PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Algorithmes de résolution du problème de plus court chemin avec contraintes de ressources
Auteur: Author:	Yassir Miladi
Date:	2007
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Miladi, Y. (2007). Algorithmes de résolution du problème de plus court chemin avec contraintes de ressources [Ph.D. thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8000/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/8000/
Directeurs de recherche: Advisors:	François Soumis
Programme: Program:	Unspecified

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

NOTE TO USERS

This reproduction is the best copy available.

UMI®

UNIVERSITÉ DE MONTRÉAL

ALGORITHMES DE RÉSOLUTION DU PROBLÈME DE PLUS COURT CHEMIN AVEC CONTRAINTES DE RESSOURCES

YASSIR MILADI
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION DU DIPLÔME DE
PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES DE L'INGÉNIEUR)
DÉCEMBRE 2007

© Yassir Miladi, 2007.

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-37131-2
Our file Notre référence
ISBN: 978-0-494-37131-2

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

Algorithmes de résolution du problème de plus court chemin avec contraintes de ressources

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

M. DESAULNIERS Guy, Ph.D., président
M. SOUMIS François, Ph.D., membre et directeur de recherche
M. ROUSSEAU Louis-Martin, Ph.D., membre
M. RENAUD Jacques, Ph.D., membre externe

À mon père...

REMERCIEMENTS

En premier lieu, je voudrais exprimer mon extrême gratitude envers M. François Soumis pour le support qu'il m'a apporté et la confiance qu'il m'a témoignée tout au long de cette thèse. Sa disponibilité, ses précieux conseils, sa grande rigueur, sa sensibilité humaine et ses encouragements m'ont permis d'avancer de façon constructive et rapide, tout en conservant une certaine autonomie dans mon travail. Il m'a permis de travailler sur un projet d'une aussi grande envergure et de découvrir la pratique de la génération de colonnes pour résoudre des problèmes de grande taille.

Je remercie également M. Guy Desaulniers de bien vouloir présider mon jury, ainsi que M. Louis-Martin Rousseau d'avoir accepté d'être membre de ce jury.

Merci à tous mes collègues de travail, dont Issmail Elhallaoui pour son aide précieuse dans l'utilisation des modules de GENCOL et M. Maurice Morel pour les échanges constructifs sur nos problèmes respectifs.

Enfin, un merci spécial à ma famille pour leur encouragement, leur soutien et leur patience durant tous les moments difficiles de mes années d'études.

RÉSUMÉ

Le problème de plus court chemin avec contraintes de ressources consiste à trouver un chemin d'un noeud origine à un noeud destination, de coût minimum et respectant les contraintes sur les consommations de ressources en chaque noeud du réseau.

Ce problème apparaît comme sous-problème à l'intérieur d'un algorithme de génération de colonnes. Lors de la résolution des applications de grande taille, par un algorithme de génération de colonnes, on peut avoir des centaines de sous-problèmes ayant des milliers de noeuds et des dizaines de milliers d'arcs.

Les algorithmes les plus efficaces pour résoudre ce problème sont souvent les algorithmes du type programmation dynamique. Un tel algorithme consiste à associer un état (étiquette) à chaque chemin. Ainsi, à chaque itération, on prolonge l'ensemble des étiquettes en un noeud vers les noeuds successeurs et, par la suite, on élimine les étiquettes dominées. La complexité d'un tel algorithme augmente exponentiellement avec le nombre de ressources. Dans les applications industrielles de grande taille, la résolution de chacun de ces problèmes peut nécessiter beaucoup de temps, même que, dans bien des cas et lorsqu'on a un grand nombre de ressources, on n'est pas capable de résoudre ces problèmes à l'optimalité.

Afin d'obtenir rapidement de bonnes solutions, il est impératif de trouver des heuristiques pour réduire l'espace des états, sans détériorer la qualité des solutions. Une première heuristique utilisée consiste à dominer sur seulement un petit sous-ensemble de ressources. Cette approche est statique et ne fournit pas toujours des solutions de bonne qualité. Ainsi, on se propose dans ce travail d'explorer différentes méthodes dans le but d'améliorer cette heuristique dans le contexte d'un algorithme de génération de colonnes. Ces heuristiques qu'on se propose d'étudier doivent fournir de meilleures solutions que les approches statiques, de plus elles ne doivent pas être coûteuses en temps de calcul.

Dans une première partie, on se propose d'étudier la méthode de Nagih et Soumis (2006), qui a pour but la réduction de l'espace des états par agrégation des ressources. Cette méthode consiste à projeter les ressources sur un vecteur de dimension inférieure, puis d'ajuster dynamiquement la matrice de projection en utilisant des multiplicateurs de Lagrange sur les noeuds, ou aussi sur les arcs. À la fin de cette partie, on prouve que cette méthode a une convergence aléatoire et ne fournit pas les résultats escomptés. On argumente le tout par quelques essais numériques.

Dans la deuxième partie de ce travail, on propose une nouvelle méthode dynamique et flexible de résolution pour le problème de plus court chemin avec fenêtres de temps (une seule ressource). Cette méthode consiste à commencer par une borne supérieure obtenue par la résolution d'un plus court chemin sur un réseau des états restreint, puis on augmente graduellement ce réseau, à l'aide d'opérations de mises à jour, afin d'améliorer la borne supérieure. Cette méthode, en plus d'être dynamique donne de meilleures solutions que la méthode qui consiste en la dominance uniquement sur les coûts. On présente à la fin de cette partie plusieurs résultats numériques pour différentes variantes heuristiques de cette méthode.

Dans la troisième partie, on introduit une amélioration de la méthode proposée en utilisant des variables duales sur les noeuds. Cette méthode garantit une amélioration de la borne supérieure à chaque itération, en utilisant une sélection plus dynamique des méthodes de mise à jour et ceci en utilisant les valeurs duales. Plusieurs résultats numériques montrant les améliorations obtenues sont par la suite exposés.

Dans la quatrième partie, on propose une méthode lagrangienne qui permet d'obtenir une bonne borne inférieure. Cette méthode, inspirée de celle qu'on a explorée lors de la première partie, peut être très efficace pour réduire l'espace des états en chaque noeud lors de l'application de la méthode d'amélioration de la borne supérieure.

Finalement, on propose des extensions de la méthode d'amélioration de la borne

supérieure pour le problème général de plus court chemin avec contraintes de ressources et pour des fonctions d'extension non linéaires.

ABSTRACT

The shortest path problem with resource constraints consists in finding a path from an origin point to a destination point at minimum cost while respecting constraints on the use of resources on each node. This problem appears as a subproblem in a large number of column generation algorithms. When we solve large-scale industrial problems, using a column generation algorithm, we can have hundreds of subproblems with thousands of nodes and hundreds thousands of arcs.

An efficient algorithm for solving this problem is a label setting algorithm belonging to the class of dynamical programming algorithms. This algorithm associates a label (state) to every path, prolongs every label to the successor nodes and eliminates the dominated labels. The complexity of a dynamic programming algorithm increases exponentially with the number of resources. In the large-scale industrial applications, solving those problems usually takes a long time and in many cases we are not able to solve them to optimality.

A first heuristic currently used to quickly produce feasible solutions consists in dominating only on a subset of resources. This method is static and doesn't provide a good quality solutions. Our goal in this work is to explore new methods to improve the latter heuristic and provide better solutions in the column generation algorithm context. Moreover the approaches we want to develop have to be relatively low time consuming.

In the first part of this thesis we study the method proposed by Nagih and Soumis (2006). This method consists of aggregating resources at each node by projecting them on a vector of smaller dimension and using a Lagrangian relaxation algorithm to determine the coefficients of the projection. We then evaluate the method described by Nagih and Soumis (2006) by proving that the method for selecting the multipliers

of the projection at the nodes does not give results as good as the heuristic that it proposes to improve and that it is sensitive to random parameters. Moreover, we explain why the results obtained are very unstable. Then we provide some numerical results.

In the second part, we propose a flexible and dynamic algorithm for solving the shortest path problem with time windows (one resource only). This method consists of starting with an upper bound obtained by solving a shortest path problem on a restrained state space. Then we gradually increase the size of this network by applying update operations on the nodes, which decreases the bound value. In the end of this part, we present some numerical results based on some heuristics of this method.

In the third part we propose an improvement of the latter method by introducing dual variables on the nodes. This improvement insures that the upper bound will decrease at every iteration of the method. At the end of this part, we support these improvements with few numerical results.

Finally, we propose a Lagrangian method which gives a good lower bound. This method, which is inspired by the first part of this work, will lead to reducing the state space and consequently having less iterations in the method we already proposed. We also propose some extensions of the method to the general case of the shortest path problem with resources constraints and for any type of extension functions. We prove that the size of the state space networks increase linearly along the iterations.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ALGORITHMES

CHAPITRE 1

INTRODUCTION

Le problème de plus court chemin est l'un des problèmes de graphe les plus étudiés. Depuis les algorithmes de Bellman et de Dijkstra, plusieurs chercheurs se sont penchés sur différentes variantes des problèmes de plus court chemin et sur des algorithmes efficaces pour les résoudre. Selon le contexte et l'application, plusieurs de ces problèmes sont résolus avec des algorithmes de complexité polynomiale. Cependant dans plusieurs cas, il n'existe pas d'algorithmes polynomiaux et efficaces pour les très grandes instances. Les temps de résolution peuvent devenir trop grands lorsque ces problèmes de plus court chemin apparaissent comme des sous-problèmes et doivent être résolus plusieurs fois. C'est dans cette optique qu'on s'est proposé d'étudier ces problèmes. En particulier, dans cette thèse, on s'est intéressé aux problèmes de plus court chemin avec contraintes de ressources (voir ci-bas pour la définition de ressources). Pour ce problème, il n'existe aucun algorithme polynomial qui le résout de façon optimale. Dans les applications industrielles où on a de très grandes instances avec un grand nombre de ressources, la résolution de ces problèmes devient très fastidieuse et il arrive souvent qu'on ne soit pas capable de les résoudre jusqu'à l'optimalité.

Étant donné un réseau connexe ayant des coûts et des consommations de ressources sur les arcs et des contraintes en chaque noeud sur ces ressources, le problème de plus court chemin avec contraintes de ressources consiste à trouver un chemin de coût minimum entre un noeud source et un noeud destination qui respecte les contraintes de chacune des ressources en chaque noeud. Ainsi, à chaque chemin de l'origine au noeud i, on fait correspondre un vecteur de consommation de ressources $(T_i^1, T_i^2, ..., T_i^n)$, n étant le nombre de ressources considérées, qui est

le cumul de consommation de ressources sur chacun des arcs constituant le chemin. Chaque consommation de ressource T_i^r doit respecter les fenêtres de ressource en chaque noeud : $T_i^r \in [a_i^r, b_i^r]$, $r = 1, ..., n$. Dans le cas où on a une seule ressource, le problème est appelé plus court chemin avec fenêtres de temps (PCC-FT).

Selon les applications, ces contraintes de ressources peuvent être des fenêtres de temps (nombre de minutes de travail, temps alloué à la maintenance des véhicules, temps de pause, etc.). Cela peut être aussi un nombre de quarts de travail, un nombre de tâches effectuées par un véhicule, un nombre de vols ou même aussi la taille du chargement d'un véhicule. Les coûts sur les arcs sont généralement des coûts d'opération ou même de repos.

Le problème du plus court chemin avec contraintes de ressources est un problème NP-difficile, même dans le cas d'une seule ressource, comme le montrent Handler et Zang (1980).

Dans ce travail, l'étude du problème du plus court chemin avec contraintes de ressources en chaque noeud s'inscrit dans un problème plus général. Le problème général porte sur la résolution par génération de colonnes de problèmes, comportant un très grand nombre de variables, associées à des tournées de véhicules, des horaires d'équipages, etc. Les problèmes de plus court chemin avec contraintes de ressources apparaissent comme sous-problèmes dans les algorithmes de génération de colonnes pour résoudre ces problèmes.

La résolution de ces derniers vise à minimiser les coûts d'opération d'une flotte de véhicules (avions, autobus, ambulances, etc.) ou d'une équipe de travail lors de la réalisation d'un ensemble de tâches prédéterminées. En 1998, Desaulniers et al. ont proposé un modèle générique de flot non linéaire multi-commodité, appelé le *modèle unifié*, dans le but de modéliser une grande classe de problèmes déterministes de tournées de véhicules et d'horaires d'équipages. Les auteurs ont proposé un algorithme de génération de colonnes pour résoudre ces problèmes.

L'algorithme de génération de colonnes consiste à résoudre alternativement un problème maître restreint et plusieurs sous-problèmes. Les sous-problèmes sont des problèmes de plus court chemin avec contraintes de ressources (PCC-CR). Plusieurs recherches sur la résolution des différentes variantes des problèmes de plus court chemin ont été effectuées.

Desrochers (1986) et Desrochers et Soumis (1988a) proposent un algorithme de programmation dynamique du type "pulling" afin de résoudre les problèmes de plus court chemin avec contraintes de ressources. En chaque noeud, l'algorithme génère de nouvelles étiquettes qui correspondent aux coûts et aux consommations de ressources des chemins atteignant ces noeuds. Ce type d'algorithme est reconnu pour son efficacité lorsque le problème de plus court chemin avec contraintes de ressources apparaît comme sous-problème dans un algorithme de génération de colonnes, étant donné que l'algorithme peut générer plusieurs solutions réalisables en même temps. Cependant, pour les problèmes de très grande taille, il est souvent impossible de finir complètement la résolution.

La complexité de cette méthode augmente exponentiellement avec le nombre de ressources du problème. Des stratégies d'accélération sont nécessaires. Ainsi la recherche s'est essentiellement orientée vers le développement d'algorithmes, heuristiques et exacts, pour la réduction de l'espace des étiquettes (appelé aussi espace des états) dans le but de générer des solutions réalisables qui seront envoyées au problème maître.

Selon l'application, des stratégies «statiques» ont été utilisées. Dans le cas pratique, on utilise la dominance sur un sous-ensemble de ressources. Ces sous-ensembles sont modifiés, au cours des itérations afin de générer un ensemble de colonnes de qualités variées tout en gardant des temps de résolution raisonnables et jusqu'à épuisement du temps de résolution alloué ou jusqu'à épuisement de la mémoire disponible. Étant «gloutonnes», ces stratégies sont souvent difficiles à définir et à ajuster de façon efficace et doivent être révisées et validées régulièrement.

C'est dans cette optique que Nagih et Soumis (1999), (2000) et (2006) ont développé un algorithme de «relaxation» qui consiste en la projection de l'espace des étiquettes sur un espace de dimension plus petite. Bien que cet algorithme ne fournisse pas une solution optimale, il permet de générer des chemins réalisables à ajouter au problème maître.

Puisque le présent travail s'inscrit dans le cadre d'un problème plus général qui est la résolution de grandes applications de transport, en utilisant un algorithme de génération de colonnes, nos recherches se sont surtout concentrées sur les outils et moyens de trouver de nouvelles techniques efficaces pour réduire l'espace des états dans les problèmes de plus court chemin avec contraintes de ressources.

Les méthodes de résolution qu'on se propose de développer doivent être «robustes». Cette «robustesse» est relative aux principales méthodes qui ont déjà été développées dans la littérature et en industrie. En effet, les méthodes développées, surtout en industrie, (par KRONOS *Inc*) qui consistent en la dominance sur un sous-ensemble de ressources (en changeant de sous-ensemble de ressources au besoin), sont statiques ; la qualité des solutions fournies dépend des stratégies choisies par l'analyste et des jeux de données disponibles. Ces stratégies peuvent être, par exemple, de considérer, lors de la dominance seulement, une ressource parmi un sous-ensemble de ressources corrélées. Aussi, on peut ne pas dominer sur une ressource peu contraignante (c'est-à-dire qui ne permet pas l'élimination d'une quantité importante de chemins lors de la résolution).

Ainsi, une méthode «robuste», selon nos critères, est telle que la valeur Z^{Meth} de l'objectif fournie par une telle méthode vérifie $Z^{Opt} \leq Z^{Meth} < Z^{ND}$, où Z^{Opt} est la valeur de la solution optimale et Z^{ND} est la valeur de la solution fournie par les stratégies de dominance sur un sous-ensemble de ressources. C'est-à-dire, une méthode qui fait mieux que la dominance sur un sous-ensemble de ressources et s'approche de la solution optimale sans pour autant nécessiter trop de temps de calculs.

La méthode proposée par Nagih et Soumis (2006) avait cet objectif. Cependant, comme on va le montrer plus tard, la qualité des solutions trouvées par cette méthode est très variable. De plus, on prouvera que cette méthode est très instable. Elle peut même produire de moins bonnes solutions que les stratégies de dominance sur un sous-ensemble de ressources tout en utilisant beaucoup plus de temps de calculs que cette dernière.

La relaxation de l'espace des états a été très peu traitée dans la littérature, il n'y a pas eu de discussion dans ces travaux sur la notion de robustesse.

Même si on n'est pas toujours capable de résoudre ces problèmes jusqu'à optimalité à cause de leur grande taille, notre but principal est de fournir les meilleures solutions réalisables, qui seront fournies par la suite au problème maître dans un algorithme de génération de colonnes.

Dans une première partie de ce travail, on a exploré la méthode proposée par Nagih et Soumis (1999), (2000) et (2006), qui est une méthode d'agrégation du type lagrangien. Cette méthode consiste en la relaxation des contraintes de ressources à chaque noeud en utilisant une projection dynamique sur un espace d'états de dimension inférieure à celle de l'espace original. Les auteurs utilisent des matrices de projection dont les multiplicateurs sont donnés par une relaxation lagrangienne calculée en chaque noeud du réseau. Le problème est ensuite résolu par un algorithme de programmation dynamique, similaire à celui de Desrochers et Soumis (1988a), en considérant une fonction de coût lagrangienne et en dominant sur le reste des ressources non projetées. Par la suite, la méthode proposée effectue, au fil des itérations de l'algorithme de génération de colonnes, l'ajustement des multiplicateurs par la méthode lagrangienne dans un algorithme de réoptimisation. Une étude détaillée de cette approche sera exposée ainsi que les variantes possibles qui utilisent cette approche.

Une critique de ce travail a été par la suite formulée. Cette critique porte sur la sensibilité de la méthode, proposée par Nagih et Soumis (1999), (2000) et (2006),

à la variation même minime des paramètres dont elle dépend. Par conséquent, il a été impossible d'appliquer cette théorie étant donné les failles théoriques et algorithmiques. Des tableaux de valeurs seront exposés à la fin de cette partie avec différents problèmes tests pour appuyer la critique de façon expérimentale.

Convaincu de l'inefficacité de la méthode «duale» proposée par Nagih et Soumis (1999), (2000) et (2006), on a exploré, dans une deuxième partie, une nouvelle approche de résolution du problème qui est du type «primale». Dans cette approche, on commence par résoudre un problème de plus court chemin sur un réseau restreint associé au réseau initial considéré. Cette résolution initiale fournit généralement des solutions réalisables. Après, on augmente itérativement ce réseau en d'autres réseaux sur lesquels on applique un algorithme de plus court chemin ordinaire, pour obtenir de meilleures solutions. Théoriquement, ce processus itératif permet d'obtenir la solution optimale.

Cet algorithme, bien qu'étant lui aussi pseudo-polynomial, possède plusieurs avantages. Premièrement, on peut obtenir rapidement des solutions réalisables qu'on peut envoyer au problème maître dans le cadre de l'algorithme de génération de colonnes. On peut aussi arrêter la résolution selon la capacité de nos ressources informatiques ou selon les temps alloués à la résolution, contrairement à un algorithme de programmation dynamique où on doit attendre la fin de la résolution pour pouvoir avoir des solutions.

De plus, dans le cas où on résout le problème en dominant uniquement sur le coût, lorsque le chemin optimal est dominé par un petit nombre de chemins, la complexité peut être très basse, contrairement à l'algorithme de programmation dynamique proposé par Desrochers (1986) et Desrochers et Soumis (1988a).

L'algorithme original proposé procède, à chaque itération, de façon systématique à la génération de nouveaux chemins réalisables. L'algorithme ne privilégie aucun

type particulier de chemins. Il est par la suite plus fructueux de donner des critères préalables de génération de ces chemins afin d'améliorer la solution courante au fil des itérations. Étant donné la grande flexibilité qu'offre cet algorithme, on peut imposer des stratégies d'accélération plus dynamiques.

Une première stratégie qu'on propose est de considérer les améliorations les plus fructueuses en premier lieu. On attribue à chaque noeud des variables duales qui mesurent leurs potentiels pour améliorer la solution courante. On donne priorité ainsi à des solutions avant d'autres afin de réduire plus vite l'écart avec la valeur optimale. On peut ainsi arrêter la résolution avant d'arriver à l'optimalité si l'amélioration de la solution courante devient fastidieuse, ou lorsqu'on a une solution «satisfaisante» ou tout simplement lorsqu'on ne dispose plus de ressources informatiques.

Une deuxième stratégie est de combiner la résolution avec une relaxation lagrangienne. En effet, étant donné que la relaxation lagrangienne peut fournir des bornes inférieures de qualité, on peut exploiter cette approche en la combinant avec l'algorithme qu'on propose afin de réduire l'espace des états et par la suite le domaine de recherche de solutions. Cette borne est utile pour approximer l'écart de la valeur de la solution courante par rapport à la valeur de la solution optimale. On pourra, par conséquent, arrêter au besoin la résolution avant l'arrêt de l'algorithme lorsqu'on juge que l'écart est «raisonnable».

Dans ce travail, on commencera d'abord par une revue de littérature sur les différents problèmes de plus court chemin en général. Dans cette partie, on essayera d'établir une classification de ces différents problèmes et des différentes méthodes de résolution. Par la suite, on présentera le contexte général de la thèse, soit le modèle unifié et ses applications. Une revue de littérature plus détaillée des différentes approches de résolution proposées par Nagih et Soumis sera présentée au chapitre 4, ainsi que les méthodes et heuristiques générales basées sur la programmation dynamique. À la fin du chapitre 4, on présente une critique de la méthode de Nagih et

Soumis. Dans le chapitre 5, on exposera une nouvelle approche de résolution avec plusieurs résultats numériques. Dans le chapitre 6 on présente des améliorations intéressantes de cet algorithme, ainsi que plusieurs résultats numériques afin de valider ces approches. Dans le chapitre 7, on propose une généralisation des méthodes introduite au chapitre 5 au cas général des PCC-CR et dans le cas où les fonctions d'extension sont non linéaires. En conclusion, on fera une synthèse globale de nos travaux ainsi que des propositions pour de futures recherches.

CHAPITRE 2

REVUE DE LA LITTÉRATURE

Plusieurs spécifications du problème de plus court chemin avec contraintes de ressources ont été traitées en utilisant différentes approches et algorithmes de résolution, variant selon le contexte et la taille des problèmes considérés. La littérature s'est surtout concentrée sur le problème de plus court chemin avec une seule contrainte de ressource (problème de plus court chemin contraint) ou aussi sur les problèmes de plus court chemin avec fenêtres de temps (une fenêtre de temps à chaque noeud). Par contre, le problème général de plus court chemin avec contraintes de ressources n'a pas été fréquemment traité dans la littérature.

On se propose dans cette section de faire une revue de la littérature générale de tous ces problèmes, entre autres, des cas particuliers du problème de plus court chemin avec contraintes de ressources. Notons qu'on proposera, dans les deux chapitres qui suivent celui-ci, une revue de la littérature plus détaillée du problème spécifique de plus court chemin avec contraintes de ressources et on traite ainsi, en détail, les approches de résolution de ces problèmes. Tout d'abord, on exposera les méthodes de résolution exactes, puis les méthodes heuristiques.

2.1 Méthodes exactes

Dans la première partie de cette revue de littérature, on explorera les différentes approches de résolution exactes qui sont du type programmation dynamique et les différents types de problèmes de base qui ont été résolus en utilisant ces approches. Par la suite, on explorera différents travaux portant sur les extensions de ces problèmes qui ont été aussi résolus de façon exacte par des algorithmes du type programmation

dynamique. Cette partie sera présentée en quatre principales sous-sections, trois de ces sous-sections seront réservées aux différentes méthodes du type programmation dynamiques appliquée sur différentes variantes du problème général du PCC-CR et de leurs extensions. Une autre section sera dédiée pour les différentes approches qui font appel à la relaxation lagrangienne.

2.1.1 Problèmes du plus court chemin avec fenêtres de temps et ses variantes

Plusieurs recherches se sont concentrées au début sur le cas particulier du problème de plus court chemin avec fenêtres de temps (PCC-FT) et ses variantes et plusieurs algorithmes du type programmation dynamique ont été développés. C'est ainsi que Desrosiers et al. (1983) ont développé un algorithme pseudo-polynomial du type *label-correcting* pour résoudre le cas particulier du problème PCC-FT avec des durées t_{ij} positives. Si on définit la taille τ d'un problème donné comme étant $\sum_{i \in N}(a_i - b_i + 1)$ (N étant l'ensemble des noeuds), leur algorithme est alors de complexité $O(\tau^3)$. Desrochers et Soumis (1988a) ont exploité plus à fond l'hypothèse que $t_{ij} > 0$ et ont obtenu un algorithme du type *label-setting* de complexité $O(\tau^2)$. Desrochers et Soumis (1988b) ont présenté ensuite un algorithme de réoptimisation du problème PCC-FT basé sur une approche primale-duale. Pour le même type de problèmes, Christofides et al. (1981) ont proposé un algorithme exact, pour résoudre les TSPTW (*Traveling Salesman Problem with Time Windows*). Cet algorithme consiste en la relaxation de l'espace des états afin d'améliorer l'efficacité de l'algorithme de programmation dynamique. La méthode qu'ils ont proposé fournit une borne inférieure que les auteurs exploitent dans un algorithme de «branch and bound» pour avoir une solution optimale. Cette approche de relaxation a été par la suite généralisée dans le but de réduire l'espace des états à l'intérieur d'un algorithme de programmation dynamique dans le contexte de la résolution du problème de voyageur de commerce avec fenêtres de temps par Mingozzi et al. (1997). De façon similaire, et

afin d'accélérer l'algorithme de programmation dynamique, Abdul-Razak et Potts (1988) ont utilisé une approche de relaxation de l'espace des états dans le cadre de la résolution du problème d'ordonnancement d'une machine. Pour le même type de problèmes, Aneja et al. (1981) ont proposé un algorithme d'énumération implicite, exact, du type Dijkstra. Kolen et al. (1987) ont développé à leur tour un algorithme d'étiquetage pour résoudre le problème dans le but d'améliorer les bornes pour un problème de tournées de véhicules.

2.1.2 Problèmes du plus court chemin avec contraintes de ressources et ses variantes

Bon nombre de recherches ont essayé d'étendre ces approches au cas général des problèmes de plus court chemins avec contraintes de ressources PCC-CR et ses variantes. Plusieurs algorithmes et méthodes exactes ont été proposés. Irnich et Desaulniers (2005) proposent un article de synthèse, dans lequel ils présentent une formulation générique du PCC-CR et ses variantes et les différentes approches de résolution présentes dans la littérature, ainsi que les différentes applications des PCC-CR.

C'est ainsi que, Desrochers (1986) a développé une première généralisation du problème PCC-FT en traitant le cas avec plusieurs variables de ressources lexicographiquement positives (voir aussi Desrosiers et al., 1995). Il utilise la structure de données des SPLAY (Sleator et Tarjan, 1985) et les algorithmes de Kung et al. (1975) pour identifier efficacement les états non dominés, pour toutes les ressources. Desaulniers et al. (1998) montrent que l'algorithme de Desrochers (1986) permet de résoudre des problèmes avec fonctions d'extension f_{ij}^{kr} (fonctions responsables de la prolongation des valeurs des ressources sur le réseaux) non décroissantes.

D'un autre côté, Vovor (1997) étudie une variante du problème PCC-CR pour réseaux acycliques et valeurs de ressources discrètes. Des bornes inférieures et supérieures sur les valeurs des ressources sont associées aux arcs et des fonctions de mise à

jour, pas nécessairement non décroissantes, sont associées aux noeuds. Vovor (1997) a montré 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) a proposé, entre autres, une approche en deux phases. Il construit d'abord un réseau discrétisé ne contenant que les états pouvant faire partie des 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. Cependant, en pratique, lorsque l'espace des états est grand, cette approche nécessite beaucoup de mémoire pour l'entreposage des réseaux discrétisés et ce surtout lorsque le nombre de sous-problèmes est élevé.

Dans le cadre de la programmation multicritère, White (1982) a utilisé une approche qui combine programmation linéaire et programmation dynamique afin de résoudre un problème de plus court chemin à plusieurs objectifs. D'un autre côté, Henig (1985) a présenté des méthodes pour résoudre le problème de plus court chemin à deux objectifs avec fonctions d'utilité quasi-concaves en combinant un algorithme de programmation dynamique et un algorithme de plus court chemin.

Une autre classe de méthodes exactes du type programmation dynamique sont les méthodes de k plus court chemin. Plusieurs recherches se sont intéressées à l'élaboration d'algorithmes efficaces pour k-plus courts chemins. Pollack (1961) a trouvé un algorithme de complexité $O(n^k)$ alors que Yen (1961) a proposé un algorithme de complexité $O(kn^3)$. Katoh et al. (1982) ont amélioré cette complexité à $O(k(m + nlog(n)))$ sur les réseaux non orientés où m est le nombre d'arcs. Fox (1978) a développé un algorithme pour résoudre les k-plus courts chemin avec cycles

de complexité $O(n^2 + knlog(n))$. Epstein (1978) a utilisé une représentation implicite des chemins pour améliorer significativement cette borne à $O(m + nlog(n) + kn)$. Ces méthodes ont été utilisées par la suite pour résoudre les problèmes de plus courts chemins avec contraintes de ressources et ses variantes. Dans ce type d'algorithme le nombre de ressources ou de contraintes importe peu dans la façon avec laquelle on résout le problème. En effet, on résout séquentiellement un problème relâché et la première solution réalisable qu'on trouve est éventuellement la solution optimale. C'est ainsi que Chen et Yang (2004) trouvent k-plus courts chemins en ignorant les contraintes de ressources pour ensuite choisir le moins coûteux réalisable qui est forcément la solution optimale pour le problème de plus court chemin avec contraintes de ressources. L'algorithme qu'ils proposent est de complexité $O(n^3kr)$, où n est le nombre des noeuds du réseau et r est le nombre de ressources.

Remarquons que la complexité de chacun de ces algorithmes est en fonction de k, le nombre de chemins. Or, pour le problème de plus court chemin avec contraintes de ressources, k peut être très grand avant qu'on trouve un chemin qui vérifie toutes le fenêtres de ressources, ce qui rend la méthode des k-plus courts chemins relativement fastidieuse.

2.1.3 Extensions des PCC-FT et PCC-CR

Selon le contexte et les applications, plusieurs chercheurs se sont penchés sur les extensions des problèmes de plus court chemin avec contraintes de ressources (aussi bien le problème de base que le problème général). Dans le cadre de la résolution du problème VRPTW (*Vehicle Routing Problem with Time Windows*) comportant deux ressources, Kolen et al. (1987), Desrochers et al. (1992) et Kohl et Madsen (1997) ont incorporé des contraintes d'élimination des 2-cycles au problème PCC-FT comme stratégie d'accélération globale. L'élimination des 2-cycles implique une augmentation du nombre d'états (au pire cas par un facteur de 2) qui est compensée

par une réduction sensible de l'écart d'intégrité et du nombre de problèmes résiduels à résoudre dans le processus d'énumération. D'autre part Irnich et Villeneuve (2006) utilisent des techniques d'élimination de k-cycles ($k \geq 3$) lors de la résolution du PCC-CR élémentaire, par un algorithme du type programmation dynamique en utilisant de nouvelles règles de dominance. Ceci a permis d'améliorer la borne inférieure dans le cadre de la résolution par un algorithme de génération de colonnes du problème VRPTW. Enfin, deux algorithmes de types différents ont été développés pour traiter d'autres extensions du problème PCC-FT. D'une part, Ioachim et al. (1993) ont traité le problème PCC-FT avec un objectif incluant à la fois les variables de flot X_{ij}^k sur les arcs et des variables de ressources T_i^{kr} sur les noeuds. D'autre part, Dumas et al. (1991) ont proposé également un algorithme de résolution pseudo-polynomial pour résoudre un cas particulier du problème PCC-FT avec cueillette et livraison.

2.1.4 Méthodes lagrangiennes exactes

Plusieurs méthodes utilisent la relaxation lagrangienne comme approche de résolution aussi bien pour les PCC-CR que les PCC-FT et leurs variantes. Ces méthodes sont le plus souvent hybrides et sont utilisées avec d'autres méthodes de résolution. En effet, souvent on utilise la relaxation lagrangienne dans le but de trouver des bornes inférieures, afin de réduire l'espace des états ou aussi dans le but de définir des fonctions d'utilité lagrangiennes. C'est ainsi que Beasley et Christofides (1989) ont utilisé à leur tour une formulation en nombres entiers $(0-1)$ et ont utilisé une relaxation lagrangienne pour trouver une borne inférieure à l'intérieur d'un algorithme de «branch and bound». Handler et Zang (1980) proposent de leur côté un algorithme pour résoudre le problème de plus court chemin avec une contrainte du type sac à dos dans lequel ils utilisent un algorithme de k-plus courts chemins pour obtenir une première solution qui satisfait la contrainte pour ensuite utiliser une relaxation lagrangienne afin de réduire le nombre k.

2.2 Méthodes heuristiques

Dans les différentes recherches qu'on a présenté auparavant, lors de la résolution du PCC-CR et ses variantes par des algorithmes exactes, la complexité de ces algorithmes est toujours très grande. Des communications privées avec les développeurs de KRONOS *Inc.* nous ont appris que plusieurs grands problèmes de PCC-CR avec plusieurs ressources ne peuvent pas être résolus jusqu'à l'optimalité. Ainsi plusieurs recherches se sont penchées sur des méthodes heuristiques ayant pour but de trouver rapidement des solutions réalisables de bonne qualité. Dans ce qui suit, on commencera par présenter la littérature sur des variantes simples du PCC-FT. La première est le problème de plus court chemin avec une seule contrainte linéaire de ressources au noeud destination (CSP), la seconde est le MCSP qui est l'extension de ce dernier à plusieurs contraintes linéaires de ressources au noeud destination. Par la suite, on exposera la revue de littérature pour les PCC-FT et les PCC-CR.

2.2.1 Problèmes de plus court chemin avec une ou plusieurs contraintes linéaires

Les problèmes CSP et MCSP peuvent des fois être encore plus difficiles à résoudre que les PCC-CR. En effet, lors de la résolution par un algorithme du type programmation dynamique, il n'y a pas beaucoup d'élimination d'étiquettes (chemins) contrairement au PCC-CR où plusieurs chemins deviennent non réalisables à cause des contraintes de ressources sur chaque noeud, et sont donc éliminés lors de la résolution. Ainsi, plusieurs heuristiques ont été développées afin de résoudre ce problème.

Warburton (1987) a développé une heuristique pour résoudre le problème à plusieurs objectifs sans contraintes, par un algorithme de programmation dynamique. Il

fournit, par la suite, une adaptation de cet algorithme pour le CSP. À partir des travaux de Warburton (1987), Hassine (1992) a proposé deux algorithmes de résolution pseudo-polynomiaux qui sont aussi du type programmation dynamique. Le premier est de complexité $O(mnC)$, où C est le coût maximal des arcs. Cet algorithme est, par la suite, utilisé pour développer un deuxième algorithme d'approximation de complexité $\frac{mn}{\epsilon}$, où ϵ est l'écart de la solution courante par rapport à la solution optimale, en utilisant des bornes supérieures et inférieures afin de réduire l'espace des états. Lorenz et Raz (2001) ont donné à leur tour une extension élégante de l'algorithme exact de Hassine (1992) pour le cas acyclique. D'un autre côté, Korkmaz et Krunz (2001) utilisent l'algorithme de k plus courts chemins avec des fonctions d'utilité non linéaires afin de trouver rapidement des solutions pour le problème MCSP, dans le cadre du problème général de QoS (qualité du service) pour le trafic sur les réseaux de télécommunications.

La relaxation lagrangienne a été aussi utilisée à plusieurs reprises afin de trouver rapidement des solutions heuristiques. En effet, Dumitrescu et Boland (2003) utilisent comme approche de résolution un algorithme d'énumération des chemins avec coûts lagrangiens pour le problème de CSP. D'autre part, Carlyle et Wood (2003) utilisent une approche similaire à celle de Handler et Zang (1980) pour résoudre les problèmes de MCSP. Dans cette approche, les auteurs utilisent la relaxation lagrangienne et un algorithme d'énumération afin d'identifier des plus courts chemins à ε près.

2.2.2 Méthodes heuristiques pour la résolution du PCC-FT et PCC-CR

Des adaptations heuristiques des algorithmes d'étiquetage, du type programmation dynamique, ont été utilisées en industrie. Ces méthodes consistent en la dominance sur un sous-ensemble de ressources bien choisi. Ces sous-ensembles de ressources (qu'on appelle communément modèles) peuvent être changées alternativement, à l'intérieur de la résolution afin de varier l'espace de recherche pour obtenir des solutions différentes.

Dans le cadre de la résolution des problèmes de tournées de véhicules avec fenêtres de temps, Desaulniers et al. (2006) utilisent une approche hybride pour la résolution des sous-problèmes qui sont des plus courts chemins élémentaires avec contraintes de ressources. Cette approche consiste à résoudre le problème, de façon alternative, en utilisant un algorithme heuristique du type recherche tabou et une approche de programmation dynamique.

Pour la première fois dans la littérature, une heuristique qui combine à la fois l'approche de programmation dynamique d'étiquetage et l'approche lagrangienne a été proposée par Nagih et Soumis (2006) pour résoudre les PCC-CR. Le but initial de cette approche heuristique était de trouver rapidement des solutions réalisables et ceci dans le cadre d'un algorithme de génération de colonnes. Cette approche avait pour but de produire de meilleures solutions que l'approche de dominance décrite un peu plus haut et qui ne soit pas très coûteuse en termes de temps de calcul. On propose une étude plus détaillée de cette approche dans le chapitre 4.

Dans le cadre du problème de tournées de véhicules avec fenêtre de temps, Feillet et al. (2005) développent plusieurs heuristiques afin d'accélérer la résolution des sous-problèmes qui sont des PCC-FT élémentaires. Ces heuristiques ont été développées à l'intérieur d'une méthode du type programmation dynamique. Une première heuristique est une méthode de recherche locale, où l'ensemble des noeuds est divisé en potentiellement «bons» et potentiellement «mauvais» selon leurs valeurs duales. Une ressource est ajoutée pour limiter le nombre de chemins passant par les «mauvais» noeuds. Une deuxième heuristique consiste en l'ajout d'étiquettes initiales à l'espace des états de l'algorithme de programmation dynamique qui permettront d'éliminer, par dominance, un grand nombre d'étiquettes générées au cours du processus. Une troisième heuristique développée consiste à calculer des bornes supérieures en utilisant une relaxation du problème de plus court chemin en un problème de sac à dos. Ces bornes permettent de réduire l'espace des états lors de la résolution par un algorithme de programmation dynamique.

CHAPITRE 3

PROBLÉMATIQUE GÉNÉRALE

Dans cette partie, on expose d'abord le problème général qui englobe un grand ensemble de problèmes d'optimisation sous un modèle générique appelé modèle unifié (Desaulniers et al., 1998). La résolution de ces problèmes vise à minimiser les coûts d'opération d'une flotte de véhicules ou d'équipes de travail lors de la réalisation d'un ensemble de tâches prédéterminées. C'est un modèle qui est valable pour un grand éventail d'applications, par exemple, les problèmes de tournées de véhicules, d'horaires de personnel, etc. Une approche de résolution pour ces problèmes est un algorithme de génération de colonnes. L'algorithme de génération de colonnes se divise en deux problèmes : le problème maître qui est généralement résolu par l'algorithme du simplexe et les sous-problèmes qui sont des problèmes de plus court chemin avec contraintes de ressources. On fera aussi un survol des applications de cette formulation.

3.1 Le modèle unifié

Le problème consiste à couvrir, à coût minimum, un ensemble de tâches avec des chemins de certaines commodités. D'une part, chaque chemin d'une commodité est limitée localement par la structure du réseau et par diverses contraintes de ressources. L'utilisation des commodités peut être également limitée par des relations de préséance ou de couplage entre les tâches à effectuer. D'autre part, cette utilisation de commodités est limitée globalement par des contraintes de coordination entre celles-ci.

On commence tout d'abord par une formulation mathématique générique du problème. Une commodité $k \in \mathcal{K}$ circule sur le réseau $\mathcal{G}^k = (\mathcal{N}^k, \mathcal{A}^k)$, où $\mathcal{N}^k$ et $\mathcal{A}^k$ sont

des ensembles de noeuds et d'arcs. L'ensemble $\mathcal{N}^k$ contient au moins deux noeuds, un noeud origine o^k et un noeud puits d^k. À chaque commodité k est également associée un ensemble de ressources $\mathcal{R}^k$.

Les variables de flot de la commodité k dans $\mathcal{G}^k$ sont dénotées par le vecteur $\mathbf{X}^k = \{X_{ij}^k | (i,j) \in \mathcal{A}^k\}$, où X_{ij}^k est une variable binaire selon que l'arc (i,j) est utilisé par la commodité k ou non. Les variables représentant l'état des ressources à chaque noeud de $\mathcal{N}^k$ sont dénotées par le vecteur $\mathbf{T}^k = \{T_i^{kr} | i \in \mathcal{N}^k, r \in \mathcal{R}^k\}$, où T_i^{kr} prend la valeur de la ressource r cumulée depuis le noeud o^k si le noeud i est visité par la commodité k et la valeur 0 sinon. L'agrégation intermédiaire $\mathbf{T}_i^k = \{T_i^{kr} | r \in \mathcal{R}^k\}$ représente le vecteur de toutes les ressources au noeud $i \in \mathcal{N}^k$.

Les variables supplémentaires Y_s, $s \in S$, sont utilisées pour compléter la structure du problème. Les variables Y_s peuvent servir de variables d'écart (Barnhart et al., 1998) ou de surplus (Graves et al., 1993). Elles peuvent aussi compter le nombre de commodités dans une solution (Desaulniers et al., 1998), ou encore accumuler certaines variables de ressources sur un ensemble de commodités (Ioachim et al., 1993). Ces variables Y_s sont limitées par une borne inférieure l_s, et une borne supérieure u_s

On désigne par a_{wij}^k les coefficients binaires des variables de flot X_{ij}^k et valent 1 si l'arc $(i,j) \in A^k$ couvre la tache w et 0 sinon, et ceci pour chaque commodité $k \in K$; a_{ws} et b_{hs} sont les coefficients de la variable Y_s, $s \in S$ dans la contrainte de couverture de tâches $w \in W$ et dans la contrainte liante $h \in H$

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

Ainsi le modèle unifié s'écrit sous la forme du programme mathématique :

$$z_{IP} = \min \sum_{k \in \mathcal{K}, (i,j) \in \mathcal{A}^k} c_{ij}^k X_{ij}^k + \sum_{k \in \mathcal{K}, i \in \mathcal{N}^k, r \in \mathcal{R}^k} c_i^{kr} T_i^{kr} + \sum_{s \in \mathcal{S}} c_s Y_s \qquad (3.1)$$

sous les contraintes :

$$\sum_{\substack{k\in\mathcal{K}\\(i,j)\in\mathcal{A}^k}} a^k_{wij}X^k_{ij} + \sum_{s\in\mathcal{S}} a_{ws}Y_s = \quad a_w \quad \forall w \in W \tag{3.2}$$

$$\sum_{\substack{k\in\mathcal{K}\\i\in\mathcal{N}^k\\r\in\mathcal{R}^k}} b^{kr}_{hi}T^{kr}_i + \sum_{\substack{k\in\mathcal{K}\\(i,j)\in\mathcal{A}^k}} b^k_{hij}X^k_{ij} + \sum_{s\in\mathcal{S}} b_{hs}Y_s = \quad b_h \quad \forall h \in H \tag{3.3}$$

$$l_s \leq Y_s \leq \quad u_s \quad \forall s \in \mathcal{S} \tag{3.4}$$

$$\sum_{j:(o^k,j)\in\mathcal{A}^k} X^k_{o^kj} = 1 = \quad X^k_{o^k} \quad \forall k \in \mathcal{K} \tag{3.5}$$

$$\sum_{i:(i,d^k)\in\mathcal{A}^k} X^k_{id^k} = 1 = \quad X^k_{d^k} \quad \forall k \in \mathcal{K} \tag{3.6}$$

$$\sum_{j:(i,j)\in\mathcal{A}^k} X^k_{ij} = \sum_{j:(j,i)\in\mathcal{A}^k} X^k_{ji} = \quad X^k_i \quad \forall i \in \mathcal{N}^k \setminus \{o^k, d^k\} \tag{3.7}$$

$$X^k_{ij} \in \quad \{0,1\} \quad \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}^k \tag{3.8}$$

$$X^k_{ij}(f^{kr}_{ij}(\mathbf{T}^k_i) - T^{kr}_j) \leq \quad 0 \quad \begin{array}{c}\forall k \in \mathcal{K}, (i,j) \in \mathcal{A}^k,\\ r \in \mathcal{R}^k\end{array} \tag{3.9}$$

$$\sum_{\substack{i\in\mathcal{N}^k\\r\in\mathcal{R}^k}} b^{kr}_{hi}T^{kr}_i + \sum_{(i,j)\in\mathcal{A}^k} b^k_{hij}X^k_{ij} \leq \quad b_h \quad \forall k \in \mathcal{K}, h \in H \tag{3.10}$$

$$l^{kr}_i X^k_i \leq T^{kr}_i \leq \quad u^{kr}_i X^k_i \quad \begin{array}{c}\forall k \in \mathcal{K}, i \in \mathcal{N}^k,\\ r \in \mathcal{R}^k\end{array}. \tag{3.11}$$

Les contraintes (3.2)-(3.4) permettent les interactions entre toutes les commodités du modèle, dénommées *multiple path linking constraints* dans Desaulniers et al. (1998). Les contraintes (3.2) forment un problème de recouvrement généralisé, où a_w est le nombre de fois que la tâche w doit être effectuée. Les contraintes (3.3) lient globalement plusieurs commodités. Les valeurs des coefficients b^{kr}_{hi}, b^k_{hij} et b_h dépendent

du problème à résoudre et plusieurs exemples sont décrits dans Desaulniers et al. (1998).

Les contraintes (3.5)-(3.11) regroupent les variables par commodité, dénommées *single path constraints* dans Desaulniers et al. (1998). Les contraintes (3.5)-(3.8) imposent la structure classique d'un chemin unique pour la commodité k dans un réseau G^k de la source au puits de ce dernier.

Les contraintes (3.9) donnent le mécanisme général utilisant une fonction d'extension f_{ij}^{kr} quelconque pour restreindre les valeurs admissibles de la variable de ressource T_j^{kr} au noeud j en fonction du vecteur de ressources $\mathbf{T}_i^k$ d'un noeud i qui le précède. Les inégalités en (3.9) admettent des solutions où $T_j^{kr} \geq f_{ij}^{kr}(\mathbf{T}_i^k)$ dès que la variable X_{ij}^k est strictement positive. Cette opportunité permet pour une ressource, représentant le temps par exemple, d'introduire un certain délai entre le temps d'arrivée au noeud j, représenté par $f_{ij}^{kr}(\mathbf{T}_i^k)$, et le temps de début du service à ce même noeud, représenté par T_j^{kr} (Desrochers et al., 1992). Notons que l'expression des fonctions d'extension f_{ij}^{kr} varient selon le contexte et l'application. De plus, elles peuvent être croissantes, décroissantes, continues ou discontinues.

Les contraintes (3.10) sont le pendant par commodité des contraintes (3.3). Elles peuvent limiter localement la construction de chemins en imposant par exemple des contraintes de préséance, de couplage entre les tâches ou d'élimination de cycles (Desaulniers et al., 1998).

Les contraintes (3.11) permettent de restreindre le domaine d'une variable de ressources T_i^{kr} à un intervalle donné par les bornes inférieure l_i^{kr} et supérieure u_i^{kr}.

3.2 Méthode de résolution

L'objectif (3.1) et les contraintes (3.5)-(3.11) sont séparables par commodité, formant une structure diagonale en $|K|$ blocs avec les contraintes liantes (3.2)-(3.4). La méthode de décomposition de Dantzig et Wolfe (1960) tire le plus profit de ce type de structure. Cette méthode est principalement associée à des algorithmes de génération de colonnes ou de plans coupants (voir Dantzig et Wolfe, 1960; Goffin et Vial, 1990; Kelley, 1960; Veinott, 1967).

3.2.1 Méthode de génération de colonnes

L'algorithme de génération de colonnes (GC) consiste à résoudre alternativement un problème maître restreint et plusieurs sous-problèmes, un pour chaque commodité. Le problème maître restreint dérive du problème maître en considérant un sous-ensemble de ses variables de chemin qui est mis à jour à chaque itération. Son rôle consiste à retrouver la meilleure solution avec les variables de chemin et est résolu en utilisant l'algorithme du simplexe primal ou dual. Chaque sous-problème PCC-CR permet la génération de tous les chemins réalisables pour la commodité correspondante. En utilisant l'information duale associée à la solution courante du problème maître restreint, le rôle des sous-problèmes consiste en la génération de chemins réalisables de coût marginal négatif qui va améliorer la solution courante. Le processus de génération de colonnes s'arrête lorsqu'il n'y a plus de chemins de coût marginal négatif à générer par les sous-problèmes.

Pour accélérer la résolution des problèmes de grande taille, on utilise des heuristiques pour résoudre les sous-problèmes et générer des colonnes réalisables. Ces heuristiques fournissent des solutions de plus en plus proches de l'optimum au cours des itérations de GC tout en maintenant des temps de résolution réduits.

3.2.2 Le problème maître

Le problème maître permet d'obtenir une borne inférieure z_{LP} sur la solution optimale z_{IP} d'un problème résiduel.

Sans perte de généralité, notons Ω^k l'ensemble des chemins admissibles de G^k, $k \in \mathcal{K}$. Pour chaque chemin $p \in \Omega^k$, on associe les trois paramètres suivants : a^k_{wp} qui prend la valeur 1 si p couvre la tâche $w \in W$ et 0 sinon, b^k_{hp} la contribution de p à la contrainte globale $h \in H$, et c^k_p le coût de p. Une variable binaire θ^k_p est associée à chaque chemin $p \in \Omega^k$. Elle prend la valeur 1 si la commodité k est affectée au chemin p et prend 0 sinon. Ainsi le problème maître s'écrit :

$$\min \sum_{\substack{k \in \mathcal{K} \\ p \in \Omega^k}} c^k_p \theta^k_p + \sum_{s \in \mathcal{S}} c_s Y_s \tag{3.12}$$

sous les contraintes :

$$\sum_{\substack{k \in \mathcal{K} \\ p \in \Omega^k}} a^k_{wp} \theta^k_p + \sum_{s \in \mathcal{S}} a_{ws} Y_s = \quad a_w \quad \forall w \in W \tag{3.13}$$

$$\sum_{\substack{k \in \mathcal{K} \\ p \in \Omega^k}} b^k_{hp} \theta^k_p + \sum_{s \in \mathcal{S}} b_{hs} Y_s = \quad b_h \quad \forall h \in H \tag{3.14}$$

$$l_s \leq Y_s \leq \quad u_s \quad \forall s \in \mathcal{S} \tag{3.15}$$

$$\sum_{p \in \Omega^k} \theta^k_p = \quad 1 \quad \forall k \in \mathcal{K} \tag{3.16}$$

$$\theta^k_p \geq \quad 0 \quad \forall k \in \mathcal{K}, p \in \Omega^k \tag{3.17}$$

$$X^k_{ij} = \sum_{p \in \Omega^k} x^k_{ijp} \theta^k_p \quad \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}^k \tag{3.18}$$

$$X^k_{ij} \quad \text{binaire} \quad \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}^k \tag{3.19}$$

Voir Desaulniers et al. (1998) pour plus de détails.

3.2.3 Les sous-problèmes

Étant donné que les contraintes (3.5)-(3.11) du problème sont décomposables en bloc par commodité, alors, par la décomposition de Dantzig-Wolfe, on a un sous-problème pour chaque commodité k. Soit $\alpha = \{\alpha_w | w \in W\}$, $\beta = \{\beta_h | h \in H\}$ et $\gamma = \{\gamma_k | k \in \mathcal{K}\}$ les vecteurs des variables duales associées aux ensembles de contraintes (3.13), (3.14) et (3.16), respectivement. Le coût réduit $\bar{c}_p^k(\alpha, \beta, \gamma)$ de chaque variable de chemin θ_p^k en fonction des vecteurs α, β et γ est donné par :

$$\bar{c}_p^k(\alpha, \beta, \gamma) \quad = \quad c_p^k - \sum_{w \in W} a_{wp}^k \alpha_w - \sum_{h \in H} b_{hp}^k \beta_h - \gamma^k. \tag{3.20}$$

Alors l'objectif du sous-problème pour la commodité k s'écrit comme suit (voir Desaulniers et al., 1998, pour plus de détails)

$$Obj(X^k, T^k) = T_{d(k)}^{k0} - \sum_{(i,j) \in \mathcal{A}^k} (\sum_{w \in N} a_{w,ij}^k \alpha_w + \sum_{h \in H} b_{h,ij}^k \beta_h) X_{ij}^k - \sum_{\substack{i \in V^k \\ r \in R^k \\ h \in H}} b_{h,i}^{kr} \beta_h T_i^{kr} - \gamma^k.$$

Cet objectif linéaire permet ainsi d'identifier le chemin $p \in P^k$ de coût marginal minimum pour chaque commodité $k \in \mathcal{K}$. Ainsi pour chaque commodité k, le sous-problème s'écrit :

$$\min Obj(X^k, T^k) \tag{3.21}$$

sous les contraintes :

$$\sum_{i:(i,d^k)\in\mathcal{A}^k} X^k_{id^k} = 1 = \quad X^k_{d^k} \quad \forall k \in \mathcal{K} \tag{3.22}$$

$$\sum_{j:(i,j)\in\mathcal{A}^k} X^k_{ij} = \sum_{j:(j,i)\in\mathcal{A}^k} X^k_{ji} = \quad X^k_i \quad \forall i \in \mathcal{N}^k \setminus \{o^k, d^k\} \tag{3.23}$$

$$X^k_{ij} \in \quad \{0,1\} \quad \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}^k \tag{3.24}$$

$$X^k_{ij}(f^{kr}_{ij}(\mathbf{T}^k_i) - T^{kr}_j) \leq \quad 0 \quad \begin{array}{c}\forall k \in \mathcal{K}(i,j) \in \mathcal{A}^k \\ r \in \mathcal{R}^k\end{array} \tag{3.25}$$

$$l^{kr}_i X^k_i \leq T^{kr}_i \leq \quad u^{kr}_i X^k_i \quad \begin{array}{c}\forall k \in \mathcal{K}, i \in \mathcal{N}^k, \\ r \in \mathcal{R}^k\end{array}. \tag{3.26}$$

Comme on l'a déjà mentionné, on a un sous-problème par commodité. Dans bon nombre d'applications, on a des centaines de commodités. De plus, dans un algorithme de génération de colonnes, on doit résoudre ces k problèmes à chaque itération. Ainsi, un algorithme rapide de résolution est nécessaire afin de fournir des colonnes de qualité au problème maître.

Pour bon nombre d'applications (par exemple, les problèmes de rotations d'équipages), les fonctions d'extension qui figurent dans les contraintes (3.26) du modèle unifié sont non-décroissantes. En particulier, lorsqu'elles sont linéaires, elles s'écrivent : $f^{kr}_{ij}(T^{kr}_i) = T^{kr}_i + t^{kr}_{ij}$. La prochaine section présente une formulation simplifiée du problème de plus court chemin avec contraintes de ressources dans le cas des fonctions d'extension linéaires.

3.2.4 Le problème de plus court chemin avec contraintes de ressource et fonctions d'extension linéaires

Soit $G = (N, A)$ un réseau orienté acyclique avec A l'ensemble des arcs (i, j) et $N = V \cup \{o, d\}$ l'ensemble des noeuds rangés dans l'ordre topologique, où V est l'ensemble des noeuds i qui peuvent être visités entre l'origine o et la destination d. Soit R l'ensemble des ressources, avec $|R| = n$. À chaque noeud $i \in N$ est associé des intervalles (fenêtres de ressource) $[a_i^r, b_i^r]$, $r \in R$, restreignant les quantités des ressources utilisées pour atteindre le noeud i. Un arc $(i, j) \in A$ consomme une quantité t_{ij}^r de chaque ressource $r \in R$ et a un coût c_{ij}. Tout arc ne permettant pas de visiter le noeud j après le noeud i, c'est-à-dire ne respectant pas les fenêtres de ressource :

$$a_i^r + t_{ij}^r \leq b_j^r, \quad \forall r \in R$$

est supprimé. On associe à un chemin P_j, de l'origine o au noeud j, un coût C_j qui est la somme des coûts des arcs le composant, et un n-vecteur d'état T_j correspondant aux consommations de chaque ressource $r \in R$, cumulées T_j^r selon les fonctions d'extension $f_{ij}^r(T_i^r) = T_i^r + t_{ij}^r$. Un état $T_j^R = (T_j^1, ..., T_j^n)$ est dit réalisable ou légal si :

$$a_j^r \leq T_j^r \leq b_j^r, \quad \forall r \in R.$$

Ainsi lorsque qu'on a des fonctions de prolongation linéaires, on peut formuler le problème de plus court chemin avec contraintes de ressources comme suit :

$$\min \quad \sum_{i,j} c_{ij} x_{ij} \tag{3.27}$$

$$s.c. \quad \sum_i x_{ij} - \sum_i x_{ji} = e_j \quad \forall j \in N \tag{3.28}$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A \tag{3.29}$$

$$x_{ij} \left(T_i^r + t_{ij}^r - T_j^r\right) \leq 0, \quad \forall (i, j) \in A, \forall r \in R \tag{3.30}$$

$$T_j^r \in [a_j^r, b_j^r], \quad \forall j \in V, \forall r \in R. \tag{3.31}$$

avec

$$e_j = \begin{cases} -1 & \text{si } j = o \\ 0 & \text{si } j \in V \\ 1 & \text{si } j = d \end{cases}$$

Dans le cas d'une seule ressource, l'indice r peut être omis.

3.3 Applications

Les applications du modèle unifié sont principalement les problèmes de transport. Ainsi on peut citer les problèmes suivants : transport scolaire (Desrosiers et al., 1984), transport urbain (Desrochers et Soumis, 1989; Ribeiro et Soumis, 1994), transport de personnes et de marchandises (Desrosiers et al., 1988; Ioachim et al., 1995; Desrochers et al., 1992), transport aérien (Lavoie et al., 1988; Gamache et Soumis, 1993; Desrosiers et al., 1999), transport ferroviaire (Ziarati et al., 1997).

Le point commun entre toutes ces applications est leurs sous-problèmes, qui sont des PCC-CR. Le nombre de ressources dans ces derniers varie considérablement selon les applications. En pratique, les PCC-CR deviennent coûteux à résoudre dans un algorithme de génération de colonnes pour des problèmes de grande taille et où le nombre de ressources est supérieur à deux ou trois. Même que, pour certains problèmes, il est impossible de les résoudre à l'optimalité. Par exemple, le problème de blocs mensuels de chauffeurs d'autobus comporte trois à cinq ressources (Desrochers et Soumis, 1989) ou encore pour le problème de blocs mensuels d'équipages aériens, le nombre de ressources varie entre dix et vingt (Gamache et al., 1999). De plus, pour ces problèmes, on a souvent des milliers de noeuds, des dizaines de milliers d'arcs, des centaines de sous-problèmes. Lors de la résolution de ces problèmes, l'algorithme de génération de colonnes fera des centaines d'itérations. Le problème est encore plus complexe pour les rotations d'équipages, surtout en industrie. En effet, des communications privées avec les développeurs de *KRONOS Inc.*, ont montré que,

dans un grand nombre de cas, on se heurte à des problèmes de rotations d'équipages où le nombre de ressources dans les sous-problèmes, lors d'une résolution par GC, varie de dix à vingt, le nombre de sous-problèmes dépasse la trentaine et le nombre d'itérations de GC est de l'ordre d'un millier.

CHAPITRE 4

APPROCHE PAR RELAXATION LAGRANGIENNE

Dans cette partie, on se propose tout d'abord d'exposer en détail l'approche proposée par Nagih et Soumis (2006), sous ses différents aspects théoriques et algorithmiques, en s'appuyant sur les différentes publications des auteurs. Puis, on présentera brièvement les approches algorithmiques que les auteurs ont expérimenté. Par la suite, on expose les améliorations possibles qu'on a expérimenté pour exploiter l'approche de Nagih et Soumis (2006) à la lumière des travaux disponibles. Une critique globale des approches du type Nagih et Soumis, appuyée de résultats numériques sera alors émise. Pour finir, on présentera des conclusions sur cette approche appuyées par des essais numériques avec différents paramètres.

4.1 Le problème de plus court chemin avec contraintes de ressource : espace des états

Dans ce qui suit on va utiliser la formulation (3.27)-(3.31) énoncée à la fin du chapitre précédent.

Notons que les contraintes d'intégrité (3.29) peuvent êtres relâchées tel qu'énoncé dans le théorème suivant :

Théorème 1 *(Nagih et Soumis, 2006) Le problème ayant la formulation (3.27)-(3.31) en relâchant les contraintes d'intégrité (3.29) possède une solution optimale entière.*

Pour résoudre les problèmes de PCC-CR, Desrochers et Soumis (1988a) proposent un algorithme de programmation dynamique de type pulling. Dans cet algorithme,

les noeuds sont traités séquentiellement dans l'ordre topologique, de la source jusqu'à la destination. En chaque noeud, l'algorithme génère des étiquettes en prolongeant les chemins correspondants aux étiquettes efficaces présentes aux noeuds prédécesseurs. Un prolongement est validé s'il fournit un chemin légal, sinon il est supprimé. Puis on applique la règle de dominance pour éliminer tous les chemins correspondants à des étiquettes non efficaces. Désignons par APD un tel algorithme.

Ainsi, cet algorithme procède en deux grandes étapes. À chaque noeud $j \in N$, il effectue les opérations suivantes :

1. prolongation des chemins (génération des étiquettes et test de réalisabilité),
2. dominance (élimination des étiquettes non efficaces).

Formellement, pour un noeud j donné, des étiquettes sont créées en prolongeant celles présentes aux noeuds i, tels que $(i,j) \in A$. Plus précisément, une nouvelle étiquette (C_j, T_j^R) donnée par :

$$\begin{aligned} C_j &= C_i + c_{ij} \\ T_j^r &= \max\{a_j^r, T_i^r + t_{ij}^r\},\ r \in R \end{aligned}$$

est créée au noeud j si $T_i^r + t_{ij}^r \leq b_j^r,\ \forall\, r \in R$.

Selon Desrochers (1986), la résolution du problème PCC-CR en utilisant un tel algorithme (APD) fournit la solution optimale du problème. Soit $\mathcal{Z}_{Opt} = \mathcal{Z}_{APD}$ la valeur de cette solution.

Soit $P_i^{(k)}$ le *k-ième* chemin de l'origine o au noeud i. On lui associe une étiquette $(C_i^{(k)}, T_i^{(k)}) = (C_i^{(k)}, T_i^{1(k)}, T_i^{2(k)}, ..., T_i^{n(k)})$ représentant son état des ressources et son coût. Désignons par ξ l'espace des états relatif à tout le réseau.

Définition 1 *Soient $(C_i^{(1)}, T_i^{(1)})$ et $(C_i^{(2)}, T_i^{(2)})$ deux étiquettes associées à deux chemins réalisables de l'origine o à un noeud i. On dit que $(C_i^{(1)}, T^{(1)})$ domine $(C_i^{(2)}, T_i^{(2)})$ et on note $(C_i^{(1)}, T^{(1)}) \preceq (C_i^{(2)}, T_i^{(2)})$ si et seulement si*

$$C_i^{(1)} \leq C_i^{(2)} \qquad et \qquad T_i^{r(1)} \leq T_i^{r(2)},\ \forall\, r \in R.$$

Remarque 1 *La relation $\preceq$ définit une relation d'ordre sur l'espace des étiquettes ξ qui est un sous-ensemble de dimension $n+1$. Cette relation d'ordre n'est pas totale lorsque le nombre de ressources est non nul ($n > 0$).*

Définition 2 *Une étiquette associée à un chemin réalisable de o à i est dite efficace si elle est minimale au sens de la relation d'ordre $\preceq$. Un chemin est dit efficace s'il est associé à une étiquette efficace.*

La relation de dominance $\preceq$ étant une relation d'ordre partiel, il devient de moins en moins probable, avec l'augmentation du nombre de ressources, d'éliminer des étiquettes dominées. Le nombre d'étiquettes efficaces à traiter augmente d'une façon exponentielle en fonction du nombre de ressources, ce qui rend la mémoire requise et les temps de calcul prohibitifs.

4.2 Projection de l'espace des étiquettes

Afin d'obtenir des solutions primales réalisables, une alternative proposée par Nagih et Soumis (2006) consiste à projeter l'espace des étiquettes, de dimension $n+1$, sur un espace de dimension plus petite, puis appliquer les règles de dominance dans cet espace de dimension réduite pour définir les étiquettes efficaces.

Soient $\Pi = (\pi_p^q)$ une matrice réelle $m \times (n+1)$, avec $m \leq n$, et $\tilde{T}$ le nouveau vecteur de ressources tel que

$$\begin{cases} \tilde{T}^1 &= \pi_1^0 C + \pi_1^1 T^1 + ... + \pi_1^n T^n \\ \vdots & \\ \tilde{T}^m &= \pi_m^0 C + \pi_m^1 T^1 + ... + \pi_m^n T^n. \end{cases} \tag{4.1}$$

Par exemple, une projection orthogonale sur les coûts et les m premières composantes de l'espace des étiquettes (ce qui est équivalent à dominer sur le coût et sur les

m premières ressources) permet de réduire le nombre d'étiquettes en chaque noeud. Cependant, on risque d'éliminer en cours de traitement des chemins optimaux. Ainsi, bien que la réalisabilité par rapport à toutes les ressources soit maintenue, l'optimalité n'est plus garantie.

Cependant, trouver de bons multiplicateurs pour cette projection est une tâche non triviale. C'est pour cela que Nagih et Soumis (2006) proposent une approche lagrangienne afin de trouver des multiplicateurs adéquats.

4.2.1 Projection par noeuds

On se propose dans cette section d'étudier l'approche de projection par noeuds, introduite par Nagih et Soumis (2006), dans le cas où on a plusieurs ressources.

Nagih et Soumis (2006) proposent deux techniques de projections à l'intérieur desquelles les multiplicateurs sont ajustés dynamiquement. La première est du type relaxation lagrangienne, dans laquelle on relâche une ou plusieurs contraintes dans l'objectif. Dans cette approche, on forme tout d'abord une partition constituée de deux sous-ensembles R_1 et R_2, R_1 représente les ressources relâchée et R_2 les ressources non relâchées. La partition de l'ensemble des ressources se présente en m $(m < n)$ sous-ensembles. L'entier m représente le nombre de lignes de la matrice de projection. Les auteurs proposent des matrices de projection en escalier qui permettraient l'ajustement des multiplicateurs par des algorithmes de sous-gradients. La projection en utilisant ce type de matrices reviendrait ainsi à faire une partition des ressources R.

La deuxième approche que les auteurs ont proposé est du type relaxation surrogate où les ressources sont partitionnées en plusieurs sous-ensembles, les ressources de chaque sous-ensemble sont combinées entre elles pour en former une nouvelle.

On n'étudiera pas en détail cette approche étant donné qu'elle est très similaire à l'approche lagrangienne.

Afin d'obtenir des variables duales par noeud, les auteurs se proposent de récrire le problème PCC-CR sous une autre forme en agrégeant dans une seule contrainte toutes les contraintes de consommation de ressources aux différents noeuds j du réseau et ceci pour un sous-ensemble de ressources $R_1 \subset R_2$. En multipliant les contraintes de borne supérieure sur la consommation de ressources par x_{ij}, $(i,j) \in A$ puis en sommant sur tous les prédécesseurs i de j, on obtient :

$$\sum_{i|(i,j)\in A} x_{ij}(\max\{a_j, T_i^{r_1} + t_{ij}^{r_1}\} - b_j^{r_1}) \leq 0,\ r \in R_1.$$

Ainsi la nouvelle formulation s'écrit :

$$\min \quad \sum_{i,j} c_{ij} x_{ij} \tag{4.2}$$

$$s.c. \quad \sum_i x_{ij} - \sum_i x_{ji} = e_j \quad \forall j \in N \tag{4.3}$$

$$x_{ij} \geq 0 \quad \forall\,(i,j) \in A \tag{4.4}$$

$$x_{ij}\left(T_i^r + t_{ij}^r - T_j^r\right) \leq 0 \quad \forall\,(i,j) \in A,\ \forall r \in R \tag{4.5}$$

$$T_j^r \geq a_j^r \quad \forall\, j \in N,\ \forall r \in R \tag{4.6}$$

$$\sum_{i|(i,j)\in A} x_{ij}\left(\max\{a_j^{r_1}, T_i^{r_1} + t_{ij}^{r_1}\} - b_j^{r_1}\right) \leq 0 \quad \forall j \in N, r_1 \in R_1. \tag{4.7}$$

$$T_j^{r_2} \leq b_j^{r_2} \quad \forall\, j \in N,\ \forall r_2 \in R_2. \tag{4.8}$$

Notons que la nouvelle formulation (4.2)-(4.8) est équivalente à la formulation standard (3.27)-(3.31) en présence des contraintes d'intégrité (pour les deux formulations). En effet, afin de satisfaire la condition de flot sur le noeud j, les variables

x_{ij} sur les arcs (i, j) incidents à j doivent être toutes nulles, sauf une qui vaut 1. Les contraintes (3.31) avec $x_{ij} = 0$ sont évidemment satisfaites. Les contraintes (3.31) avec $x_{ij} = 1$ s'obtiennent directement de (4.7) après avoir retiré de la somme les termes $x_{ij} = 0$.

Cependant, lorsqu'on relâche les contraintes d'intégrité, cette équivalence n'est plus vraie comme le montre l'exemple suivant (voir figure 4.1). Supposons que, dans cet exemple on a un réseau de deux noeuds i et j. On a une seule ressource r, ainsi $R_1 = \{r\}$ et $R_2 = \emptyset$. La fenêtre de temps en i est de $[0, 10]$ et celle en j est de $[0, 5]$. On a deux arcs joignant ces deux noeuds, le coût et la consommation de ressources respectifs du premier sont $(0, 7)$ et du deuxième sont de $(7, 3)$. Supposons de plus que $T_i = 0$ et $C_i = 0$.

Du noeud i au noeud j, on a deux chemins (arcs) dont les étiquettes associées sont $(0, 7)$ et $(7, 3)$ dont aucune ne domine l'autre. Ainsi, si on relâche les contrainte d'intégrité, la solution ayant un flot de $\frac{1}{2}$ sur les deux arcs doit être considérée. Ainsi, pour cette solution, la contrainte (4.7) est vérifiée car $\frac{1}{2}(7-5) + \frac{1}{2}(3-5) = 0$. Cette solution qui est réalisable pour la formulation (4.2)-(4.8) a un coût $\frac{1}{2}.0 + \frac{1}{2}.7 = 3.5$. Pour la formulation (3.27)-(3.31), le premier chemin n'est pas réalisable car (3.31) n'est pas vérifiée, car $7 \notin [0, 5]$. Donc, pour cette formulation la solution optimale a un flot de 1 sur le premier arc et de 0 sur le deuxième et son coût est de $7+0 = 7$. Ce qui prouve que la formulation (4.2)-(4.8) a une solution de coût inférieur à la valeur optimale pour la formulation (3.27)-(3.31).

Ainsi, la nouvelle formulation est une relaxation de la formulation originale. Toutefois, comme nous résolvons par programmation dynamique, nous ne considérerons que les solutions entières, cas dans lequel les deux formulations sont équivalentes.

La relaxation lagrangienne consiste à relaxer une ou plusieurs contraintes, puis ajouter ces contraintes dans l'objectif, pondérées avec des multiplicateurs positifs.

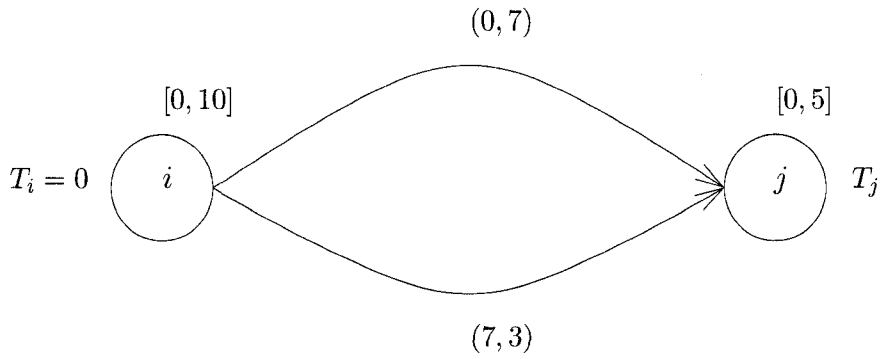

Figure 4.1 – Exemple où la formulation (4.2)-(4.8) n'est pas équivalente à (3.27)-(3.31)

Lorsque ces contraintes sont des contraintes de non positivité, les multiplicateurs associés sont positifs ou nuls. Afin d'obtenir la solution optimale, il suffit alors d'utiliser un algorithme itératif de sous-gradients qui permet de faire converger les multiplicateurs vers leurs valeurs optimales, ceci dans le cas où il n'y a pas d'écart d'intégrité.

C'est dans cette optique que les auteurs envoient les contraintes (4.7) vers l'objectif, pondérées avec des multiplicateurs de Lagrange.

Plus précisément, en dualisant les contraintes (4.7), on obtient le problème lagrangien suivant :

$$\mathcal{L}(u) = \min \sum_{i,j} \left(c_{ij} + \sum_{r_1 \in R_1} u_j^{r_1} (\max\{a_j, T_i^{r_1} + t_{ij}^{r_1}\} - b_j^{r_1}) \right) x_{ij} \tag{4.9}$$

$$
\begin{aligned}
s.c. \quad \sum_i x_{ij} - \sum_i x_{ji} &= e_j \quad \forall j \in N && (4.10)\\
x_{ij} &\geq 0 \quad \forall (i,j) \in A && (4.11)\\
x_{ij}\left(T_i^r + t_{ij}^r - T_j^r\right) &\leq 0 \quad \forall (i,j) \in A,\ \forall r \in R && (4.12)\\
T_j^r &\geq a_j^r \quad \forall j \in N,\ \forall r \in R && (4.13)\\
T_j^{r_2} &\leq b_j^{r_2} \quad \forall j \in N,\ \forall r_2 \in R_2. && (4.14)
\end{aligned}
$$

où $u_j^{r_1} \geq 0, j \in N,\ r_1 \in R_1$, sont les multiplicateurs de Lagrange associés.

Le dual lagrangien s'écrit alors :

$$
(DL) \qquad \begin{cases} \max_u \quad \mathcal{L}(u) \\ \\ s.c. \quad u_j^{r_1} \geq 0, \quad \forall j \in N,\ \forall r_1 \in R_1. \end{cases} \qquad (4.15)
$$

La dominance par rapport au coût lagrangien uniquement revient à garder, en chaque noeud j, les étiquette Pareto-optimales donnée par :

$$
\left[\begin{array}{l} \min \left\{ C_i + c_{ij} + \sum_{r_1 \in R_1} u_j^{r_1} (\max\{a_j^{r_1}, T_i^{r_1} + t_{ij}^{r_1}\} - b_j^{r_1}) \right\} \\ (i,j) \in A \\ T_i^{r_2} + t_{ij}^{r_2} \leq b_j^{r_2}\ ; r_2 \in R_2 \end{array} \right] \qquad (4.16)
$$

Remarque 2 *Puisqu'on dualise des contraintes de bornes sur les ressources, et contrairement à l'approche décrite dans la section 4.1, la résolution du problème lagrangien (4.9)-(4.14) avec les multiplicateurs optimaux, peut fournir des solutions non réalisables, et la valeur $v(DL)$ est un minorant de la valeur optimale $v^\star$ du problème PCC-CR.*

Proposition 1 *La résolution du PCC-CR en utilisant la formulation lagrangienne, en variant les multiplicateurs, peut fournir des solutions réalisables mais pas forcément optimales.*

Preuve :

Considérons la figure 4.2 représentant les étiquettes Pareto-optimales au noeud d d'un problème de PCC-FT. Supposons de plus, qu'on a une borne supérieure de b_d sur la consommation de temps en ce noeud. La résolution du PCC-CR, et en particulier le PCC-FT, en utilisant la dominance par rapport à la combinaison $C_d + \pi_d \times T_d$ avec différentes valeurs de π_d fournirait, respectivement les étiquettes (C_d^1, T_d^1), (C_d^3, T_d^3) et (C_d^4, T_d^4) pour des valeurs de π_d égales à α_1, α_3 et α_4. Bien que l'étiquette (C_d^2, T_d^2) pourrait éventuellement nous mener à un chemin optimal, il n'existe pas d'instanciation de π qui permettrait d'obtenir cette dernière. Les valeurs de multiplicateurs α_3 et α_4 fournissent des solutions réalisables. Le multiplicateur α_1 fournit une solution non réalisable. Aucune valeur des différents multiplicateurs ne permet d'obtenir la solution (C_d^2, T_d^2) qui est la solution optimale.

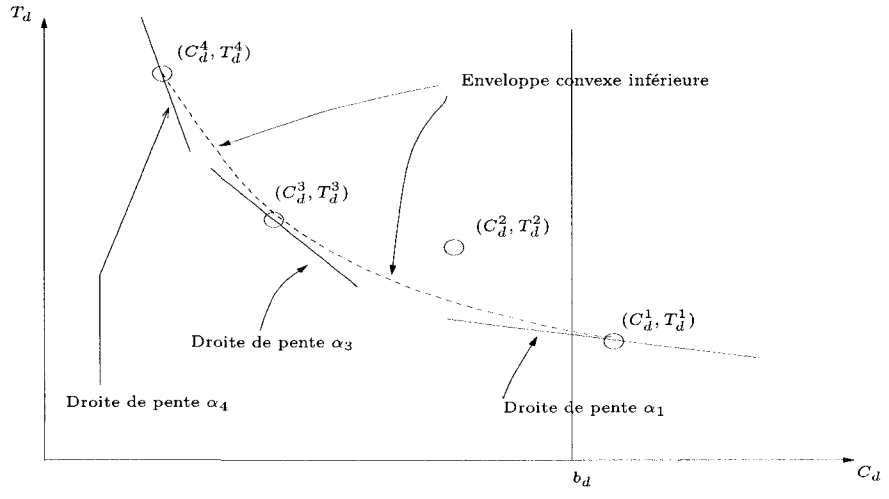

Figure 4.2 – Représentation des étiquettes en un noeud i.

4.2.2 Approche de projection par arcs

Bien que la projection par noeuds peut ne pas fournir la solution optimale, il existe une matrice de projection, dont les coefficients dépendent non seulement du noeud traité mais aussi de ses prédécesseurs, autrement dit une matrice qui dépend des arcs, qui pourra nous garantir l'optimalité, comme le montre le théorème 2.

En effet, soit $\phi_d(\widetilde{\pi})$ ($\widetilde{\pi} = \{\widetilde{\pi}\}_{(i,j)\in A}$) le coût du plus court chemin obtenu en dominant en chaque noeud $j \in N$ par rapport à la valeur de $\phi_j(\widetilde{\pi}_{ij}) = C_j + \widetilde{\pi}_{ij} \times T_i$, où les coefficients de projection $\widetilde{\pi}_{ij}$ dépendent cette fois du noeud traité j et de ses prédécesseurs, c'est-à-dire, des arcs (i,j) qui aboutissent au noeud j. La valeur optimale, en utilisant cette approche pour la résolution PCC-FT, est donnée par le problème :

$$\begin{cases} \min_{\widetilde{\pi}} \quad \phi_d(\widetilde{\pi}) \\ \\ s.c. \quad \widetilde{\pi}_{ij} \geq 0, \quad \forall\,(i,j) \in A. \end{cases} \tag{4.17}$$

Si on désigne par $v^\star$ la valeur optimale du problème PCC-FT, on obtient le théorème suivant :

Théorème 2 *(Nagih et Soumis (1999)) Il existe $\widetilde{\pi}$, un vecteur de multiplicateurs sur les arcs, tel que :*

$$\min_{\widetilde{\pi}\geq 0} \phi_d(\widetilde{\pi}) \;=\; v^\star.$$

Bien que ce théorème garantisse l'optimalité, trouver les multiplicateurs par arcs qui garantissent l'optimalité n'est pas une tâche évidente. En particulier, des multiplicateurs par arcs optimaux ne sont pas nécessairement les multiplicateurs de Lagrange.

Corollaire 1 *Soit $\widetilde{\pi}$ le vecteur de multiplicateurs donné par le théorème précédent, alors, pour tout vecteur de multiplicateurs π dépendant uniquement des noeuds, on a :*

$$v^{\star} \quad = \quad \min_{\widetilde{\pi}} \phi_d(\widetilde{\pi}) \quad \leq \quad \min_{\pi} \phi_d(\pi).$$

Notons que, contrairement à l'approche de projection par noeuds, qui est basée sur une relaxation de la formulation du problème de plus court chemin original, l'approche de projection par arcs ne nécessite aucune modification dans la formulation du problème original.

Afin d'exploiter plus profondément le théorème 2 de Nagih et Soumis (1999), on a utilisé l'approche de relaxation lagrangienne afin de trouver des multiplicateurs par arcs. On considère la formulation par arc suivante, équivalente à la formulation originale du PCC-CR :

$$\min \qquad \sum_{i,j} c_{ij} x_{ij} \tag{4.18}$$

$$s.c. \qquad \sum_{i} x_{ij} - \sum_{i} x_{ji} = e_j \quad \forall j \in V \tag{4.19}$$

$$x_{ij} \geq 0 \quad \forall (i,j) \in A \tag{4.20}$$

$$x_{ij} \left(T_i^r + t_{ij}^r - T_j^r\right) \leq 0 \quad \forall (i,j) \in A,\ \forall r \in R \tag{4.21}$$

$$T_j^r \geq a_j^r \quad \forall j \in V,\ \forall r \in R \tag{4.22}$$

$$x_{ij} \left(\max\{a_j^r, T_i^r + t_{ij}^r\} - b_j^r\right) \leq 0 \quad \forall (i,j) \in A,\ \forall r \in R. \tag{4.23}$$

En dualisant un sous-ensemble $R_1 \subset R$ des contraintes (4.23), on obtient la fonction de Lagrange suivante :

$$\mathcal{L}(u) = \min_{\substack{\text{S.C(4.19)-(4.23)} \\ \text{avec } r \in R \setminus R_1}} \sum_{i,j} \left(c_{ij} + \sum_{r_1 \in R_1} u_{ij}^{r_1} (\max\{a_j^{r_1}, T_i^{r_1} + t_{ij}^{r_1}\} - b_j^{r_1}) \right) x_{ij}.$$

où $u_{ij}^{r_1} \geq 0, (i,j) \in A$, $r_1 \in R_1$, sont les multiplicateurs de Lagrange par arcs associés.

Le dual lagrangien est :

$$\begin{cases} \max_u & \mathcal{L}(u) \\ s.c. & u_{ij}^{r_1} \geq 0, \ \forall (i,j) \in A, \ \forall r_1 \in R_1 \end{cases} \tag{4.24}$$

La dominance par rapport au coût (lagrangien) uniquement revient à garder, en chaque noeud j, l'étiquette donnée par :

$$\left[\begin{array}{l} \min_i \left\{ C_i + c_{ij} + \sum_{r_1 \in R_1} u_{ij}^{r_1} (\max\{a_j^{r_1}, T_i^{r_1} + t_{ij}^{r_1}\} - b_j^{r_1}) \right\} \\ (i,j) \in A \\ T_i^{R_2} + t_{ij}^{R_2} \leq b_j^{R_2} \end{array} \right] \tag{4.25}$$

Cette méthode a été expérimentée durant ce projet de thèse et les résultats n'ont pas été plus satisfaisants que ceux de la méthode utilisant les multiplicateurs sur les noeuds. Les sections qui suivent permettront d'éclaircir pourquoi la méthode de projection sur les noeuds est instable. Cette analyse s'applique aussi à la méthode de projection par arcs, car comme on va le prouver, les multiplicateurs de Lagrange obtenus par agrégation des contraintes ne sont pas optimaux.

4.3 Comparaison des différents algorithmes de résolution

Dans cette section, on se propose d'exposer, d'abord, en détail les différents algorithmes utilisés pour la résolution des PCC-CR. Par la suite, on se propose de comparer ces méthodes à l'approche de résolution de Nagih et Soumis et essayer de situer cette méthode par rapport à ses précédentes.

4.3.1 Algorithme de programmation dynamique avec ressources non dominées

L'algorithme de programmation dynamique avec ressources non dominées APD-ND est une heuristique de APD, utilisée afin de réduire l'espace des états et produire rapidement des solutions réalisables. Dans cet algorithme, la prolongation est effectuée comme pour APD. La dominance se fait sur un sous-ensemble de ressources $\mathcal{R}_1 \subseteq \mathcal{R}$.

Désignons par $\mathcal{Z}_{ND}$ le coût du meilleur chemin de o à d produit par cet algorithme.

Proposition 2 *On a :*

$$\mathcal{Z}_{OPT} \leq \mathcal{Z}_{ND}.$$

Preuve :
Si $\mathcal{P}$ est le chemin fourni par APD-ND, alors $\mathcal{P}$ est réalisable pour APD.

Remarque 3 *Dans certains cas, on a $\mathcal{Z}_{OPT} < \mathcal{Z}_{ND}$ comme le montre le réseau de l'exemple sur la figure 4.3. Notons par (c_{ij}, t_{ij}) le couple correspondant au coût et à la consommation de ressource, respectivement, sur l'arc (i, j) et par $[a_i, b_i]$ les fenêtres de temps. Si on applique APD-ND, en dominant sur le coût uniquement, c'est l'étiquette $E_1 = (6, 3)$ qui est générée, alors que l'étiquette optimale est $E_2 = (5.5, 2.5)$ (de coût 5.5).*

4.3.2 Algorithme de relaxation lagrangienne : APD-L

En utilisant la formulation (4.2)-(4.7), la prolongation des étiquettes lors de la résolution en utilisant APD-L se fait comme pour APD, la dominance se fait en

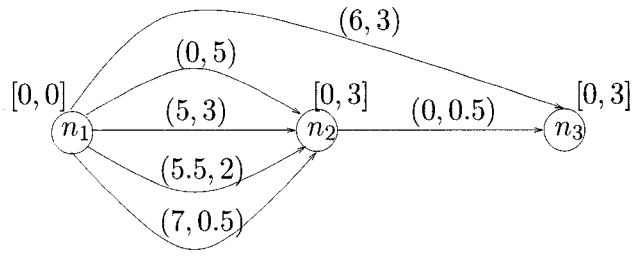

Figure 4.3 – Exemple où $\mathcal{Z}_{OPT} < \mathcal{Z}_{ND}$

utilisant le sous-ensemble de ressources $\mathcal{R}_1$ et les nouveaux coûts lagrangiens donnés par :

$$\min_{(i,j)\in\mathcal{A}} \left(C_i + c_{ij} + \sum_{r_1\in\mathcal{R}_1} u_j^{r_1}(\max\{a_j^{r_1}, T_i^{r_1} + t_{ij}^{r_1}\} - b_j^{r_1}) \right) ; r_2 \in \mathcal{R}_2$$

où $\mathcal{R}_2 = \mathcal{R} \setminus \mathcal{R}_1$. Désignons par APD-L un tel algorithme.

4.3.3 APD-L à l'intérieur d'un algorithme de génération de colonnes

Dans un algorithme de génération de colonnes, les PCC-CR apparaissent comme des sous-problèmes. À chaque itération de génération de colonnes, une colonne réalisable représente un chemin réalisable de coût réduit négatif. Ainsi APD et APD-ND peuvent être utilisés pour résoudre les sous-problèmes dans un algorithme de génération de colonnes. Par contre, APD-L ne peut être utilisé étant donné que les chemins générés par ce dernier peuvent être non réalisables. D'où l'introduction par Nagih et Soumis (2000) de l'algorithme APD-LND, qui n'est autre qu'un algorithme APD-ND avec des coûts lagrangiens sur les arcs.

Désignons par $\mathcal{Z}_{LND}(u)$ le coût du meilleur chemin généré par cet algorithme, u étant le vecteur de multiplicateurs de Lagrange optimaux.

4.3.4 Algorithme de Nagih et Soumis

Nagih et Soumis (2000) présentent un algorithme (qu'on notera dans ce qui suit par N-S) qui opère en deux étapes à chaque itération k de l'algorithme de génération de colonnes, afin de produire des colonnes de coût marginal négatif. La première utilise APD-L afin de trouver des multiplicateurs de Lagrange efficaces u_k. La deuxième consiste à appliquer APD-LND en utilisant les multiplicateurs de Lagrange trouvés dans la première étape afin de générer des colonnes réalisables.

Étant donné, qu'à une itération k de l'algorithme de génération de colonnes, trouver les multiplicateurs optimaux u_k^* nécessiterait plusieurs itérations successives de APD-L, cette méthode peut s'avérer coûteuse. La méthode N-S utilisée par Nagih et Soumis consiste à appliquer une seule fois APD-L puis à appliquer APD-LND en utilisant les multiplicateurs u_k trouvés afin de produire des colonnes réalisables et de coût marginal négatif à chaque itération k de génération de colonnes. Afin de mettre à jour les multiplicateurs de Lagrange, on utilise une méthode de sous-gradients. Dans cette méthode, on choisit d'abord une suite de pas (a_k) qui vérifient :

$$\sum_{k=0}^{\infty} \theta_k \rightarrow \infty$$

et

$$\theta_k \rightarrow 0.$$

aussi appelées les conditions de Pollack (1961). On applique en premier lieu APD-L en utilisant les multiplicateurs u_{k-1} de l'itération précédente, on trouve les sous-gradients $Sg_k(i) = T_i - b_i$, correspondants au noeud i, ensuite on calcule les nouveaux multiplicateurs de Lagrange $u_k(i) = u_{k-1}(i) + a_k \times Sg_k(i)$. Cette heuristique est basée sur le fait que lorsque k est grand, le vecteur C_k des coûts réduits sur les arcs du réseau ne change pas beaucoup d'une itération à une autre de l'algorithme de génération de colonnes. Ainsi, lorsque k est grand, on peut espérer voir u_k converger vers une valeur optimale.

4.3.5 Remarques sur la convergence de l'algorithme de Nagih et Soumis

On se propose dans cette section de comparer qualitativement la convergence de APD-ND et la convergence de l'algorithme N-S proposé par Nagih et Soumis (2006). Rappelons qu'un algorithme efficace et robuste doit fournir une solution dont la valeur Z^{Meth} vérifie $Z^{ND} > Z^{Meth} \geq Z^{Opt}$ où Z^{Opt} est la valeur optimale et Z^{ND} est la valeur de la solution fournie par les stratégies de dominance sur un sous-ensemble de ressources qu'on a présenté auparavant. Ainsi, on montrera dans cette section que l'algorithme de Nagih et Soumis ne vérifie pas les critères d'efficacité et de robustesse voulues. De plus, cet algorithme est très sensible aux paramètres dont il dépend.

Afin de confirmer ou infirmer la théorie proposée, on a étudié plusieurs variantes de l'algorithme en changeant à chaque fois la stratégie d'ajustement des multiplicateurs sur différents problèmes réels (horaires d'équipages), et en changeant les paramètres de l'algorithme tels que les pas de la méthode de sous-gradients.

Le premier facteur qui peut avoir le plus d'impact sur la qualité des solutions est la façon avec laquelle les multiplicateurs sont mis à jour. En effet, étant donné qu'on a des multiplicateurs par noeuds et non par arcs et étant donné que la nouvelle formulation (4.2)-(4.7) n'est qu'une relaxation de la formulation originale du PCC-CR, le choix d'une stratégie d'ajustement des multiplicateurs est une opération délicate, lors de la résolution par un algorithme de programmation dynamique. On a exploré et testé plusieurs stratégies sur des problèmes différents en nature et en taille. Voici quelques unes des plus importantes stratégies qu'on a exploré.

Pour commencer, on s'est proposé d'étudier, avant tout, les possibles améliorations qu'on peut effectuer pour une seule itération de l'algorithme des sous-gradients dans le réseau de la façon suivante :

1. Mettre à jour les multiplicateurs sur les noeuds au fur et à mesure qu'on génère des étiquettes lors de l'algorithme de programmation dynamique et mettre à

jour le multiplicateur selon la plus grande violation de la contrainte de borne supérieure au noeud correspondant. Ceci implique qu'on suppose de mettre un flot égal à 1 sur l'arc entrant au noeud considéré correspondant à la plus grande violation de la contrainte. Ainsi, la contrainte $\sum_i x_{ij} \left(\max\{a_j^r, T_i^r + t_{ij}^r\} - b_j^r\right) \leq 0$ devient $\left(\max\{a_j^r, T_i^r + t_{ij}^r\} - b_j^r\right) \leq 0$ où (i,j) est l'arc portant le flot égal à 1. Le multiplicateur de Lagrange qui est calculé à partir de cette contrainte est donné par $u_{k+1} = u_k + \theta_k \left(\max\{a_j^r, T_i^r + t_{ij}^r\} - b_j^r\right)$, où $\{\theta_k\}_{k\geq 1}$ est une suite de pas. Cette stratégie a l'avantage d'être facile à implémenter mais suppose une optimalité locale.

2. On peut supposer aussi qu'étant donné un noeud j, le flot est non nul pour tout $(i,j) \in A$ correspondant à la contrainte $\sum_i x_{ij} \left(\max\{a_j^r, T_i^r + t_{ij}^r\} - b_j^r\right) \leq 0$. Ainsi, on additionne la somme des violations de la contrainte pour mettre à jour le multiplicateur.

3. Traiter les multiplicateurs à reculons, c'est-à-dire, à partir du noeud destination vers le noeud source. Ceci peut se faire de plusieurs façons :

   (a) Considérer uniquement les étiquettes (chemins) qui violent la contrainte de borne supérieure au noeud destination, les ordonner en ordre décroissant de violation de la contrainte. Par la suite, on remonte le chemin correspondant à cette étiquette jusqu'au noeud source, en ajustant les multiplicateurs à chaque fois qu'il y a une violation d'une contrainte en un noeud intermédiaire. Un noeud déjà traité n'est pas traité de nouveau en parcourant un autre chemin. L'avantage de cette stratégie est de considérer le moins de chemins possible afin de minimiser les temps de calculs, en espérant que la plupart des chemins qui violent la contrainte de borne supérieure au noeud destination, violent aussi quelques contraintes dans les noeuds intermédiaires. L'inconvénient de cette stratégie est qu'on n'attribue pas nécessairement un multiplicateur à chaque noeud où on a eu une violation de l'une des contraintes des fenêtres de ressources.

(b) On peut faire exactement comme ci-haut, mais en remontant tous les chemins à partir du noeud destination. Cette méthode peut s'avérer ardue car on risque d'explorer inutilement plusieurs fois les mêmes sous-chemins.

(c) Dans les deux dernières stratégies décrites, on ne fait pas de mise à jour des multiplicateurs une fois changés à l'intérieur d'une itération de l'algorithme de sous-gradients. Ainsi, on peut faire une mise à jour du multiplicateur de Lagrange à chaque fois qu'un chemin a violé une des contraintes des fenêtres de ressources et, par la suite, on peut prendre le multiplicateur qui a la plus grande valeur. On peut aussi considérer la somme cumulative des multiplicateurs.

Le deuxième facteur qui risque d'influencer la convergence de l'algorithme de sous-gradients est la suite de pas $\{\theta_k\}_{k\geq 1}$. Il est bien connu que le pas de l'algorithme des sous-gradients n'influe pas sur la valeur vers laquelle l'algorithme converge mais la vitesse avec laquelle il converge tant que la suite de pas vérifie les conditions de Pollack (1961). Afin de vérifier la convergence de l'algorithme, on s'est proposé d'abord de le vérifier avec plusieurs types de pas qui vérifient les conditions de Pollack (1961). Plusieurs suites vérifiant ces conditions existent dans la littérature. En premier lieu, on a expérimenté la suite normalisée donnée par : $\theta_k = \frac{\lambda_k[UB_i - \mathcal{L}(u_i^k)]}{\|T_i^k - b_i\|^2}$, UB_i étant une borne supérieure de la fonction de coût au noeud i, $\mathcal{L}(u_i^k)$ la valeur de la fonction lagrangienne au noeud i, λ_k est une sous-suite qu'on divise par un facteur supérieur à 2 à chaque fois qu'on remarque que l'objectif lagrangien arrête de décroître et T_i^k est la valeur de la ressource au noeud i à l'itération k de l'algorithme de sous-gradients.

Cette suite de pas, bien qu'elle soit connue pour être efficace en théorie, possède cependant l'inconvénient d'impliquer plusieurs calculs : norme, borne supérieure, mise à jour de la sous-suite λ_k.

Ainsi, on a choisi, d'abord et afin de tester la méthode, la suite de pas normalisée $\theta_k = \frac{\lambda_k[UB_i - \mathcal{L}(u_i^k)]}{\|T_i^k - b_i\|^2}$ en itérant plusieurs fois l'algorithme de sous-gradients à l'intérieur d'une même itération de génération de colonnes.

En utilisant un procédé de réoptimisation à l'intérieur de l'algorithme de génération de colonnes, la mise à jour des pas pour l'algorithme de sous-gradients s'impose en deux endroits : d'abord à l'intérieur d'une même itération de génération de colonnes (GC), puis dans les différentes itérations de GC et ceci en utilisant les multiplicateurs de l'itération précédente (Nagih et Soumis (2000)). Cette approche est encore plus efficace pendant les dernières itérations de l'algorithme de génération de colonnes dans lesquelles les coûts réduits sur les arcs ne changent pas de façon significative.

Étant donné que notre but principal n'est pas uniquement de faire converger l'algorithme des sous-gradients mais surtout d'améliorer et de varier la qualité des solutions fournies par les sous-problèmes au problème maître, on s'est proposé de considérer des suites de pas faciles à calculer (du point de vue informatique), qui satisfont les conditions de Pollack afin de varier le plus possible les solutions envoyées au problème maître, sans pour autant que ce soit coûteux en calculs. Ainsi, on a choisi des suites de pas du type $\theta_k(\varepsilon) = \frac{\varepsilon}{k}$, où ε est un coefficient assez petit, qui est de l'ordre de l'inverse des coûts réduits courants, afin d'éviter les oscillations de l'algorithme pendant les premières itérations de GC.

Dans la même optique et étant donné que les multiplicateurs peuvent varier sensiblement selon l'amplitude de la contrainte de ressource considérée, on s'est aussi proposé de considérer une suite de pas différente pour chaque ressource en considérant $\theta_r = \frac{1}{\max_i(b_i^r - a_i^r)}$ qui est la longueur maximale que pourrait prendre la fenêtre de ressource pour chaque noeud, valeur qu'on peut connaître avant même de commencer la résolution.

4.3.6 Résultats numériques sur la convergence de l'algorithme N-S

Pour valider la théorie proposée, on a choisi d'essayer numériquement les différentes approches qu'on a proposé pour différents problèmes tests. Dans cette section,

on présentera des résultats numériques qu'on a obtenus sur quatre problèmes. Ces problèmes tests sont des problèmes de rotations d'équipages pour la compagnie aérienne *Air Canada.* Dans ces problèmes, on essaye de construire des suites de quarts (rotations) pour les équipages. Les problèmes sont de tailles différentes (moyenne à grande). Ils ont un point commun qui est relatif aux conventions collectives dont bénéficient les employés. On a une ressource pour le temps total de travail d'un équipage calculé en minutes et l'intervalle de cette ressource est de $[0, 720]$ pour tous les noeuds. Une deuxième ressource qui est le temps de vol calculé aussi en minutes et son intervalle de contrainte est de $[0, 480]$ pour tous les noeuds, une troisième ressource pour le nombre de vols dont l'intervalle de contraintes est $[0, 5]$ pour tous les noeuds et une quatrième pour le nombre de quarts de travail dont l'intervalle de contrainte est $[0, 4]$, pour tous les noeuds aussi.

Il est clair que, pour ce type de problème, il n'est pas très pertinent de relâcher la ressource 3 ou la ressource 4 puisque leur ensemble de valeurs est très petit par rapport aux deux autres ce qui impliquera une altération de la solution sans pour autant avoir un gain significatif dans les temps de calcul.

Ainsi, pour tester l'efficacité de l'algorithme, on effectuera des tests en relâchant les fenêtres de la première, la deuxième et la troisième ressource alternativement ou simultanément.

Dans ce qui suit, on propose des tableaux de valeurs, un pour chaque problème considéré, avec différentes suites de pas. Dans ces tableaux, on désignera par *Pas(k)* la suite de pas considérée, *Ress. Dom.* étant le nombre de ressources sur lesquels on a dominé, *Itrérations GC* le nombre d'itérations de l'algorithme de génération de colonnes, *Temps(SPP)* est le temps total (en secondes) de la résolution des sous-problèmes à l'intérieur de GC et *Objectif* est la valeur de l'objectif atteinte à la dernière itération de GC.

4.3.6.1 Problème I

Il s'agit d'un petit problème de rotations d'équipages d'Air Canada ayant 21 sous-réseaux, 928 noeuds, 15494 arcs et 112 tâches (vols). Pour tirer des résultats concluants sur la méthode dans un algorithme de génération de colonnes, on a choisi comme critère de comparaison, la valeur de l'objectif du problème général. On a choisi de relâcher 3 ressources en dominant sur une ressource assez significative soit le temps de vol de façon qu'il n'y ait pas d'écart significatif entre la valeur de l'objectif lorsqu'on domine sur toutes les ressources et lorsqu'on domine uniquement sur la ressource non relâchée.

Pour faire une étude comparative, on a fait d'abord la résolution en dominant sur le coût et les 4 ressources ce qui donne la valeur optimale du problème et on a fait la résolution en relâchant les 3 ressources en dominant sur le coût et une seule ressource. Les résultats sont présentés dans le tableau 4.1.

Tableau 4.1 – Problème I : Dominance sur 4 ressources versus sur 1 ressource

Ress. dom.	Itérations GC	Temps(SPP)	Objectif
4	61	59.0	15237.5
1	60	29.9	15358.4

Dans le tableau 4.2, on montre les différents résultats qu'on obtient à chaque fois qu'on opte pour une des stratégies d'ajustement des multiplicateurs de la méthode N-S. Ces différents tests ont tous été effectués pour une même suite de pas : $\theta_k = \frac{\lambda_k[UB_i - \mathcal{L}(u_i^k)]}{\|T_i^k - b_i\|^2}$. Pour la même stratégie (par exemple la stratégie 3.a), on a choisi de faire commencer la méthode à partir d'itérations avancées de l'algorithme de génération de colonnes ; ces expérimentations seront notées par des primes (3.a' pour le même exemple).

Tableau 4.2 – Problème I : Stratégies d'ajustement des multiplicateurs

Stratégie	Itérations GC	Temps(SPP)	Objectif
1	59	74.3	15269.1
2	63	67.7	15331.3
3.a	73	71.2	15962.1
3.a'	61	68.2	15314.3
3.b	65	79.0	15251.5
3.c	63	61.4	15460.5

Dans le tableau 4.3, on a exposé différents résultats pour différentes suites de pas, ceci en utilisant la stratégie 1 qui donne les meilleurs résultats. On remarque une grande sensibilité de la valeur de l'objectif à la suite de pas considérée.

Tableau 4.3 – Problème I : Variations de l'objectif selon les suites de pas

Suite(k)	Itérations GC	Temps(SPP)	Objectif
$\frac{3.10^{-2}}{k}$	62	62.7	15714.1
$\frac{2.10^{-2}}{k}$	72	74.3	15408.0
$\frac{10^{-2}}{k}$	63	62.4	15460.5
$\frac{9.10^{-3}}{k}$	73	77.3	15768.2
$\frac{10^{-5}}{k}$	64	67.6	15731.1
$\frac{2.10^{-5}}{k}$	68	71.1	15751.5
$\frac{10^{-5}}{k^2}$	77	78.1	15687.0

Dans ces résultats, on devrait s'attendre à ce que la méthode proposée ait des temps de calcul entre 59.0 et 29.9 secondes pour un objectif entre 15237.5 et 15358.4, ce qui, comme on peut constater, est loin d'être le cas.

4.3.6.2 Problème II

On considère un autre problème du même type que le problème I avec une taille de réseau un peu plus grande, soit : 23 sous-réseaux, 1154 noeuds, 22593 arcs et 225 vols.

Dans le tableau 4.4, la première ligne donne le résultat obtenu par GC en dominant sur le coût et 4 ressources. La deuxième ligne est le résultat obtenu par GC en dominant sur le coût et une ressource.

Tableau 4.4 – Problème II : Dominance sur 4 ressources versus sur 1 ressource

Ress. dom.	Itérations GC	Temps(SPP)	Objectif
4	105	227.0	95035.2
1	93	84.2	95288.9

Dans le tableau 4.5, on montre les différents résultats qu'on a obtenus en appliquant l'algorithme GC à cette instance en utilisant la méthode N-S et en changeant à chaque fois de stratégie d'ajustement des multiplicateurs et ceci pour une même suite de pas égale à $\theta_k = \frac{\lambda_k[UB_i - \mathcal{L}(u_i^k)]}{\|T_i^k - b_i\|^2}$.

Tableau 4.5 – Problème II : Stratégies d'ajustements des multiplicateurs

Stratégie	Itérations GC	Temps(SPP)	Objectif
1	101	188.1	95288.9
2	107	218.1	95021.2
2'	92	170.1	95288.9
3.a	97	167.9	95134.1
3.b	108	212	95160.6
3.c	93	174.2	95288.9

Dans le tableau 4.6, on montre pour cette instance, des résultats numériques

illustrant la grande variation de l'objectif lorsqu'on change la suite de pas. Ceci a été appliqué pour la stratégie 2 qui donne les meilleurs résultats dans ce cas-ci.

Tableau 4.6 – Problème II : Variation de l'objectif selon les suites de pas

Suite(k)	Itérations GC	Temps(SPP)	Objectif
$\frac{10^{-7}}{k}$	97	167.9	95288.9
$\frac{10^{-5}}{k}$	100	189.2	95288.9
$\frac{10^{-3}}{k}$	98	177.2	95288.9
$\frac{9.10^{-4}}{k}$	108	212	95160.6
$\frac{10^{-2}}{k}$	102	201.3	95325.5

Encore une fois les résultats obtenus sont loin des résultats escomptés. Ainsi, on s'attendait à ce que la durée de résolution, lors de la résolution par N-S, ne dépasse pas 227.0 et dont l'objectif varie entre 95035.2 et 95288.9, alors qu'on remarque que ce n'est le cas que pour un très petit nombre de ces tests.

4.3.6.3 Problème III

On considère un autre problème de rotations d'équipages pour la compagnie aérienne Air Canada avec le même type de données que les deux considérées précédemment avec 28 sous-réseaux, 1610 noeuds, 27540 arcs et 406 vols.

Les résultats de la résolution par GC en dominant sur le coût et 4 ressources et en dominant sur le coût et une ressource uniquement sont présentés dans le tableau 4.7.

Dans le tableau 4.8, on montre différents résultats numériques de la résolution de cette instance par GC en changeant à chaque fois de stratégie d'ajustement des multiplicateurs et ceci pour une même suite de pas égale à $\theta_k = \frac{\lambda_k[UB_i - \mathcal{L}(u_i^k)]}{\|T_i^k - b_i\|^2}$.

Tableau 4.7 – Problème III : Dominance sur 4 ressources versus sur 1 ressource

Ress. dom.	Itérations GC	Temps(SPP)	Objectif
4	125	177.7	41600.0
1	182	104.2	42054.7

Tableau 4.8 – Problème III : Comparaison des approches de résolution à l'intérieur de GC

Stratégie	Itérations GC	Temps(SPP)	Objectif
1	175	209.0	42688.1
1'	169	189.1	438324.2
2	172	188.1	43431.1
3.a	188	201.8	42295.0
3.a'	181	197.8	42218.4
3.b	175	188.1	42888.4
3.c	182	199.8	42784.1

Les résultats numériques du tableau 4.9 illustrent la variation de l'objectif selon la suite de pas considérée, ceci en considérant la stratégie 3.a' qui donne les meilleurs résultats.

Tableau 4.9 – Problème III : Variation de la valeur de l'objectif avec les suites de pas

Suite(k)	Itération GC	Temps(SPP)	Objectif
$\frac{10^{-8}}{k}$	181	195.8	42585.0
$\frac{10^{-7}}{k}$	175	187.8	42888.4
$\frac{10^{-6}}{k}$	169	178.9	42468.0
$\frac{10^{-5}}{k}$	175	198.0	42608.2
$\frac{10^{-1}}{k}$	119	129.8	43324.2

Pour chacun des tests qu'on a effectué, la majorité n'ont pas répondu à nos attentes pour ce qui est de la valeur de l'objectif et des temps de résolution.

4.3.6.4 Problème IV

On considère un autre problème de rotations d'équipages pour la compagnie aérienne Air Canada avec le même type de données que les deux considérés précédemment avec 23 sous-réseaux, 1586 noeuds, 33445 arcs et 389 vols.

Comme pour les autres instances, on rapporte dans le tableau 4.10 les résultats de la résolution par un algorithme GC en utilisant, d'une part, la dominance sur le coût et 4 ressources et en utilisant, d'autre part, la dominance sur le coût et une ressource uniquement.

Tableau 4.10 – Problème IV : Dominance sur 4 ressources versus sur 1 ressource

Ress. dom.	Itérations GC	Temps(SPP)	Objectif
4	187	681.4	70755.0
1	190	247.6	72942.7

Le tableau 4.11 donne les résultats qu'on a obtenus en changeant à chaque fois de stratégie d'ajustement des multiplicateurs pour une même suite de pas égale à $\theta_k = \frac{\lambda_k[UB_i - \mathcal{L}(u_i^k)]}{\|T_i^k - b_i\|^2}$.

Tableau 4.11 – Problème IV : Approches de résolution utilisant N-S

Stratégie	Itérations GC	Temps(SPP)	Objectif
1	199	505.2	74113.4
1'	192	488.0	72881.4
2	185	492.1	72673.3
2'	195	497.3	72001.4
3.b	198	503.1	73171.5
3.b'	178	478.9	73341.1
3.c	181	474.1	72872.7

Les résultats numériques du tableau 4.12 illustrent la variation de l'objectif selon

la suite de pas considérée, ceci à l'intérieur de la stratégie 2' qui donne les meilleurs résultats pour ce problème.

Tableau 4.12 – Problème IV : Variations selon les suites de pas

Suite(k)	Itération GC	Temps(spp)	Objectif
$\frac{10^{-9}}{k}$	191	495.8	73005.1
$\frac{10^{-8}}{k}$	188	487.2	72078.3
$\frac{0,6.10^{-8}}{k}$	179	478.9	73329.1
$\frac{10^{-6}}{k}$	192	498.0	72997.3
$\frac{10^{-5}}{k}$	189	484.8	72910.7
$\frac{10^{-4}}{k}$	198	503.1	73134.0

Dans ces tableaux, on remarque encore une fois une grande variation dans les résultats obtenus lorsqu'on change de pas et de stratégie de résolution, ce qui prouve encore une fois la grande sensibilité de la méthode aux paramètres choisis.

4.3.6.5 Résultats numériques sur l'approche de projection par arcs

Dans ce qui suit, on s'est proposé de tester l'approche de projection par arcs en utilisant les mêmes stratégies d'ajustement des multiplicateurs qu'on a utilisé dans l'approche de projection par noeuds mais en considérant des multiplicateurs pour chaque arc. Des résultats numériques pour le problème IV pour différentes suites de pas sont donnés dans le tableau 4.13.

Dans ces différents tests, on remarque une grande sensibilité de la valeur de l'objectif du problème maître par rapport à un changement minime de la valeur du pas. De plus, la valeur de l'objectif dépasse souvent la valeur de l'objectif de résolution par GC en dominant sur le coût et une seule ressource. De plus, les temps de calcul ont augmenté considérablement (plus que 4 fois).

Tableau 4.13 – Comparaison des approches de résolution par N-S pour l'approche de projection par arcs

Suite(k)	Itérations GC	Temps(SPP)	Objectif
$\frac{10^{-4}}{k}$	193	749.2	73105.2
$\frac{10^{-5}}{k}$	189	717.4	73201.3
$\frac{10^{-6}}{k}$	197	770.9	71009.1
$\frac{10^{-7}}{k}$	192	738.0	72986.8
$\frac{10^{-8}}{k}$	188	714.8	71910.2
$\frac{10^{-9}}{k}$	188	713.1	72134.4

4.3.7 Étude théorique de la convergence de l'algorithme de Nagih et Soumis

Désignons par $\mathcal{Z}_{NS}^{Rec}(a_k)$, la valeur de la solution donnée par l'algorithme de génération de colonnes appliquée sur un problème de recouvrement en utilisant la méthode NS pour résoudre les sous-problèmes. Ceci pour une suite de pas $\{a_k\}_{k\geq 1}$ donnée, par $\mathcal{Z}_{ND}^{Rec}$ celle trouvée par l'utilisation de l'algorithme APD-ND pour résoudre les sous-problèmes et $\mathcal{Z}_{Opt}^{Rec}$ la valeur optimale de la solution du problème.

Le but initial de l'algorithme de Nagih et Soumis (N-S) est de fournir rapidement des chemins réalisables, de bonne qualité au problème maître de l'algorithme de génération de colonnes. Selon nos critères d'efficacité et de stabilité, la valeur de la solution fournie doit être dans l'intervalle $[\mathcal{Z}_{Opt}^{Rec}, \mathcal{Z}_{ND}^{Rec}]$, et la plus proche possible de $\mathcal{Z}_{Opt}^{Rec}$. De plus, il fallait que les temps de résolution ne soient pas beaucoup plus grands que ceux de l'algorithme APD-ND. On montrera dans cette section que :

- L'algorithme N-S peut produire dans certains cas une solution moins bonne que l'algorithme APD-ND.
- Cette détérioration de la borne peut être très grande.
- La qualité de la borne obtenue avec N-S varie de façon discontinue en fonction de la suite de pas choisie, i.e. deux suites de pas très semblables $\{a_k\}$ et $\{b_k\}$

peuvent donner une excellente borne ($\mathcal{Z}_{NS}^{Rec}(a_k) = \mathcal{Z}_{Opt}^{Rec}$) et une très mauvaise borne ($\mathcal{Z}_{NS}^{Rec}(b_k) > \mathcal{Z}_{ND}^{Rec}$).

Proposition 3 *Il existe un problème de couverture de tâches et une suite de pas* $(a_k)_{k\geq 1}$ *satisfaisant les conditions de Pollack, pour lesquels on a :*

$$\mathcal{Z}_{ND}^{Rec} < \mathcal{Z}_{NS}^{Rec}(a_k).$$

Preuve :

L'existence d'un tel problème est présentée dans l'exemple suivant :
Considérons le problème de couverture avec des chemins du noeud n_1 au noeud n_3 sur le graphe orienté de la figure 4.3 avec une seule tâche au noeud destination. Les solutions réalisables de ce problème sont les chemins réalisables de n_1 à n_3. De plus ce problème est équivalent au PCC-CR sous-jacent. On a au total 5 chemins entre n_1 et n_3 (non nécessairement tous réalisables) d'étiquettes respectives $E_1 = (6, 3)$, $E_2 = (0, 5.5)$, $E_3 = (5, 3.5)$, $E_4 = (5.5, 2.5)$ et $E_5 = (7, 1)$. Notons que si on applique un algorithme de génération de colonnes sur ce problème en résolvant les PCC-CR avec APD, l'étiquette optimale est E_4 de coût 5.5 alors que l'algorithme de génération de colonnes utilisant APD-ND donne l'étiquette E_1 de coût 6.

Appliquons N-S à ce problème avec $a_k = \frac{0.7}{k}$ qui est une suite vérifiant les conditions de Pollack. Quelques itérations de l'algorithme sont illustrées dans le tableau 4.14. Les étapes sont présentées en ordre, de gauche à droite. Par exemple, pour la deuxième itération ($k = 2$), on applique APD-L avec $u_1(n_2) = 1.4$ et $u_1(n_3) = 1$ trouvés dans la première itération ce qui donne l'étiquette optimale E_2 et un sous-gradient au noeud n_2 de $Sg_2(n_2) = 2$, $Sg_2(n_3) = 1.5$, $u_2(n_2) = u_1(n_3) + a_2 \times Sg_2(n_2) = 1.4 + \frac{0.7}{2} \times (2) = 2.1$ et $u_2(n_3) = u_1(n_3) + a_2 \times Sg_2(n_3) = 1 + \frac{0.7}{2} \times (2) = 1.8$. Les coûts lagrangiens des étiquettes pour la deuxième itération, qui sont donnés par $C_{n_3}^L(E_i) = C_{n_3}(E_i) + u_2 \times (T_{n_2} - b_{n_2}) + u_3 \times (T_{n_3} - b_{n_3})$, sont alors : $C^L(E_1) = 6$,

Tableau 4.14 – Itérations de l'algorithme N-S pour la suite de pas $a_k = \frac{0.7}{k}$

k	Solution	$Sg_k(n_2)$	$Sg_k(n_3)$	$u_k(n_2)$	$u_k(n_3)$	$\mathcal{Z}_{LND}$	Étiq Opt
0	E_2	2	2.5	0	0	-	-
1	E_2	2	1.5	1.4	1	7	E_5
2	E_2	2	1.5	2.1	1.8	7	E_5
3	E_5	-2.5	-2	1.5	1.6	7	E_5
4	E_5	-2.5	-2	1.4	1.5	7	E_5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

$C^L(E_2) = 8$, $C^L(E_3) = 5.9$, $C^L(E_4) = 2.5$, $C^L(E_5) = -1.85$. La colonne générée par l'algorithme APD-LND est alors le chemin d'étiquette E_5.

À l'intérieur d'un algorithme de génération de colonnes, lors de la première itération, la solution donnée par N-S est le chemin associé à E_5 de coût 7. La variable duale de la contrainte de couverture devient alors 7. À la deuxième itération, le coût réduit de tous les chemins est égal à $C_{n_3}(E_i) = C_{n_3}(E_i) - 7$ donc $C_{n_3}(E_5) = 0$. Ainsi, si le critère d'arrêt de l'algorithme est le signe des coûts réduits, on voit que pour $k = 1$ l'optimalité est atteinte et le chemin fourni par N-S est donc le chemin associé à E_5.

La figure 4.4 représente les coûts lagrangiens des 4 chemins passants par le noeud n_2 en fonction du choix du premier terme de la suite de pas. On observe sur cette figure que la solution de APD-L sera E_2 ou E_5 et que $u_k(n_2)$ converge vers 1.5 qui représente la valeur de μ au point d'intersection entre les deux droites. En effet, les deux droites, correspondant au coût lagrangien des étiquettes E_2 et E_5, constituent l'enveloppe convexe inférieure. La solution donnée par l'algorithme lagrangien appartient à l'une de ces deux droites ; de plus, sa valeur converge vers μ intersection des deux droites correspondant à E_2 et E_5. Par la suite, même si on poursuivait les itérations de l'algorithme de génération de colonnes, afin de laisser converger les multiplicateurs, le chemin généré va toujours être le chemin associé à E_5. Ainsi quel que soit le critère

d'arrêt, avec cette suite de pas, on obtient : $6 = \mathcal{Z}_{ND}^{Rec} < \mathcal{Z}_{NS}^{Rec}(a_k) = 7$. Ce qui met fin à la preuve.

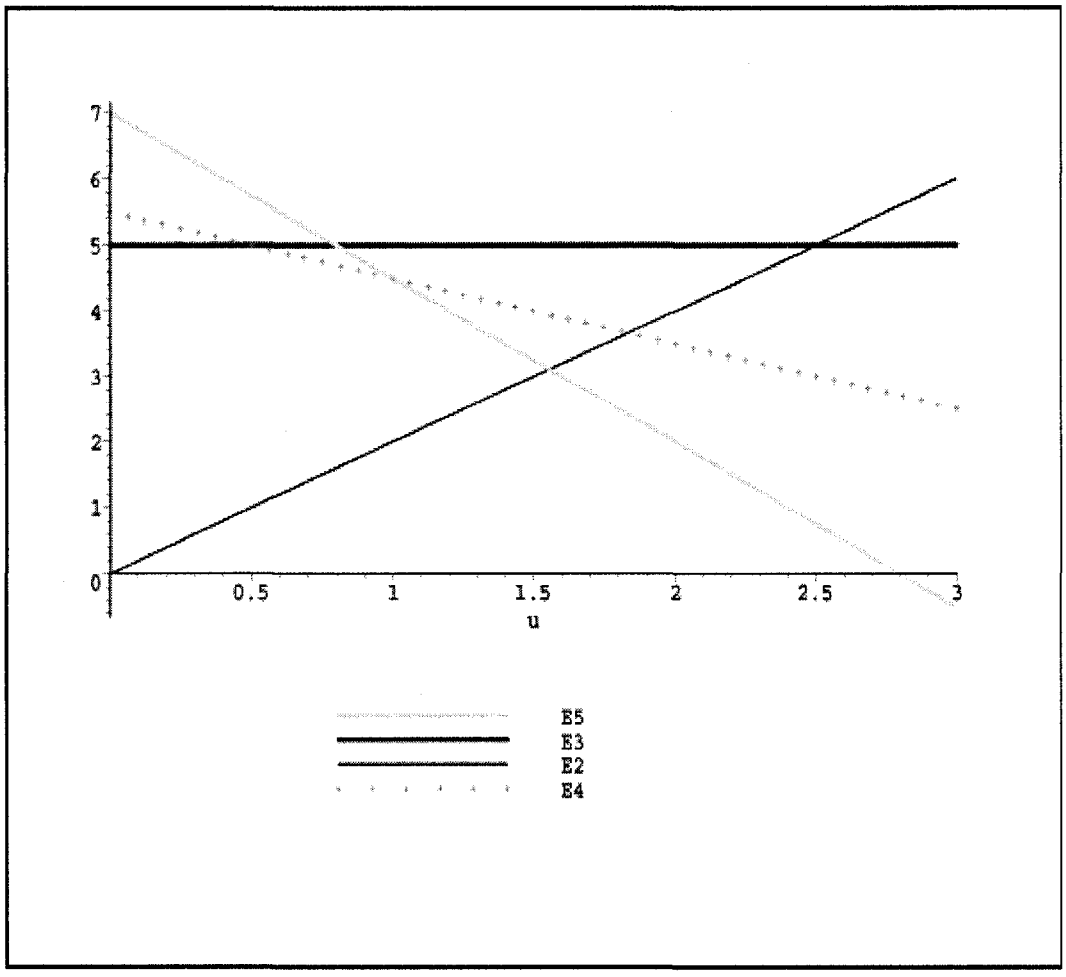

Figure 4.4 – Coûts lagrangiens en fonction du premier terme de la suite de pas.

Proposition 4 *Il existe un problème de couverture de tâches et deux suites de pas semblables,* $\{a_k\}_{k\geq 1}$ *et* $\{b_k\}_{k\geq 1}$ *satisfaisant les conditions de Pollack, pour lesquelles :*

$$\mathcal{Z}_{Opt}^{Rec} = \mathcal{Z}_{NS}^{Rec}(b_k) < \mathcal{Z}_{ND}^{Rec} < \mathcal{Z}_{NS}^{Rec}(a_k)$$

Preuve :

Les inégalités : $\mathcal{Z}_{ND}^{Rec} < \mathcal{Z}_{NS}^{Rec}(a_k)$ et $\mathcal{Z}_{Opt}^{Rec} < \mathcal{Z}_{ND}^{Rec}$ ont déjà été prouvées. Pour l'inéga-

Tableau 4.15 – Itérations de l'algorithme N-S pour la suite de pas $b_k = \frac{0.8}{k}$

k	Solution	$Sg_k(n2)$	$Sg_k(n_3)$	$u_k(n_2)$	$u_k(n_3)$	$\mathcal{Z}_{LND}$	Étiq Opt
0	E_2	2	2.5	0	0	-	-
1	E_2	2	2.5	1.6	1.4	7	E_5
2	E_5	-2.5	-2	0.6	0.5	5.5	E_4
3	E_2	2	1.5	1.1	1.02	7	E_5
4	E_2	2	1.5	1.5	1.4	7	E_5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

lité $\mathcal{Z}_{NS}^{Rec}(b_k) < \mathcal{Z}_{NS}^{Rec}(a_k)$, si on considère le même exemple que ci-dessous, d'après la figure 4.4, on voit que, pour $b_k = \frac{0.8}{k}$, l'algorithme N-S donne comme solution l'étiquette E_4 optimale. En effet, si on considère les 4 premières itérations représentées dans le tableau 4.15, on remarque qu'à la deuxième itération, l'algorithme génère le chemin d'étiquette E_4 qui est la solution optimale de coût 5.5. Or, d'après l'exemple précédent, pour $a_k = \frac{0.7}{k}$, l'algorithme génère la solution correspondante à l'étiquette E_5 de coût 7. Ce qui prouve la proposition.

Remarque 4 *1. Dans les deux exemples précédents, bien que les deux suites $a_k = \frac{0.7}{k}$ et $b_k = \frac{0.8}{k}$ soient assez semblables, les valeurs des solutions sont différentes (différence > 1). De plus, les différences entre les valeurs de $\mathcal{Z}_{NS}^{Rec}$, $\mathcal{Z}_{ND}^{Rec}$ et $\mathcal{Z}_{Opt}^{Rec}$ peuvent être aussi grandes que l'on veut. En effet, il suffit de multiplier les coûts sur les arcs dans l'exemple précédent par M pour obtenir des différences M fois plus grandes.*

2. Si à chaque itération de génération de colonnes, on faisait plusieurs itérations de APD-L afin d'obtenir le multiplicateur optimal $u^*(n_2) = 1.5$, on génèrerait toujours le chemin d'étiquette E_5. Ceci montre que l'algorithme N-S peut générer de meilleures solutions avec une suite de pas qui produit des multiplicateurs variés plutôt qu'en utilisant des multiplicateurs lagrangiens optimaux.

3. Pour cet exemple, il existe une valeur des multiplicateurs u telle que APD-LND génère la colonne optimale. Toutefois, ce n'est pas le cas pour tous les problèmes.

4.3.8 Conclusions sur les résultats de l'algorithme N-S

Bien que quelques expérimentations avec des paramètres bien choisis peuvent donner des espoirs que la méthode aboutisse à de bons résultats, comme ce qui a été montré par Nagih et Soumis (2000), les multiples tests pour différentes stratégies qu'on a effectué montrent une grande variabilité des résultats et aucune tendance de convergence ou de stabilité selon un paramètre ou l'autre, que ce soit pour l'algorithme en tant que tel ou pour l'implémentation à l'intérieur d'un algorithme de génération de colonnes.

L'étude théorique présentée dans les sections précédentes explique pourquoi les résultats peuvent être mauvais et instables. En effet, on a montré que les multiplicateurs de Lagrange ne convergent pas vers les valeurs permettant d'obtenir les meilleures solutions.

Quant à la méthode de projection par arcs, bien que prometteuse vu les résultats théoriques dans Nagih et Soumis (1999), elle s'est avérée aussi instable que la méthode de projection par noeuds. En plus, elle nécessite beaucoup plus de temps de calcul que l'approche par noeuds, étant donné le nombre considérable d'arcs dans le réseau. L'instabilité de l'approche par arcs est certainement due aux mêmes raisons que celles de l'approche par noeuds (instabilité de l'algorithme d'ajustement des multiplicateurs).

CHAPITRE 5

ALGORITHME D'AMÉLIORATION DYNAMIQUE DE LA BORNE SUPÉRIEURE

5.1 Introduction

On propose dans ce chapitre un algorithme exact du type primal pour résoudre itérativement le problème de PCC-FT, sur des réseaux acycliques et dans le cas où les fonctions d'extension sont linéaires. Cet algorithme commence par une première résolution approximative et essaye de trouver de meilleures solutions au fil des itérations. Plus précisément, la méthode commence par la résolution d'un PCC sur un petit réseau $\mathcal{G}^0$, on raffine $\mathcal{G}^0$ itérativement en des réseaux $\mathcal{G}^1$,$\mathcal{G}^2$,...,$\mathcal{G}^{\max}$ (max étant la dernière itération), de sorte qu'on améliore itérativement la solution courante et que $\mathcal{G}^{\max}$ fournisse la solution optimale. Sans perte de généralité, on supposera que la résolution du PCC sur le réseau $\mathcal{G}^0$ permet de produire une solution réalisable. Cette condition peut être facilement satisfaite dans les applications industrielles.

Cet algorithme peut s'avérer très efficace dans le contexte de génération de colonnes étant donné qu'on garde toujours la réalisabilité (étant donné une solution réalisable, chaque nouvelle solution générée au fil des itérations est elle aussi réalisable), contrairement à d'autres algorithmes du même type tels que les algorithmes de relaxation lagrangienne ou de k-plus courts chemins. Ainsi, on peut avoir le choix entre arrêter l'algorithme à l'optimalité (processus qui risque d'être fastidieux pour les problèmes de grande taille) ou on peut considérer d'autres critères d'arrêt de notre choix selon l'application ou le besoin (par exemple, les ressources informatiques). Par la suite, on proposera également d'autres techniques d'accélération.

On commencera tout d'abord par construire des nouveaux réseaux qui constituent

un raffinement du problème original afin de se débarrasser de certaines contraintes qui sont dans la formulation originale du problème. Par la suite, on se propose de définir des opérations sur ces réseaux afin de les enrichir et d'exhiber de nouvelles solutions qui s'améliorent au fil des itérations. On définira aussi des algorithmes de marquage afin de parcourir ces réseaux, de garder les solutions qui ont été découvertes et d'améliorer les solutions courantes.

De plus, on prouve que la complexité peut être assez basse dans certains cas. On montrera quelques exemples simples et, pour finir, on discutera le tout avec les résultats numériques qu'on a déjà obtenus sur différents types de problèmes.

5.2 Problème de plus court chemin avec fenêtres de temps

Soit $G(N, A)$ un réseau acyclique comme défini dans les chapitres précédents. Le problème de plus court chemin avec fenêtres de temps (PCC-FT) est un cas particulier du PCC-CR avec une seule ressource qui est le temps. Le problème de PCC-FT se formule comme suit :

$$\min \quad \sum_{i,j} c_{ij} x_{ij} \tag{5.1}$$

$$s.c. \quad \sum_{i} x_{ij} - \sum_{i} x_{ji} = e_j \quad \forall j \in N \tag{5.2}$$

$$x_{ij} \geq 0, \quad \forall (i,j) \in A \tag{5.3}$$

$$x_{ij} (T_i + t_{ij} - T_j) \leq 0, \quad \forall (i,j) \in A \tag{5.4}$$

$$T_j \in [a_j, b_j], \quad \forall j \in V. \tag{5.5}$$

Le problème PCC-FT étant un cas simplifié du PCC-CR, sa manipulation et

son écriture est plus simple. De plus, les algorithmes et les résultats que nous proposons pour PCC-FT se généralisent au problème PCC-CR. Ainsi, on commencera par étudier le PCC-FT. Par la suite, dans le chapitre 7, on discutera la généralisation des méthodes proposées pour les PCC-FT aux problèmes acycliques, à plusieurs ressources et avec des fonctions d'extension non linéaires.

5.3 Réseau réduit des états

Pour tout ce chapitre, on supposera que les noeuds de N sont ordonnés topologiquement.

Soit $\mathcal{P}$ un chemin de o à i de coût $C_i^{\mathcal{P}}$ et de consommation de temps $T_i^{\mathcal{P}}$. On note par $e_i^{\mathcal{P}}$ le couple $(C_i^{\mathcal{P}}, T_i^{\mathcal{P}})$ correspondant à l'étiquette du chemin $\mathcal{P}$.

On définit l'ensemble E_i des étiquettes en i, pour tout $i \in N$, de façon itérative selon l'algorithme 5.1

Algorithme 5.1 Construction de E_i

L'ensemble des noeuds de G étant ordonné topologiquement : $N = \{0, 1, \ldots n\}$.
$E_0 = \{e_0\}$, $e_0 = (0, 0)$.
pour tout noeud $j = 1, \ldots, n$ **faire**
  $E_1, \ldots, E_{j-1}$ ayant déjà été construits, on définit E_j comme suit :
  pour tout noeud i tel que $(i, j) \in A$ et $i \in \{0, 1 \ldots, j-1\}$ **faire**
    pour tout $e_i^{\mathcal{P}} \in E_i$ de valeur $(C_i^{\mathcal{P}}, T_i^{\mathcal{P}})$
    si $T_i^{\mathcal{P}} \leq b_i$ **alors**
      Soit le chemin $\mathcal{Q} = \mathcal{P} \cup \{(i, j)\}$, **faire**
        on rajoute une nouvelle étiquette $e_j^{\mathcal{Q}}$ à E_j, de valeur $(C_j^{\mathcal{Q}}, T_j^{\mathcal{Q}})$, $C_j^{\mathcal{Q}} = C_i^{\mathcal{P}} + c_{ij}$ et $T_j^{\mathcal{Q}} = \max\{a_j, T_i^{\mathcal{P}} + t_{ij}\}$.
  On élimine les étiquettes dominées de E_j.

Remarque 5 *Un ensemble E_i peut contenir des éléments réalisables et non réalisables. Ainsi, $\cup_{i \in N} E_i$ est différent de l'espace des états ξ défini dans le chapitre 4.*

Proposition 5 *Pour tout $i \in N$, si on trie les étiquettes de E_i par ordre croissant de leur coût alors elles sont triées par ordre strictement décroissant de leur consommation de temps.*

Preuve :
Notons d'abord qu'il n'existe pas d'étiquettes de même coût parmi les étiquettes non dominées (par la règle de dominance établie dans le chapitre 3). Supposons que la proposition n'est pas vraie, alors il existerait deux étiquettes $e_i^{\mathcal{P}} = (C_i^{\mathcal{P}}, T_i^{\mathcal{P}})$ et $e_i^{\mathcal{Q}} = (C_i^{\mathcal{Q}}, T_i^{\mathcal{Q}})$ telles que $C_i^{\mathcal{P}} < C_i^{\mathcal{Q}}$ et $T_i^{\mathcal{P}} \leq T_i^{\mathcal{Q}}$. Ainsi, $e_i^{\mathcal{P}}$ dominerait $e_i^{\mathcal{Q}}$, ce qui contredit la définition de l'ensemble E_i duquel on a déjà enlevé les étiquettes dominées.

On supposera dans tout ce qui suit que les éléments de E_i sont triés de cette façon.

5.4 Réseau de l'espace des états

Définition 3 *On définit le réseau de l'espace des états, noté $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, donné par $\mathcal{N} = \cup_{i \in N} E_i$ l'ensemble des noeuds de $\mathcal{G}$ et $\mathcal{A} = \{(e_i^{\mathcal{P}}, e_j^{\mathcal{Q}}) | (i,j) \in A, e_i^{\mathcal{P}} \in E_i, \mathcal{Q} = \mathcal{P} \cup \{(i,j)\}$ et $e_j^{\mathcal{Q}} \in E_j\}$ est l'ensemble de ses arcs. Ainsi, les noeuds de $\mathcal{N}$ sont associés aux étiquettes de l'espace des états. Le coût sur l'arc $(e_i^{\mathcal{P}}, e_j^{\mathcal{Q}})$ est exactement celui de l'arc (i,j) dans le graphe G original.*

Proposition 6 *La résolution du PCC (ordinaire) sur $\mathcal{G}$ est équivalente à la résolution du PCC-FT sur G par programmation dynamique. Autrement dit, les deux résolutions produisent des solutions optimales du PCC-FT.*

Preuve : À chaque chemin issu de l'origine vers un noeud i correspond une étiquette non dominée $(C_i^{\mathcal{P}}, T_i^{\mathcal{P}})$. De plus, à chaque étiquette $(C_i^{\mathcal{P}}, T_i^{\mathcal{P}})$ non dominée correspond

un unique noeud $e_i^{\mathcal{P}}$ de $\mathcal{G}$ dont le coût, lors de la résolution du PCC sur $\mathcal{G}$, est $C_i^{\mathcal{P}}$. En particulier, lorsque $i = d$ au plus court chemin dans G correspond une étiquette de l'espace des états et par la suite un noeud de $\mathcal{G}$. Donc la valeur du plus court chemin donné par la résolution du PCC-CR sur G est la même que celle donnée par la résolution du PCC sur $\mathcal{G}$.

5.5 Réseau restreint des étiquettes

On appelle réseau restreint des étiquettes, noté $\mathcal{G}^0 = (\mathcal{N}^0, \mathcal{A}^0)$, un réseau qui est du même type que le réseau $\mathcal{G}$ défini précédemment. Les noeuds de ce réseau sont des étiquettes. Ce réseau est construit par un procédé de *marquage* des noeuds qui est une généralisation de l'algorithme de Bellman. Dans cet algorithme, on commence par marquer le noeud origine, puis on marque progressivement les autres noeuds. Ainsi, pour tout $i \in N$ désignons par M_i l'ensemble des noeuds marqués et prolongés ; M_i est initialisé à $\emptyset$. En supposant que les noeuds du nouveau réseau ont déjà été construits en $i \in N$, on marque le noeud qui correspond à l'étiquette de moindre coût et on prolonge uniquement ce noeud marqué comme le montre la figure 5.1. Dans cette figure, les noeuds de $\mathcal{G}$ sont triés en ordre décroissant des coûts. Sur cette figure, le noeud marqué, qui correspond à l'étiquette $(1, 7)$, est celui qui a le plus petit coût. Ce noeud (représenté sur la figure en pointillé) est marqué et prolongé. Ceci se fait selon l'algorithme 5.2. Cet algorithme commence avec un ensemble $E_0^0 = \{e_0\}$ tel que e_0 correspond à l'étiquette $(0, 0)$ au noeud origine o.

Proposition 7 *Résoudre le PCC sur $\mathcal{G}^0$ est équivalent à la résolution du PCC-FT sur G, par programmation dynamique, en dominant uniquement sur les coûts (PCC-ND).*

Preuve :

Pour chaque $i \in N$ dans l'ensemble E_i^0, seul le noeud de moindre coût $e_i^{\mathcal{P}} \in M_i$

Algorithme 5.2 Algorithme de marquage

Initialisation : $E_0^0 = \{e_0\}$ et $\forall i \in N \setminus \{o\}$, $E_i^0 = \emptyset$
pour $i = 0, 1, \ldots, n$ **faire**
Soit E_i^0 l'ensemble de noeuds en i. On trie les noeuds de E_i^0 en ordre décroissant de la valeur du coût de leurs étiquettes.
Soit $e_i^{\mathcal{P}_1}, \ldots, e_i^{\mathcal{P}_k} = e_i^{\mathcal{P}}$ les k noeuds dont les étiquettes correspondantes sont non dominées et telles que $T_i^{\mathcal{P}} \leq b_i$. Soit $e_i^{\mathcal{P}_{k+1}}, \ldots, e_i^{\mathcal{P}_h}$ les noeuds de E_i^0 tels que $T_i^{\mathcal{P}_{k+1}} > b_i$.
Mettre les arcs $(e_i^{\mathcal{P}_1}, e_i^{\mathcal{P}_2}), \ldots, (e_i^{\mathcal{P}_{k-1}}, e_i^{\mathcal{P}_k})$ dans $\mathcal{A}^0$.
Marquer $e_i^{\mathcal{P}}$.
Prolonger $e_i^{\mathcal{P}}$:
pour tout $j \in N$ tel que $(i, j) \in A$ **faire**
Soit le chemin $\mathcal{Q} = \mathcal{P} \cup \{(i, j)\}$, créer $e_j^{\mathcal{Q}} = (C_j^{\mathcal{Q}}, T_j^{\mathcal{Q}})$ telle que $C_j^{\mathcal{Q}} = C_i^{\mathcal{P}} + c_{ij}$ et $T_j^{\mathcal{Q}} = \max\{a_j, T_i^{\mathcal{P}} + t_{ij}\}$
Ajouter $e_j^{\mathcal{Q}}$ à E_j^0
Enlever les noeuds de E_j^0 dont les étiquettes associées sont dominées.
Ajouter l'arc $(e_i^{\mathcal{P}}, e_j^{\mathcal{Q}})$ à $\mathcal{A}^0$.
Ajouter $e_i^{\mathcal{P}}$ à M_i.

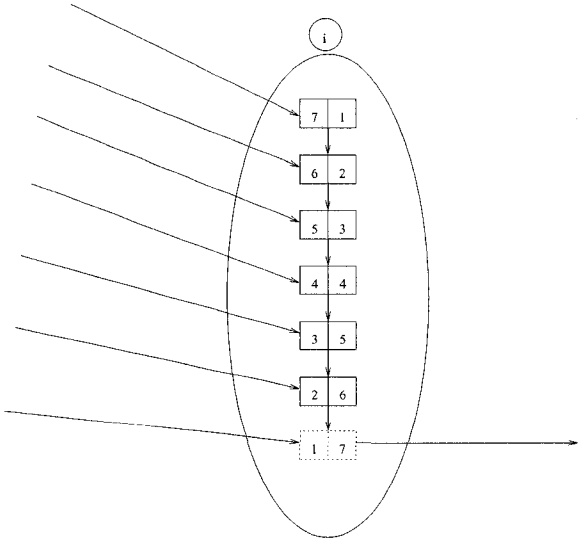

Figure 5.1 – Noeud quelconque du réseau restreint des étiquettes

est prolongé. Or $e_i^{\mathcal{P}}$ correspond à l'étiquette réalisable de moindre coût au noeud i lorsqu'on domine uniquement sur le coût. On prolonge donc la même étiquette du noeud i et on obtient les mêmes étiquettes aux noeuds successeurs.

Définition 4 *Un noeud $e_i^{\mathcal{P}} = (C_i^{\mathcal{P}}, T_i^{\mathcal{P}}) \in \mathcal{G}^0$ est dit non réalisable lorsque $T_i^{\mathcal{P}} > b_i$.*

Proposition 8 *Si le réseau $\mathcal{G}^0$ ne contient pas de noeuds non réalisables, alors la résolution du PCC sur $\mathcal{G}^0$ fournit la solution optimale de PCC-FT sur G.*

Preuve :
Si $\mathcal{G}^0$ ne contient pas de noeuds non réalisables, alors le PCC-FT sur G sera équivalent à un PCC sans fenêtres de temps. Il suffit donc d'enlever les fenêtres de temps, le problème reste le même. On peut ainsi le résoudre par un algorithme de plus court chemin ordinaire sur $\mathcal{G}^0$ (Dijkstra par exemple).

Remarque 6 *Un noeud marqué $e_i^{\mathcal{P}} \in E_i^0$ possède un seul successeur dans E_j^0, pour tout $(i,j) \in A$, alors qu'un noeud non réalisable ne possède pas de successeurs dans $\mathcal{G}^0$.*

5.6 Opération de *Mise à Jour*

On se propose dans cette partie de définir une opération qui permet de raffiner le réseau $\mathcal{G}^0$ en considérant des partitions des fenêtres de temps $[a_i, b_i]$, ce qui induit une partition des ensembles E_i. Le but de ce raffinement est d'obtenir un réseau qui satisfait des conditions semblables à celles de la proposition 8 et sur lequel on peut appliquer un algorithme de plus court chemin «standard» du type Dijkstra afin d'obtenir la solution optimale. Ce but ne sera pas atteint en une seule étape. Par

contre, le raffinement du réseau permettra d'obtenir de meilleures solutions au fil des itérations.

Soit $\lambda_1 = [a_i, f_i^1], \lambda_2 =]f_i^1, f_i^2], \ldots, \lambda_k =]f_i^{k-1}, b_i]$ une partition disjointe de sous-intervalles de $[a_i, b_i]$. On appelle partition disjointe d'ensembles de noeuds en i, les ensembles $E_i^{\lambda_1}, \ldots, E_i^{\lambda_k}$ tels que $E_i^{\lambda} = \{e_i^{\mathcal{P}} = (C_i^{\mathcal{P}}, T_i^{\mathcal{P}}) | T_i^{\mathcal{P}} \in \lambda\}$, $\lambda \in \{\lambda_1, \ldots, \lambda_k\}$.

Sans perte de généralités, on notera par $|f_i^1, f_i^2|$ un intervalle ouvert, fermé ou semi-ouvert.

Nous représenterons un sous-intervalle de la partition par $|e_i, f_i|$ au lieu de $|f_i^k, f_i^{k+1}|$ lorsqu'il n'est pas utile de connaître sa position k dans la partition.

Définition 5 *Soit $\lambda_1 = |a_i, f_i^1|, \lambda_2 = |f_i^1, f_i^2|, \ldots, \lambda_k = |f_i^{k-1}, b_i|$ une partition de $|a_i, b_i|$. Soit $E_i^{\lambda_1}, \ldots, E_i^{\lambda_k}$ la partition disjointe correspondante de E_i. Un noeud $e_i^{\mathcal{P}} = (C_i^{\mathcal{P}}, T_i^{\mathcal{P}})$ est dit relativement non réalisable par rapport à E_i^{λ}, $\lambda = |f_i^{u-1}, f_i^u|$, lorsque $T_i^{\mathcal{P}} > f_i^u$.*

Définition 6 *On dit que $e_i^{\mathcal{P}}$ est relativement non réalisable s'il existe au moins un ensemble E_i^{λ} par rapport auquel il est relativement non réalisable.*

Remarque 7 *Si $e_i^{\mathcal{P}}$ est relativement non réalisable par rapport à $E_i^{\lambda_u}$, avec $\lambda_u = |f_i^{u-1}, f_i^u|$, alors il l'est par rapport à tout $E_i^{\lambda_v}$, avec $\lambda_v = |f_i^{v-1}, f_i^v|$ tel que $f_i^v < f_i^u$.*

Remarque 8 *Un noeud non réalisable est relativement non réalisable. En effet, si $e_i^{\mathcal{P}} = (C_i^{\mathcal{P}}, T_i^{\mathcal{P}})$ est non réalisable, alors $T_i^{\mathcal{P}} > b_i$. Donc il est relativement non réalisable par rapport à E_i.*

Proposition 9 *La résolution du problème de plus court chemin ordinaire sur un réseau $\mathcal{G}$ qui ne contient pas de noeuds relativement non réalisables fournit la solution optimale du problème original de plus court chemin avec fenêtres de temps.*

Preuve : La preuve est la même que celle de la proposition 8.

5.6.1 Opération de *Mise à Jour* : MAJ$(i, E_j^{\mu}, e_j^{\mathcal{B}})$

Soit (i, j) un arc de A, E_j^{μ} un ensemble de noeuds tel que $\mu = |e_j, f_j|$. On introduit l'ensemble B_i qui est un ensemble d'étiquettes non réalisables. Cet ensemble se modifie dans les algorithmes qu'on proposera dans la suite, on peut en enlever des éléments existants ou en rajouter de nouveaux.

Soit $e_j^{\mathcal{B}} \in B_j$ un noeud relativement non réalisable par rapport à E_j^{μ} et soit $e_i^{\mathcal{B}}$ son prédécesseur. Soit E_i^{λ} un ensemble de noeuds faisant partie d'une partition disjointe d'ensembles de noeuds en i telle que $\lambda = |e_i, f_i|$ et $(f_j - t_{ij}) \in \lambda$.

Désignons par une partition au temps $f_j - t_{ij}$, l'union des deux sous-ensembles suivants : $E_i^{\lambda_1} = \{e_i^{\mathcal{S}} \in E_i^{\lambda} | T_i^{\mathcal{S}} \in |e_i, f_j - t_{ij}|\}$ et $E_i^{\lambda_2} = \{e_i^{\mathcal{S}} \in E_i^{\lambda} | T_i^{\mathcal{S}} \in |f_j - t_{ij}, f_i|\}$. Supposons que $E_i^{\lambda_1}$ est non vide. Étant donné un noeud relativement non réalisable, l'opération de *mise à jour* consiste à remonter vers son noeud prédécesseur, diviser l'ensemble des noeuds correspondant selon la partition $E_i^{\lambda_1} \cup E_i^{\lambda_2}$. Par la suite, on enlève le noeud relativement non réalisable du réseau et son arc incident afin d'alléger la structure du réseau. L'opération se décrit selon l'algorithme 5.3. (Voir aussi la figure 5.2).

Notons que, dans cet algorithme on ne prolonge pas les nouvelles étiquettes créées. Ainsi, désignons par P_i l'ensemble des noeuds marqués et non prolongés au noeud $i \in N$.

Algorithme 5.3 Algorithme de mise à jour : MAJ$(i, E_j^{\mu}, e_j^{\mathcal{B}})$

Supprimer $e_j^{\mathcal{B}}$ s'il est non réalisable sinon l'enlever de B_j
Soit $e_i^{\mathcal{B}}$ le prédécesseur de $e_j^{\mathcal{B}}$
Soit E_i^{λ} l'ensemble de noeuds contenant $e_i^{\mathcal{B}}$, tel que $\lambda = |e_i, f_i|$
Soit $E_i^{\lambda_1}$ et $E_i^{\lambda_2}$ une partition au temps $f_j - t_{ij}$
Soit $e_i^{\mathcal{P}_1}$ le noeud de coût minimum dans $E_i^{\lambda_1}$
Marquer $e_i^{\mathcal{P}_1}$
Ajouter $e_i^{\mathcal{P}_1}$ à P_i.
si $e_i^{\mathcal{P}_2}$ est le successeur de $e_i^{\mathcal{P}_1}$ dans E_i^{λ} **alors**
  Enlever l'arc $(e_i^{\mathcal{P}_1}, e_i^{\mathcal{P}_2})$ du réseau.

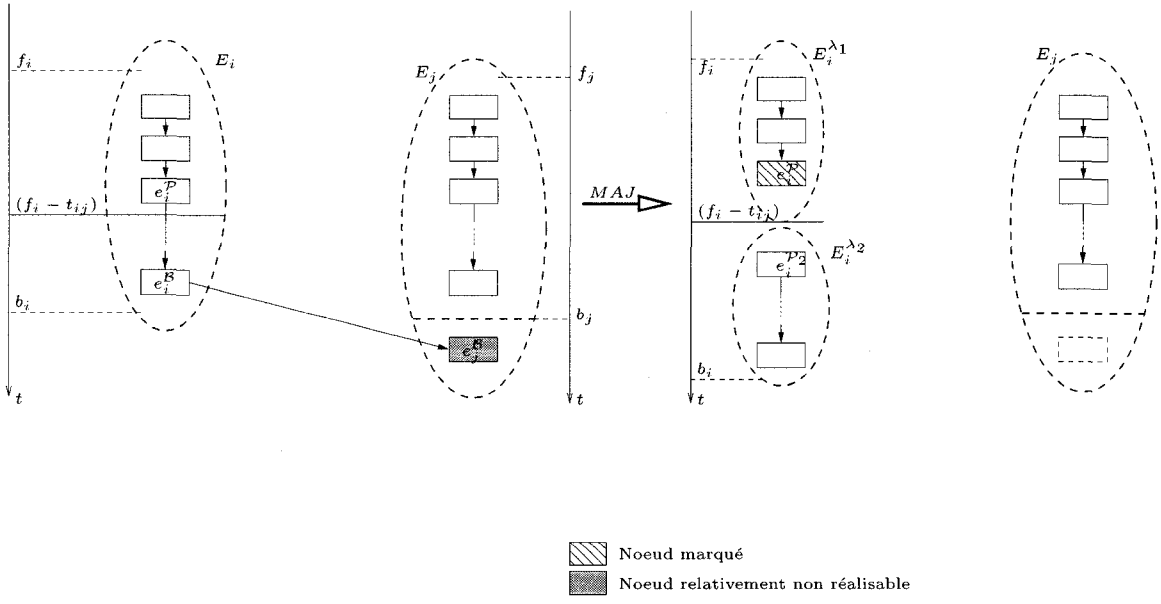

Figure 5.2 – Opération de mise à jour

Remarque 9 *Après avoir appliqué $MAJ(i, E_j^{\mu}, e_j^b)$, on a une nouvelle partition disjointe comprenant $E_i^{\lambda_1}$ et $E_i^{\lambda_2}$.*

5.7 Algorithme d'amélioration dynamique de la borne supérieure : ADBS

Cette méthode agit itérativement en deux étapes. Dans une première étape, on parcourt les noeuds en ordre topologique inverse et on applique les opérations de mise à jour sur les noeuds relativement non réalisables et en marquant temporairement des noeuds qui vont être prolongés par la suite. Dans une deuxième étape, on marque définitivement les noeuds temporairement marqués en prolongeant vers l'avant. On répète ces deux étapes jusqu'à ce qu'on n'ait plus de noeud relativement non réalisable.

En d'autres mots, on maintient trois listes de noeuds mises à jour : B_i, P_i et M_i pour tout $i \in N$. Ainsi on applique la procédure de mise à jour aux noeuds de la liste B_i qui sont relativement non réalisables. On ajoute aux ensembles P_i de nouveaux noeuds qui sont temporairement marqués obtenus par cette opération. On finit par les marquer définitivement en les ajoutant à M_i et en les enlevant de P_i, et ceci jusqu'à ce que B_i devienne vide, pour tout $i \in N$.

L'algorithme qu'on propose commence par le réseau $\mathcal{G}^0$ et opère selon 5.4

Notons $\mathcal{G}^k$ le nouveau réseau crée à la $k$ième itération de l'algorithme. Les ensembles E_i^{λ}, P_i et M_i qu'on va utiliser par la suite sont ceux présent à la $k$ième itération. À l'itération k, il faut faire un marquage vers l'avant afin de reconstruire le nouveau réseau et prolonger les nouveaux noeuds qu'on a crée. L'algorithme de marquage ressemble à l'algorithme (5.2) qu'on a utilisé pour construire le réseau $\mathcal{G}^0$. Nous conserverons les notations précédentes.

Algorithme 5.4 Algorithme d'amélioration dynamique de la borne supérieure

Soit $k = 1$ (numéro d'itération)
répéter
  pour tout $j = n, n-1, \ldots, 0$ **faire**
    pour tout $e_j^{\mathcal{B}} \in B_j$ **faire**
      pour tout E_j^{λ} tel que $e_j^{\mathcal{B}}$ est relativement non réalisable par rapport à E_j^{λ} **faire**
        Soit $i \in N$ tel que $(i,j) \in A$ et qui précède j sur le chemin $\mathcal{B}$.
        Appliquer l'opération MAJ$(i, E_j^{\lambda}, e_j^{\mathcal{B}})$
    Ajouter les nouveaux noeuds relativement non réalisables (noeuds de E_i^{λ}) à B_i
  Appliquer l'algorithme de marquage (5.5) sur le nouveau réseau $\mathcal{G}^k$ défini ci-dessous.
  $k \rightarrow k+1$
jusqu'à $B_j = \emptyset$

Algorithme 5.5 Algorithme de marquage

pour tout $i = 0, 1, \ldots, n$ **faire**
  pour tout sous-ensemble d'étiquettes E_i^{λ} en i **faire**
    Enlever les noeuds, dominés sur le coût, de E_i^{λ} et de P_i.
  Trier les noeuds de E_i^{λ} selon le coût de leurs étiquettes.
  Mettre de nouveaux arcs $(e_i^{\mathcal{P}_1}, e_i^{\mathcal{P}_2}), (e_i^{\mathcal{P}_2}, e_i^{\mathcal{P}_3}), \ldots$ entre les noeuds de E_i^{λ}.
  Soit $e_i^{\mathcal{P}}$ le noeud qui correspond à l'étiquette de moindre coût.
  Prolonger $e_i^{\mathcal{P}}$ vers ses successeurs :
  pour tout j tel que $(i,j) \in A$ **faire**
    Soit $\mathcal{Q} = \mathcal{P} \cup \{(i,j)\}$: créer $e_j^{\mathcal{Q}} = (C_j^{\mathcal{Q}}, T_j^{\mathcal{Q}})$ tel que $C_j^{\mathcal{Q}} = C_i^{\mathcal{P}} + c_{ij}$ et $T_j^{\mathcal{Q}} = \max\{a_j, T_i^{\mathcal{P}} + t_{ij}\}$ si $T_j^{\mathcal{Q}} \leq b_j$
  Enlever $e_i^{\mathcal{P}}$ de P_i et l'ajouter à M_i.

Illustrons les étapes de cette méthode par un petit exemple qu'on peut transcrire de façon simple.

Exemple d'application de la méthode ADBS sur un réseau de petite taille

L'exemple suivant illustre l'application de l'algorithme sur un petit réseau (10 noeuds, 18 arcs) illustré dans la figure 5.3. On considère le problème de plus court chemin avec fenêtres de temps sur ce réseau. Le noeud source est le noeud 1 et le

noeud destination est le noeud d. Le coût et la consommation de ressource sur un arc sont représentés par le couple (c_{ij}, t_{ij}). Le réseau est acyclique avec des coûts positifs pour simplifier la présentation. De plus, on a mis des fenêtres de temps sur chaque noeud : $[0, 0]$ pour le noeud 0, $[0, 5]$ pour le noeud 1, $[0, 5]$ pour le noeud 2, $[0, 8]$ pour le noeud 3, $[0, 10]$ pour le noeud 4, $[0, 10]$ pour le noeud 5, $[0, 12]$ pour le noeud 6, $[0, 15]$ pour le noeud 7, $[0, 20]$ pour le noeud 8 et $[0, 30]$ pour le noeud d.

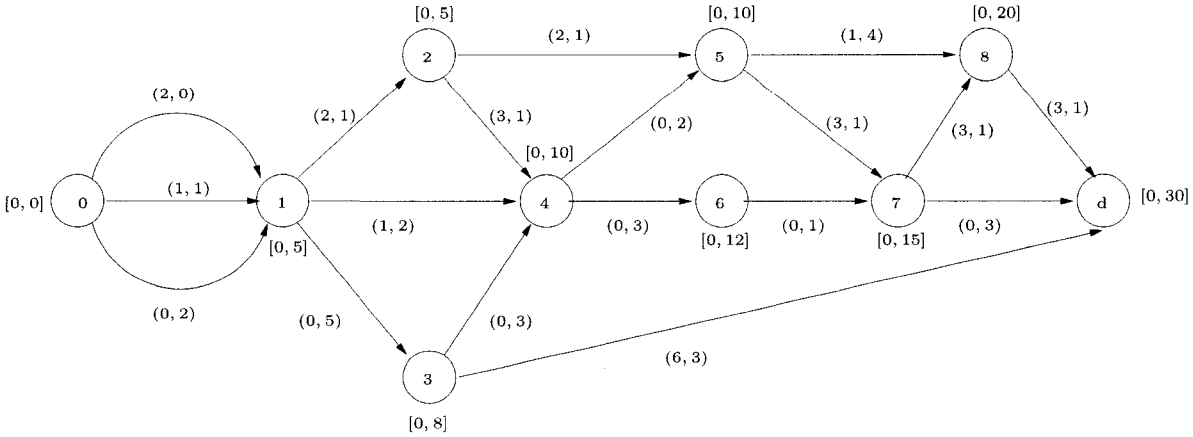

Figure 5.3 – Exemple d'application de ADBS

Afin de comparer l'approche de résolution par programmation dynamique et l'algorithme ADBS avec les autres méthodes, on a choisi comme critère le nombre d'étiquettes générées par les deux algorithmes.

La figure 5.4 illustre les étiquettes correspondants à tous les chemins réalisables, non réalisables et non dominés du réseau, c'est-à-dire l'ensemble de toutes les étiquettes qu'on peut générer à l'aide de l'algorithme de programmation dynamique. Remarquons que ce réseau comprend aussi les éléments de $\mathcal{G}$. On peut énumérer 52 étiquettes au total. On remarque que la solution optimale est donnée par l'étiquette $(2, 10)$ au noeud destination. Pour des fins de comparaison, on a illustré dans ce réseau les noeuds non réalisables (en noir) et les noeuds marqués pour la méthode ADBS (en hachuré).

La figure 5.5 montre l'itération 0 de l'algorithme, qui consiste en la génération du réseau $\mathcal{G}^0$ comme décrit dans la section précédente, ainsi que le marquage des

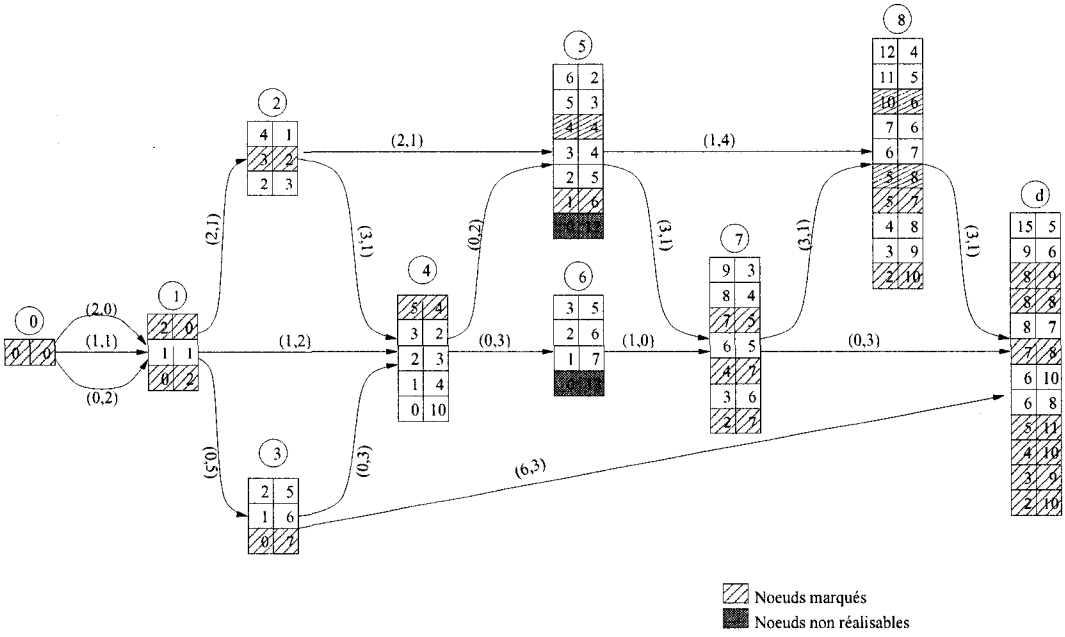

Figure 5.4 – Exemple : première étape

étiquettes et l'identification des noeuds relativement non réalisables. En effet, on a prolongé les étiquettes marquées (en remplissage hachuré sur la figure 5.5), on a identifié les noeuds relativement non réalisables (remplissage en noir) et on a enlevé les étiquettes dominées. Dans cette figure, on a au total 18 étiquettes, l'étiquette optimale au noeud destination d est $e_d^{opt} = (6, 10)$. Sur la figure 5.5, on remarque qu'on a 2 noeuds non réalisables, $e_5^p = (0, 12)$ et $e_6^q = (0, 13)$. On rajoute par la suite ces noeuds non réalisables aux ensembles des noeuds relativement non réalisables respectifs B_5 et B_6, initialisés à $\emptyset$ auparavant.

On applique maintenant ADBS une première fois à $\mathcal{G}^0$: on parcourt $\mathcal{G}^0$ en ordre topologique inverse, on applique l'opération de mise à jour pour les étiquettes non réalisables e_5^p et e_6^q qui sont dans B_5 et B_6. L'opération de mise à jour sur e_5^p produit les nouveaux sous-ensembles de noeuds E_4^1 et E_4^2. Par la suite, l'opération de marquage produit la nouvelle étiquette $(1, 6)$ au noeud 5 d'autre part. L'opération de mise à jour appliquée sur l'étiquette e_6^q donne lieu à une nouvelle étiquette $(0, 10)$ qui est relativement non réalisable au noeud 4 par rapport à E_4^1 (et par la suite à E_4^2

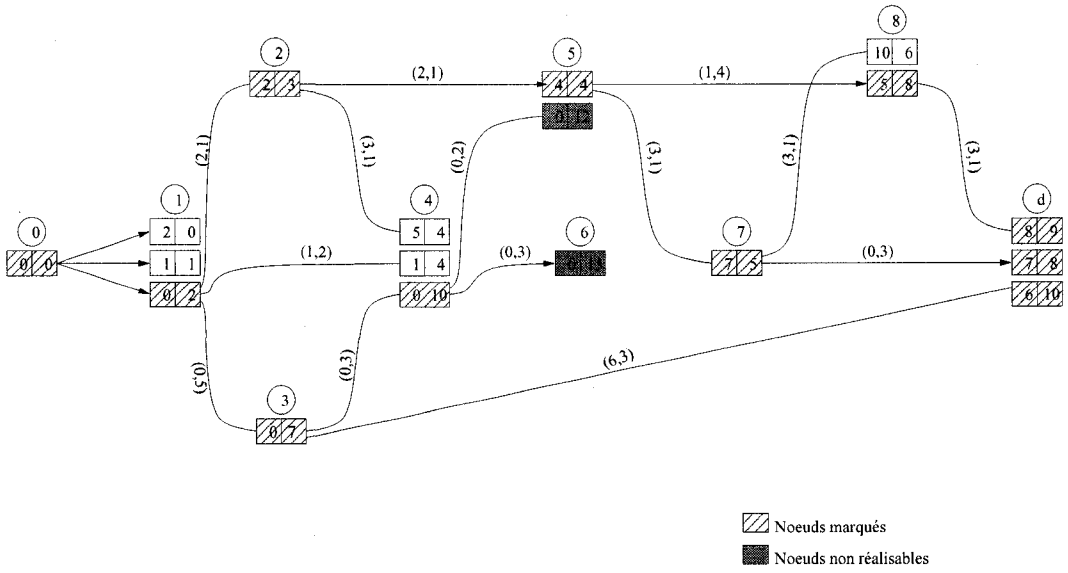

Figure 5.5 – Exemple : deuxième étape

aussi). Bien que l'algorithme n'a pas encore atteint l'optimalité, on peut prolonger vers l'avant les nouvelles étiquettes et procéder au marquage des étiquettes comme illustré sur la figure 5.6. On remarque qu'on obtient ainsi une meilleure étiquette que celle qu'on avait à l'itération 0 de l'algorithme qui est de valeur : $(4, 10)$. Or si on continue l'algorithme en appliquant l'opération de mise à jour sur la nouvelle étiquette relativement non réalisable $(0, 10)$, on génère une nouvelle étiquette $(3, 2)$ au noeud 4. Cette fois-ci, si on prolonge les étiquettes vers l'avant et on procède à l'opération de marquage, on obtient la solution optimale au problème qui est donnée par l'étiquette $(2, 10)$ au noeud destination. On remarque qu'on a uniquement 20 étiquettes générées par ADBS.

Remarquons que, bien que la cardinalité du graphe $\mathcal{G}^k$ des étiquettes, générés par ADBS, soit inférieure à celle de $\mathcal{G}$, on a $\mathcal{G}^k \not\subseteq \mathcal{G}$.

Remarque 10 *Dans l'algorithme proposé, on commence par un réseau de petite taille ($O(n)$, n étant le nombre de noeuds de G). On augmente la taille du réseau itérativement en ajoutant au réseau $\mathcal{G}^k$ des noeuds dont les étiquettes associées sont non dominées par rapport au graphe $\mathcal{G}^k$ mais qui peuvent l'être pour le réseau $\mathcal{G}$.*

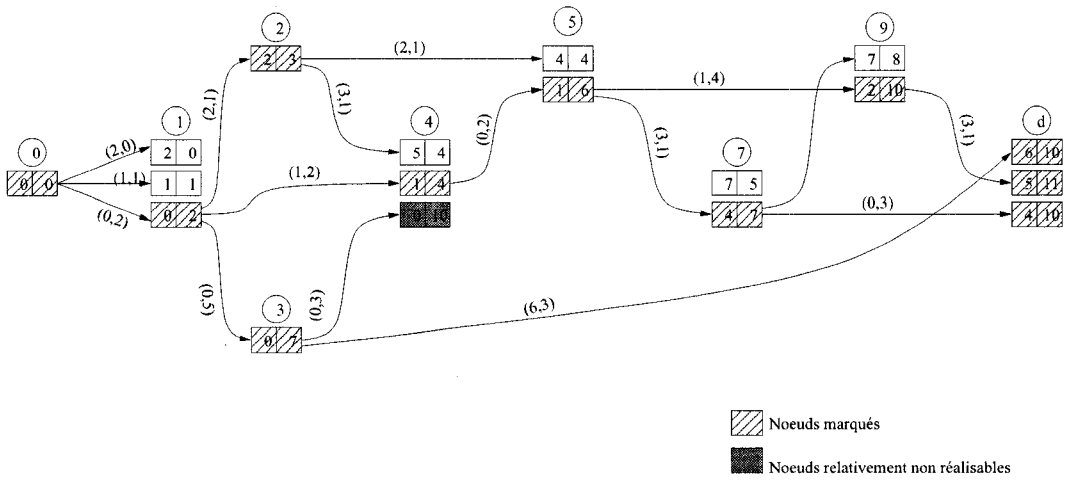

Figure 5.6 – Exemple : troisième étape

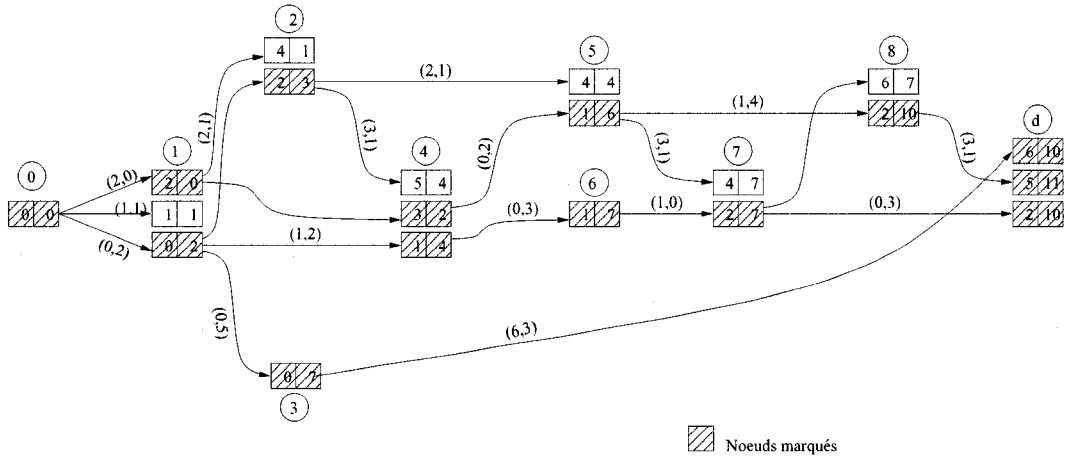

Figure 5.7 – Exemple : quatrième étape

5.8 Généralisation de l'algorithme pour les cas cycliques

Lorsque le réseau G est cyclique et les durées sur les arcs t_{ij} sont entières et non négatives, on peut appliquer de la même façon l'algorithme de marquage sur les réseaux $\mathcal{G}^k$ en utilisant un algorithme du type «reaching» de Desrochers (1986), au lieu de l'algorithme du type Bellman, sur un nouveau réseau acyclique équivalent à G.

Le nouveau réseau peut être construit en divisant les fenêtres de temps $[a_i, b_i]$, en chaque noeud i, en sous-intervalles. Cette subdivision s'effectue de façon que lorsqu'on prolonge les étiquettes vers un successeur, aucun prolongement de ce dernier ne risque de créer un cycle dans le même sous-intervalle. Ceci est dû au fait que l'étiquette prolongée ne peux plus être réalisable en i (en divisant la fenêtre de temps) et, par conséquent, ne peut pas créer de cycle.

Ainsi, pour j un noeud donné, désignons par $\delta_i = min_i\{t_{ij}; (i,j) \in A\}$. Alors, la subdivision de sous-intervalles de temps se fait comme suit : $\lambda_1 = [a_i, a_i + \delta_i[, \lambda_2 = [a_i + \delta_i, a_i + 2\delta_i[, \ldots, \lambda_p = [a_i + (p-1)\delta_i, a_i + p\delta_i[$, p étant un entier tel que $a_i + p\delta_i < b_i \leq a_i + (p+1)\delta_i$.

Par la suite, on divise le noeud i en p noeuds. Chaque nouveau noeud i_k ayant pour fenêtre de temps λ_k et on dédouble les arcs pour chaque nouveau noeud. Le nouveau réseau ainsi obtenu est traité comme un réseau acyclique.

5.9 Résultats numériques

Le but de ce travail étant l'étude des PCC-CR à l'intérieur d'un algorithme de génération de colonnes, on a implémenté la méthode ADBS à l'intérieur du logiciel *GENCOL*. L'efficacité de l'algorithme sera donc mesurée à l'intérieur de cet algorithme et par rapport aux paramètres de ce dernier.

On a fait des tests sur plusieurs réseaux. Les tests ont été effectués sur des problèmes d'horaires de véhicules avec fenêtres de temps. Trois types de réseaux ont été retenus : des petits réseaux de 100 noeuds (tâches), des réseaux de 700 noeuds et de grands réseaux de 1000 noeuds. Ces problèmes ont été créés aléatoirement en utilisant un générateur aléatoire de réseaux pour le problème d'horaires de véhicules avec fenêtres de temps et dépôts multiples.

Les tests ont été effectués sur une machine du type Linux, avec un processeur AMD-64 de 2.4GHz.

Étant donné la taille relativement grande des problèmes, on s'est donc proposé de résoudre les problèmes heuristiquement. Pour cela, on s'est proposé d'appliquer l'opération de mise à jour sur un petit nombre d'arcs (entre 2 et 20 pour chaque noeud) et un petit nombre d'itérations de l'algorithme ADBS (2 à 3). De plus, on a commencé à appliquer la méthode que dans des itérations avancées de génération de colonnes afin de ne pas alourdir inutilement le calcul lors des premières itérations.

Afin de faire cette comparaison, deux critères sont à surveiller. Le premier est le temps de calcul des sous-problèmes, le second est la valeur de l'objectif du problème maître. Afin de faire une comparaison quantitative des différentes approches, on s'est proposé de mesurer l'écart, en pourcentage, entre l'objectif des algorithmes APD, APD-ND et ADBS. Ceci est donné par l'expression : $\frac{C^* - C^{APD-ND}}{C^{ND} - C^{APD}}$, C^* étant la valeur de l'objectif en utilisant l'algorithme considéré. D'un autre côté, on a mesuré l'écart, en pourcentage, des temps de calculs totaux des sous-problèmes de GC ce qui est donné par l'expression : $\frac{T^{APD} - T^*}{T^{APD} - T^{ND}}$, T^* étant le temps total de calcul des sous-problèmes. Ainsi, par exemple, l'algorithme APD représente 100% d'écart avec la valeur optimale et 100% d'écart dans les temps de calculs, alors que APD-ND représente un écart de 0% par rapport à la valeur donnée par APD-ND, pour un écart de 0% en temps de calculs. Ce sont ces deux valeurs extrêmes qui vont nous servir de repère pour la qualité de notre algorithme.

Dans les tableaux présentés ci-dessous, on notera par *APD* la résolution par l'algorithme de programmation dynamique, *APD-ND* la résolution par programmation dynamique en dominant uniquement sur le coût, *ADBS* la résolution par l'algorithme d'amélioration dynamique de la borne supérieure, *GC* le numéro de l'itération de génération de colonnes, *Temps(SPP)* le temps cumulé des sous-problèmes, *Nombre Étiq* le nombre total d'étiquettes générées, *Nombre Col* le nombre de colonnes générées et *Valeur Obj(PM)* la valeur de l'objectif du problème maître.

5.9.1 Problème I

Pour commencer, on a opté de choisir un problème relativement petit pour lequel on pourra tester facilement les performances de l'algorithme qu'on propose. De plus, un problème de plus petite taille a l'avantage qu'on pourrait pousser la convergence de l'algorithme à sa limite. Il s'agit d'un problème qui est de taille relativement petite, il possède 100 noeuds, 100 tâches (une tâche sur chaque noeud), 11550 arcs et 2 réseaux (un pour chaque dépôt).

Plusieurs tests ont été effectués sur ce problème. Malheureusement, on n'a pas pu atteindre l'optimalité lors de l'application de ADBS. Le mieux qu'on a obtenu est un écart de 8% par rapport à la valeur optimale obtenue à l'intérieur de l'algorithme de génération de colonnes. Pour cela, on a appliqué 3 itérations de l'algorithme, on a appliqué l'opération MAJ à un maximum de 20 arcs incidents à chaque noeud et on a commencé dès le début de l'algorithme de génération de colonnes. On a remarqué que même si on augmentait la valeur de ces paramètres, on ne réussissait pas à améliorer la valeur de l'objectif.

Le tableau 5.1 résume les tests qu'on a effectué sur ce problème en utilisant APD, 5.2 ceux de APD-ND et 5.3 ceux ADBS. Afin de simplifier la présentation, on a affiché les résultats donnés par l'algorithme GC à chaque 10 itérations.

Selon les critères qu'on a fixé au début, l'écart de la solution obtenue par ADBS par rapport à la solution optimale est de 28% pour les temps de résolution et 92% pour la valeur de l'objectif.

Aussi, on a remarqué que le nombre d'étiquettes générées n'est pas beaucoup plus grand pour ADBS que pour APD-ND. De plus le nombre de colonnes générées reste du même ordre pour les 3 méthodes. On va voir que cette remarque reste aussi valable pour tous les prochains problèmes test.

Tableau 5.1 – Problème I : Résolution par APD

PB I	APD			
GC	Temps(SPP)	Valeur Obj(PM)	Nombre Étiq	Nombre Col
20	1.3	244616.6	11702	238
30	2.1	214942.5	12576	350
40	2.8	210970.0	16678	451
50	3.4	209539.9	14562	329
Fin	5.7	208204.1	10982	351

Tableau 5.2 – Problème I : Résolution par APD-ND

PB I	APD-ND			
GC	Temps(SPP)	Valeur Obj(PM)	Nombre Étiq	Nombre Col
20	0.6	252257.1	3731	253
30	0.8	227172.4	3854	369
40	1.0	210970.0	4026	467
50	1.2	219852.2	3287	324
Fin	1.4	219434.0	3108	356

Tableau 5.3 – Problème I : Résolution par ADBS

PB I	ADBS			
GC	Temps(SPP)	Valeur Obj(PM)	Nombre Étiq	Nombre Col
20	1.1	247336.0	3737	235
30	1.6	219053.5	4210	343
40	2.1	212306.2	4754	443
50	2.6	209926	4017	323
Fin	4.5	208278.7	3965	363

5.9.2 Problème II

Le deuxième problème est de taille moyenne ; il possède 700 noeuds, 700 tâches, 348251 arcs et 2 réseaux.

Tableau 5.4 – Problème II : Résolution par APD

PB II	APD			
GC	Temps(SPP)	Valeur Obj(PM)	Nombre Étiq	Nombre Col
20	263	1934192.1	14543772	1478
30	406.2	1584943.6	19810219	2277
40	537.3	1535181.6	2406203	3003
50	646.1	1512298.4	2981139	2238
60	762.5	1503882.2	2766185	2839
70	863.9	1501468.4	3380452	3349
80	942.8	1500913.8	3111643	2310
90	982.3	1500871.5	2423112	2471
Fin	989.8	1500870.9	1594501	2485

Tableau 5.5 – Problème II : Résolution par APD-ND

PB II	APD-ND			
GC	Temps(SPP)	Valeur Obj(PM)	Nombre Étiq	Nombre Col
20	178.7	1806089.4	6718737	1509
30	262.8	1598331.6	7752921	2259
40	330.9	1554359.4	9290210	2841
50	389.8	1535955.6	1102648	3290
60	427.7	1528106.0	1075321	2249
70	462.6	1524305.8	1465298	2582
80	497.0	1522284.8	1207648	2865
90	515.9	1522045.2	9476310	2983
Fin	518.8	1522032.8	6425808	2998

Pour ce problème, vu que la taille est relativement grande, on a choisi plutôt d'appliquer un petit nombre d'opérations MAJ en chaque noeud, et comparer plutôt l'amélioration de la valeur de l'objectif par rapport à la résolution par APD-ND à l'intérieur de GC. Ainsi, on a effectué uniquement 2 itérations de l'algorithme et on

Tableau 5.6 – Problème II : Résolution par ADBS

PB II	ADBS			
GC	Temps(SPP)	Valeur Obj(PM)	Nombre Étiq	Nombre Col
20	178.7	1806089.4	6718737	1509
30	262.8	1598331.6	7752921	2259
40	352.9	1547099.9	12984224	2888
50	446.8	1527991.1	1324254	3417
60	508.1	1522367.3	1472091	2333
70	550.8	1519628.6	1601148	2547
80	590.1	1518343.8	1470720	2744
90	601.2	15196219.5	1191699	2784
Fin	601.2	15196219.5	802937	2784

a appliqué MAJ à uniquement 3 arcs incidents à chaque noeud. Pour ADBS, on a commencé tout d'abord par appliquer APD-ND aux premières itérations de GC, puis appliquer ADBS à partir de la 40ième itération.

Dans le tableau 5.4, on résume les résultats qu'on a obtenu sur les tests effectués sur le problème II en utilisant l'algorithme APD, 5.5 résume ceux de APD-ND et 5.6 ceux ADBS.

Selon les critères qu'on a fixé au début, l'écart de la solution obtenue par ADBS par rapport à la solution optimale est de 28% pour les temps de résolution et 87% pour la valeur de l'objectif.

5.9.3 Problème III

Le troisième problème est de taille relativement grande : il possède 1000 noeuds, 1000 tâches, 720374 arcs et 2 réseaux.

Tableau 5.7 – Problème III : Résolution par APD

PB III	APD			
GC	Temps(SPP)	Valeur Obj(PM)	Nombre Étiq	Nombre Col
20	632.8	2507851.8	63961395	1982
30	998.4	2094018.6	78034956	3030
40	1383.0	2013917.8	82152876	4021
50	1843.5	1980218.0	84173282	4925
60	2293.3	1964754.2	72981013	3738
70	2789.4	1957354.4	74075243	4460
80	3231.5	1954385.6	62852481	3063
90	3481.3	1953778.5	66759332	3511
Fin	3602.0	1953747.7	58773450	3645

Tableau 5.8 – Problème III : Résolution par APD-ND

PB III	APD-ND			
GC	Temps(SPP)	Valeur Obj(PM)	Nombre Étiq	Nombre Col
20	544.9	2273437.0	28465883	2109
30	846.5	2081265.7	37986216	3083
40	1087.5	2035513.2	35761079	3876
50	1309.1	2015042.0	36096342	4488
60	1463.8	2006284.5	31087653	4957
70	1598.5	2002128.9	29524109	3300
80	1715.4	2000099.9	24892135	3588
90	1769.1	1999822.4	25965401	3695
Fin	1774.7	1999818.5	22651928	3700

Tableau 5.9 – Problème III : Résolution par ABDS

PB III	ADBS			
GC	Temps(SPP)	Valeur Obj(PM)	Nombre Étiq	Nombre Col
20	534.4	2273437.0	28465883	2109
30	834.1	2081265.7	37986216	3083
40	1069.5	2035513.2	35761079	3876
50	1301.2	2015042.0	36096342	4488
60	1417.8	1997233.7	38082735	3128
70	1682.4	1987354.4	37285341	4460
80	1874.5	1974385.6	32665198	3063
90	1977.3	1973778.5	28276198	3511
Fin	2034.6	1973767.2	24926271	3536

Pour ce problème aussi, on a choisi plutôt d'appliquer un petit nombre d'opérations de MAJ en chaque noeud, et comparer l'amélioration de la valeur de l'objectif par rapport à la résolution par APD-ND à l'intérieur de GC. Ainsi, on a effectué uniquement 2 itérations de l'algorithme, on a commencé par appliquer la méthode APD-ND lors des premières itérations de GC, puis on a appliqué la méthode ADBS à partir de la 50ième itération de GC. De plus, on a appliqué MAJ uniquement à 4 arcs incidents à chaque noeud.

Les tableaux 5.7, 5.8 et 5.9 résument les résultats des tests qu'on a effectué sur le problème III, en utilisant APD, APD-ND et ADBS, respectivement.

Selon les critères qu'on a fixé au début, l'écart de la solution obtenue par ADBS par rapport à la solution optimale est de 20% pour les temps de résolution et 66% pour la valeur de l'objectif.

5.9.4 Problème IV

Ce problème est très semblable au problème III : il possède 1000 noeuds, 1000 tâches, 711260 arcs et 2 réseaux.

Pour ce problème aussi, on a choisi plutôt d'appliquer un petit nombre de MAJ en chaque noeud, et comparer plutôt l'amélioration de la valeur de l'objectif par rapport à la résolution par APD-ND à l'intérieur de GC. Ainsi on a effectué uniquement 2 itérations de l'algorithme, on a commencé à appliquer la méthode à partir de la 80ième itération de GC et on a appliqué MAJ à uniquement 5 arcs incidents à chaque itération de GC.

Dans le tableau 5.10, on résume les résultats des tests qu'on a effectué sur le problème IV en utilisant l'algorithme APD, 5.11 résume ceux de APD-ND et 5.12 ceux ADBS.

Tableau 5.10 – Problème IV : Résolution par APD

PB IV	APD			
GC	Temps(SPP)	Valeur Obj(PM)	Nombre Étiq	Nombre Col
20	479.3	2422810.6	43343651	2117
30	733.0	2145987.9	52095922	3214
40	976.5	2085683.5	55925432	4193
50	1215.0	2059286.1	51098147	3009
60	1456.6	2047602.8	57701211	3819
70	1684.8	2043318.8	62324419	4478
80	1884.3	2042392.8	42771590	4992
90	2002.8	2042301.6	31818514	3217
Fin	2020.2	2042300.7	21531987	3238

Tableau 5.11 – Problème IV : Résolution par APD-ND

PB IV	APD-ND			
GC	Temps(SPP)	Valeur Obj(PM)	Nombre Étiq	Nombre Col
20	472.7	2383313.8	13140671	2093
30	660.5	2174217.3	17687212	3090
40	818.2	2123277.8	18897226	3891
50	951.8	2100007.4	14582673	4531
60	1057.1	2090655.7	15705109	5006
70	1151.6	2085191.7	19876515	3385
80	1213.3	2083146.3	16854731	3633
90	1260.7	2082045.4	11760128	3815
Fin	1348.0	2080972.4	10844238	4117

Tableau 5.12 – Problème IV : Résolution par ADBS

PB IV	ADBS			
GC	Temps(SPP)	Valeur Obj(PM)	Nombre Étiq	Nombre Col
30	651.5	2174217.3	17687212	3090
40	809.0	2123277.8	18897226	3891
50	939.4	2100007.4	14582673	4531
60	1044.2	2090655.7	15705109	5006
70	1150.1	2085191.7	19876515	3385
80	1312.6	2073927.7	32073509	3950
90	1440.0	2069225.9	34873761	4346
Fin	1504.8	2068598.1	25088413	4431

Selon les critères qu'on a fixé au début, l'écart de la solution obtenue par ADBS par rapport à la solution optimale est de 12% pour les temps de résolution et 32% pour la valeur de l'objectif.

5.9.5 Comparaison de la qualité des solutions en fonction du temps des trois algorithmes

Bien que les heuristiques de résolution qu'on a utilisé soient assez élémentaires, et comme l'indiquent les tableaux de résultats, on peut voir la valeur de l'objectif du problème maître s'approcher de façon significative de la valeur optimale.

Comme le montre la figure 5.8, qui compare les trois algorithmes pour le problème III (choisi étant donné qu'il a la plus grande taille), ADBS produit de meilleures solutions que APD-ND. De plus, si on restreignait son temps de calcul à celui de APD-ND, il produirait de meilleures solutions que ce dernier. D'autre part, la courbe de ADBS est inférieure à celle de APD pour la plupart des temps. Toutefois, on observe qu'il aurait été préférable d'arrêter ADBS plus tôt, et peut-être changer la valeur des paramètres.

Ainsi, on voit que ADBS a un certain potentiel de fournir des solutions de qualité, mais qu'il y a encore place pour amélioration.

5.10 Méthodes d'accélération : choix de i lors de l'opération MAJ$(i, E_j^{\mu}, e_j^{\mathcal{B}})$

Lors des expérimentations numériques de l'algorithme ADBS, étant donné la grande taille des réseaux considérés, on a choisi d'appliquer un petit nombre d'itérations de l'algorithme ADBS. De plus, lors de l'application de ADBS, lorsqu'on a fait le parcours du réseau dans l'ordre topologique inverse, on a plutôt opté de choisir un petit nombre de noeuds i pour lesquels on applique l'opération de mise à jour MAJ$(i, E_j^{\mu}, e_j^{\mathcal{B}})$. Ceci avait pour but d'obtenir rapidement des solutions réalisables de bonne qualité à envoyer au problème maître.

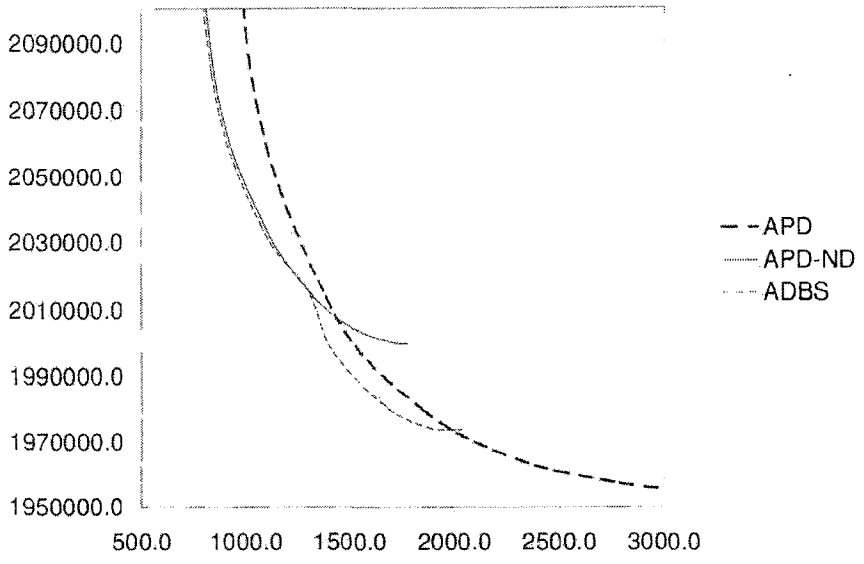

Figure 5.8 – Problème III : Comparaison de la qualité des solutions en fonction du temps des trois algorithmes (valeur de l'objectif en fonction du temps)

Cependant, notre choix pour ce noeud i était assez aléatoire, ainsi une meilleure stratégie s'impose. C'est dans cette optique qu'on propose d'introduire une méthode dynamique de choix du noeud i. Rappelons que, lors de la procédure de mise à jour MAJ$(i, E_j^{\mu}, e_j^{\mathcal{B}})$ (voir algorithme 5.3), on avait une partition de l'ensemble des étiquettes en i, $E_i^{\lambda_1} \cup E_i^{\lambda_2}$. On choisit l'étiquette $e_i^{\mathcal{P}_1}$ de coût minimum dans $E_i^{\lambda_1}$. Or rien ne garantit que ce noeud existe et possède un successeur réalisable dans E_j^{μ}. On a ainsi introduit l'algorithme ADBS-I (5.6) qui opère de façon similaire à l'algorithme ADBS, à la différence que pour tout noeud j, on trie d'abord tous les arcs entrant au noeud j et on applique la procédure MAJ$(i, E_j^{\mu}, e_j^{\mathcal{B}})$ à l'arc (i, j) de coût minimum et tel que l'étiquette $e_i^{\mathcal{P}_1}$ au noeud i existe et possède un successeur réalisable en j. Ceci nous permet, d'une part, d'éviter de créer inutilement un nouvel ensemble d'étiquettes et, par la suite, un ensemble de nouveaux chemins qui risquent d'être non pertinents pour la résolution du problème et, d'autre part, de choisir l'arc entrant de coût minimum lors de l'application de l'opération de mise à jour. Ainsi, pour tout j, désignons par N_j l'ensemble des noeuds i tels que $(i, j) \in A$, qui sera utilisé dans l'algorithme 5.6.

Algorithme 5.6 Algorithme d'amélioration dynamique de la borne supérieure modifié : ADBS-I

pour tout $j = n, n-1, \ldots, 0$ **faire**
  pour tout $e_j^{\mathcal{B}} \in B_j$ **faire**
    pour tout E_j^{λ} tel que $e_j^{\mathcal{B}}$ est relativement non réalisable par rapport à E_j^{λ} **faire**
      Soit $i \in N$ tel que $(i,j) \in A$ et qui précède j sur le chemin $\mathcal{B}$.
    Trier N_j en ordre croissant du coût des arcs $(i,j) \in A$
    Prendre i_0 le noeud de moindre coût de N_j.
    si $E_{i_0}^{\lambda}$ possède une étiquette $e_{i_0}^{\mathcal{P}_1} = (C_{i_0}^{\mathcal{P}_1}, T_{i_0}^{\mathcal{P}_1})$ tel que $T_i^{\mathcal{P}_1} \leq b_j - t_{ij}$ **alors**
      Appliquer l'opération MAJ$(i_0, E_j^{\lambda}, e_j^{\mathcal{B}})$.
    Choisir un nouveau i_0, qui est le noeud suivant dans N_j
    Revenir à l'étape précédente.
  Ajouter les nouveaux noeuds relativement non réalisables à B_i
Appliquer l'algorithme de marquage (5.5) sur le nouveau réseau.

5.10.1 Résultats numériques

Afin de vérifier l'efficacité de ces heuristiques, on a effectué des tests sur différents problèmes avec les mêmes paramètres. Ces paramètres consistent à appliquer la méthode à partir de l'itération 50, l'appliquer uniquement deux fois et pour chaque noeud j, et on choisit un seul noeud i, lors de l'application de l'opération MAJ, selon l'algorithme 5.6.

Dans les tableaux de valeurs 5.13 et 5.14 , on notera par *APD* la résolution par l'algorithme de programmation dynamique, *APD-ND* la résolution par programmation dynamique en dominant uniquement sur les coûts, *ADBS* la résolution par l'algorithme d'amélioration dynamique de la borne supérieure, *ADBS-I* la résolution par l'algorithme d'amélioration dynamique de la borne supérieure amélioré selon l'algorithme 5.6, *GC* l'itération de génération de colonnes, par *T(SPP)* le temps cumulé de la résolution des sous-problèmes, *Nbre Col* le nombre de colonnes générées, *Val Obj(PM)* la valeur de l'objectif du problème maître, % C le pourcentage d'écart de la valeur du problème maître, obtenu en résolvant avec ADBS-I, par rapport à APD et % T le pourcentage d'écart des temps de résolution par ADBS-I, par rapport à APD.

Les premiers tests ont été effectués sur les problèmes III et IV de la section précédente, les résultats numériques sont rapportés dans le tableau 5.13.

Tableau 5.13 – Comparaison des approches de résolution pour les problèmes III et IV

PB	ALGO	GC	T(SPP)	Val Obj(PM)	Nbre Col	%C	%T
III	ADBS	98	2034.6	1973767.2	3536	64 %	27%
	ADBS-I	108	2209	1972823.4	3981	58%	32%
IV	ADBS	112	1504.8	2068598.1	4431	22%	24%
	ADBS-I	119	1965	2061054.3	4347	51%	28%

On remarque un écart par rapport à APD, en pourcentage, de la valeur de l'objectif pour les deux problèmes (environ 58% pour le premier et 58% pour le second). C'est ainsi qu'on a retenu cette heuristique pour de prochaines expérimentations sur une plus grande échelle.

Le tableau 5.14 montre les résultats qu'on a obtenu sur une série de problèmes du même type que ceux de la section précédente. Tous ces problèmes ont 1000 noeuds, 1000 tâches et cinq réseaux.

Dans ces expérimentations, on remarque qu'on a des améliorations de la valeur de l'objectif qui peuvent atteindre 59% par rapport à la résolution par APD-ND.

5.10.2 Comparaison de la qualité des solutions en fonction du temps des trois algorithmes lors de la résolution par ADBS-I

La figure 5.9 illustre le rapport, en pourcentage, d'amélioration de l'objectif et le pourcentage de consommation du temps pour chacun des problèmes qu'on vient de présenter. L'axe des abscisses est le pourcentage de détérioration des temps de

Tableau 5.14 – Comparaison des approches de résolution pour quelques problèmes

PB	ALGO	GC	T(SPP)	Val Obj(PM)	Nbre Col	%AM	%T
V	APD	104	2725.9	1942063.8	3946	100 %	100%
	APD-ND	108	596.4	1985393.6	4081	0 %	0%
	ADBS-I	115	829.4	1966922.5	3034	42 %	11%
VI	APD	101	2749.0	1951948.6	3779	100 %	100%
	APD-ND	104	547.5	1992466.4	3304	0%	0%
	ADBS-I	115	730.1	1972097.6	3570	50%	8%
VII	APD	95	2556.4	1925140.9	3372	100 %	100%
	APD-ND	108	466.6	1968718.6	3860	0%	0%
	ADBS-I	120	730.2	1949306.3	3007	45%	12%
VIII	APD	99	2999.1	1940576.5	3677	100 %	100%
	APD-ND	119	904.8	1978839.6	4683	0 %	0%
	ADBS-I	128	1359.2	1961570.3	3977	45 %	21%
IX	APD	102	3051.4	1933714.3	4090	100 %	100%
	APD-ND	147	1054.6	1966706.4	4890	0 %	0%
	ADBS-I	135	1317.5	1951170.6	4191	47 %	13%
X	APD	106	3234.4	1878393.3	3942	100 %	100%
	APD-ND	146	798.7	1918880.9	3355	0 %	0%
	ADBS-I	124	1169.4	1897091.3	4382	53 %	15%
XI	APD	100	3035.1	1896193.0	3646	100 %	100%
	APD-ND	111	618.9	1932272.0	4872	0 %	0%
	ADBS-I	139	1131.8	1911036.3	4704	59 %	21%
XII	APD	104	3222.0	1928370.4	3865	100 %	100%
	APD-ND	111	984.8	1962269.1	4844	0 %	0%
	ADBS-I	115	1343.0	1943117.9	4413	56 %	16%

5.10.2 Comparaison de la qualité des solutions en fonction du temps des trois algorithmes lors de la résolution par ADBS-I

La figure 5.9 illustre le rapport, en pourcentage, d'amélioration de l'objectif et le pourcentage de consommation du temps pour chacun des problèmes qu'on vient de présenter. L'axe des abscisses est le pourcentage de détérioration des temps de résolution par rapport au temps de calculs pour APD-ND, l'axe des ordonnées est le pourcentage de la valeur de l'objectif. Ainsi, le point situé à l'origine du repère correspond à la solution obtenue en utilisant l'algorithme APD-ND, le point au coin supérieur droit correspond à la solution obtenue en utilisant l'algorithme APD. Ainsi une bonne solution serait située au dessus de la diagonale du graphe ; en particulier, plu ıe, me

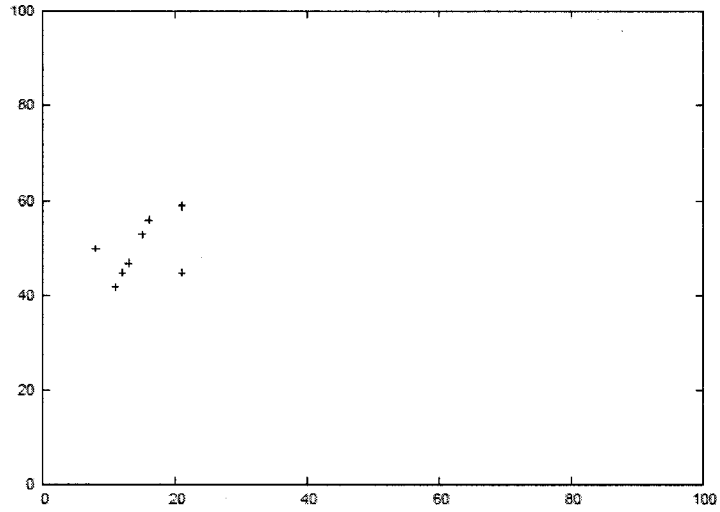

Figure 5.9 – Pourcentage d'amélioration de la valeur de l'objectif en fonction du pourcentage de détérioration des temps pour ADBS-I

On voit sur cette figure une tendance à ce que les points correspondant aux solutions obtenues par ADBS-I pour les différents problèmes soient proches du coin supérieur gauche. Ceci montre une bonne qualité des solutions obtenues.

Comme le montre la figure 5.10, représentant une comparaison des 3 algorithmes pour le problème V, ADBS-I produit de meilleures solutions que APD-ND. De plus, sa courbe représentative est toujours bien inférieure à celle de APD. L'algorithme ADBS-I fournirait de bien meilleures performances que APD, si on arrêtait APD au même temps où s'arrête ADBS-I. Ainsi, on voit que ADBS en général semble très prometteur pour fournir des solutions de qualité.

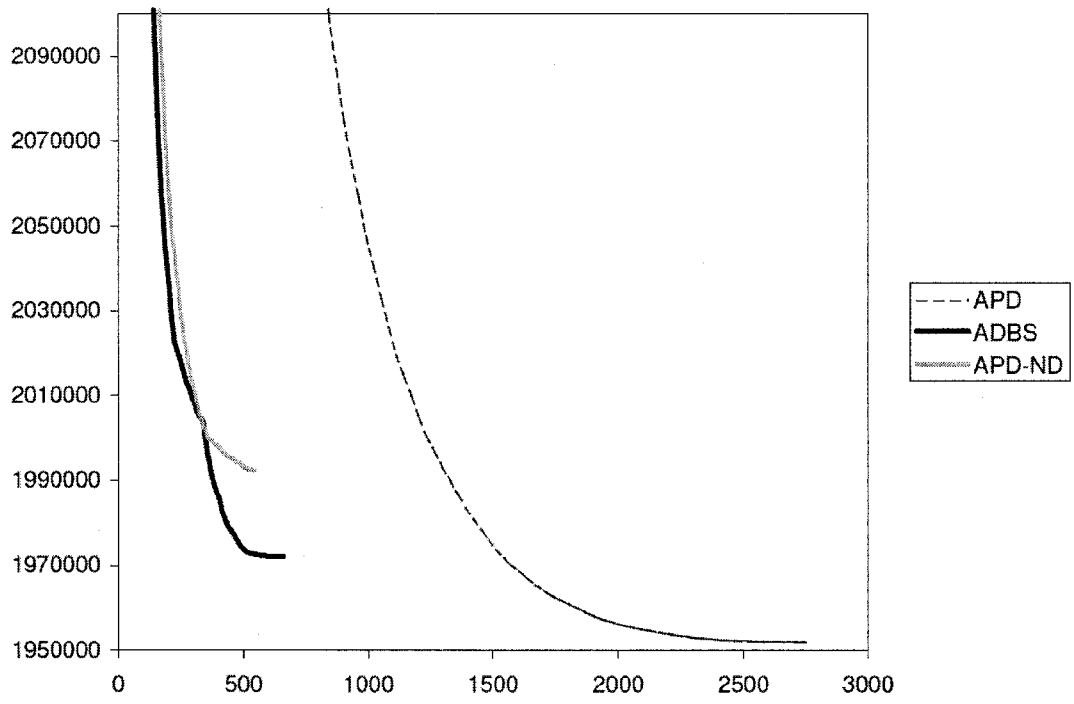

Figure 5.10 – Problème V : Comparaison des performances des 3 algorithmes (valeur de l'objectif en fonction du temps)

CHAPITRE 6

MÉTHODES D'ACCÉLÉRATION

L'algorithme d'amélioration dynamique de la borne supérieure qu'on a proposé dans la section précédente est assez prometteur. Cependant, dans certains cas, l'atteinte de l'optimalité peut s'avérer une tâche fastidieuse. Ceci est dû au fait qu'on doit parfois explorer un très grand nombre d'étiquettes avant de s'approcher de la solution optimale.

Dans ce chapitre, on se propose d'explorer deux méthodes d'accélération. La première est basée sur l'attribution de variables duales à chaque noeud du graphe G. Ceci permet de faire un choix plus judicieux quant aux noeuds sur lesquels on applique la procédure de mise à jour ceci en privilégiant, la plus grande amélioration de la valeur de l'objectif plutôt que l'enrichissement du graphe des états. Cette méthode va être appuyée par plusieurs résultats numériques montrant son efficacité.

Une deuxième stratégie consiste à exploiter la borne inférieure que peut fournir la relaxation lagrangienne, afin d'établir de meilleurs critères d'arrêt d'une part, et d'éviter certaines opérations de mise à jour inutiles d'autre part. En effet, ceci nous permettra d'arrêter la résolution lorsque la marge d'erreur deviendra assez petite (ou relativement petite dans certains cas) afin d'éviter des calculs qui risquent d'apporter peu d'améliorations au niveau de l'algorithme de génération de colonnes.

6.1 Méthode d'accélération en utilisant les variables duales

On propose dans cette section une première méthode d'accélération de l'algorithme dynamique d'amélioration de la borne supérieure. Cette méthode consiste à attribuer des variables duales aux noeuds. Les variables duales sont obtenues à partir des coûts associés aux noeuds, en utilisant l'algorithme d'étiquetage APD. Nous verrons par la suite, plus précisément, comment on obtient ces valeurs duales.

Par la suite, à l'intérieur de l'algorithme, on peut privilégier l'application de l'opération MAJ aux arcs dont les coûts réduits sont les plus prometteurs, c'est-à-dire ceux qui peuvent améliorer rapidement la borne supérieure du problème, comme on le montre dans le théorème 3.

6.1.1 Fonctions duales et variables duales

Définition 7 *Soit E_i^λ un sous-ensemble de noeuds, comme défini dans les sections précédentes, i étant un noeud de N et $\lambda = [f_i, g_i]$ un sous-intervalle de $[a_i, b_i]$ la fenêtre de temps en i du problème PCC-FT. On définit la fonction duale sur E_i^λ par :*

$$F_i^\lambda : \lambda \rightarrow \mathbb{R}$$

$$F_i^\lambda(t) = \begin{cases} \min\{C_i \mid T_i \leq t \; et \; (C_i, T_i) \in E_i^\lambda\} \; si \; E_i^\lambda \neq \emptyset \\ +\infty \; si \; E_i^\lambda = \emptyset. \end{cases}$$

Remarque 11 *Remarquons que $F_i^\lambda(g_i)$ constitue la valeur du plus court chemin courant de l'origine vers un noeud de E_i^λ. Ainsi, si $\lambda = [a_i, b_i]$, alors $F_i^\lambda(b_i)$ est la valeur du plus court chemin du problème PCC-FT au noeud i.*

Définition 8 *Soit (i, j) un arc de A, et soient E_i^λ et E_j^μ deux sous-ensembles d'étiquettes en i et j tels que $\lambda = [f_i, g_i]$ et $\mu = [f_j, g_j]$, deux sous-intervalles des*

fenêtres de temps en i et j respectivement. Soit $\mathcal{P}$ et $\mathcal{Q}$ deux chemins tels que $\mathcal{Q} = \mathcal{P} \cup \{(i,j)\}$. On définit le coût réduit sur l'arc $(e_i^{\mathcal{P}}, e_j^{\mathcal{Q}}) \in \mathcal{A}$, et qu'on note $\bar{c}_{ij}^{\mathcal{P}}$, par : $\bar{c}_{ij}^{\mathcal{P}} = c_{ij} + F_i^{\lambda}(g_j - t_{ij}) - F_j^{\mu}(g_j)$.

Notons par $F_i^k(t)$ la valeur de la fonction duale, à l'itération k de l'algorithme ADBS, lorsqu'il n'y a pas de confusion sur l'intervalle sur lequel cette fonction est définie.

Notons aussi que, lors d'une itération de ADBS, plus le coût réduit sur un arc est négatif, plus il a de potentiel pour améliorer la solution courante, comme le montre le théorème qui suit.

Théorème 3 *Pour tout arc $(i,j) \in A$ et pour n'importe quelle itération k de l'algorithme d'amélioration dynamique de la borne, les coûts réduits $\bar{c}_{ij}^{\mathcal{P}}$ vérifient l'une des propositions suivantes :*

- *Cas 1 : Si $\bar{c}_{ij}^{\mathcal{P}} < 0$ alors $F_j^{k+1}(g_j) < F_j^k(g_j)$.*
- *Cas 2 : Si $\bar{c}_{ij}^{\mathcal{P}} \geq 0$ alors on n'améliore pas la solution courante en appliquant MAJ sur l'arc (i,j).*

Preuve :

Si $\bar{c}_{ij}^{\mathcal{P}} < 0$, alors $c_{ij} + F_i^k(g_j - t_{ij}) < F_j^k(g_j)$. Donc, la nouvelle étiquette en j, créée par l'opération MAJ, appliquée à l'arc (i,j) a un coût $C_j^* \leq c_{ij} + F_i^k(g_j - t_{ij})$, ainsi on $F_j^{k+1}(g_j) \leq C_j^* = c_{ij} + F_i(g_j - t_{ij}) < F_j^k(g_j)$ et alors $F_j^{k+1}(g_j) < F_j^k(g_j)$.
D'autre part, on a $\bar{c}_{ij}^{\mathcal{P}} = c_{ij} + F_i^k(g_j - t_{ij}) - F_j^k(g_j)$, or si $\bar{c}_{ij}^{\mathcal{P}} \geq 0$ alors $c_{ij} + F_i^k(g_j - t_{ij}) \geq F_j^k(g_j)$, mais par définition, la nouvelle étiquette en j, créée par l'opération MAJ, a un coût C_j^* tel que $C_j^* = c_{ij} + F_i^k(g_j - t_{ij})$ donc $C_j^* \geq F_j^k(g_j)$, ce qui prouve que rien ne garantit qu'on peut améliorer la solution courante.

Remarque 12 *Le cas 2 du théorème 3 montre que lorsque $\bar{c}_{ij} \geq 0$, l'application de l'opération de mise à jour MAJ à l'itération courante ne peut pas garantir une*

amélioration de la solution courante. Ainsi, lors de l'application de l'opération MAJ, il faut éviter de traiter les arcs qui ont un coût réduit non négatif.

Remarque 13 *S'il existe un arc (i, j) qui vérifie la condition du cas 1 du théorème 3, c'est-à-dire $\bar{c}_{ij} < 0$ et si, de plus, j fait partie du chemin optimal à l'itération courante de ADBS, alors l'application de l'opération de mise à jour MAJ à cet arc améliorera la valeur de la solution courante. D'autre part, si j fait partie d'un chemin $\mathcal{P}$ de o à d et si $\bar{c}_{ij}$ est plus petit que la différence entre la valeur de ce chemin et la valeur de la solution courante, alors l'application de l'opération de mise à jour MAJ à cet arc améliorera aussi la valeur de la solution courante.*

Ainsi, ce théorème nous fournit non seulement un outil d'accélération efficace dans la façon d'appliquer ADBS, mais aussi une méthode avec laquelle on est sûr d'avoir au moins une amélioration à chaque étape de l'algorithme. En effet, en mettant en ordre croissant les valeurs des $\bar{c}_{ij}$, on applique l'opération de mise à jour selon cet ordre prédéfini, ce qui permet une amélioration plus rapide surtout au début de l'algorithme. Ainsi, on propose l'algorithme 6.1.

L'algorithme 6.1 peut assurer une amélioration efficace de la borne supérieure à chaque itération, du moins aux premières itérations dans lesquelles on peut avoir des coûts réduits négatifs sur les arcs.

L'algorithme proposé peut ne pas nécessairement mener à l'optimalité, étant donné qu'on traite uniquement les arcs qui ont un coût réduit strictement négatif. On peut atteindre l'optimalité si on relâche cette condition. Une autre stratégie qui peut nous mener à l'optimalité est de considérer, en dernier lieu, les arcs qui ont un coût réduit positif ou nul.

Algorithme 6.1 Algorithme d'amélioration de la borne supérieure accéléré

Soit $\Delta < 0$ un critère d'arrêt.
répéter
  pour $j = n, n-1, \ldots, 0$ **faire**
    pour tout $e_j^{\mathcal{B}} \in B_j$ **faire**
      Soit $i \in N$ tel que $(i,j) \in A$ et qui précède j sur le chemin $\mathcal{B}$.
      pour tout E_j^{λ} tel que $e_j^{\mathcal{B}}$ est relativement non réalisable par rapport à E_j^{λ} **faire**
        Calculer $\bar{c}_{ij}^{\lambda}$.
    Classer les $\bar{c}_{ij}^{\lambda}$ en ordre croissant.
  pour La valeur la plus petite de $\bar{c}_{ij}$ jusqu'à la valeur la plus grande et $< \Delta$. **faire**
    Appliquer l'opération MAJ$(i, E_j^{\lambda}, e_j^{\mathcal{B}})$.
  Appliquer l'algorithme de marquage (5.5) sur le nouveau réseau ainsi obtenu.
jusqu'à $B_j = \emptyset$

6.2 Méthodes d'accélération utilisant les variables duales : heuristiques et résultats numériques

Dans la partie d'expérimentations du chapitre précédent, on a appliqué des heuristiques assez simples pour obtenir rapidement des solutions réalisables, de bonne qualité à envoyer au problème maître. Ces heuristiques consistaient, en premier lieu, d'appliquer la procédure MAJ à un sous-ensemble de petite taille d'arcs et non à tous les arcs incidents à des noeuds relativement non réalisables. De plus, afin de réduire les temps de calcul, on a choisi d'effectuer un petit nombre d'itérations de ADBS.

Bien que ceci a donné de bonnes améliorations par rapport à l'heuristique qui constitue en la dominance uniquement par rapport au coût, l'algorithme restait à l'état brut, c'est-à-dire que malgré sa grande flexibilité, on ne peut pas garantir de meilleures améliorations des solutions au fil des itérations.

On se propose dans cette section d'apporter des améliorations à l'algorithme original à la lumière des résultats du théorème 3 et de l'algorithme 6.1. Ces améliorations se concrétisent comme suit.

6.2.1 Choix de l'arc entrant (i, j) lors de la procédure de mise à jour MAJ$(i, E_j^{\mu}, e_j^{\mathcal{B}})$

Lorsqu'on applique l'algorithme ADBS, on parcourt le réseau en ordre topologique inverse et on applique la procédure MAJ à chaque fois qu'on a un nouveau noeud relativement non réalisable $e_j^{\mathcal{B}}$. Mais en considérant le théorème 3 et l'algorithme 6.1, seuls les arcs (i, j) qui ont des coûts réduits strictement négatifs peuvent améliorer directement la solution courante.

Dans cette optique, on s'est proposé d'utiliser une heuristique d'accélération compatible avec la grande taille des réseaux considérés. Ainsi, on calculera, à la fin de chaque itération de l'algorithme, des variables duales à chaque noeud. Ceci s'effectue en initialisant les variables duales en chaque noeud à ∞. Par la suite, on remonte d'abord le plus court chemin trouvé à partir du noeud puits vers le noeud source et on associe comme variable duale au noeud i la valeur du plus petit coût sur les étiquettes disponibles en ce noeud.

Par la suite, on fait la même chose pour le reste des chemins arrivant au noeud destination, en ordre croissant du coût des chemins correspondants. Lors du parcours inverse, dès qu'on trouve un noeud qui a déjà une variable duale finie, on arrête car on n'a pas besoin de retraiter les sous-chemins deux fois. Puis, lors du parcours du réseau en ordre topologique inverse, lorsqu'on a un noeud j qui contient un ou plusieurs noeuds relativement non réalisables $e_j^{\mathcal{B}}$ en j, on calcule tous les coûts réduits des arcs (i, j), tels que i fait partie du chemin $\mathcal{B}$. On procède de la même façon pour le reste des chemins arrivant jusqu'au noeud destination.

Par la suite, on trie ces arcs en ordre croissant de leur coût réduit. On prend l'arc de plus petit coût réduit (et peut être aussi le deuxième comme on verra plus tard) et on applique MAJ uniquement lorsque le coût réduit de ce dernier est négatif. Finalement, on applique l'algorithme de marquage. Plus précisément, on procède selon

ADBS mais en utilisant les variables duales comme on l'illustre dans l'algorithme 6.2 qu'on notera par ADBS-D.

Algorithme 6.2 Algorithme d'amélioration de la borne supérieure accéléré : ADBS-D

pour $k = 1, ..., N$ (N critère d'arrêt) **faire**
  Soit $e_n^{\mathcal{P}^{Opt}}$ l'étiquette correspondant au plus court chemin en $n = d$ à l'itération courante.
  pour $j = n, n-1, \ldots, 0$ noeuds du chemin $\mathcal{P}^{Opt}$ **faire**
    Calculer la variable duale au noeud j
  pour $j = n, n-1, \ldots, 0$ **faire**
    pour tout $e_j^{\mathcal{B}} \in B_j$ **faire**
      Soit $i \in N$ tel que $(i,j) \in A$ et qui précède j sur le chemin $\mathcal{B}$.
      pour tout E_j^{λ} tel que $e_j^{\mathcal{B}}$ est relativement non réalisable par rapport à E_j^{λ} **faire**
        Calculer $\bar{c}_{ij}^{\lambda}$.
      Classer les $\bar{c}_{ij}^{\lambda}$ en ordre croissant.
    Soit (i,j) l'arc ayant la valeur la plus petite pour $\bar{c}_{ij}$.
    si $\bar{c}_{ij} < 0$ **alors**
      Appliquer l'opération MAJ$(i, E_j^{\lambda}, e_j^{\mathcal{B}})$.
  Appliquer l'algorithme de marquage 5.5 sur le nouveau réseau ainsi obtenu.

Remarquons que, lors des expérimentations numériques, on a fixé le nombre d'itérations de l'algorithme ADBS-D à $N = 2$ afin de fournir rapidement des solutions réalisables au problème maître.

6.2.2 Accélération à l'intérieur de l'algorithme de génération de colonnes

Pendant la résolution à l'intérieur de l'algorithme de génération de colonnes, on a remarqué que lors des premières itérations, l'algorithme APD-ND pour les sous-problèmes permet une plus grande décroissance de l'objectif par unité de temps. Ainsi, on a opté de résoudre les sous-problèmes lors des premières itérations de génération de colonnes, en utilisant APD-ND puis changer à l'algorithme 6.2 ultérieurement.

De plus, d'après les expérimentations qu'on a déjà effectué, on a aussi remarqué qu'après quelques itérations de génération de colonnes même pour l'algorithme 6.2, le nombre de colonnes générées cesse d'assurer convenablement la décroissance de la valeur de l'objectif.

C'est à ce moment qu'on introduit une variante de l'algorithme 6.2. Dans cette variante, on considère les deux arcs qui ont les coûts réduits les plus bas au lieu d'un seul. Ceci permettra la génération de colonnes de meilleure qualité et assurera par la suite un bonne décroissance de la valeur de l'objectif. Ceci est dû au fait qu'on rajoute de nouveaux chemins de meilleurs coût qui assurent la décroissance le l'objectif du problème maître de GC, tout en gardant des temps de résolution assez bas. Ceci s'est avéré très efficace comme on le montrera dans les résultats numériques qui suivent.

6.2.3 Résultats numériques

Lors des expérimentations préliminaires, on a remarqué que l'efficacité de l'algorithme ADBS augmente avec la taille des réseaux. Ainsi, pour cette partie, on va considérer uniquement des problèmes de taille 1000 noeuds et 1000 tâches et 5 réseaux. Ces problèmes son différents de ceux du chapitre précédent. Les tests ont été effectués sur des machines 64 bits avec un processeur AMD de 2.4GHz.

Pour les problèmes qui suivent, afin d'obtenir des résultats préliminaires efficaces sans avoir à manipuler trop de paramètres, on a choisi d'appliquer l'algorithme à partir de la 40ième itération de l'algorithme de génération de colonnes. Ce choix a été dicté par l'allure de la courbe donnant la valeur de l'objectif en fonction du temps pour l'algorithme APD-ND. En effet, on a remarqué que cet algorithme commence à ralentir à partir de cette itération. C'est pour les mêmes raisons qu'on passe à la deuxième phase de l'algorithme (considérer deux arcs par noeud lors de la procédure MAJ au lieu d'un seul) à l'itération 70 de l'algorithme de génération de colonnes et ce pour tous les problèmes considérés.

Avec les mêmes notations que celles du chapitre précédent, on donne les tableaux de résultats pour le temps total d'exécution des sous-problèmes et la valeur de l'objectif final du problème maître. Afin d'illustrer la différence entre les trois algorithmes, on a établi des figures comparatives exprimant la valeur de l'objectif du problème maître en fonction du temps.

6.2.3.1 Problème I

Le problème I est un problème de horaires de véhicules avec fenêtres de temps ayant 1000 noeuds et 74845 arcs. On présente dans le tableau 6.1 les résultats des différentes approches.

Tableau 6.1 – Problème I : Comparaison des approches de résolution pour le problème I

PB	ALGO	GC	T(SPP)	Val Obj(PM)	Nbre Col	%AM	%T
I	APD	93	2706.3	1977666.6	3344	100 %	100%
	APD-ND	98	417.4	2011013.0	3930	0%	0%
	ADBS-D	109	1153.3	1984037.9	3633	81%	32%

Remarquons que nous avons amélioré la valeur de l'objectif de 81% par rapport à la valeur de APD-ND, alors que les temps ont augmenté de 30% uniquement. La courbe représentative (figure 6.1) montre clairement que l'algorithme ADBS-D est bien plus rapide que les autres algorithmes.

6.2.3.2 Problème II

Le problème II est un problème de horaires de véhicules avec fenêtres de temps ayant 1000 noeuds et 76326 arcs. On représente sur le tableau 6.2 les résultats des différentes approches.

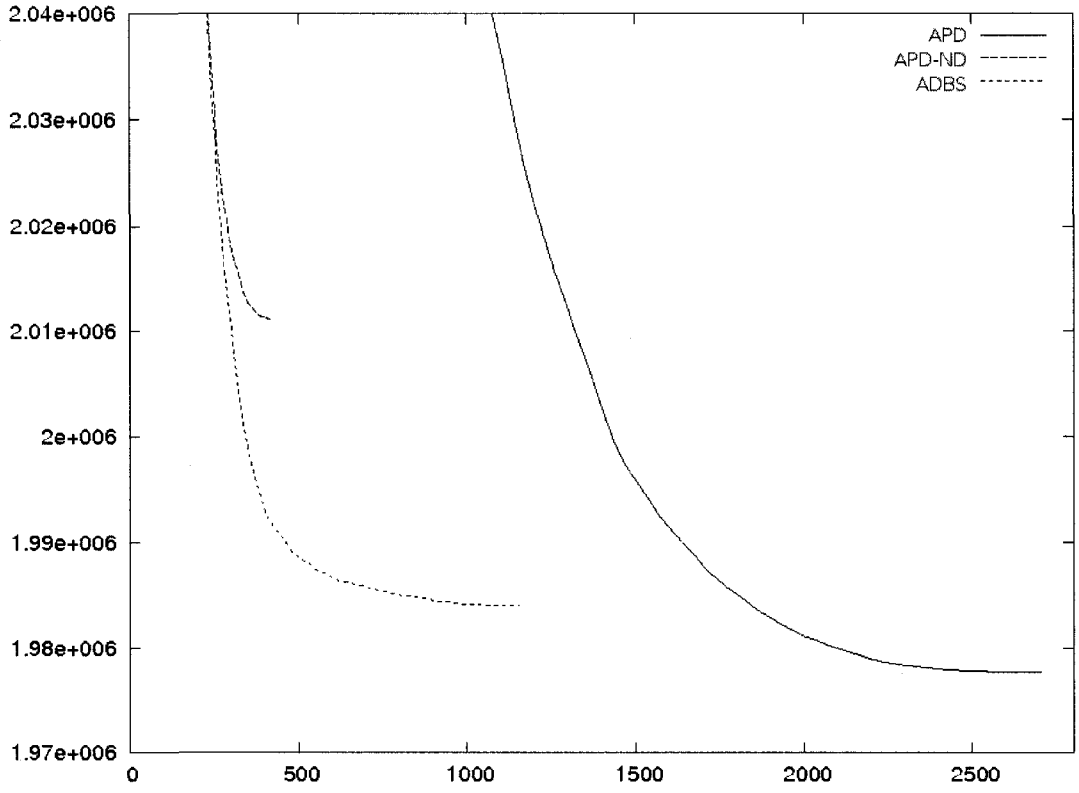

Figure 6.1 – Problème I : Comparaison de la vitesse des trois algorithmes (valeur de l'objectif en fonction du temps)

Tableau 6.2 – Comparaison des approches de résolution pour le problème II

PB	ALGO	GC	T(SPP)	Val Obj(PM)	Nbre Col	%AM	%T
II	APD	98	2729.2	1938114.3	3428	100 %	100%
	APD-ND	136	594.8	1987100.1	3786	0%	0%
	ADBS-D	124	1174.7	1950322.0	3566	75%	27%

Remarquons que nous avons amélioré la valeur de l'objectif de 75%, alors que les temps ont augmenté de 28% uniquement. La courbe représentative (figure 6.2) montre clairement que l'algorithme ADBS-D est bien plus rapide que les autres algorithmes.

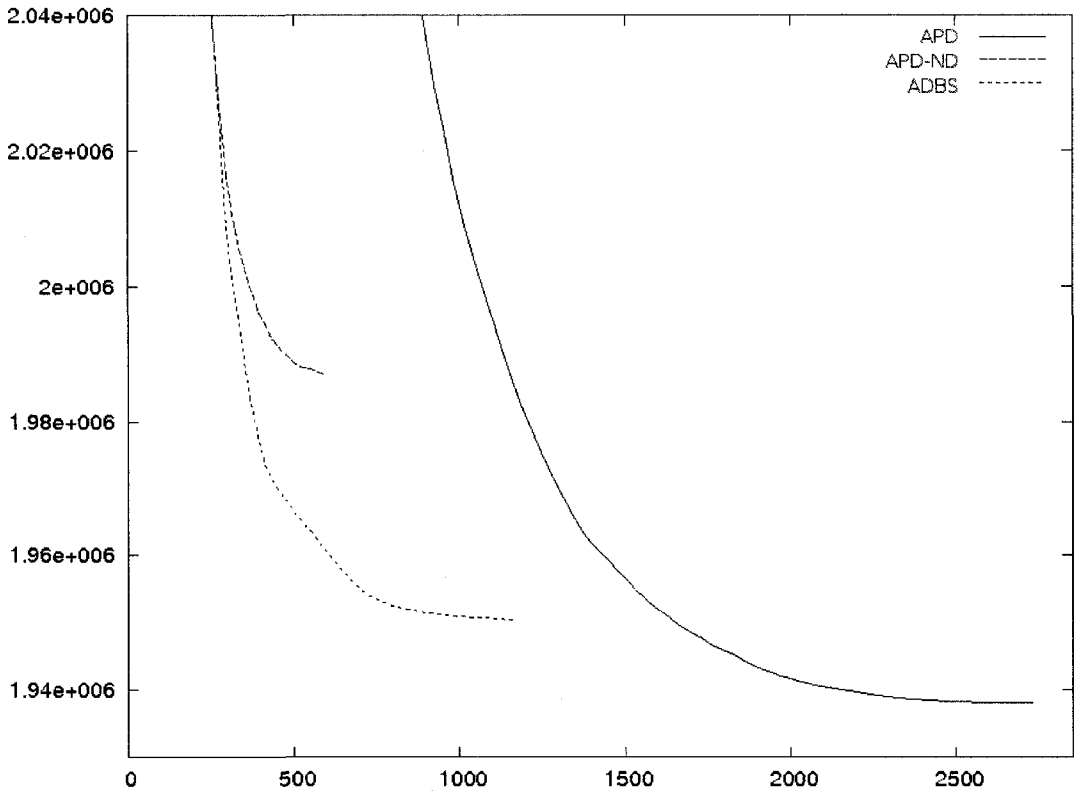

Figure 6.2 – Problème II : Comparaison de la vitesse des trois algorithmes (valeur de l'objectif en fonction du temps)

6.2.3.3 Problème III

Le problème qu'on considère dans cette section est un problème de horaires de véhicules avec fenêtres de temps ayant 1000 noeuds et 69875 arcs. Les résultats qu'on a obtenu pour ce problème sont présentés dans le tableau 6.3.

Remarquons que nous avons amélioré la valeur de l'objectif de 71%, alors que les temps ont augmenté de 18% uniquement. La courbe représentative (figure 6.3) montre clairement que l'algorithme ADBS-D est bien plus rapide que les autres algorithmes.

Tableau 6.3 – Comparaison des approches de résolution pour le problème III

PB	ALGO	GC	T(SPP)	Val Obj(PM)	Nbre Col	%AM	%T
III	APD	100	2389.6	1938114.3	3550	100 %	100%
	APD-ND	134	853.4	1986258.6	4128	0%	0%
	ADBS-D	111	1094.7	1952032.2	3474	71%	15%

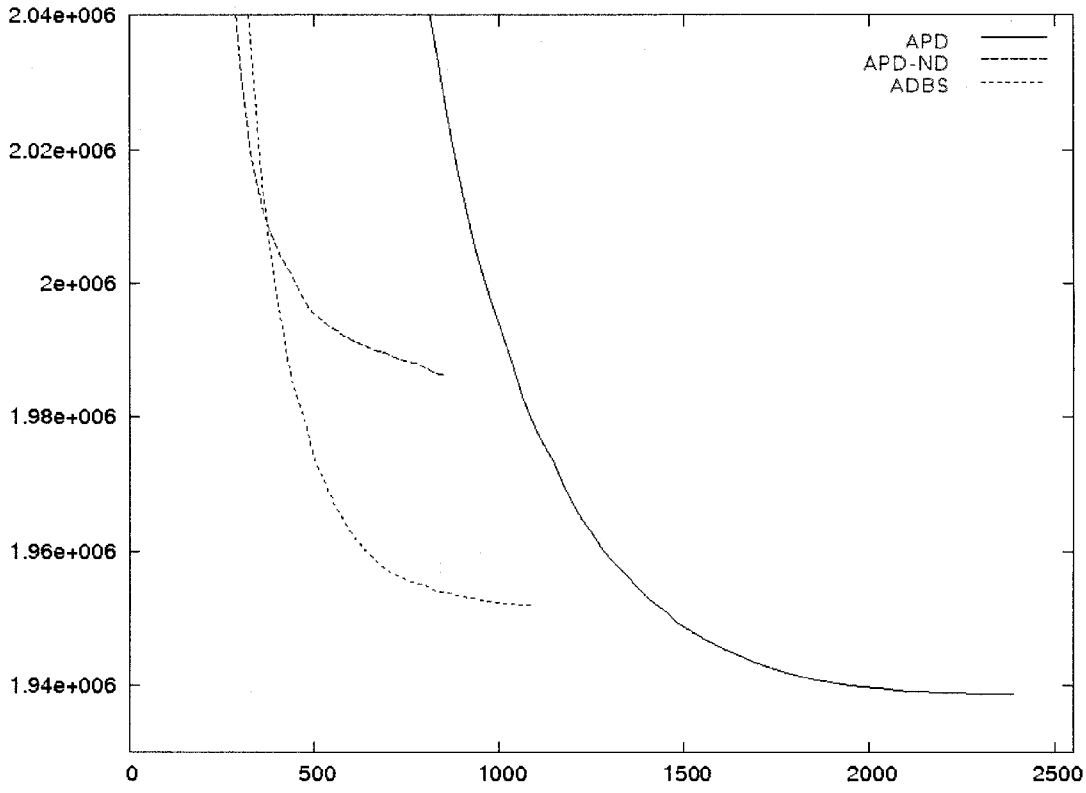

Figure 6.3 – Problème III : Comparaison entre la vitesse des trois algorithmes (valeur de l'objectif en fonction du temps)

6.2.3.4 Problème IV

Ici encore, on considèrera le même type de problèmes. Le problème suivant possède 1000 noeuds et 71785 arcs. Les résultats qu'on a obtenu pour ce problème sont

présentés dans le tableau 6.4.

Tableau 6.4 – Comparaison des approches de résolution pour le problème IV

PB	ALGO	GC	T(SPP)	Val Obj(PM)	Nbre Col	%AM	%T
IV	APD	97	2801.2	1982951.1	3388	100 %	100%
	APD-ND	111	532.8	2030786.8	3853	0%	0%
	ADBS-D	117	1401.7	1993391.7	3946	78%	38%

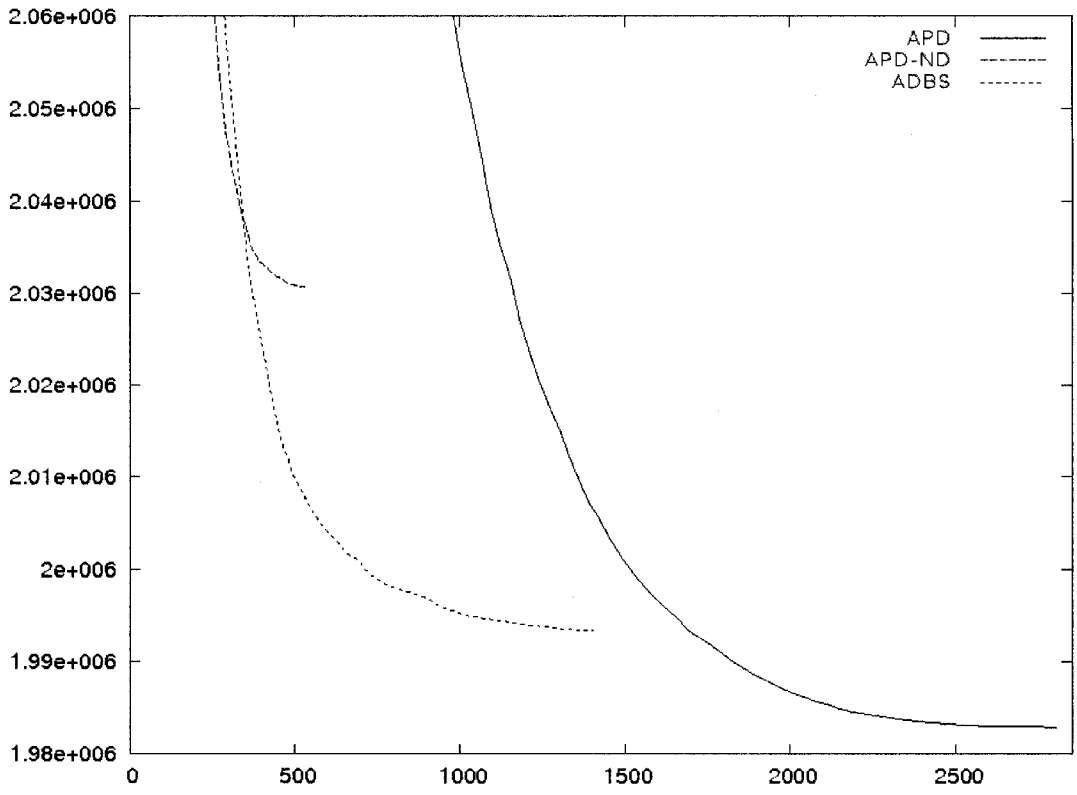

Figure 6.4 – Problème IV : Comparaison entre la vitesse des trois algorithmes (valeur de l'objectif en fonction du temps)

Remarquons que nous avons amélioré la valeur de l'objectif de 79%, alors que les temps ont augmenté de 57%. La courbe représentative montre clairement que l'algorithme ADBS-D est bien plus rapide que les autres algorithmes.

6.2.3.5 Problème V

Pour le même genre de problèmes avec 1000 noeuds et 71785 arcs, le tableau 6.5 montre les résultats obtenus pour les 3 algorithmes.

Tableau 6.5 – Comparaison des approches de résolution pour le problème V

PB	ALGO	GC	T(SPP)	Val Obj(PM)	Nbre Col	%AM	%T
V	APD	98	2693.0	2013792.6	3630	100 %	100%
	APD-ND	126	557.8	2055167.2	3820	0%	0%
	ADBS-D	111	1122.6	2027936.6	3270	78%	26%

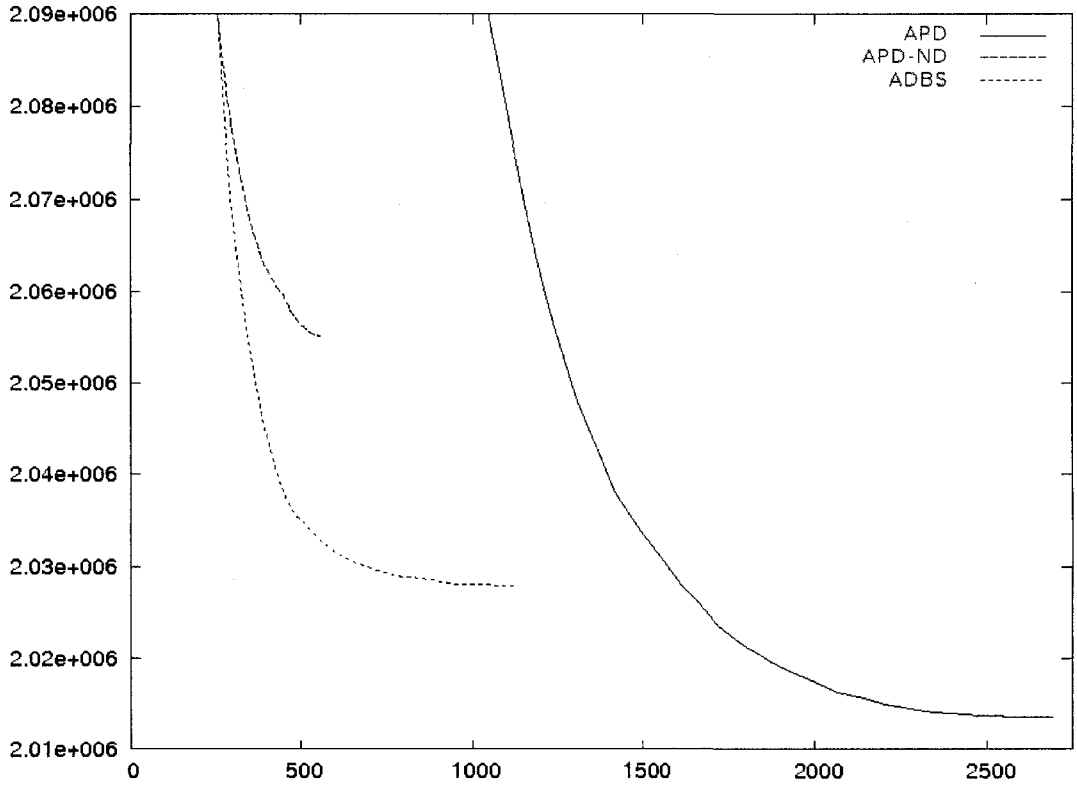

Figure 6.5 – Problème V : Comparaison de la vitesse des trois algorithmes (valeur de l'objectif en fonction du temps)

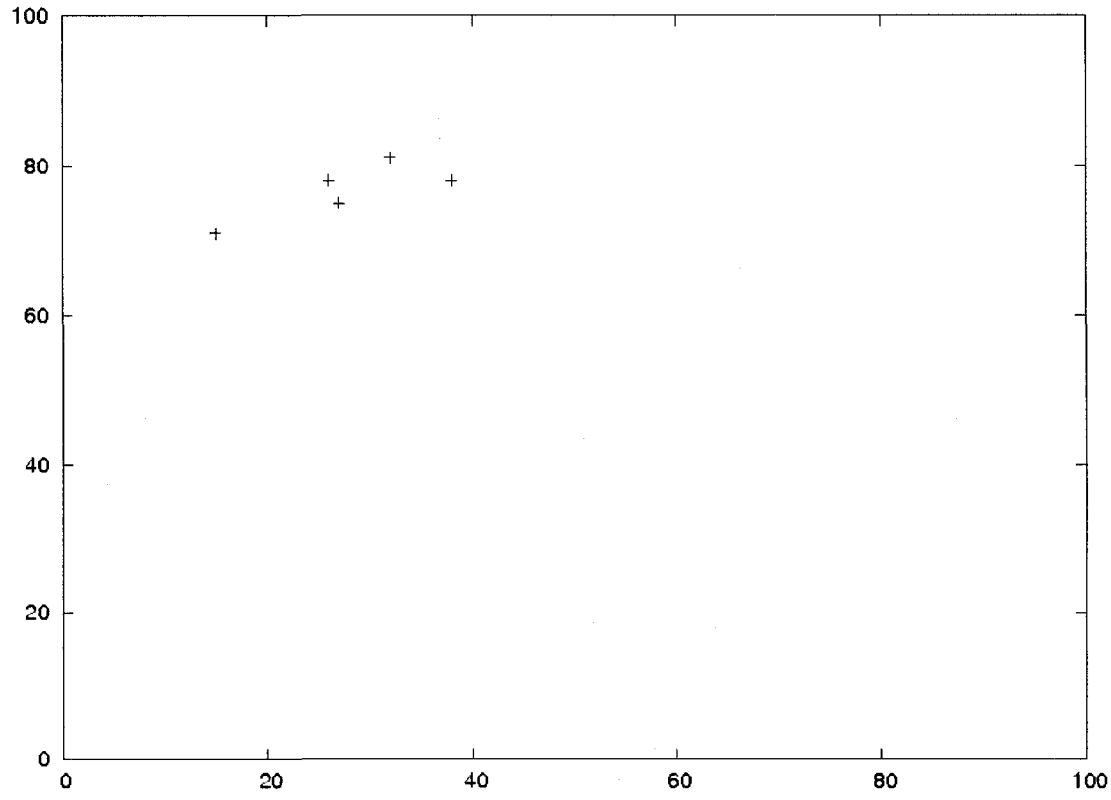

Figure 6.6 – Pourcentage d'amélioration de la valeur de l'objectif en fonction du pourcentage de détérioration du temps

Remarquons que lors de la résolution du problème V par ADBS-D la solution obtenue est à environ 66% de la valeur optimale alors qu'on a environ 28% des temps de calculs. De plus, la courbe représentative (figure 6.5) montre clairement que l'algorithme ADBS-D est bien plus rapide que les autres algorithmes.

Dans la figure 6.6, on a illustré le pourcentage d'amélioration de l'objectif, lors de l'application de ADBS-D, pour les différents problèmes testés ci-dessus. Si on compare cette figure avec 5.9, on peut remarquer une amélioration dans la qualité des solutions.

6.2.3.6 Justification du choix des paramètres de résolution à l'intérieur de l'algorithme de génération de colonnes

Dans les différents problèmes tests qu'on a considérés ci-haut, on a choisi de commencer par appliquer APD-ND pour les premières itérations de GC, puis on applique, dans une première phase, l'algorithme ADBS-D à partir de la 40ième en considérant un seul arc lors de l'application de MAJ, puis on passe à une deuxième phase de l'algorithme, en appliquant MAJ à 2 arcs par noeud considéré. Pour justifier ce choix, on s'est appuyé sur un exemple numérique qui est un problème semblable aux précédents. On illustre les résultats relatifs à ce problème par les deux figures 6.7 (la deuxième est un agrandissement de la première n'incluant pas la courbe de APD). En effet, ces deux figures représentent une comparaison de la méthode ADBS-D telle qu'on l'a proposé dans les problèmes tests précédents. On d'abord appliqué l'algorithme dès les premières itérations (ADBS-DI et ADBS-DII sur la figure 6.7). Ainsi, on a commencé par APD-ND puis a appliqué la méthode ADBS-D à la 40ième itération représenté par le point 1 (ADBS-DIII sur la figure 6.7) sur la figure et on passe à la deuxième phase de la méthode à l'itération 70 représentée par le point 2 (ADBS-D sur la figure 6.7).

Sur ces figures (6.7), on peut remarquer un léger ralentissement de l'algorithme APD-ND aux alentours de la 40ème itération (environ à 350 sec après le commencement). De plus, on remarque qu'il y a eu un ralentissement de l'algorithme proche de l'itération 70. C'est pour cela qu'on a décidé d'introduire la deuxième phase de l'algorithme à ce niveau là. De plus, pour le même problème on a fait des tests sur l'algorithme appliqué, à partir de la 50ème et 60ème itération de l'algorithme de génération de colonnes. C'est ainsi qu'on conclut que notre choix initial de la valeur des paramètres est le plus adéquat.

Remarquons que, pour de futures recherches, on pourrait automatiser l'ajustement de ces paramètres en changeant à chaque fois de méthode lorsque celle-ci ne montre plus assez d'améliorations.

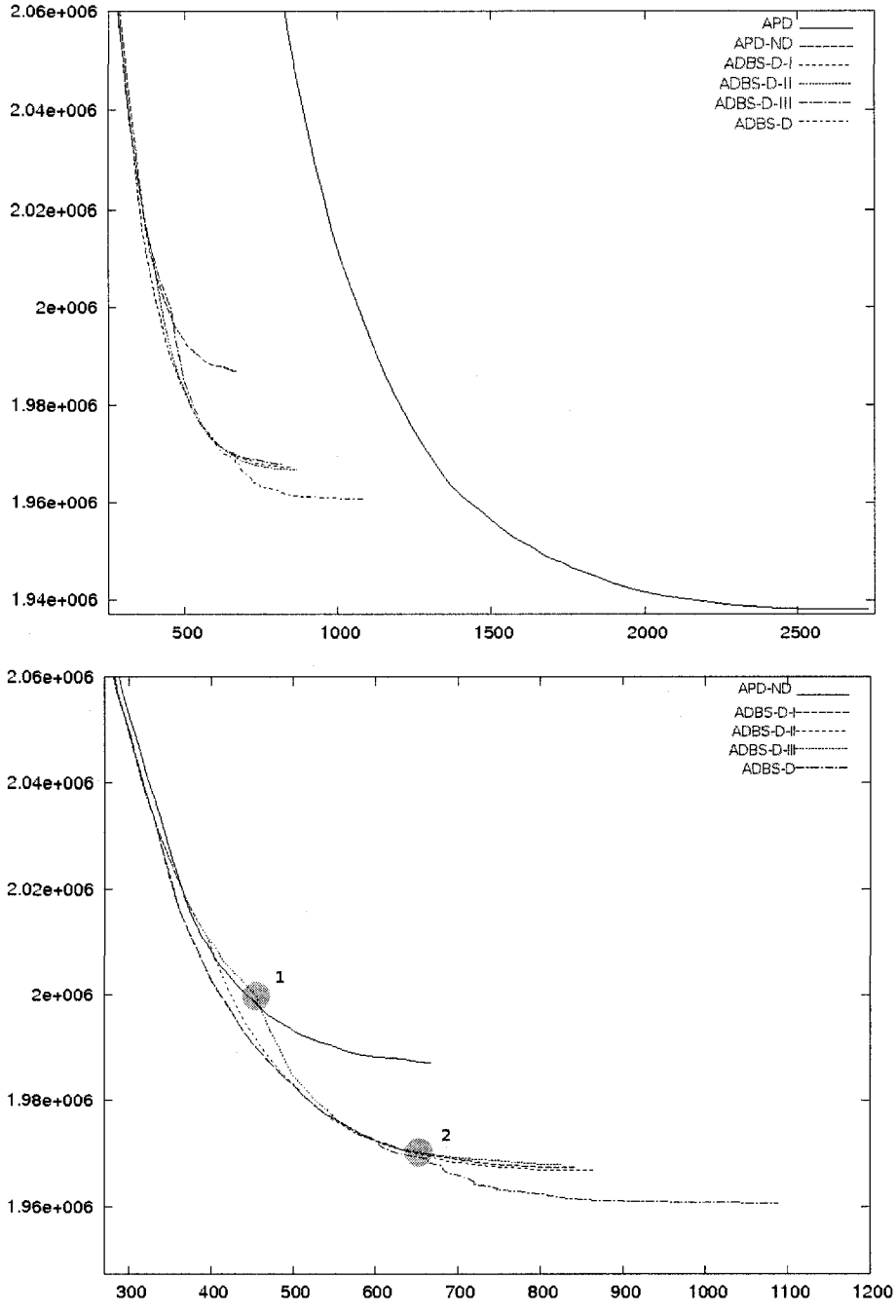

Figure 6.7 – Problème V : Justification comparative du choix des paramètres de résolution par ADBS-D (valeur de l'objectif en fonction du temps)

Ainsi, les résultats obtenus constituent une amélioration considérable par rapport à l'algorithme ADBS (en moyenne 40%) et par rapport à l'algorithme ADBS-I (en moyenne 25%). De plus, ceci n'a pas trop affecté les temps de calculs.

6.3 Méthode d'accélération en utilisant la relaxation lagrangienne

Dans cette partie, on se propose d'utiliser l'approche lagrangienne pour établir une borne inférieure lors de la résolution du problème par ADBS et ses variantes. Ceci permettra d'évaluer approximativement l'écart entre la solution courante et la solution optimale et se créer ainsi des mécanismes et des critères pour arrêter la résolution au besoin afin d'éviter une suite d'opérations où la borne supérieure ne s'améliore pas.

Dans ce qui suit, on introduit tout d'abord une première borne inférieure en relâchant toutes les bornes supérieures sur les fenêtres de temps du problème et en dominant uniquement sur les coûts. On définit l'ensemble $\bar{E}_i$ des *étiquettes relâchées* en i, pour tout $i \in N$ par l'algorithme 6.3.

Algorithme 6.3 Construction de $\bar{E}_i$

L'ensemble des noeuds de G étant ordonné topologiquement : $N = \{0, 1, \ldots n\}$.
$\bar{E}_0 = \{\bar{e}_0\}$, $\bar{e}_0 = (0, 0)$.
pour tout $j = 1, \ldots, n$ **faire**
  $\bar{E}_1, \ldots, \bar{E}_{j-1}$ ayant déjà été construits, on définit $\bar{E}_j$ comme suit :
  pour tout i tel que $(i, j) \in A$ et $i \in \{0, 1 \ldots, j-1\}$ **faire**
    pour tout $\bar{e}_i^{\mathcal{P}} \in \bar{E}_i$ de valeur $(C_i^{\mathcal{P}}, T_i^{\mathcal{P}})$ **faire**
      Pour $\mathcal{Q} = \mathcal{P} \cup \{(i, j)\}$.
      on ajoute une nouvelle étiquette $e_j^{\mathcal{Q}}$ à $\bar{E}_j$,de valeur $(C_j^{\mathcal{Q}}, T_j^{\mathcal{Q}})$, $C_j^{\mathcal{Q}} = C_i^{\mathcal{P}} + c_{ij}$
      et $T_j^{\mathcal{Q}} = \max\{a_j, T_i^{\mathcal{P}} + t_{ij}\}$.
  On élimine les étiquettes dominées de $\bar{E}_j$.

Définition 9 *On définit le réseau de l'espace des états sur le problème relâché noté*

$\bar{\mathcal{G}} = (\bar{\mathcal{N}}, \bar{\mathcal{A}})$ *comme suit :* $\bar{\mathcal{N}} = \cup_{i \in N} \bar{E}_i$ *est l'ensemble des noeuds de* $\bar{\mathcal{G}}$ *et* $\bar{\mathcal{A}} = \{(e_i^{\mathcal{P}}, e_j^{\mathcal{Q}}) | (i,j) \in A, e_i^{\mathcal{P}} \in \bar{E}_i, \mathcal{Q} = \mathcal{P} \cup (i,j) \text{ et } \mathrm{e_j^{\mathcal{Q}}} \in \bar{\mathrm{E}}_\mathrm{j}\}$.

Lemme 1 *La résolution du problème de plus court chemin avec fenêtres de temps sur le réseau* $\bar{\mathcal{G}}$ *fournit une borne inférieure pour le problème PCC-FT. Désignons par* $\bar{C}$ *cette valeur.*

6.3.1 Réseau d'espace des états lagrangien

On se propose dans ce qui suit de définir un réseau dans lequel on résout un problème de plus court chemin en dominant sur les coûts lagrangiens.

Considérons le cas de la relaxation lagrangienne où on a un unique multiplicateur de Lagrange u, associé à la contrainte sur la fenêtre de temps du noeud destination d, qu'on note par $[a_d, b_d]$. Ce cas est un simple sous-cas de la relaxation lagrangienne étudiée aux chapitre 4. Soit u_k un multiplicateur de Lagrange à l'itération k. Le coût lagrangien, en un noeud donné i, faisant partie du plus court chemin fourni par la résolution du problème lagrangien à l'itération précédente, est ainsi calculé comme suit : $C_i(u_k) = C_i + u_k^d(T_d^k - b_d)$, T_d^k étant la consommation de temps au noeud destination d, à l'itération k. Ainsi on définit $\bar{\mathcal{G}}^{Lag}(u)$ le réseau défini de la même façon que $\bar{\mathcal{G}}$ mais en dominant sur les coûts lagrangiens associés au multiplicateur u_k selon l'algorithme 6.4 qui est semblable à 5.1.

Rappelons qu'on obtient la valeur du multiplicateur de Lagrange u_{k+1} à l'itération $k+1$ en utilisant un algorithme de sous-gradients qui donne ce multiplicateur selon la formule : $u_{k+1} = u_k + p_k(T_d^k - b_d)$, p_k étant une suite de pas. Ainsi, on peut construire itérativement une suite de réseaux $\bar{\mathcal{G}}^{Lag}(u_k)$, chacun dépend du multiplicateur u_k calculé à partir de l'itération précédente. Remarquons, qu'à chaque itération k, les pénalisations par les sous-gradients ne sont appliquées que pour les chemins optimaux de l'itération $k-1$.

Algorithme 6.4 Construction de $E_i(u_k)$

L'ensemble des noeuds de G étant ordonné topologiquement : $N = \{0, 1, \ldots n\}$.
$E_0(u_k) = \{e_0\}$, $e_0 = (0, 0)$.
pour tout $j = 1, \ldots, n$ **faire**
  $E_1(u_k), \ldots, E_{j-1}(u_k)$ ayant déjà été construits, on définit $E_j(u_k)$ comme suit :
  pour tout i tel que $(i, j) \in A$ et $i \in \{0, 1 \ldots, j-1\}$ **faire**
    pour tout $e_i^{\mathcal{P}} \in E_i(u_k)$ de valeur $(C_i^{\mathcal{P}}, T_i^{\mathcal{P}})$. Soit $\mathcal{Q} = \mathcal{P} \cup \{(i, j)\}$ **faire**
      on ajoute une nouvelle étiquette $e_j^{\mathcal{Q}}$ à E_j,de valeur $(C_j^{\mathcal{Q}}, T_j^{\mathcal{Q}})$, $C_j^{\mathcal{Q}} = C_i^{\mathcal{P}} + c_{ij}$ et $T_j^{\mathcal{Q}} = \max\{a_j, T_i^{\mathcal{P}}\} + t_{ij}$.
  On calcule les coût lagrangiens de toutes les étiquettes en j données par $C_j(u_k) = C_j + u_k(T_d^k - b_d)$.
  On enlève les étiquettes dominées sur le coût lagrangien.

Théorème 4 *Si u est un multiplicateur de Lagrange u tel que, si $(C(u), T(u))$ est une étiquette qui vérifie $T(u) > b_d$, alors $C(u)$ est une borne inférieure pour le PCC-FT. De plus, elle constitue une meilleure borne que celle du lemme 1.*

Preuve :

Soit (C^*, T^*) l'étiquette correspondant à la solution optimale du PCC-FT. Par définition de $\bar{\mathcal{G}}^{Lag}(u)$, l'étiquette correspondant au plus court chemin sur le réseau lagrangien $\bar{\mathcal{G}}^{Lag}(u)$ est non réalisable possède un coût lagrangien plus petit ou égal à C^*. Autrement dit, si on désigne par $(C(u), T(u))$ une telle étiquette, on a alors $C^* + u(T^* - b_d) \geq C(u) + u(T(u) - b_d)$. Or, $T^* - b_d \leq 0$ (car réalisable) et $T(u) - b_d > 0$, donc $C^* \geq C(u)$, et par la suite $C(u)$ constitue une borne inférieure pour le problème PCC-FT. De plus $C(u) > C$, C étant la borne donnée par le lemme 1, car il existe toujours u tel que $C(u) = C + u(T - b_d)$.

6.3.2 Généralisation de l'approche lagrangienne

On se propose dans cette section d'établir une méthode qui garantit une borne inférieure lagrangienne en utilisant un multiplicateur de Lagrange u_i pour chaque noeud i.

Théorème 5 *Il existe $\mathcal{T}$ un arbre des plus courts chemins, trouvé par un algorithme lagrangien, tel que, en chaque noeud de cet arbre, $T_i^{\mathcal{T}} \geq b_i$ ou $T_i^{\mathcal{T}} < b_i$ et $u_i = 0$.*

Preuve :

Un arbre de plus courts chemins, trouvé par un algorithme lagrangien, peut être obtenu par l'ajustement des multiplicateurs avec un sous-gradient et un pas donné par une suite décroissante. Or au noeud i, la solution optimale u_i^* et $(C_i^{\mathcal{T}}, T_i^{\mathcal{T}})$ vérifie les propriétés suivantes :

Propriété 1 $C_i^{\mathcal{T}} + u_i^*(T_i^{\mathcal{T}} - b_i) = \max_{u_i} \min_{\mathcal{P}} \{C_i^{\mathcal{P}} + u_i(T_i^{\mathcal{P}} - b_i)\}$, *$\mathcal{P}$ étant un chemin de o à i.*

Propriété 2 *Si $T_i^{\mathcal{T}} < b_i$ et $u_i^* > 0$, alors il existe un chemin $\mathcal{Q}$ de o à i tel que $C_i^{\mathcal{T}} + u_i^*(T_i^{\mathcal{T}} - b_i) = C_i^{\mathcal{Q}} + u_i^*(T_i^{\mathcal{Q}} - b_i)$ et $T_i^{\mathcal{Q}} \geq b_i$.*

Preuve de la propriété 2 :

Si la propriété n'était pas vraie, il existerait $\Delta > 0$ tel qu'on peut remplacer u_i^* par $u_i^* - \Delta$ et le sous-chemin de o à i dans $\mathcal{T}$ reste de coût minimum pour la fonction $C_i + (u_i^* - \Delta)(T_i - b_i)$.

Comme $C_i^{\mathcal{T}} + u_i^*(T_i^{\mathcal{T}} - b_i) > C_i^{\mathcal{T}} + (u_i^* - \Delta)(T_i^{\mathcal{T}} - b_i)$, il n'y a pas de contradiction avec la propriété 1. Donc la propriété 2 est vraie.

En remplaçant le chemin de o à i de $\mathcal{T}$ par $\mathcal{Q}$, on obtient une solution optimale qui satisfait le théorème. Ce qui prouve le théorème.

Remarque 14 *Pour assurer que l'arbre $\mathcal{T}$ obtenu avec l'ajustement des multiplicateurs satisfasse le théorème 5, il suffit, au moment de l'arrêt de l'algorithme, de faire une dernière itération en diminuant légèrement les u_i.*

Théorème 6 *Soit $\mathcal{S}$ l'arbre des plus courts chemins obtenu par l'algorithme APD. Et soit $(C_i^{\mathcal{S}}, T_i^{\mathcal{S}})$ l'étiquette optimale au noeud i. Soit $\mathcal{T}$ un arbre de plus courts chemins, obtenu par un algorithme lagrangien, satisfaisant le théorème 5, alors $C_i^{\mathcal{S}} \geq C_i^{\mathcal{T}}$ pour tout i.*

Preuve :
Comme $\mathcal{T}$ est une solution qui minimise le coût lagrangien avec les multiplicateurs u_j^*, alors $C_i^{\mathcal{S}} + u_i^*(T_i^{\mathcal{S}} - b_i) \geq C_i^{\mathcal{T}} + u_i^*(T_i^{\mathcal{T}} - b_i)$. Ainsi $C_i^{\mathcal{S}} \geq C_i^{\mathcal{T}} + u_i^*(T_i^{\mathcal{T}} - b_i) - u_i^*(T_i^{\mathcal{S}} - b_i)$. Le terme $u_i^*(T_i^{\mathcal{T}} - b_i)$ étant positif puisque $T_i^{\mathcal{T}} - b_i \geq 0$ et $u_i^* \geq 0$. Le terme $u_i^*(T_i^{\mathcal{S}} - b_i)$ étant négatif puisque $T_i^{\mathcal{S}} - b_i \leq 0$ et $u_i^* \geq 0$. Donc $C_i^{\mathcal{S}} \geq C_i^{\mathcal{T}}$.

Le théorème 6 peut nous fournir une méthode d'accélération efficace comme on le montre dans la section 6.3.3, qu'on pourra exploiter en l'intégrant à l'algorithme d'amélioration dynamique de la borne supérieure. De plus, on pourra à l'aide de ce théorème maintenir en même temps des bornes inférieure et supérieure et essayer de réduire l'écart entre ces deux valeurs. Ceci nous permet de fournir un bon critère d'arrêt heuristique lors de la résolution.

Avec les mêmes notations que les algorithmes de la section précédente, on définit l'algorithme d'amélioration dynamique de la borne supérieure en utilisant la relaxation lagrangienne.

6.3.3 Algorithme d'amélioration dynamique de la borne supérieure en utilisant la relaxation lagrangienne

Cet algorithme est semblable à l'algorithme 5.4 à la différence que ce dernier vérifie à chaque itération l'écart entre la solution courante et la borne inférieure lagrangienne. Si cet écart est plus grand que le critère ε, on applique l'opération MAJ. Sinon on passe à l'opération suivante.

Algorithme 6.5 Algorithme lagrangien d'amélioration de la borne supérieure

Soit $\varepsilon \geq 0$ un critère d'arrêt.
répéter
  pour $j = n, n-1, \ldots, 0$ **faire**
    pour tout $e_j^{\mathcal{B}} \in B_j$ **faire**
      pour tout E_j^{λ} tel que $e_j^{\mathcal{B}}$ est relativement non réalisable par rapport à E_j^{λ} **faire**
        Soit $i \in N$ tel que $(i,j) \in A$ et qui précède j sur le chemin $\mathcal{B}$.
        Calculer $\theta = |C_i(u_i) - C_i|$.
        si $\theta > \varepsilon$ **alors**
          Appliquer l'opération MAJ$(i, E_j^{\lambda}, e_j^{\mathcal{B}})$.
    Ajouter les nouveaux noeuds relativement non réalisables aux ensembles B_i
    Appliquer l'algorithme de marquage lagrangien (décrit ci-dessous) sur le nouveau réseau.
jusqu'à $B_i = \emptyset$.

À l'intérieur de l'algorithme 6.5, on fait appel à l'algorithme 6.6 qui suit qui est un algorithme de marquage semblable à l'algorithme 5.5 en utilisant les coûts lagrangiens.

Algorithme 6.6 Algorithme de marquage lagrangien

Soit u le multiplicateur de Lagrange courant.
pour $i = 0, 1, \ldots, n$ **faire**
  pour tout $e_i^{\mathcal{P}} \in \bar{E}_i$ **faire**
    pour tout E^{λ} sous-ensemble d'étiquettes en i **faire**
      Enlever les noeuds dominés sur le coût, de E_i^{λ} et de P_i.
      Enlever les noeuds dominés sur les coûts lagrangiens de $\bar{E}_i(u)$.
      Trier les noeuds de E_i^{λ} selon le coût de leurs étiquettes.
      Mettre des arcs $(e_i^{\mathcal{P}_1}, e_i^{\mathcal{P}_2}), (e_i^{\mathcal{P}_2}, e_i^{\mathcal{P}_3}), \ldots$ entre les noeuds de E_i^{λ}.
      Soit $e_i^{\mathcal{P}}$ le noeud qui correspond à l'étiquette moindre coût de $\bar{E}_i(u)$, ajouter $e_i^{\mathcal{P}}$ à P_i..
      Prolonger de de la façon suivante $e_i^{\mathcal{P}}$:
      pour tout j tel que $(i, j) \in A$ **faire**
        Soit $\mathcal{Q} = \mathcal{P} \cup \{(i, j)\}$, créer $e_j^{\mathcal{Q}} = (C_j^{\mathcal{Q}}, T_j^{\mathcal{Q}})$ tel que $C_j^{\mathcal{Q}} = C_i^{\mathcal{P}} + c_{ij}$ et $T_j^{\mathcal{Q}} = \max\{, a_j, T_i^{\mathcal{P}} + t_{ij}\}$ si $T_j^{\mathcal{Q}} \leq b_j$.
        Créer $e_j^{\mathcal{Q}}(u) = (C_j^{\mathcal{Q}}(u), T_j^{\mathcal{Q}}(u))$ tel que $C_j^{\mathcal{Q}}(u) = C_i^{\mathcal{P}} + c_{ij}$ où $T_j^{\mathcal{Q}} = \max\{a_j, T_i^{\mathcal{P}} + t_{ij}\}$ si $T_j^{\mathcal{Q}} \leq b_j$.
        Enlever $e_i^{\mathcal{P}}$ de P_i et le ajouter à M_i.

Remarque 15 *Remarquons qu'on peut combiner les deux algorithmes en maintenant deux critères d'arrêt. Soit un sur la borne inférieure lagrangienne et un autre sur les coûts réduits des arcs, ce qui constitue un algorithme qui peut accélérer encore plus la résolution et éventuellement arrêter la résolution sans avoir à encourir d'énormes calculs.*

CHAPITRE 7

GÉNÉRALISATION DE L'ALGORITHME ADBS

Le but initial de ce travail était l'étude des PCC-CR. Étant donné que les algorithmes ADBS, ADBS-I et ADBS-D ont montré qu'on peut trouver rapidement de bonnes solutions pour les problèmes PCC-FT, on envisagera de généraliser ces différents algorithmes. Notre approche devrait apporter des réductions de temps par rapport à la résolution optimale par l'algorithme APD dans le cas des PCC-CR avec multiples ressources.

L'algorithme ADBS-D pour le problème PCC-FT divise un noeud en deux lors de l'application d'une opération MAJ. Nous proposerons ici une généralisation pour le problème PCC-CR qui consiste à diviser un noeud en $R + 1$ (R est le nombre de ressources) lors d'une opération MAJ. Dans cette généralisation on n'ajoute pas, à chaque opération MAJ, un nombre de noeuds qui augmente exponentiellement en fonction de R. Ceci est avantageux par rapport à l'algorithme APD dans lequel le nombre d'étiquettes augmente exponentiellement avec le nombre de ressources.

De plus, jusqu'à maintenant, on a uniquement considéré les problèmes ayant des fonctions d'extension linéaires. il sera intéressant d'explorer les cas où les fonctions d'extension ne sont pas nécessairement linéaires.

7.1 Généralisation pour les PCC-CR

Dans ce travail, on a exploré l'approche de résolution des PCC-FT par les algorithmes du type ADBS dans le cas où les fonctions d'extension sont linéaires non décroissantes. On se propose alors de revoir l'algorithme ADBS pour PCC-FT et

expliquer les modifications qu'il faudra apporter à certaines étapes de l'algorithme afin de l'adapter pour PCC-CR.

Tout d'abord, on propose de construire l'ensemble des étiquettes E_i en i selon un algorithme semblable à l'algorithme 5.1 de construction des E_i, mais en tenant compte de la valeur de toutes les ressources. Par la suite, avant l'application de l'algorithme ADBS pour PCC-FT, on a trié l'ensemble des étiquettes E_i en i, en ordre décroissant de leurs coûts et, par conséquent, par ordre croissant de leurs consommations de ressources. Ce tri, dans le cas où l'espace des états est de dimension plus grande, n'a pas les mêmes propriétés. En effet, dans le cas où on a R ressources ($R > 1$), lorsqu'on trie l'ensemble des étiquettes en ordre croissant du coût, on n'a aucun ordre sur les consommations des ressources. Supposons dans ce qui suit que E_i est trié selon les coûts.

Par la suite, on définit une partition d'un noeud et de la liste de ses étiquettes de la façon qui suit. Soit E_j un ensemble de noeuds en j et soit $e_j^{\mathcal{B}} \in B_j$ un noeud relativement non réalisable, par rapport à la ressource r c'est-à-dire qui viole la contrainte relative à la ressource r : $T_j^{\mathcal{B}} > b_j^r$ (b_j^r étant la borne supérieure relative à l'ensemble E_j). Notons que $e_j^{\mathcal{B}} \in B_j$ peut violer plusieurs contraintes de ressources en même temps, soit $r_1, ..., r_k$ ces ressources. Soit $g_i^r = b_j^r - t_{ij}^r \ \forall r = r_1, ..., r_k$. Par la suite, on parcourt l'ensemble des étiquettes de E_i, qui a permis de générer $e_j^{\mathcal{B}}$, en ordre croissant de leurs coûts. Et on subdivise l'ensemble E_i en $k+1$ sous-ensembles :

- E_i^0 étant l'ensemble des étiquettes réalisables en i. Désignons par $e_i^{\mathcal{P}_0}$ l'élément de moindre coût de ce sous-ensemble.
- E_i^1 étant l'ensemble des étiquettes qui ne satisfont pas la première contrainte de ressource r_1 uniquement, c'est-à-dire que $T_i^{r_1} > g_i^{r_1}$ et $T_i^r \leq g_i^r$ pour $r \neq r_1$. Désignons par $e_i^{\mathcal{P}_1}$ l'élément de moindre coût de ce sous-ensemble.
- E_i^2 étant l'ensemble des étiquettes qui ne satisfont pas la contrainte de ressource r_2 et peut-être aussi la contrainte de ressource r_1 et sans violer les autres contraintes de ressources, c'est-à-dire $T_i^{r_2} > g_i^{r_2}$, $T_i^{r_1}$ est quelconque et

$T_i^r \leq g_i^r$ pour $r \neq r_1$ et $r \neq r_2$. Désignons par $e_i^{\mathcal{P}_2}$ l'élément de moindre coût de ce sous-ensemble.

- E_i^3 étant l'ensemble des étiquettes qui ne satisfont pas la contrainte de ressource r_3 et peut-être aussi r_1 ou r_2 et sans violer les autres contraintes de ressources, c'est-à-dire $T_i^{r_3} > g_i^{r_3}$, $T_i^{r_1}$ et $T_i^{r_2}$ sont quelconques et $T_i^r \leq g_i^r$ pour $r \neq r_1$, $r \neq r_2$ et $r \neq r_3$. Désignons par $e_i^{\mathcal{P}_3}$ l'élément de moindre coût de ce sous-ensemble.
- ...
- E_i^k étant l'ensemble des étiquettes qui ne satisfont pas la contrainte de ressource r_k, et peut-être aussi les contraintes de ressources de $r = r_1, r_2, ..., r_{k-1}$. Désignons par $e_i^{\mathcal{P}_k}$ l'élément de moindre coût de ce sous-ensemble.

Notons que les sous-ensembles $E_i^0, E_i^1, ..., E_i^k$ sont eux aussi définis par des fenêtres de ressources et peuvent être divisés lors de nouvelles opérations MAJ.

Le reste des étapes de l'algorithme ADBS (aussi ADBS-I et ADBS-D) restent les mêmes dans le cas de plusieurs ressources. Ainsi, on prolonge en j uniquement l'étiquette $e^{\mathcal{P}_0}$ et on prolonge vers les autres successeurs de i les étiquettes $e_i^{\mathcal{P}_0}, e_i^{\mathcal{P}_1}, ..., e_i^{\mathcal{P}_k}$.

Remarquons que la complexité de l'algorithme ADBS pour les PCC-CR n'augmente pas exponentiellement par rapport à l'algorithme ADBS pour les PCC-FT. En effet, les parcours des étiquettes de E_i se font en des temps similaires. De plus, on a une subdivision d'un noeud en au plus $R + 1$ sous-ensembles lors d'une opération MAJ pour le cas des PCC-CR. Donc, la généralisation au PCC-CR n'ajoute pas un nombre d'étiquettes qui augmente exponentiellement avec le nombre de ressources. Ainsi, la complexité de l'algorithme ADBS est a priori bien inférieure à celle de APD qui croît exponentiellement avec le nombre de ressources.

7.2 Généralisation aux fonctions d'extension non linéaires

Comme on a vu dans l'introduction de ce travail, les fonctions d'extension ne sont pas linéaires pour un bon nombre d'applications. On se propose dans cette partie d'explorer les modifications qu'on devrait apporter aux algorithmes ADBS et ses variantes afin de les adapter aux problèmes ayant des extensions non linéaires dans le cadre des PCC-FT et, plus généralement, des PCC-CR.

Dans le cas des fonctions d'extension non linéaires, on appliquera la même idée de partition des étiquettes en un noeud i. Toutefois, pour tout $(i,j) \in A$, les critères de partition ne s'appliquent pas à la valeur des ressources en i. Ils s'appliquent sur la valeur des étiquettes en j. Ainsi, il faut appliquer les fonctions d'extension à chaque étiquette en i pour obtenir les étiquettes en j et vérifier si elles satisfont les contraintes de ressources. En d'autres mots, pour chaque étiquette $e_i \in E_i$, on la prolonge en j et on vérifie les contraintes de ressources : $f_{ij}^r(T_i^r) \leq b_i^r$. C'est ainsi qu'on forme les sous-ensembles des noeuds qui ne violent aucune contrainte de ressource, violent la première contrainte de ressource, violent la deuxième contrainte de ressource, etc. Ces ensembles $E_i^0, E_i^1, ..., E_i^k$ sont définis de façon semblable à ceux de la section précédente :

- E_i^0 étant l'ensemble des étiquettes qui vérifient $f_{ij}^r(T_i) \leq b_j^r$ pour tout r. Désignons par $e_i^{\mathcal{P}_0}$ l'élément de moindre coût de ce sous-ensemble.
- E_i^1 étant l'ensemble des étiquettes telles que $f_{ij}^{r_1}(T_i^{r_1}) > b_i^{r_1}$ et $f_{ij}^r(T_i^r) \leq b_i^r$ pour tout $r \neq r_1$.
- E_i^2 étant l'ensemble des étiquettes telles que $f_{ij}^{r_2}(T_i^{r_2}) > b_i^{r_2}$, $T_i^{r_1}$ est quelconque et $f_{ij}^r(T_i^r) \leq g_i^r$ pour $r \neq r_1$ et $r \neq r_2$.
- E_i^3 étant l'ensemble des étiquettes telles que $f_{ij}^{r_3}(T_i^{r_3}) > b_i^{r_3}$, $T_i^{r_1}$ et $T_i^{r_2}$ sont quelconques et $f_{ij}^r(T_i^r) \leq b_i^r$ pour $r \neq r_1$, $r \neq r_2$ et $r \neq r_3$.

- ...
- E_i^k étant l'ensemble des étiquettes telles que $f_{ij}^{r_k}(T_i^{r_k}) > b_i^{r_k}$.

Remarquons que la complexité augmente de façon constante avec le nombre d'étiquettes par noeud. En effet, en chaque noeud, on doit parcourir toutes les étiquettes, en plus d'appliquer les fonctions d'extension afin de déterminer lesquelles sont réalisables, et ceci pour chacune des ressources.

CHAPITRE 8

CONCLUSION

Le but initial de ce travail était de trouver rapidement de bonnes solutions pour le problème de plus court chemin avec contraintes de ressources à l'intérieur d'un algorithme de génération de colonnes. Malheureusement, l'approche de projection proposée par Nagih et Soumis (2006) s'est avérée infructueuse. Tant sur le plan théorique que sur le plan numérique, comme on l'a prouvé dans les chapitres 5 et 6, cette approche peut donner, de façon aléatoire, de bons résultats comme elle peut donner aussi de mauvais résultats. En effet, le but initial de l'algorithme de Nagih et Soumis est d'obtenir des solutions dont le coût est meilleur que la résolution utilisant l'algorithme APD-ND ; le coût des solutions fournies en utilisant N-S doit être le plus proche que possible du coût fourni en utilisant l'algorithme APD. Les temps de résolution ne doivent pas être beaucoup plus grands que lorsqu'on utilise APD-ND pour résoudre les sous-problèmes. On a montré que les solutions fournies lorsqu'on utilise N-S peuvent être moins bonnes que les solutions fournies lorsqu'on utilise APD-ND, même que cette variation peut être très grande. On a appuyé nos résultats théoriques par des résultats numériques prouvant le comportement assez aléatoire de la méthode N-S.

En somme, on a investi beaucoup d'efforts et de longues années de recherches pour une recherche qui est le moins qu'on puisse dire décevante, surtout qu'elle a été basée sur des travaux antérieurs qui semblaient a priori très intéressants.

Cependant, cette approche nous a informé sur d'éventuelles applications de ces projections et sur le comportement de la relaxation lagrangienne à l'intérieur des algorithmes de résolution des problèmes de plus court chemin avec contraintes de

ressources. On a pu recycler cette approche par la suite pour créer une heuristique afin d'accélérer la méthode qu'on a proposé en raffinant la borne supérieure lors de la résolution et choisir un bon critère d'arrêt heuristique.

Ainsi on a décidé, pour la suite de la thèse, de s'orienter vers d'autres approches. Dans la deuxième partie de ce travail, la nouvelle approche qu'on a proposé a donné des résultats préliminaires satisfaisants. Dans cette partie, on a introduit une nouvelle méthode de type primal qui permet d'améliorer graduellement la borne supérieure au fil des itérations. Cette approche permet d'introduire progressivement et de façon dynamique des nouvelles solutions qui peuvent être de plus en plus bonnes et, par la suite, d'améliorer la borne. Elle nous a permis de faire des gains considérables par rapport à la méthode statique, des gains qui peuvent varier entre 58% et 65% de l'écart entre l'algorithme APD et APD-ND pour une légère augmentation des temps de calcul et ceci en utilisant différentes variantes heuristiques de notre méthode. Par la suite, on a proposé des améliorations de notre méthode en considérant des opérations de mises à jour plus ciblées et qui garantissent une amélioration de la borne supérieure à chaque itération et ceci en utilisant les variables duales. Cette deuxième amélioration a permis des gains qui peuvent varier entre 72% et 82% de l'écart entre l'algorithme APD et APD-ND pour une légère augmentation des temps de calcul. Aussi, on a montré que la méthode peut offrir bien plus d'améliorations si on étudiait les autres aspects et paramètres. En somme, la méthode qu'on a proposé a l'avantage d'être très flexible. On peut choisir plusieurs paramètres de résolution à l'avance, selon nos ressources informatiques ou aussi selon les temps de calculs voulus.

Par la suite, on s'est proposé de généraliser théoriquement notre méthode au cas des PCC-CR ayant des fonctions d'extension quelconques. On a montré que notre approche peut s'étendre au PCC-CR quelconques en utilisant des approches semblables à celles qu'on a utilisé pour le PCC-FT. Ainsi, on a montré qu'on peut appliquer les opérations MAJ sans pour autant augmenter la taille du réseau de façon exponentielle. En effet, à chaque fois qu'on applique l'opération MAJ, on crée au plus

k nouveaux sous-ensembles de noeuds, tel que $k \leq R$, R étant le nombre total des ressources. Bien sûr, le théorème 3 selon lequel on privilégie les mises à jour qui impliquent une amélioration directe de l'objectif reste un excellent critère pour les choix des arcs entrants.

De plus, on a montré que l'extension de notre approche aux cas où des PCC-CR avec fonctions d'extension non linéaires peut se faire en utilisant des approches semblables aux cas linéaires sans pour autant augmenter de façon exponentielle la taille des réseaux. Ces extensions peuvent donner de bons résultats pour de futures recherches impliquant des applications encore plus complexes que celles qu'on a traité.

Il faudra aussi faire une étude expérimentale sur l'algorithme lagrangien qui fournit une borne inférieure, ce qui va permettre de mesurer, à chaque itération, l'écart entre la borne inférieure lagrangienne et la borne supérieure. Ceci aidera à avoir un bon critère d'arrêt et d'éviter des calculs qui n'améliorent que de peu la borne supérieure. Éventuellement, on pourra développer des méthodes plus raffinées qui utiliseront ces deux approches en même temps.

Ainsi cette méthode peut poser des défis intéressants autant sur le plan théorique que sur le plan algorithmique et expérimental et ça ouvre de bons volets pour de futures recherches.

BIBLIOGRAPHIE

ABDUL-RAZAK, T. S. et POTTS, C. N. (1988). Dynamic programming state-space relaxation for single-machine scheduling. *Journal of Operational Research Society.* 39:2. P. 141–152.

ANEJA, Y. P., AGGARWAL, V. et NAIR, K. P. K. (1981). Shortest chain subject to side constraints. *Networks.* 13. P. 295–302.

BARNHART, C., JOHNSON, E., NEMHAUSER, G., SAVELSBERGH, M. et VANCE, P. (1998). Branch-and-price : Column generation for solving huge integer programs. *Operations Research.* 46. P. 316–329.

BEASLEY, J. et CHRISTOFIDES, N. (1989). An algorithm for the resource constrained shortest path problem. *Networks.* 19. P. 379–394.

CARLYLE, M. et WOOD, R. K. (2003). Lagrangian relaxation and enumeration solving for constrained shortest path problems. *Proceedings of the 38th Annual ORSNZ Conference, Cambridge, Massachusetts, USA.*

CHEN, Y. L. et YANG, H. H. (2004). Finding the $k-$shortest paths in a time-window network. *Computers and Operations Research.* 31. P. 499–513.

CHRISTOFIDES, N., MINGOZZI, A. et P.TOTH. (1981). Exact algorithm for the vehicle routing problem based on spanning tree and shortest path relaxations. *Mathematical Programming.* 20. P. 255–282.

DANTZIG, G. et WOLFE, P. (1960). Decomposition principle for linear programs. *Operations Research.* 8. P. 101–111.

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. P. 57–93.

DESAULNIERS, G., HADJAR, A. et LESSARD, F. (2006). Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Cahiers du GEARD G-2006-45, HEC Montréal, Canada.*

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.

DESROCHERS, M., DESROSIERS, J. et SOLOMON, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research.* 40. P. 342–354.

DESROCHERS, M. et SOUMIS, F. (1988a). A generalized permanent labeling algorithm for the shortest path problem with time windows. *INFOR.* 26. P. 191–212.

DESROCHERS, M. et SOUMIS, F. (1988b). A reoptimization algorithm for the shortest pat problem with time windows. *European Journal of Operational Research.* 35. P. 242–254.

DESROCHERS, M. et SOUMIS, F. (1989). A column generation approach to the urban transit crew scheduling problem. *Transportation Science.* 23. P. 1–13.

DESROSIERS, J., DUMAS, Y. et SOUMIS, F. (1988). The multiple vehicle dial-a-ride problem. *Computer-Aided Transit Scheduling.* J.R. Daduna and A. Wren (éd), Lecture Notes in Economics and Mathematical Systems 308, Springer, Berlin. P. 15–27.

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

DESROSIERS, J., LASRY, A., MCINNIS, D., SOLOMON, M. et SOUMIS, F. (1999). *The airline operations management system at Air Transat.* École des Hautes Études Commerciales, Canada : Cahiers du GERAD. G-95-23.

DESROSIERS, J., PELLETIER, P. et SOUMIS, F. (1983). Plus courts chemins avec contraintes d'horaire. *RAIRO.* 17. P. 357–377.

DESROSIERS, J., SOUMIS, F. et DESROCHERS, M. (1984). Routing with time windows by column generation. *Networks.* 14. P. 545–565.

DUMAS, Y., DESROSIERS, J. et SOUMIS, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research.* 54. P. 7–22.

DUMITRESCU, I. et BOLAND, N. (2003). Improved preprocessing, labeling and scaling algorithm for the weight-constrained shortest path problem. *Networks.* 42. P. 135–153.

EPSTEIN, D. (1978). Finding the k shortest paths. *SIAM Journal of Computing.* 28:2. P. 652–673.

FEILLET, D., GENDREAU, M. et ROUSSEAU, L. (2005). New refinements for the solution of vehicle routing problems with branch and price. *Cahiers du CRT, Université de Montréal.*

FOX, B. L. (1978). Data structures and computer science techniques in operations research. *Operations Research.* 26. P. 686–717.

GAMACHE, M. et SOUMIS, F. (1993). *A method for optimally solving the rostering problem.* École des Hautes Études Commerciales, Canada : Cahiers du GERAD. G-93-40.

GAMACHE, M., SOUMIS, F., MARQUIS, G. et DESROSIERS, J. (1999). A column generation approach for large scale aircrew rostering problems. *Operations Research.* 47. P. 247–262.

GOFFIN, J. et VIAL, J. (1990). Cutting planes and column generation techniques with the projective algorithm. *Journal of Optimization Theory and Applications.* 65. P. 409–429.

GRAVES, G., MCBRIDE, R., GERSHKOFF, I., ANDERSON, D. et MAHIDHARA, D. (1993). Flight crew scheduling. *Management Science.* 39. P. 736–745.

HANDLER, G. et ZANG, I. (1980). A dual algorithm for the constrained shortest path problem. *Networks.* 10. P. 293–310.

HASSINE, R. (1992). Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research.* P. 36–42.

HENIG, M. I. (1985). The shortest path problem with two objective functions. *European Journal of Operational Research.* 25. P. 281–291.

IOACHIM, I., DESROSIERS, J., DUMAS, Y., SOLOMON, M. et VILLENEUVE, D. (1995). A request clustering algorithm for door-to-door handicapped transportation. *Transportation Science.* 29. P. 63–78.

IOACHIM, I., GÉLINAS, S., DESROSIERS, J. et SOUMIS, F. (1993). A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks.* 39. P. 736–745.

IRNICH, S. et DESAULNIERS, G. 2005. « *« Shortest Path Problems with Resource Constraints »* ». GERAD 25th Anniversary Series. Springer. Chapitre 2, P. 33–65.

IRNICH, S. et VILLENEUVE, D. (2006). The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing.* 18:3. P. 391–406.

KATOH, N., IBARAKI, T. et MINES, H. (1982). An efficient algorithm for k shortest paths in a network. *Networks.* 12. P. 411–427.

KELLEY, J. (1960). The cutting-plane method for solving convex programs. *SIAM.* 8. P. 703–712.

KOHL, N. et MADSEN, O. (1997). An optimization algorithm for the vehicle routing problem with time windows based on lagrangian relaxation. *Operations Research.* 45. P. 395–406.

KOLEN, A., KAN, A. R. et TRIENKENS, H. (1987). Vehicle routing with time windows. *Operations Research.* 35. P. 266–273.

KORKMAZ, T. et KRUNZ, M. (2001). Multi-constrained optimal path selection. *INFOCOM.* P. 834–843.

KUNG, H., LUCCIO, F. et PREPARATA, F. (1975). On finding the maxima on set of vectors. *Journal of the ACM.* 22. P. 469–476.

LAVOIE, S., MINOUX, M. et ODIER, E. (1988). A new approach for crew pairing problems by column generation and application to air transport. *European Journal of Operational Research.* 35. P. 45–58.

LORENZ, D. et RAZ, D. (2001). A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters.* 28. P. 213–219.

MINGOZZI, A., BIANCO, L. et RICCIARDELLI, S. (1997). Dynamic programming strategies for the traveling salesman problem with time windows and precedence constraints. *Networks.* 13:3. P. 365–377.

NAGIH, A. et SOUMIS, F. (1999). L'agrégation des contraintes de ressources dans un problème de plus court chemin avec contraintes de ressources. *Les cahiers du GERAD, École des Hautes Études Commerciales, Montréal, Canada.*

NAGIH, A. et SOUMIS, F. (2000). L'agrégation des contraintes de ressources en chaque nœud dans un problème de plus court chemin. *Les cahiers du GERAD, École des Hautes Études Commerciales, Montréal, Canada.*

NAGIH, A. et SOUMIS, F. (2006). Nodal aggregation of resource constraints in a shortest path problem. *European Journal of Operational Research.* 172(2). P. 500–514.

POLLACK, M. (1961). The kth best route through a network. *Operations Research.* 9:3. P. 578–589.

RIBEIRO, C. et SOUMIS, F. (1994). A column generation approach to the multiple depot vehicle scheduling problem. *Operations Research.* 42. P. 41–52.

SLEATOR, D. et TARJAN, R. (1985). Self-adjusting binary serach trees. *Journal of the ASM.* 32. P. 652–686.

VEINOTT, A. (1967). The supporting hyperplane method for unimodal programming. *Operations Research.* 15. P. 147–152.

VOVOR, T. (1997). *Problèmes de chemins bicritères ou avec contraintes de ressources.* Thèse de doctorat, École Polytechnique de Montréal.

WARBURTON, A. (1987). Approximation of pareto optima in multiple-objective shortest-path problems. *Operations Research.* 35:1. P. 7–79.

WHITE, D. J. (1982). The set of efficient solutions for multiple objective shortest path problems. *Computers and Operations Research.* 9:2. P. 101–107.

YEN, J. Y. (1961). Finding the k shortest loopless paths in a network. *Operations Research.* 9:3. P. 578–589.

ZIARATI, K., SOUMIS, F., DESROSIERS, J., GÉLINAS, S. et SAINTONGE, A. (1997). Locomotive assignement with heterogeneous consists at CN North America. *European Journal of Operational Research.* 97. P. 281–292.