complexity Analysis of an Interior Cutting Plane Method for Convex Feasibility Problems. *SIAM Journal on Optimization* **6**, 638–652.

GOFFIN, J.-L. et VIAL, J.-P. (1990). Cutting Planes and Column Generation Techniques with the Projective Algorithm. *Journal of Optimization Theory and Applications* **65**, 409–429.

GOFFIN, J.-L. et VIAL, J.-P. (1999). Convex Nondifferentiable Optimization : A Survey Focussed on the Analytic Center Cutting Plane Method. Cahiers du GERAD G-99-17, École des Hautes Études Commerciales, Canada.

GRAVES, G.W., MCBRIDE, R.D., GERSHKOFF, I., ANDERSION, D. et MAHIDHARA, D. (1993). Flight Crew Scheduling. *Management Science* **39**, 736–745.

HAASE, K., DESAULNIERS, G. et DESROSIERS, J. (1998). Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems. Soumis en octobre 1998 à *Transportation Science.*

HANSEN, P., JAUMARD, B. et POGGI DE ARAGÃO, M. (1991). Un algorithme primal de programmation linéaire généralisée pour les programmes mixtes. *Comptes Rendus de l'Académie des Sciences de Paris* **313**, 557–560.

HANSEN, P. et MLADENOVIĆ, N. (1998a). Variable Neighborhood Search for the p-Median. *Location Science* **5**. 207–226.

HANSEN, P. et MLADENOVIĆ, N. (1998b). An Introduction to Variable Neighborhood Search. In *Proceedings of the 2nd International Conference on Metaheuristics—MIC97*, S. Voss *et al.* (éd.), Kluwer, Dordrecht.

HANSEN, P., PEETERS, D., RICHARD, D. et THISSE, J.-F. (1985). The Minisum and Minimax Location Problems Revisited. *Operations Research* **33**, 1251–1265.

HARRIS, P.M.J. (1973). Pivot Selection Methods of the Devex LP Code. *Mathematical Programming* 5, 1–28.

HIRIART-URRUTY, J. et LEMARÉCHAL, C. (1993). *Convex Analysis and Minimization Algorithms II : Advanced Theory and Bundle Methods.* A Series of Comprehensive Studies in Mathematics. Springer-Verlag.

HOFFMAN, K.L. et PADBERG, M. (1993). Solving Airline Crew Scheduling Problems by Branch-and-Cut. *Management Science* **39**, 657–682.

HOUCK, D.J.JR, PICARD, J.C., QUEYRANNE, M. et VEMUGANTI, R.R. (1980). The Traveling Salesman Problem as a Constrained Shortest Path Problem : Theory and Computational Experience. *Opsearch* **17**, 93–109.

IOACHIM, I. (1994). *Planification des itinéraires d'une flotte d'avions avec contraintes de synchronisation d'horaires.* Thèse de doctorat, École Polytechnique de Montréal, Canada.

IOACHIM, I., DESROSIERS, J., DUMAS, Y., SOLOMON, M.M. et VILLENEUVE, D. (1995). A Request Clustering Algorithm for Door-to-Door Handicapped Transportation. *Transportation Science* **29**, 63–78.

IOACHIM, I., GÉLINAS, S., DESROSIERS, J. et SOUMIS, F. (1998). A Dynamic Programming Algorithm for the Shortest Path Problem with Time Windows and Linear Node Costs. *Networks* **31**, 193–204.

JAUMARD, B., SEMET, F. et VOVOR, T. (1999) A Two-Phase Resource Constrained Shortest Path Algorithm for Acyclic Graphs. Cahiers du GERAD G-96-48, École des Hautes Études Commerciales, Canada.

JEAN, A. (1996). *Plans de coupure pour des problèmes de multiflots dans des graphes acycliques.* Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

JUNGER, M., REINELT, G. et RINALDI, G. (1995). The Traveling Salesman Problem. *Handbooks in Operations Research and Management Science, Volume 7 : Network Models*, Ball M.O. *et al.* (éds.), North Holland, Amsterdam.

KARP, R.M. (1994). Probabilistic Recurrence Relations. *Journal of the ACM* **41**, 1136–1150.

KELLEY, J.E.JR (1960). The Cutting-Plane Method for Solving Convex Programs. *SIAM* **8**, 703–712.

KIWIEL, K.C. (1989). A Survey of Bundle Methods for Nondifferentiable Opimization. *Mathematical Programming : Recent Developments and Applications*, M. Iri et K. Tanabe (éd.), KTT/Kluwer, Tokyo, 263–282.

KOHL, N. (1995). *Exact Methods for Time Constrained Routing and Related Scheduling Problems.* Thèse de doctorat, Technical University of Denmark, Danemark, Institute of Mathematical Modelling IMM-PHD-1995-16.

KOHL, N., DESROSIERS, J., MADSEN, O.B.G., SOLOMON, M.M. et SOUMIS, F. (1999). 2-Path Cuts for the Vehicle Routing Problem with Time Windows. *Transportation Science* **33**, 101–116.

KOHL, N. et MADSEN, O.B.G. (1997). An Optimization Algorithm for the Vehicle Routing Problem with Time Windows based on Lagrangean Relaxation. *Operations Research* **45**, 395–406.

KOLEN, A.W.J., RINOOY KAN, A.H.G. et TRIENEKENS, H.W.J.M. (1987). Vehicle Routing with Time Windows. *Operations Research* **35**, 266–273.

KOTY, L. (1996). *Construction d'itinéraires d'une flotte d'avions hétérogène avec contraintes d'entretien.* Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

KRAU, S. (1997). *Extensions du problème de Weber.* Thèse de doctorat, École Polytechnique de Montréal, Canada.

KUNG, H.T., LUCCIO, F. et PREPARATA, F.P. (1975). On Finding the Maxima of a Set of Vectors. *Journal of the ACM* **22**, 469–476.

LACOURSIÈRE, F. (1992). *Heuristiques pour l'obtention de solutions entières à partir de solutions continues.* Mémoire de maîtrise. École Polytechnique de Montréal, Canada.

LAND, A. et POWELL, S. (1973). *Fortran Codes for Mathematical Programming : Linear, Quadratic and Discrete.* John Wiley and Sons, New York, NY.

LAPORTE, G. (1992). The Vehicle Routing Problem : An Overview of Exact and Approximate Algorithms. *European Journal of Operational Research* **59**, 345–358.

LASDON, L.S. (1970). *Optimization Theory for Large Systems.* MacMillan, New York, NY.

LASRY, A. (1996). *Rotations d'équipages pour un transporteur aérien régional.* Mémoire de maîtrise, École des Hautes Études Commerciales, Montréal, Canada.

LAURENT, J.-C., DESAULNIERS, G., MALHAMÉ, R. et SOUMIS, F. (1995). A Column Generation Method for Optimal Load Management via Control of Water Heaters. *IEEE Transactions on Power Systems* **10**, 1389–1400.

LAVOIE, S., MINOUX, M. et ODIER, É. (1988). A New Approach of Crew Pairing Problems by Column Generation and Application to Air Transport. *European Journal of Operational Research* **35**, 45–58.

LEMARÉCHAL, C., NEMIROVSKII, A. et NESTEROV, Y. (1991). *New Variants of Bundle Methods.* Rapport technique 1508, Institut National de Recherche en Informatique et en Automatique, France.

LINGAYA, N.C. (1996). Communication privée.

LÖBEL, A. (1997). Recent Computational Developments for Large-Scale Multiple-Depot Vehicle Scheduling Problems. In *Proceedings of the 7th International Workshop on Computer-Aided Scheduling of Public Transport.*

LUENBERGER, D.G. (1989). *Linear and Nonlinear Programming.* Second Edition, Addison-Wesley, Reading, Mass.

MARSTEN, R.E. (1981). The Design of the XMP Linear Programming Library. *ACM Transactions on Mathematical Software* **7**, 481–497.

MARSTEN, R.E., HOGAN, W.W. et BLANKENSHIP, J.W. (1975). The BOXSTEP Method for Large-Scale Optimization. *Operations Research* **23**, 389–405.

MESSIE, K. (1995) *Problème hebdomadaire d'affectation de locomotives aux trains.* Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

MILLER, C.E., TUCKER, A.W. et ZEMLIN, R.A. (1960). Integer Programming Formulations and Traveling Salesman Problems. *Journal of the ACM* **7**, 326–329.

MLADENOVIĆ, N. et HANSEN, P. (1997). Variable neighborhood search. *Computers and Operations Research* **24**, 1097–1100.

NAGIH, A. et SOUMIS, F. (1999). *L'agrégation des contraintes de ressources dans un problème de plus court chemin.* Cahiers du GERAD G-99-02, École des Hautes Études Commerciales, Canada.

NAZARETH, J.L. (1987). *Computer Solution of Linear Programs.* Oxford University Press, New York, New York.

NEMHAUSER, G.L. et WIDHELM, W.B. (1971). A Modified Linear Program for Columnar Methods In Mathematical Programming. *Operations Research* **19**, 1051–1060.

NEMHAUSER, G.L. et WOLSEY, L.A. (1988). *Integer and Combinatorial Optimization.* John Wiley and Sons, New York, NY.

PIRES, J.M. (1997). *Développement de méthodes parallèles pour des problèmes de grande taille.* Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

PIRES, J.M. et VILLENEUVE, D. (1999). *The* UTIL *Library (Version util-4.4.12).* Rapport technique, GERAD.

POLYAK, B.T. (1967). A General Method of Solving Extremum Problems. *Soviet Mathematics* **8**, 593–597.

PSARAFTIS, H. (1980). A Dynamic Programming Solution to the Single-Vehicle, Many-to-Many, Immediate Request Dial-a-ride Problem. *Transportation Science* **14**, 130–154.

PSARAFTIS, H. (1983). An Exact Algorithm for the Single-Vehicle Many-to-Many Dial-a-ride Problem with Time Windows. *Transportation Science* **17**, 351–357.

RIBEIRO, C. et SOUMIS, F. (1994). Column Generation Approach to the Multiple Depot Vehicle Scheduling Problem. *Operations Research* **42**, 41–52.

ROCHON, V. (1997). *Ajustement des variables duales dans le contexte d'une méthode génération des colonnes.* Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

ROSING, K.E. (1992). An Optimal Method for Solving the (Generalized) Multi-Weber Problem. *European Journal of Operational Research* **58**, 414–426.

ROUSSEAU, J.-M. et DESROSIERS, J. (1995). Results Obtained with CREW-OPT, a Column Generation Method for Transit Crew Scheduling. *Computer-Aided Transit Scheduling*, J.R. Daduna *et al.* (éd.), Lecture Notes in Economics and Mathematical Systems 430, 349–358.

ROUSSEAU, J.-M., LESSARD, R. et BLAIS, J.-Y. (1985). Enhancement to the HASTUS Crew Scheduling Algorithm. *Computer Scheduling of Public Transport 2*, J.-M. Rousseau (éd.), Elsevier, North-Holland.

RYAN, D.M. et FOSTER, B.A. (1981). An Integer Programming Approach to Scheduling. *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, A. Wren (éd.), North-Holland, Amsterdam, 269–280.

SGAIER, N. (1993). *Ordonnancement de la production dans un atelier de type « job-shop » avec machines à traitement par lots.* Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

SHOR, N.W. (1985). *Minimization Methods for Nondifferentiable Function.* Springer-Verlag, Berlin.

SLEATOR, D.D. et TARJAN, R.E.. (1985). Self-adjusting Binary Search Trees. *Journal of the ACM* **32**, 652-686.

SOLOMON, M.M. et DESROSIERS, J. (1988). Time Window Constrained Routing and Scheduling Problems. *Transportation Science* **22**, 1–13.

STOJKOVIĆ, G. (1999). *Gestion journalière d'une flotte d'avions.* Thèse de doctorat, École Polytechnique de Montréal, Canada.

STOJKOVIĆ, M., SOUMIS, F. et DESROSIERS, J. (1998). The Operational Airline Crew Scheduling Problem. *Transportation Science* **32**, 232–245.

VANCE, P.H., BARNHART, C., JOHNSON, E.L. et NEMHAUSER, G.L. (1994). Airline Crew Scheduling : A New Formulation and Decomposition Algorithm. Working paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia.

VANDERBECK, F. (1994). *Decomposition and Column Generation for Integer Programs.* Thèse de doctorat, Université Catholique de Louvain, Faculté des Sciences Appliquées, Louvain-la-Neuve, Belgique.

VANDERBECK, F. et WOLSEY, L.A. (1996). An Exact Algorithm for IP Column Generation. *Operations Research Letters* **19**, 151–160.

VOVOR, T. (1997). Problèmes de chemins bicritères ou avec contraintes de ressource : algorithmes et applications. Thèse de doctorat, École Polytechnique de Montréal, Canada.

VEINOTT, A.F.JR (1967). The Supporting Hyperplane Method for Unimodal Programming. *Operations Research* **15**, 147–152.

YE, Y. (1997). *Interior Point Algorithms : Theory and Analysis.* Wiley-Interscience series in Discrete Mathematics and Optimization. John Wiley and Sons, New York, NY.

ZIARATI, K., SOUMIS, F., DESROSIERS, J., GÉLINAS, S. et SAINTONGE, A. (1997). Locomotive Assignment with Heterogeneous Consists at CN Noth America. *European Journal of Operational Research* **97**, 281–292.

ZIARATI, K., SOUMIS, F., DESROSIERS, J. et SOLOMON, M.M. (1999). A Branch-First, Cut-Second Approach for Locomotive Assignment. *Management Science* **45**, 1156–1168.

Annexe A

Décomposition de Dantzig-Wolfe et énumération incomplète des points extrêmes

Le problème P étudié au chapitre 3 est caractérisé par un oracle dont la fonction consiste à retourner les coefficients d'une variable de coût réduit minimum, étant donné un vecteur de variables duales. Ce chapitre montre par un exemple (le problème Ω) qu'il est possible qu'un tel oracle ne puisse pas générer les coefficients de toutes les variables de la relaxation linéaire P_R du problème P. On note par J l'ensemble des indices des variables du problème P_R et par J^* le sous-ensemble des indices correspondant aux variables pouvant être générées par l'oracle. Le résultat précédent s'écrit ainsi : $J^* \subseteq J$ et il existe des cas où $J^* \neq J$.

On suppose maintenant que le problème P_R est obtenu en appliquant le principe de décomposition de Dantzig-Wolfe (1960) sur une formulation linéaire compacte. Ce principe de décomposition consiste à remplacer la description du domaine du sous-problème comme intersection de demi-espaces par une description équivalente comme combinaison convexe des points extrêmes et combinaison linéaire des rayons extrêmes de ce domaine. On note par $\widehat{J}$ le sous-ensemble des indices des variables du problème P_R qui correspondent à des points extrêmes du domaine, et on peut écrire l'affirmation suivante : $\widehat{J} \subseteq J$ et il existe des cas où $\widehat{J} \neq J$.

On remarque que le principe de décomposition de Dantzig-Wolfe n'implique pas

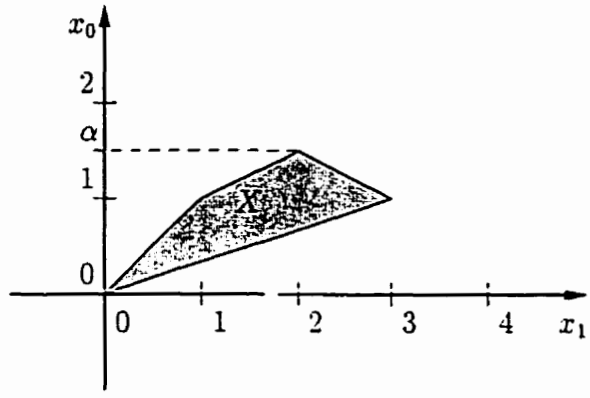


Figure A.1 – Domaine X de l'oracle.

que le problème décomposé soit résolu par génération de colonnes au moyen d'un oracle, bien que ce soit l'approche utilisée en pratique à cause du trop grand nombre de variables souvent obtenu. Cependant, dans le cas de la génération de colonnes, il n'est pas garanti que l'oracle utilisé puisse générer les coefficients de toutes les variables de l'ensemble $\widehat{J}$. En fait, l'exemple suivant prouve le contraire, à savoir qu'il existe des problèmes obtenus selon le principe de décomposition de Dantzig-Wolfe pour lesquels l'oracle ne peut générer tous les points extrêmes. La figure A.1 illustre graphiquement le domaine utilisé comme sous-problème dans l'exemple du problème Γ suivant, avec $\alpha > 1$:

$$(\Gamma) \qquad \min x_0$$

sujet à :

$$x_1 = 2$$
$$(x_0, x_1) \in X.$$

L'application du principe de décomposition de Dantzig-Wolfe sur le problème Γ mène au problème décomposé suivant, où les variables y_0, y_1, y_2 et y_3 correspondent aux

4 points extrêmes du domaine X :

$$\min y_1 + \alpha y_2 + y_3$$

sujet à :

$$\begin{aligned} y_1 + 2y_2 + 3y_3 &= 2 \\ y_0 + y_1 + y_2 + y_3 &= 1 \\ y_0, y_1, y_2, y_3 &\geq 0. \end{aligned}$$

Or, par comparaison avec le problème Ω du chapitre 3, on voit que l'oracle ne peut générer la variable y_2 lorsque $\alpha > 1$.

On peut donc ajouter aux relations déjà obtenues qu'il existe des cas où $\widehat{J} \subsetneq J^*$. Ce résultat complète celui obtenu au chapitre 3, dans le cas particulier où le problème de génération de colonnes provient de l'application du principe de décomposition de Dantzig-Wolfe sur une formulation compacte.