Thèse de doctorat de l'UNIVERSITE PARIS VI

Spécialité :

INFORMATIQUE

Présentée par :

Nabil ABSI

pour obtenir le grade de DOCTEUR de l'UNIVERSITE PARIS VI

Sujet de thèse :

Modélisation et résolution de problèmes de lot-sizing à capacité finie

Date de soutenance : 01 décembre 2005

devant le jury composé de :

Monsieur	Philippe Chrétienne	Directeur de thèse
Madame	Safia Kedad-Sidhoum	Co-Directrice de thèse
Monsieur	Stéphane Dauzère-Pérès	Rapporteur
Monsieur	Gérard Plateau	Rapporteur
Monsieur	Jean-Yves Jaffray	Examinateur
Madame	Colette Mercé	Examinatrice
Monsieur	Cédric Hutt	Membre invité

Remerciements

Je tiens tout d'abord à remercier sincèrement Philippe Chrétienne, Professeur à l'Université Pierre et Marie Curie, et Safia Kedad-Sidhoum, Maître de conférences au sein de cette même université, pour m'avoir encadré et apporté leur soutien tout au long de cette thèse. Merci Safia pour tes conseils qui m'ont guidé durant ces trois années, pour le temps que tu m'as consacré, ainsi que pour ta patience.

Je remercie Stéphane Dauzère-Pérès, Professeur à l'Ecole des Mines de Saint-Étienne, et Gérard Plateau, Professeur à l'Université Paris 13, qui m'ont fait l'honneur d'être rapporteurs de cette thèse.

Je tiens à remercier Jean-Yves Jaffray, Professeur à l'Université Pierre et Marie Curie, Colette Mercé, Professeur à l'INSA de Toulouse, Cédric Hutt, Docteur et Chef produit à DynaSys, pour avoir accepté de faire partie de mon jury de thèse.

Mes remerciements vont également à DynaSys qui m'a offert la chance d'effectuer cette thèse en convention CIFRE au sein de la Direction R&D. Je remercie tout particulièrement, Augustin Holveck, Directeur Général, et Ariel Weil, Directeur R&D, de m'avoir accueilli au sein de DynaSys. Je remercie aussi Jean-Luc Rominger, Responsable R&D, pour sa bienveillance.

Je tiens à remercier Henri Laulhere pour son soutien durant ces trois années de thèse. J'adresse également mes remerciements à tout le personnel de DynaSys, et je pense tout particulièrement à l'équipe R&D : Julien, Pti Fred, Camille, Greg, Mathieu, Thierry, Antoine, Adrien, Régis, Alex, Gaétan, Hervé et Laurent, pour l'ambiance chaleureuse. Je n'oublie pas de remercier Sylvie, Françoise, Raph, Grand Fred, Richard, Valérie, Caroline, Christiane et j'en oublie.

Je remercie tous les membres de l'équipe SYSDEF du Laboratoire d'Informatique Paris 6 qui m'ont offert un environnement de travail agréable et m'ont apporté joie et bonne humeur. Merci à Yasmin, Yann, Anna, Francis, Bénédicte, Nicolas, Lucie, Paul, Nathalie, Louis-Xavier, Sergio, Pierre, Emmanuel, Claire, Olivier et Nina.

Tous mes remerciements vont à ma Maman et mon Papa sans qui tout cela ne serait jamais arrivé.

Table des matières

Liste des tableaux

Table des figures

Liste des algorithmes

Liste des abréviations

AGG	Formulation agrégée.
APS	Advanced Planning and Scheduling.
BOM	Bill Of Materials.
CCLS	Problème de lot-sizing à capacité constante (Constant Capacity Lot-Sizing problem).
CLS	Problème de lot-sizing à capacité finie (Capacitated Lot-Sizing problem).
FL	Formulation basée sur le problème de localisation (Facility Location Formulmation).
GAP	Pourcentage d'optimalité, $GAP = (UB - LB)/UB$.
LB	Borne inférieure (Lower Bound).
MCLS	Problème de lot-sizing à plusieurs références, sous des contraintes de capacité et des temps de setup (Multi-item Capacitated Lot-sizing problem with Setup times).
MCLS2	Problème de lot-sizing à plusieurs références, avec des contraintes de capacité, des temps de setup, ainsi que des coûts de rupture sur la demande (Multi-item Capacitated Lot-sizing problem with Setup times and Shortage costs).
MCLS4	Problème de lot-sizing à plusieurs références, avec des contraintes de capacité, des temps de setup, des coûts de rupture sur la demande ainsi que des coûts de déficit sur le stock de sécurité (Multi-item Capacitated Lot-sizing problem with Setup times, Shortage costs and Safety Stock deficit costs).
MCLSG	Problème MCLS4 multi-groupes.
MCLSR	Problème MCLS4 multi-ressources.
MCLSV	Problème MCLS4 multi-gammes.
MIP	Programme linéaire mixte ou programmation linéaire mixte (Mixed Integer Program ou Mixed Integer Programming).
MRP	Material Requirements Planning.
PDP	Programme Directeur de Production.
PIC	Plan Industriel et Commercial.
SPMCLS2	Problème MCLS2 relaxé sur une seule période (Single Period relaxation of MCLS2).
UB	Borne supérieure (Upper Bound).
ULS	Problème de lot-sizing sans contrainte de capacité (Uncapacitated Lot-Sizing problem).
ULS2	Problème ULS avec des coûts de rupture sur la demande (Uncapacitated Lot-Sizing problem with Shortage costs).
ULS2B	Problème ULS2 avec des coûts de Backlog.
ULS4	Problème de lot-sizing avec des coûts de rupture sur la demande ainsi que des coûts de déficit sur le stock de sécurité (Uncapacitated Lot-Sizing problem with Shortage costs and Safety Stock deficit costs).
WW	Wagner et Whitin

Chapitre 1

Introduction générale

Pendant des décennies, le souci principal des producteurs était de produire des quantités toujours plus grandes pour diminuer le coût de production unitaire par économie d'échelle. Cependant, la production de masse présente un inconvénient majeur qui se traduit par la rigidification du système de production. En présence de forte demande de produits standardisés, ce problème a peu d'importance. En revanche, il devient crucial dans un environnement où l'offre dépasse largement la demande. Ainsi, la nature de la demande évolue rapidement et les entreprises doivent s'adapter à ces changements.

La réflexion s'est donc peu à peu orientée vers les moyens de diminuer les coûts fixes. Ceux-ci prennent une importance capitale, lorsqu'il est nécessaire de fabriquer des petites séries de produits différents. Les temps de changement de matériel sont alors prépondérants et la notion classique d'économie d'échelle n'est plus pertinente. Cette constatation conduit à aborder les problèmes de production sous un angle différent.

C'est donc au niveau des moyens de rationalisation de la production que l'on a constaté les évolutions les plus significatives ; les objectifs actuels sont de tenter de bénéficier de la flexibilité des petites unités, tout en gardant les avantages (en terme de coût) de la production de masse, afin de satisfaire la demande par une action au niveau de la réduction des coûts fixes.

Dans un environnement de fabrication, les problèmes de planification de la production traitent des décisions de dimensionnement de lots des différents produits à fabriquer ou traiter, des périodes auxquelles ces lots doivent être produits, et éventuellement de la ou des machines sur lesquelles les productions doivent avoir lieu. Les décisions de production sont prises au regard du meilleur rapport entre les objectifs financiers et les objectifs de satisfaction des demandes.

Le problème de la planification de la production dans une entreprise de manufacture peut être défini comme étant un ensemble de décisions, qui doivent être prises à court et à moyen terme afin de convertir des matières premières en produits finis dans une entreprise de production. La planification de la production détermine qui doit effectuer telle ou telle opération, où et quand.

Les matières premières sont achetées à des fournisseurs, et les produits finis sont vendus à des clients externes. La transformation des matières premières en produits finis peut consister en plusieurs étapes de production durant lesquelles des produits intermédiaires sont fabriqués. La planification de la production fait donc partie d'une chaîne de décisions dans laquelle, les

flux d'information circulent des clients aux fournisseurs et les flux physiques des fournisseurs jusqu'aux clients finaux (cf. figure 1.1).

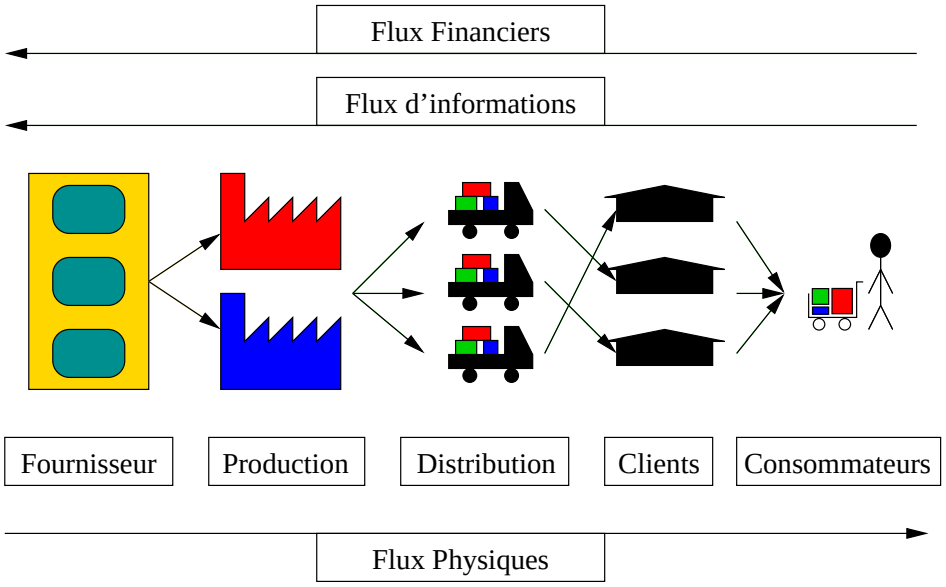


FIG. 1.1 – Chaine logistique d'approvisionnement

La notion de *chaîne logistique d'approvisionnement* (*Supply Chain Management*) est apparue dans les années 80 dans le but de répondre à cette nouvelle problématique. De nos jours, la concurrence est intense et internationale. L'offre des entreprises est en perpétuel mouvement sous l'impulsion d'une demande qui fait jouer la concurrence. La production doit s'adapter à cette instabilité chronique et l'entreprise doit avoir connaissance de la demande avant de fabriquer le produit. Si dans le passé, il fallait produire ce qui allait se vendre, il faut désormais produire ce qui est déjà vendu. La production doit s'adapter de plus en plus rapidement aux fluctuations. Ce qui conduit à une nouvelle diminution de la taille des lots de production, et à une nouvelle approche dans la définition du produit-service proposé au client.

Les décideurs cherchent alors à minimiser le coût de fonctionnement de le chaîne d'approvisionnement, en réduisant le coût des flux physiques, les coûts de production, les coûts de stocks intermédiaires ainsi que les coûts engendrés par l'insatisfaction des clients.

Les travaux de ma thèse s'inscrivent dans le cadre du développement du logiciel n.SKEP[1] de la société DynaSys qui a pour but l'optimisation de la Supply Chain Management.

En effet, DynaSys est un éditeur de logiciels qui propose une solution globale n.SKEP pour l'optimisation de la chaîne logistique. n.SKEP gère les flux d'informations qui circulent entre les clients et les fournisseurs, et dispose de plusieurs modules qui permettent d'optimiser les flux physiques circulant des fournisseurs vers les clients (cf. figure 1.2). Ce terme regroupe toutes les opérations qui permettent la création d'un produit et de sa valeur ajoutée, des fournisseurs jusqu'à la livraison finale, en passant par les ordres de fabrication, le contrôle qualité, etc. Les modules de la solution n.SKEP peuvent être décrits de la façon suivante :
n.SKEP Demand Planning : Solution collaborative de pilotage de la demande prévisionnelle, point d'entrée des modules de distribution et de la production. Les prévisions de ventes sont

[1] n.SKEP : Supply Chain Management Software, DynaSys S.A.

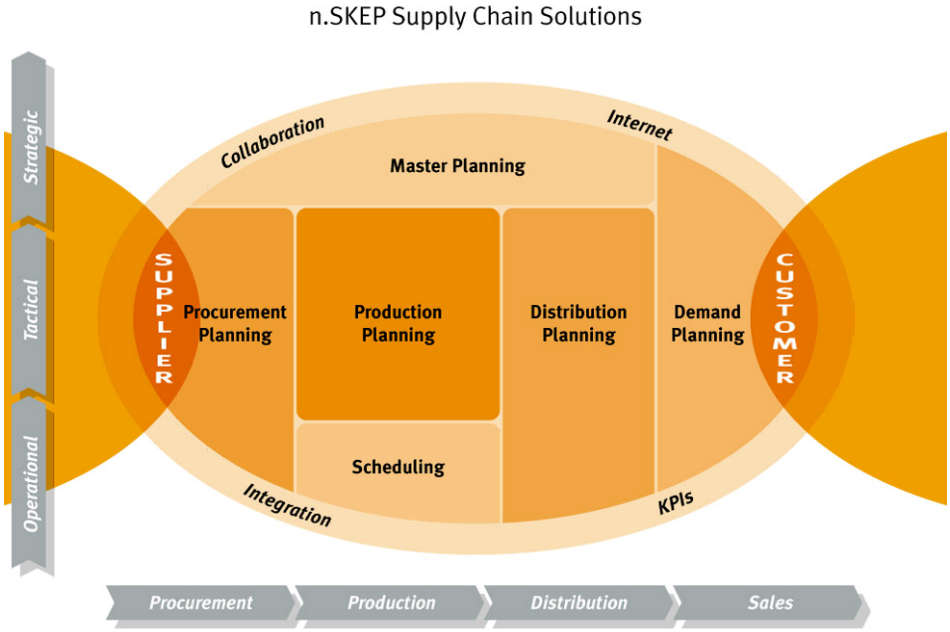


FIG. 1.2 – La solution n.SKEP de DynaSys

élaborées à partir des historiques, en prenant en compte l'ensemble des évènements (Marketing, Promotions, Cannibalisation).
n.SKEP Distribution Planning : Solution de planification centrale multi-sites qui coordonne les flux de produits à travers l'ensemble du réseau logistique (sites de distribution, de stockage et de production). L'équilibrage des flux est réalisé par simulation et/ou par optimisation des coûts.
n.SKEP Production Planning : Solution de planification de l'usine, de programmation des capacités de production nécessaires et de projection des stocks prévisionnels. Le Programme Directeur de Production (PDP) est optimisé en fonction de la demande et des stocks, compte tenu des contraintes de production.
n.SKEP Scheduling : Solution de programmation des lignes/machines de production permettant d'optimiser les délais et la capacité utilisée aux niveaux les plus détaillés.
n.SKEP Procurement Planning : Solution permettant de planifier les approvisionnements en composants et matières premières, et de garantir leur disponibilité pour la réalisation des plans de production.
n.SKEP Collaborative e-business : Solution permettant à tous les collaborateurs distants d'accéder à l'ensemble des fonctionnalités des solutions n.SKEP via internet.
n.SKEP Integration : Solution permettant l'intégration aisée et standard de n.SKEP avec le Système d'Information de l'entreprise.
n.SKEP Performance Scoring : Solution de mesure des performances de la *Supply Chain*.

Mes travaux de recherche portent essentiellement sur le module *Production Planning*, ils consistent à améliorer et à développer des outils d'optimisation de la planification de la production dans une entité de manufacture. Ces outils sont essentiellement basés sur le solveur

de programmation linéaire en nombres entiers CPLEX 9.0 de la société Ilog [110].

Le plan de production établit les besoins en ressources pour effectuer des transformations, dans le but de satisfaire les clients de la manière la plus efficace et la plus économique possible. En d'autres termes, les décisions de produire sont prises au meilleur rapport entre l'objectif financier et celui de satisfaction des clients.

Le problème de dimensionnement de lots tient une place centrale dans la planification de la production. Il est connu sous le nom de problème de *lot-sizing* dans la littérature, il répond aux questions suivantes en ayant un profit maximum :

- Quelles sont les références (produits) à produire à chaque période ?
- Quelle est la quantité à produire pour chaque référence ?
- Sur quelle machine produire chaque référence ?
- Quelle est la gamme utilisée pour produire chaque référence ?

Le problème de lot-sizing abordé dans le cadre de mes travaux de recherche consiste à déterminer un plan de production d'un ensemble de N références pour un horizon de planification constitué de T périodes. Ce plan est à capacité finie et doit tenir compte d'un ensemble de contraintes additionnelles. En effet, on dispose de R ressources dont les capacités disponibles sont limitées. Le lancement de production d'une référence à une période donnée entraîne outre la consommation variable en ressources, une consommation dite fixe (ou setup).

Les problèmes de lot-sizing traités dans ma thèse ont des spécificités industrielles issues de besoins réels exprimés par les clients de DynaSys. En effet, les clients cherchent à quantifier les pertes dues aux capacités limitantes. Ces quantités doivent être exprimées par référence et par période. Les principales caractéristiques de ces problèmes sont décrites dans les paragraphes qui suivent. Des descriptions plus détaillées sont faites dans le chapitre 3.

Dans ces problèmes de lot-sizing, les demandes ne peuvent être livrées en retard. En effet, il arrive que le client ne tolère aucun retard dans les livraisons. Dans ce cas les commandes sont annulées et les demandes perdues. Il arrive également que les capacités ne soient pas suffisantes pour satisfaire toutes les demandes aux dates prévues. Il se peut aussi qu'il soit plus profitable de perdre la demande plutôt que de la satisfaire à tout prix. En effet, dans une situation où les capacités sont limitées, on peut être contraint à anticiper la production de plusieurs périodes, cette anticipation peut ne pas être rentable en raison de coûts de stockage élevés. Dans ces cas, on parle de *ruptures sur les demandes*. Une telle situation n'est pas sans conséquence pour le fabricant. En effet, le risque majeur est la perte du client. Une autre conséquence peut être une mauvaise réputation qui peut faire fuir des clients et par conséquent diminuer les ventes. Ces problèmes poussent le décideur à fixer comme objectif principal la minimisation des ruptures sur la demande. Les ruptures sont exprimées par référence et par période. Un coût de rupture unitaire est associé à chaque référence et à chaque période.

Souvent, l'un des objectif du décideur est d'avoir un profil de stocks supérieur à un seuil appelé *stock de sécurité*. Le stock de sécurité est une partie du stock qui n'est normalement pas utilisée. Il permet de faire face à des événements imprévus (retard de livraison, augmentation de la consommation...). Les limitations des ressources peuvent pousser le décideur à satisfaire la demande pour éviter d'avoir des ruptures plutôt que de respecter le stock de

sécurité. Dans ce cas on parle de *déficit sur le stock de sécurité*. Cet objectif est secondaire par rapport aux ruptures sur la demande. En effet, le décideur préfère avoir des déficits sur le stock de sécurité plutôt que d'avoir des ruptures sur la demande. Un coût de déficit unitaire est associé à chaque référence et à chaque période.

Un objectif secondaire est de minimiser les coûts de production (coûts unitaires, coûts fixes ou coûts de setup) et les coûts de stockage. Généralement le coût de production est modélisé par une fonction décomposée en deux coûts, un coût fixe indépendant du lot produit, et un coût unitaire encouru pour chaque unité produite d'une référence. Les coûts de stockage sont des coûts unitaires. Les coûts de rupture sur la demande et de déficit sur le stock de sécurité sont considérés comme pénalités en comparaison avec les autres coûts.

L'une des originalités des problèmes de lot-sizing abordés dans mes travaux de recherche est la prise en compte des coûts de ruptures sur la demande et des coûts de déficits sur le stock de sécurité. A notre connaissance, ce problème n'a jamais été traité dans la littérature.

Les problèmes de lot-sizing que nous traitons sont de nature générique, puisque les approches développées pour les résoudre sont intégrées à un logiciel d'APS[2]. Nous présentons dans ce qui suit, les particularités additionnelles de ces problèmes.

Nous distinguons J groupes de références. Une référence peut appartenir à un ou plusieurs groupes et un groupe peut contenir une ou plusieurs références. Cette classification est faite le plus souvent selon les consommations fixes en ressources des références. Des références font partie du même groupe si elles ont une consommation fixe en ressource commune. Tout lancement de production d'une référence induit une consommation fixe en ressources ainsi qu'un coût fixe (setup) pour le (ou les) groupe(s) auxquels elle appartient. Le plan de production doit également respecter une production minimale et un lancement minimum s'il y a production. Celle-ci doit s'effectuer en respectant des contraintes de taille de lots.

Une autre particularité des problèmes traités, réside dans le fait que le décideur puisse être confronté à choisir entre des modes alternatifs de production dès lors qu'interviennent plusieurs lignes de production, plusieurs vendeurs ou plusieurs sous-traitants. Chacune de ces alternatives a des coûts de production et des consommations en ressource différents. Dans ce cas, on parle de plusieurs processus (ou gammes) de production. V est le nombre de gammes.

L'objectif des problèmes de lot-sizing étudiés est donc de minimiser les coûts de rupture sur la demande, les coûts de déficit sur le stock, les coûts de production, ainsi que les coûts de stockage en tenant compte des contraintes additionnelles décrites ci-dessus.

Les contraintes décrites précédemment ne sont pas forcément toutes prises en compte simultanément. Les utilisateurs du logiciel d'APS décident, en fonction de la nature du problème traité, des contraintes à considérer. Ce caractère générique des problèmes étudiés ainsi que l'obligation d'intégrer les résultats dans le logiciel n.SKEP nous ont conduit à nous poser plusieurs questions :

[2]Advanced Planning and Scheduling.

- Quelles sont les particularités du problème de lot-sizing que nous traitons par rapport à la littérature ?
- Comment formuler ces différentes particularités ?
- Comment aborder les problèmes de nature générique ?
- Comment résoudre ces problèmes ?

Afin d'apporter des réponses à ces questions, nous avons tout d'abord identifié le problème de lot-sizing avec des coûts de rupture sur la demande et des coûts de déficit sur le stock de sécurité comme étant le problème noyau, nous avons insisté sur les particularités de ce nouveau modèle par rapport aux formulations classiques. Nous avons également proposé un algorithme polynomial pour résoudre le problème sans contraintes de capacité ainsi qu'une approche par relaxation lagrangienne des contraintes de capacité afin de déterminer une borne inférieure du problème.

Les modules d'optimisation de n.SKEP utilisent le solveur CPLEX 9.0 qui offre un algorithme de branch-and-bound en proposant la possibilité d'utiliser des coupes classiques. La seconde approche consiste à améliorer ces modules en renforçant les relaxations linéaires continues à chaque nœud de l'arbre de branch-and-bound. Pour ce faire, nous avons développé des inégalités valides basées sur la structure des problèmes de lot-sizing étudiés.

Les caractéristiques du problème identifié le rendent difficile à décomposer. Lorsqu'il faut les prendre en considération, des approches de résolution heuristiques sont appropriées. Nous nous sommes donc orientés vers des approches heuristiques à base de programmation linéaire mixte et d'une décomposition basée sur la structure de l'horizon de planification.

Cette thèse tente de répondre aux questions posées ci-dessus, nous l'avons organisée en cinq chapitres. Le chapitre 2 est un état de l'art sur les problèmes de lot-sizing à une seule et à plusieurs références avec ou sans contraintes de capacité. Le chapitre suivant est consacré à la description et à la modélisation des différentes particularités des problèmes de lot-sizing traités dans le cadre de mes travaux de recherche. Les trois autres chapitres sont consacrés à la résolution de ces problèmes. Plus précisement, le chapitre 4 est une approche polyédrique. En effet, nous avons développé un algorithme de branch-and-cut afin de résoudre le problème de lot-sizing à plusieurs références avec des contraintes de capacité, des coûts de setup, ainsi que des coûts de rupture sur la demande. Cet algorithme est basé sur les inégalités (l, S), les inégalités couvrantes ainsi que les inégalités couvrantes inverses. Ces inégalités sont une généralisation d'autres inégalités rencontrées dans la littérature (Barany *et al.* [14, 15], Miller *et al.* [132], Marchand et Wolsey [123]). Sous certaines conditions, celles-ci décrivent des facettes de l'enveloppe convexe du problème traité. Des heuristiques de séparation de ces inégalités sont également proposées. Ces inégalités sont également généralisées afin de tenir compte des contraintes industrielles telles que, les groupes de références, le cas de plusieurs ressources en parallèle et celui de plusieurs gammes de production.

Au chapitre 5, une approche basée sur la relaxation lagrangienne est proposée afin fournir une borne inférieure de bonne qualité aux problèmes de lot-sizing à plusieurs références avec des contraintes de capacité, des coûts de setup, ainsi que des coûts de rupture sur la demande et de déficit sur le stock de sécurité. Après un rappel de la modélisation du problème traité, nous développons un algorithme polynomial pour résoudre les sous-problèmes induits par la relaxation lagrangienne des contraintes de capacité.

Dans le chapitre 6 nous proposons une approche de résolution pour les problèmes lot-sizing à capacité finie impliquant plusieurs références, des temps de setup, des coûts de rupture, des coûts de déficit sur le stock de sécurité, plusieurs groupes de références, plusieurs res-

sources, plusieurs gammes de production et des contraintes de production minimum. Le but de cette approche est de fournir une bonne borne supérieure dans des temps de calcul raisonnables. Pour ce faire, nous avons développé des heuristiques à base de programmation linéaire mixte MIP (Mixed Integer Programming). Celles-ci sont une hybridation d'une approche de décomposition de l'horizon de planification et d'une méthode branch-and-bound. L'horizon de planification est partitionné en plusieurs sous-horizons pour lesquels une stratégie de relaxation ou de gel est appliquée. Dans chaque chapitre, des résultats expérimentaux ainsi que des analyses de sensibilité sont présentés.

Enfin, nous terminons cette thèse par une conclusion générale et des perspectives de recherche pour des travaux futurs.

Chapitre 2

Etat de l'art

Depuis les travaux fondateurs de Wagner et Whitin [189], plusieurs chercheurs se sont intéressés au problème de lot-sizing sous ces différentes formes, en proposant des modélisations, des reformulations ainsi que des méthodes de résolution. Faire un état de l'art exhaustif sur tous les travaux de recherche effectués depuis la fin des années 50 est une tâche ardue. Nous allons présenter dans ce chapitre, un état de l'art des principales contributions apportées dans ce domaine. Nous proposons une classification des approches par type de problèmes (une seule référence, plusieurs références), puis par méthode de résolution (méthodes exactes, méthodes approchées).

Introduction

Depuis les travaux de Manne [121] et Wagner et Whitin [189] à la fin des années 50, plusieurs travaux ont été réalisés dans le domaine du lot-sizing. Le problème du lot-sizing à une seule référence a reçu une attention particulière en raison de sa simplicité et de son importance comme étant un sous-problème de problèmes de lot-sizing plus complexes.

Wolsey [192] a présenté un état de l'art sur les avancées concernant le problème de lot-sizing à une seule référence et sans contrainte de capacité, ainsi que son utilisation pour la résolution des problèmes de lot-sizing à plusieurs références. Brahimi *et al.* [27] ont proposé un état de l'art récent sur le problème de lot-sizing à une seule référence. Ils ont présenté différentes modélisations mathématiques et méthodes de résolution. Un état de l'art plus général et moins détaillé sur les problèmes de lot-sizing à capacité finie est proposé par Karimi *et al.* [91]. Jans et Degraeve [89] ont proposé un état de l'art sur l'application des méta-heuristiques aux problèmes de lot-sizing.

La diversité des modélisations et des approches de résolution a poussé les chercheurs à classer ces problèmes par modèle mathématique et par méthode de résolution. Belvaux et Wolsey [18, 19] ont proposé une classification par type de problèmes. En effet, les auteurs ont classé les modèles en problèmes à courtes (small bucket) et à longues périodes (big bucket)[1]. Ils les ont également classifié par problèmes à une seule et plusieurs références. Les méthodes de résolution sont généralement classifiées par méthodes exactes et méthodes approchées (ex. Brahimi *et al.* [27], Karimi *et al.* [91]). D'autres classifications sont proposées par Drexl et Kimms [52], Salomon [157], Staggemeier et Clark [165] et Wolsey [195].

[1] Les notions de small et big buckets sont développées au chapitre 3, section 3.1.

Voß et Woodruff [187] ont publié récemment un livre d'introduction à la modélisation des problèmes de planification de la production. Les auteurs ont présenté plusieurs modèles mathématiques allant du problème MRP[2] à des problèmes de lot-sizing plus complexes.

Dans ce chapitre, les problèmes de lot-sizing sont classés en deux catégories. Les problèmes de lot-sizing à une seule référence, et les problèmes de lot-sizing à plusieurs références. Pour chaque classe, la distinction est faite entre les méthodes exactes et les méthodes approchées. Toutefois, nous accordons plus d'attention aux problèmes de lot-sizing à un seul niveau de production, ainsi qu'aux problèmes à périodes longues.

Cette classification dépend essentiellement des problèmes que nous étudions. En effet, nous présentons au chapitre 4 une méthodes exacte basée sur la méthode de coupes pour la résolution d'un problème de lot-sizing à plusieurs références sous des contraintes de capacité. Au chapitre 5, nous présentons une méthode exacte pour résoudre un problème de lot-sizing à une seule référence et sans contrainte de capacité, puis une méthode approchée pour le même problème à plusieurs références sous des contraintes de capacité. Au chapitre 6, nous présentons une méthode approchée pour un problème de lot-sizing à plusieurs références sous des contraintes de capacité. Plusieurs des approches et des propriétés présentées dans ce chapitre seront reprises dans les chapitres suivants.

2.1 Problèmes à une seule référence

Les méthodes les plus utilisées pour résoudre les problèmes de lot-sizing à une seule référence sont la programmation dynamique, les méthodes des coupes, ainsi que les méthodes basées sur les modèles renforcés (*Strong formulations*). Ce sont généralement des méthodes exactes. Dans ce qui suit, nous commençons par présentation du modèle mathématique de base traité par Wagner et Whitin [189], noté ULS[3], puis les approches de résolution exactes et approchées appliquées au modèle ULS ainsi qu'à des modèles ULS étendus.

2.1.1 Modélisation du problème ULS

Le premier problème que nous présentons est le problème de lot-sizing à une seule référence, un seul niveau de production et sans contrainte de capacité (ULS), connu également sous le nom de problème de Wagner et Whitin. L'indice $t = 1, \ldots, T$ représente les périodes discrètes de l'horizon de planification, T représente le nombre de périodes. Le but est de planifier la production sur cet horizon de planification (i.e. dimensionner la taille de lot pour chaque période) afin de satisfaire la demande et de minimiser la somme des coûts de production et des coûts de stockage. Le coût de production d'un lot est décomposé en deux coûts, un coût fixe indépendant du lot produit, et un coût unitaire encouru pour chaque unité produite. Les coûts de stockage sont modélisés par des coûts unitaires pour chaque unité stockée à la fin de chaque période. La demande de chaque période est satisfaite grace à la production ou par le stock. Les retards sur la demande[4] ne sont pas autorisés. La capacité de production est infinie dans ce modèle. Pour toute période $t = 1, \ldots, T$, les variables de décision sont x_t, y_t et s_t. Elles représentent respectivement, la quantité produite à la période t si une production a été lancée à cette même période ($y_t = 1$ si $x_t > 0$), et le stock à la fin de la période t.

[2]MRP : Material Requirements Planning.

[3]ULS est une abréviation en anglais de : Uncapacitated Lot-Sizing problem.

[4]Le retard sur la demande est usuellement dénommé : backlog.

Pour toute période t, les données du problème sont : le coût de production unitaire (α_t), le coût fixe de production ou setup (β_t), le coût unitaire de stockage (γ_t), et la demande à satisfaire (d_t). La demande est exprimée à partir des commandes fermes complétées par les commandes prévisionnelles. Les demandes sont fermes à court terme, puis prévisionnelles à plus long terme.

Ce modèle est le sous-problème noyau dans la planification de la production, puisqu'il est résolu d'une manière répétitive pour chaque référence (des produits finis vers les matières premières) dans le MRP.

En utilisant ces variables et paramètres, nous formulons le problème ULS comme suit :

$$\min \sum_t \alpha_t x_t + \sum_t \beta_t y_t + \sum_t \gamma_t s_t \tag{2.1}$$

s.c :

$$x_t - s_t + s_{t-1} = d_t, \quad \forall t \tag{2.2}$$
$$x_i \leq M y_t, \quad \forall t \tag{2.3}$$
$$x_t, s_t \geq 0, \quad \forall t \tag{2.4}$$
$$y_t \in \{0,1\}, \quad \forall t \tag{2.5}$$

La fonction objectif (2.1) est la minimisation de la somme des coûts de production et des coûts de stockage. La contrainte (2.2) représente la satisfaction de la demande à chaque période t. Elle est appelée, la contrainte de conservation des flux à travers l'horizon. La contrainte (2.3) permet de modéliser la condition suivante : s'il y a lancement de production, alors la quantité produite ne devra pas dépasser le majorant de la production M (M est un majorant, il représente généralement la somme des demandes de la période t à la période T). La contrainte (2.4) signifie que les variables x_t et s_t sont continues non négatives pour chaque période t. La dernière contrainte (2.5) exprime le fait que y_t est une variable binaire pour chaque période t.

Ce modèle est appelé également formulation agrégée, car la production de la référence n'est définie que par sa période de production, contrairement à la formulation basée sur le problème de localisation d'entrepôts (Facility Location-based formulation, Krarup et Bilde [101]) où la production est définie par sa période de production et par sa période de consommation.

En rajoutant une contrainte de capacité sur la production à chaque période, on crée un problème de lot-sizing à une seule référence et à capacité finie, noté CLS[5]. Pour modéliser la limitation des ressources, la contrainte (2.3) est remplacée par la contrainte suivante :

$$x_i \leq C_t y_t, \quad \forall t \tag{2.6}$$

Dans le cas où les capacités de production sont constantes ($C_t = C$), on parle de problème de lot-sizing à capacité constante, noté CCLS[6].

[5]CLS est une abréviation en anglais de : Capacitated Lot-Sizing problem.
[6]CCLS est une abréviation en anglais de : Constant Capacity Lot-Sizing problem.

2.1.2 Complexité

Bitran et Yanasse [23] et Florian *et al.* [65] ont prouvé que le problème de lot-sizing à capacité finie CLS est NP-difficile pour le cas général et d'autres cas particuliers. Chen *et al.* [29] ont prouvé que le problème n'est pas NP-difficile au sens fort en proposant un algorithme pseudo-polynomial pour le résoudre. Le tableau 2.1 présente les principales complexités des problèmes ULS et CCLS. Les complexités du problème ULS ainsi que celles des problèmes ULS étendus sont présentées dans la section suivante.

Problème	Complexité	Références
ULS	$O(T^2)$	[189]
	$O(T \log T)$	[5], [59], [188]
CCLS	$O(T^4)$	[64]
	$O(T^3)$	[178]

TAB. 2.1 – Complexités ULS, CCLS

2.1.3 Méthodes exactes

Les méthodes exactes sont généralement appliquées aux problèmes faciles à résoudre. C'est-à-dire, en un temps polynomial. Ces problèmes apparaissent souvent comme sous-problèmes de problèmes plus difficiles.

2.1.3.1 Programmation dynamique

Les méthodes de programmation dynamique ont des applications importantes aussi bien dans l'industrie que dans la gestion. La programmation dynamique est une méthode d'optimisation opérant par phase. L'efficacité de cette méthode repose sur le principe d'optimalité de Bellman [17] : toute politique optimale est composée de sous-politiques optimales. Pour une description détaillée du principe de la programmation dynamique, le lecteur peut se référer à [194].

Les recherches sur le problème de lot-sizing dynamique ULS ont commencé en 1958 avec les papiers fondateurs de Manne [121] et Wagner et Whitin [189]. Le tableau 2.2 résume les principales références bibliographiques utilisant des méthodes de programmation dynamique pour la résolution de problèmes de lot-sizing à une seule référence.

Problème traité	Références
Problèmes de lot-sizing à une seule référence, sans contrainte de capacité	[5], [8], [26], [36], [59], [60], [68], [87], [109], [112], [115], [116], [120], [124], [137], [160], [167], [176], [182], [184], [185], [188], [189], [198], [200],
Problèmes de lot-sizing à une seule référence sous contraintes de capacité constante	[23], [33], [64], [80], [86], [105], [106], [159], [175], [178], [183], [180]
Problèmes de lot-sizing à une seule référence sous contraintes de capacité	[29], [32], [65], [96], [159], [161]

TAB. 2.2 – Problèmes de lot-sizing à une seule référence : Programmation dynamique

Wagner et Whitin [189] sont parmi les premiers à proposer un algorithme de programmation dynamique pour résoudre le problème de lot-sizing à une seule référence et sans

contrainte de capacité (ULS). Plusieurs auteurs ont généralisé cette approche pour résoudre des problèmes de lot-sizing avec des contraintes additionnelles. Les auteurs ont montré qu'il existe une solution optimale qui satisfait la propriété suivante :

$$s_{t-1}x_t = 0 \qquad \forall t$$

Ce qui signifie que pour une solution optimale, on ne produit jamais à une période en ayant un stock non nul provenant de la période précédente. Cette propriété est appelée la propriété de Wagner et Whitin (WW). Elle signifie également qu'on ne produit que pour satisfaire la demande d'un nombre entier de périodes consécutives. En se basant sur ces propriétés optimales, Wagner et Whitin ont formulé un algorithme de programmation dynamique en $O(T^2)$ pour résoudre le problème ULS.

Veinott [185] a montré que même si les coûts de production et de stockage sont des fonctions concaves quelconques, le problème de Wagner et Whitin reste solvable par un algorithme de programmation dynamique en $O(T^2)$.

Zangwill [200] a modélisé le problème ULS en un problème de réseau à coûts fixes. L'auteur fournit une preuve équivalente à celle de Wagner et Whitin, en utilisant la théorie des graphes. Dans un réseau à une seule source, un flot extrême[7] ne peut avoir qu'un seul arc avec un flot positif à chaque nœud, cette caractéristique est équivalente à la propriété de Wagner et Whitin.

Zangwill [198] a étendu le modèle de Wagner et Whitin en permettant le retard sur les demandes (*le backlogging*), pour plus de détails sur le backlogging le lecteur peut se reporter à la page 142. Ainsi, les demandes peuvent être satisfaites à des périodes futures. Zangwill a décrit la structure de la solution optimale pour ce problème. La propriété principale est que la production à une période donnée doit satisfaire la demande d'un nombre entier de périodes consécutives, que ce soit dans le futur ou dans le passé. Donc, la demande d'une période donnée doit toujours être satisfaite par une production, le stock ou le backlogging, mais jamais par une combinaison de ces trois possibilités. Zangwill a décrit un algorithme de programmation dynamique basé sur ces propriétés pour calculer la solution optimale en $O(T^2)$.

Love [115] a étendu l'approche de Wagner et Whitin en étudiant le problème avec des bornes inférieures et supérieures sur la production et sur le stock. L'auteur a proposé un algorithme de programmation dynamique en $O(T^3)$.

Pour la résolution du problème ULS, Aggarwal et Park [5], Federgruen et Tzur [59] et Wagelmans *et al.* [188] ont proposé des algorithmes différents mais qui aboutissent à la même complexité, ces algorithmes permettent d'améliorer la complexité de l'algorithme de Wagner et Whitin en faisant passer son temps de calcul à $O(T \log T)$. Federgruen et Tzur [60] ainsi que Wagelmans *et al.* [188] ont proposé des algorithmes en $O(T \log T)$ pour le cas du backlogging.

Magnanti et Vachani [120] ont traité le problème ULS avec des coûts de startup. Contrairement aux coûts de setup qui sont comptabilisés à chaque période de production, les coûts de startup sont comptabilisés une seule fois si la production est lancée sur plusieurs périodes consécutives. Pour plus de détails sur la modélisation du startup, le lecteur peut se reporter à la page 141. Les auteurs ont montré que les propriétés de Wagner et Whitin restent vérifiées. Ceci permet de dériver un algorithme de programmation dynamique en $O(T^2)$. Aggarwal et Park [5] et Van Hoesel [176] ont proposé un algorithme plus efficace en $O(T \log T)$.

[7] La notion de flots extrêmes sera abordée au chapitre 5 page 98.

Aksen *et al.* [8] ont généralisé l'algorithme de Wagner et Whitin pour le problème ULS avec des ruptures sur la demande. Dans leur cas, la demande ne peut être retardée mais peut être perdue. Ils ont proposé un algorithme de programmation dynamique qui donne la solution optimale en $O(T^2)$. Sandbothe et Thompson [159] ont étudié le même problème avec des contraintes de capacité. Ils ont proposé un algorithme en $O(T^3)$ pour le problème de lot-sizing avec des capacités constantes à travers l'horizon de planification, et des ruptures sur la demande.

Martel et Gascon [124] ont proposé un algorithme en $O(T^2)$ pour résoudre le problème de lot-sizing quand le coût de stockage est corrélé du coût de production. Différentes extensions du problème ULS sont proposées dans la littérature, e.g. produits périssables, détériorés ou obsolètes. Les principales références sont Cohen [36], Ghare et Schrader [68], Jain et Silver [87], Nahmias et Wang [137], Shah [160], Tadikamalla [167], Van Zyl [182] et Veinott [184].

Florian et Klein [64] ont traité le problème de lot-sizing à une seule référence avec des contraintes de capacité constante à travers l'horizon (CCLS). Les auteurs ont caractérisé la structure de la solution optimale avec des coûts concaves, ils ont traité également le cas avec backlogging. Ils ont défini les points de régénération comme étant les périodes avec un stock sortant nul, ainsi que des périodes de production fractionnaire comme étant les périodes avec une production strictement positive, et strictement inférieure à la capacité disponible. Les auteurs ont montré qu'il existe une solution optimale, avec la propriété qu'il existe au moins une période de régénération entre deux périodes à production fractionnaire. A partir de cette propriété, ils ont proposé un algorithme de programmation dynamique qui fournit la solution optimale en $O(T^4)$. L'approche que nous proposons au chapitre 5 est proche de celle-ci. Jagannathan et Rao [86] ont étendu ces résultats de Florian et Klein pour une structure générale de la fonction de coût de production qui n'est ni convexe, ni concave.

Récemment, Van Hoesel et Wagelmans [178] ont amélioré l'algorithme de Florian et Klein en faisant passer sa complexité à $O(T^3)$. Les auteurs ont proposé une méthode plus efficace pour résoudre les sous-problèmes. En effet, les sous-problèmes engendrés par la décomposition proposée par Florian et Klein sont résolu en $O(T^2)$, Van Hoesel et Wagelmans ont proposé un algorithme glouton pour les résoudre en $O(T)$. Hill [80] a adapté l'algorithme de Wagner et Whitin pour le problème CCLS quand les coûts de production et de stockage sont constants à travers l'horizon de planification.

Bitran et Yanasse [23] ont proposé un algorithme en $O(T^3)$ pour le problème CCLS avec des coûts de setup et des coûts de production non croissants. Les mêmes auteurs ont proposé un algorithme en $O(T^4)$ pour résoudre le problème avec des capacités non décroissantes dans le temps. Cet algorithme a été amélioré par Chung et Lin [33] pour pour atteindre une complexité de $O(T^2)$. Bitran et Yanasse [23] ont développé un algorithme en $O(T \log T)$ pour le problème avec des coûts de stockage nuls, des coûts de préparation et de production constants, et une structure générale de la fonction capacité. Ils ont également proposé un algorithme simple et trivial en $O(T)$ pour résoudre le problème de lot-sizing avec des coûts de setup et de production non décroissants, un coût de stockage nul et des capacités non croissantes dans le temps. Lambrecht et VanderVeken [106] ont étendu cette analyse pour le problème avec des structures de capacité arbitraire.

Van Vyve [180] a proposé un algorithme en $O(T^3)$ pour résoudre le problème CCLS avec backlogging et un nombre général de batch installables, i.e. à chaque période t, la variable de production x_t est bornée par une variable entière y_t. y_t prend des valeurs dans $\{1, \ldots, m_t\}$ et un coût unitaire lui est associé dans la fonction objectif. Dans le cas discret, i.e. $x_t = y_t$, le problème avec et sans backlogging est résolu en $O(T^2)$ et $O(T \log T)$ respectivement.

Vanderbeck [183] a proposé un algorithme en $O(T^6)$ pour le problème de lot-sizing à capacité constante et avec des coûts de startup.

Florian *et al.* [65] ont prouvé que le problème de lot-sizing avec des contraintes de capacité CLS est NP-Difficile, ils ont proposé une procédure pseudo-polynomiale en $O(T^2\bar{c}\bar{d})$ pour résoudre le problème, $\bar{c}$ et $\bar{d}$ représentent respectivement la capacité moyenne et la demande moyenne. Kirca [96] a proposé une amélioration de cet algorithme. Chen *et al.* [29] ont proposé une approche de programmation dynamique pseudo-polynomial pour résoudre le même problème. Récemment, Shaw et Wagelmans [161] ont étudié le problème CLS avec une fonction linéaire par morceaux pour les coûts de production et du backlogging. Ils ont proposé un algorithme en $O(T^2\bar{q}\bar{d})$ où $\bar{d}$ est la demande moyenne et $\bar{q}$ le nombre moyen de morceaux linéaires représentant la fonction de coût de production.

Loparic *et al.* [112] ont considéré le stock de sécurité en imposant une borne inférieure sur le stock, ils ont également considéré des variables qui représentent les ventes au lieu de demandes fixes, leurs objectif est de maximiser les ventes. Les auteurs ont proposé une approche de programmation dynamique et de flots pour résoudre le problème ULS incluant ces deux contraintes. D'autres contributions pour des problèmes de lot-sizing restreint sont décrites dans Bitran et Matsuo [22], et Chung *et al.* [32].

Le problème de lot-sizing à plusieurs niveaux de production et sans contrainte de capacité peut être résolu de manière polynomiale par la programmation dynamique (Love [116], Zangwill [200]) en utilisant les mêmes propriétés de décomposition que le problème ULS.

Brahimi [26] et Dauzère-Pérès *et al.* [45] ont étudié le problème de lot-sizing avec des fenêtres de temps, des intervalles durant lesquels la demande doit être produite. Les auteurs ont proposé un algorithme polynomial en $O(T^4)$ pour résoudre le problème.

On analyse dans ce qui suit, le cas spécial des problèmes de lot-sizing traités précédemment où les coûts doivent satisfaire aux conditions suivante dites de Wagner et Whitin. Un problème de lot-sizing satisfait les conditions de Wagner et Whitin, si $\alpha_t + \gamma_t \geq \alpha_{t+1}$ avec α_t et γ_t représentent respectivement, le coût de production unitaire et le coût unitaire de stockage à la période t. Cette propriété signifie que les coûts de production et de stockage sont non spéculatifs dans le sens où, si la demande d_t est satisfaite à partir de la période t ou toute période ultérieure, alors cette production est produite le plus tard possible avant t en respectant les décisions de lancement de production prises auparavant. Parce que produire et stocker à des périodes antérieures coûte plus cher que de produire à des périodes ultérieures. Un raisonnement similaire peut être effectué dans le cas du backlogging.

Quand les conditions de Wagner et Whitin sont satisfaites, Aggarwal et Park [5], Federgruen et Tsur [59] et Wagelmans *et al.* [188] ont montré que les algorithmes de programmation dynamique pour les problèmes ULS, et ULS avec des startups peuvent s'exécuter en $O(T)$.

2.1.3.2 Méthodes des coupes

Pour la résolution des problèmes de lot-sizing, les chercheurs se sont intéressés à la description de l'enveloppe convexe de l'ensemble des solutions réalisables en proposant des inégalités valides. Ainsi, en rajoutant toutes ces inégalités valides, la solution optimale obtenue en relaxant les contraintes d'intégrité est entière. Pour une description complète des méthodes de coupes le lecteur peut se référer à [194]. Le tableau 2.3 résume les principales références bibliographiques utilisant des méthodes de coupes pour la résolution de problèmes de lot-sizing à une seule référence.

Problème traité	Références
Problèmes de lot-sizing à une seule référence, sans contrainte de capacité	[9], [14], [15], [112], [146], [152], [150], [152], [177], [181], [191], [193]
Problèmes de lot-sizing à une seule référence à capacité constante	[151], [152], [40], [10], [111]
Problèmes de lot-sizing à une seule référence sous contraintes de capacité	[119], [122], [147], [149]

TAB. 2.3 – Problèmes de lot-sizing à une seule référence : Méthodes de coupes

Plusieurs auteurs se sont intéressés à la description partielle ou totale de l'enveloppe convexe de différents problèmes de lot-sizing. Les premiers travaux sont ceux de Barany *et al.* [14, 15]. Les auteurs ont montré que la famille des inégalités (l, S), avec des inégalités valides standards issues de la relaxation linéaire continue de programmes en variables mixtes, décrivent l'enveloppe convexe des solutions réalisables du problème ULS. Les inégalités (l, S) ont la forme suivante :

$$\sum_{t\in\bar{S}} x_t + \sum_{t\in S} \delta_{t,l} y_t \geq \delta_{1,l} \tag{2.7}$$

où $l \in \{1, \ldots, T\}$, $S \subseteq \{1, \ldots, l\}$, $\bar{S} = \{1, \ldots, l\} \setminus S$ et $\delta_{i,j} = \sum_{t=i}^{j} d_t$. L'idée derrière les inégalités (l, S) est qu'en supposant qu'aucun lancement de production ne soit fait durant les périodes de S, alors la demande totale $\delta_{1,l}$ doit être produite à des périodes appartenant à $\bar{S}$ (i.e. $\sum_{t\in\bar{S}} \delta_{t,l} y_t \geq \delta_{1,l}$). En supposant qu'un lancement de production soit effectué à une période $k \in S$, alors la demande $d_{1,k-1}$ doit être produite à des périodes dans $\bar{S}$. Il est cependant possible que la demande $\delta_{k,l}$ soit produite à une seule période dans S, ce qui explique les coefficients des variables y_t. La classe des inégalités (l, S) est exponentielle. Les auteurs ont également proposé un algorithme efficace pour résoudre en un temps polynomial le problème de séparation.

Pochet et Wolsey [150] ont étendu la classe des inégalités (l, S) pour le problème ULS avec backlogging. Ces inégalités décrivent l'enveloppe convexe des solutions réalisables du problème. Les mêmes auteurs [152] ont montré que lorsque les coûts satisfont les conditions de Wagner et Whitin, il est possible de résoudre le problème par la programmation linéaire en rajoutant un sous-ensemble de ces inégalités. La cardinalité de ce sous-ensemble est exponentielle par rapport à la taille du problème, mais le problème de séparation pour cet ensemble est solvable en temps polynomial. Dans le cas de la présence de la propriété de coûts de Wagner-Whitin, des inégalités valides pour de tels problèmes sont présentées dans Pereira et Wolsey [146] pour le problème ULS, et dans Pochet et Wolsey [152] pour les problèmes ULS avec et sans le backlogging, avec startup, et avec des capacités constantes.

Pour les problèmes ULS avec des coûts de startup, Van Hoesel *et al.* [177] ont généralisé la classe des inégalités valides (l, S) en une classe d'inégalités (l, R, S), qui avec les facettes standard de la relaxation linéaire continue décrivent l'enveloppe convexe des solutions réalisables. Pour plus de détails sur la modélisation des startup le lecteur peut se reporter à la page 141.

Pour le problème ULS avec des contraintes de startup, des inégalités valides ainsi que des reformulations sont présentées dans Wolsey [191] pour le cas sans contrainte de capacité. Wolsey [193] a étudié le problème avec des changeovers en proposant différentes formulations et inégalités valides.

Pochet et Wolsey [151] ont introduit les inégalités (k, l, S, I), et ils ont montré qu'elles définissent toutes les facettes non triviales de l'enveloppe convexe du problème CCLS. Elles ont la forme suivante :

$$s_{k-1} + \sum_{j \in U} x_j + \sum_{i \in V} B_j y_j \geq B_0 \tag{2.8}$$

où $1 \leq k \leq l \leq T$, (U, V) est une partition de $[k, \ldots, l]$, et $B_j \in \mathbb{R}_+$, $j \in V$. Quand les coûts satisfont aux conditions de Wagner et Whitin, les auteurs [152] ont montré qu'il était possible de résoudre le problème en rajoutant ces inégalités valides avec des inégalités valides standard de programmes linéaires mixtes. Constantino [40] a étendu ces inégalités pour le même problème avec des coûts de startup. Ces inégalités ont également été généralisées pour le problème de localisation d'entrepôts Aardal *et al.* [1].

Leung *et al.* [119], Marchand [122] et Pochet [147] ont proposé plusieurs classes d'inégalités valides pour le problème de lot-sizing à capacité finie. D'après les résultats expérimentaux, Leung *et al.* ont montré l'efficacité de leurs inégalités valides. La classe d'inégalités valides proposée par Pochet [147] est basée les inégalités de flots couvtants, tandis que la classe des inégalités valides proposée par Marchand [122] est basée sur des inégalités valides du problème de sac-à-dos.

Atamtürk et Muñoz [10] ont étudié le polytope du problème CLS. Les auteurs ont identifié les inégalités valides définissant des facettes qui coupent tous les points extrêmes fractionnaires de la relaxation linéaire du problème. Ils ont également proposé un algorithme de séparation polynomial dans le cas où les capacités sont constantes.

Loparic *et al.* [111] ont proposé des inégalités valides en se basant sur la structure de sac à dos du problème CLS. Les mêmes auteurs [112] ont proposé des inégalités valides pour le problème de lot-sizing dans lequel ils considèrent des variables de ventes au lieu de demandes fixes et des bornes inférieures sur les stocks. Des études polyédriques sur les problèmes de lot-sizing avec des contraintes de setup sur les stocks ont été proposées par Atamtürk et Kucukyavuz [9] et Van Vyve et Ortega [181].

Plusieurs études ont porté sur la recherche de structures polyédriques et d'inégalités valides pour différents modèles de lot-sizing. Une synthèse de l'ensemble des résultats de plusieurs inégalités valides pour une variété de problème de lot-sizing, avec contraintes de capacités, startups ou problèmes multi-niveaux est donné par Pochet et Wolsey [149].

2.1.3.3 Les formulations renforcées (*Strong formulations*)

Le principe de ces approches est de reformuler les modèles mathématiques en variables mixtes et de résoudre leur relaxation linéaire continue. Le but est d'avoir une solution du problème relaxé où toutes les contraintes d'intégrité sont respectées. De telles reformulations ont été développées pour le problème ULS.

Formulation basée sur le problème de localisation d'entrepôts L'une des premières reformulations du problème de lot-sizing a été proposée par Krarup et Bilde [101]. Les auteurs ont proposé la formulation du problème ULS en un problème de localisation d'entrepôts (Facility Location Problem). Les variables de production sont modélisées plus finement en introduisant la période à laquelle ces productions sont consommées. Cette modélisation nous

permet d'éliminer les variables de stock et les variables de production du modèle en les exprimant en fonction de ces nouvelles variables. Le coût associé à cette nouvelle variable est donc la somme du coût de production à la période de production concerné et du coût de stockage entre la période de production et la période de consommation. Les auteurs ont montré que la relaxation linéaire continue du problème ULS ainsi formulé possède une solution optimale où toutes les contraintes d'intégrité sont respectées.

Formulation basée sur le problème du plus court chemin Eppen et Martin [57] ont décrit une autre reformulation du problème ULS. Cette reformulation est une représentation en réseau de transport de la formule de récursivité du programme dynamique proposé par Wagner et Whitin [189]. Les auteurs ont considéré un graphe orienté de $T+1$ nœuds $0, 1, \ldots, T$. Un arc (t, t') existe pour tout $t < t'$. Le coût associé à chaque arc (t, t') représente le coût de lancement de la production à la période $t + 1$ et de la satisfaction de la demande du tronçon d'horizon $[t + 1, \ldots, t']$. Le plus court chemin du nœud 0 au nœud T dans ce réseau est équivalent à la solution optimale du problème ULS. Les auteurs ont montré que la relaxation linéaire continue du problème ULS formulé en utilisant ce principe possède une solution optimale où toutes les contraintes d'intégrité sont respectées.

Pochet et Wolsey [150] ont utilisé les reformulations en localisation d'entrepôts, en plus court chemin pour obtenir des programmes linéaires à $O(T^2)$ variables et contraintes pour le problème ULS avec backlogging. La technique de reformulation en un problème de plus court chemin proposée par Martin [125] et Eppen et Martin [57] peut également être utilisée pour résoudre le problème CCLS avec $O(T^4)$ variables et contraintes. Pochet et Wolsey [151] ont proposé une formulation en programme linéaire avec $O(T^3)$ variables et contraintes pour résoudre le problème CCLS, leur problème correspond à une séquence de plus courts chemins sur des intervalles de régénération définie par Florian et Klein [64].

Formulation basée sur la notion d'echelon stock La formulation du problème de lot-sizing à plusieurs niveaux de production ne contient pas explicitement des sous-problèmes ULS (excepté pour les produits finis). Ceci est dû à la présence de deux types de demandes, dépendantes et indépendantes. En utilisant la notion d'*echelon stock*, introduite par Clark et Scart [34], il est possible d'obtenir une formulation qui contient explicitement des sous-problèmes ULS. En effet, cette dernière est effectuée en remplaçant les variables de stock par des variables d'echelon stock. Ces nouvelles variables représentent le stock total du composant dans le système de production. Cette formulation permet d'utiliser des algorithmes de décomposition basés sur les résultats connus du problème ULS. Pour plus de détails sur les problèmes de lot-sizing à plusieurs niveaux de production, le lecteur peut se reporter à la page 143.

2.1.3.4 Approches branch-and-bound

La méthode de branch-and-bound consiste à déterminer une solution optimale à travers un arbre d'énumération par séparation et évaluation progressive. Pour une description détaillée de la méthode de branch-and-bound, le lecteur peut se référer à [194].

Jacobs et Khumawala [85] ont développé un algorithme polynomial basé sur la méthode de branch-and-bound comme alternative à la programmation dynamique pour résoudre le problème ULS. Baker *et al.* [13] ont proposé un algorithme de branch-and-bound pour résoudre

le problème CLS avec des coûts de production et de stockage constants. Pour la résolution du problème CLS avec des coûts de production et de stock concaves, Lotfi et Yoon [113] ont développé un algorithme de branch-and-bound basé sur les propriétés de décomposition énoncées par Florian et Klein [64]. Dans le cas où les capacités sont constantes Chung *et al.* [32] ont proposé une méthode de branch-and-bound couplée à un algorithme de programmation dynamique.

Afin de résoudre les problèmes de lot-sizing à plusieurs niveaux de production et sans contrainte de capacité, plusieurs auteurs ont proposé des approches basées sur la méthode de branch-and-bound, on peut citer les travaux de Afentakis et Gavish [3], Erenguc [58], Kirca [97], Robinson et Gao [155].

2.1.4 Méthodes approchées

Plusieurs auteurs ont proposé des heuristiques afin de fournir de bonnes solutions réalisables au problème ULS. Ces méthodes fonctionnent période par période et se basent sur des régles de priorité de coûts. On peut citer la méthode du moindre coût par période (*least cost per period*) proposée par Silver et Meal [162], *The EOQ Based Period Order Quantity heuristic* Berry [20], la méthode du moindre coût unitaire proposée par Gorham [72] (*Least Unit Cost*), la méthode *part period balance criterion* De Matteis et Mendoze [126], la méthode de la différence des coûts marginaux Groff [73] et la technique de placement et de dimensionnement de lots de Coleman et McKnew [38]. Une comparaison de ces méthodes peut être trouvée dans Coleman [37] et Simpson [163]. Axsäter [11], Bitran *et al.* [21] et Vachani [174] ont montré des résultats sur des bornes pire cas de quelques unes de ces heuristiques.

Maes *et al.* [117] ont proposé une heuristique basée sur la programmation linéaire pour résoudre le problème de lot-sizing à capacité finie et à plusieurs niveaux de production. Les auteurs ont utilisé la formulation basée sur le problème de localisation [101].

Hardin *et al.* [77] ont analysé plusieurs algorithmes afin de fournir des bornes supérieures et inférieures pour une variante du problème CLS. Les auteurs ont proposé plusieurs algorithmes basés sur la relaxation linéaire continue du problème, ils ont également fait une analyse pire cas.

2.2 Problèmes à plusieurs références

Les méthodes les plus fréquemment utilisées pour résoudre les problèmes de lot-sizing à plusieurs références sont les méthodes basées sur la relaxation lagrangienne, les méthodes de branch-and-cut, les méthodes basées sur les modèles renforcés et différentes heuristiques de construction de solutions réalisables. Ce sont généralement des méthodes approchées. Dans ce qui suit, nous commençons par présenter le modèle mathématique de base décrit dans Miller *et al.* [131] et Trigeiro *et al.* [172]. Ce problème est noté MCLS[8]. Nous présentons alors les approches de résolution exactes et approchées appliquées à des modèles MCLS étendus.

2.2.1 Modélisation du problème MCLS

Manne [121] est parmi les premiers à modéliser le problème de lot-sizing à plusieurs références soumis à des contraintes de capacité. L'auteur a proposé un modèle basé sur les

[8]MCLS est une abréviation en anglais de : Multi-item Capacitated Lot-sizing problem with Setup times.

conditions d'optimalité de Wagner et Whitin ($s_{t-1}x_t = 0,\ \forall t$), c'est-à-dire qu'on ne produit jamais à une période en ayant un stock non nul provenant de la période précédente. Nous présentons dans ce qui suit la modélisation mathématique du problème MCLS. Rappelons que le problème consiste à déterminer un plan de production d'un ensemble de N références, pour un horizon de planification constitué de T périodes. Les capacités de production doivent être respectées. De plus, les cadences de production dépendent de la machine, mais également des références fabriquées, la cadence peut être définie par la mesure heure/unité produite, c'est ce que l'on appelle un besoin variable unitaire, ainsi que des temps de non productivité induits par les changements de références, ce temps est appelé besoin fixe ou setup. La capacité des ressources est généralement exprimée en nombre d'heures disponibles dans la période. Ces ressources peuvent être matérielles ou humaines. Le lancement de la production d'une référence à une période donnée induit un coût fixe ainsi qu'un coût variable unitaire.

La demande est exprimée à partir des commandes fermes complétées par les commandes prévisionnelles, à la période et pour chaque référence. Les demandes sont fermes à court terme, puis prévisionnelles à plus long terme.

Une particularité importante des problèmes de planification de production est la variation de la demande dans le temps. Ces variations poussent le décideur à anticiper la fabrication afin de les satisfaire. Afin d'amortir les coûts fixes, le décideur est également contraint de lancer des campagnes de production. Ces deux décisions ainsi que les capacités limitantes entraînent l'obligation de constituer des stocks. On introduit donc un coût de stockage unitaire pour chaque référence et à chaque période.

La modélisation du problème MCLS se base sur les hypothèses suivantes :

- Tous les calculs se font à la fin de chaque période, tel que l'incrémentation des stocks, satisfaction de la demande...) ;
- Les coûts ainsi que les besoins fixes sont comptabilisés à chaque période de production, même si les productions sont lancées à des périodes consécutives.

Afin de décrire le modèle mathématique du problème MCLS, on utilise les notations suivantes :

Les paramètres :
d_{it} : Demande (commandes et prévisions) pour la référence i à la période t.
f_{it} : Besoin fixe en ressources de la référence i à la période t.
v_{it} : Besoin variable en ressources de la référence i à la période t.
c_t : Capacité disponible à la période t.
α_{it} : Coût de production unitaire variable de la référence i à la période t.
β_{it} : Coût fixe relatif à un lancement de la référence i à la période t.
γ_{it} : Coût unitaire de stockage pour la référence i à la période t.

Les variables :
x_{it} : Quantité produite pour la référence i à la période t.
$y_{it} = \begin{cases} 1 \text{ s'il y a production de la référence } i \text{ à la période } t \text{ (i.e. si } x_{it} > 0). \\ 0 \text{ sinon} \end{cases}$
s_{it} : Valeur du stock en fin de période t pour la référence i.

En utilisant ces variables et paramètres, nous formulons le problème MCLS comme suit :

$$\min \sum_{i,t} \alpha_{it} x_{it} + \sum_{i,t} \beta_{it} y_{it} + \sum_{i,t} \gamma_{it} s_{it} \tag{2.9}$$

s.c :

$$x_{it} - s_{it} + s_{i(t-1)} = d_{it}, \qquad \forall i, \forall t \tag{2.10}$$

$$\sum_{i=1}^{N} v_{it} x_{it} + \sum_{i=1}^{N} f_{it} y_{it} \leq c_t, \qquad \forall t \tag{2.11}$$

$$x_{it} \leq M y_{it}, \qquad \forall i, \forall t \tag{2.12}$$

$$x_{it}, s_{it} \geq 0, \qquad \forall i, \forall t \tag{2.13}$$

$$y_{it} \in \{0, 1\}, \qquad \forall i, \forall t \tag{2.14}$$

La fonction objectif (2.9) minimise le coût total induit par le plan de production à savoir les coûts de production et de stockage, ainsi que les coûts fixes de lancement. La contrainte (2.10) exprime la conservation des flux à travers l'horizon. La contrainte (2.11) exprime le fait que le plan que nous souhaitons calculer doit être à capacité finie. En effet, pour la réalisation d'un plan, nous disposons d'une quantité de ressources qui sera consommée par la production d'une ou plusieurs références. La consommation totale doit rester inférieure à la capacité disponible. La contrainte (2.12) permet de modéliser la condition suivante : s'il y a lancement de production, alors la quantité produite ne devra pas dépasser un majorant de la production M. Celui-ci représente le minimum entre la quantité maximale de la référence pouvant être produite et la demande totale sur le tronçon de l'horizon $[t, .., T]$. La contrainte (2.13) signifie que les variables x_{it} et s_{it} sont continues non négatives pour toute référence i, pour chaque période t. La dernière contrainte (2.14) exprime le fait que y_{it} est une variable binaire pour toute référence i, pour chaque période t.

Ce modèle est appelé également formulation agrégée, car la production d'une référence n'est définie que par sa période de production, contrairement à la formulation basée sur le problème de localisation d'entrepôts [101]) où la production est définie par sa période de production et par sa période de consommation.

2.2.2 Complexité

Chen et Thizy [31] ont montré que le problème MCLS est NP-difficile au sens fort. Le problème de décision du même problème a été prouvé NP-complet par Maes *et al.* [117].

2.2.3 Méthodes exactes

Le problème MCLS étant NP-Difficile au sens fort [31], les auteurs se sont principalement intéressés aux méthodes approchées. Nous décrivons ici les travaux portant sur les méthodes exactes, principalement basées sur les méthodes de branch-and-cut. D'autre approches tel que la decomposition de Dantzig-Wolf ainsi que la génération de colonnes sont abordées. Les principales références bibliographiques proposant des méthodes exactes pour des problèmes de lot-sizing à plusieurs références sont résumées dans le tableau 2.4.

Approches	Références
Branch-and-cut	[14], [18], [19], [41], [119], [131], [135], [132], [133], [134], [183]
Autres méthodes (Decomposition de Dantzig-Wolf, génération de colonnes, branch-and-bound)	[12], [28], [30], [54], [76], [88], [90], [99], [108], [121]

TAB. 2.4 – Problèmes de lot-sizing à plusieurs références : Méthodes exactes

2.2.3.1 Approches branch-and-cut

La méthode de branch-and-cut est une méthode de branch-and-bound couplée à une méthode de coupes, utilisée pour résoudre les problèmes linéaires en nombre entiers. En effet, à chaque nœud de l'arbre de recherche, les relaxations linéaires continues sont renforcées par des inégalités valides afin d'améliorer la qualité de la borne inférieure. Le fait d'avoir une meilleure borne inférieure du problème au cours du parcours de l'arbre de recherche, permet de réduire la taille de ce dernier lors de la recherche d'une solution optimale ou réalisable. Si les inégalités valides ne sont rajoutées qu'au premier nœud (ou racine) de l'arbre de recherche, la méthode est appelée cut-and-branch. Pour une présentation complète de cette méthode, le lecteur peut se référer à [194].

Crowder *et al.* [42] sont parmi les premiers à utiliser l'approche de branch-and-cut pour résoudre des problèmes à variables binaires. On peut citer également les travaux de Grötschel *et al.* [74] et Padberg et Rinaldi [145].

Plusieurs auteurs ont montré que l'utilisation des inégalités (l, S), introduites par Barany *et al.* [14, 15], permettaient d'améliorer la borne inférieure et de réduire considérablement le pourcentage d'optimalité des problèmes MCLS. Belvaux et wolsey [18] ont proposé un système de branch-and-cut incluant des prétraitements et des inégalités valides pour une variété de modèles de lot-sizing.

Vanderbeck [183] a développé une méthode de branch-and-price qui combine une méthode de génération de colonnes et une méthode de coupes pour la résolution du problème MCLS avec startup.

Constantino [41] a étudié le problème MCLS avec des bornes inférieures sur les variables de production. L'auteur a proposé des inégalités valides pour ce problème en se basant sur des relaxations sur une seule période. Des résultats expérimentaux intéressants ont été donnés.

D'autres inégalités valides pour le problème MCLS ont été proposées par Miller *et al.* [132, 133, 135], les auteurs ont défini des inégalités valides en relaxant le problème sur une seule période en considérant un stock précédent, le stock précédent est un terme de l'équation de flux assimilé à la variable de stock entrant. Ils ont reformulé les contraintes de ressources en contraintes de sac à dos afin d'utiliser les résultats de Marchand et Wolsey [123] sur le problème de sac à dos continu ainsi que des procédures de liftings séquentiels. Les auteurs ont également montré que ces inégalités représentaient des facettes du polyèdre des solutions réalisables sous certaines conditions. Miller [131] et Miller *et al.* [134] ont présenté un algorithme de branch-and-cut efficace pour le problème MCLS en utilisant des inégalités valides de la relaxation sur une seule période avec un stock précédent.

2.2.3.2 Autres approches

Dzielinski et Gomory [54] et Manne [121] ont étudié la décomposition de Dantzig-Wolfe pour le problème MCLS sans besoin fixe. Les auteurs ont montré que la plupart des variables

étaient entières si le nombre de références est plus important que le nombre de périodes. Kleindorfer et Newson [99] ont proposé une décomposition lagrangienne pour le problème MCLS basée sur la relaxation des contraintes de ressources, ils ont montré que leur approche fournissait la même borne inférieure que celle trouvée par la décomposition de Dantzig-Wolf proposée par Manne [121] et Dzielinski et Gomory [54]. En effet, les deux méthodes sont duales.

Haase et Kimms [76] ont proposé un algorithme de branch-and-bound pour la résolution de problème MCLS avec des contraintes qui tiennent compte de l'ordre de passage des références sur la ressource. Les variables de lancement ainsi que les séquences de passages sont fixées durant le parcours de l'arbre de recherche.

Degraeve et Jans [88] ont proposé une méthode de décomposition de Dantzig et Wolfe basée sur la relaxation des contraintes de flux pour résoudre le problème MCLS. Ils ont résolu les sous-problèmes issus en utilisant les propriétés linéaires du problème de sac-à-dos à choix multiples pour trouver une borne inférieure. Ils ont montré que leurs bornes sont meilleures que celles proposées par Belvaux et Wolsey [18] et Miller *et al.* [134].

Lasdon et Terjung [108] ont proposé une reformulation du programme linéaire pour fournir une approche basée sur une technique de génération de colonnes plus efficace. Chen et Thizy [30] ont proposé une méthode de génération de colonnes pour le problème MCLS sans temps de setup. Bahl [12] et Cattrysse *et al.* [28] ont également utilisé la méthode de génération de colonnes pour la résolution du problème MCLS. Kang *et al.* [90] ont étendu le problème MCLS en rajoutant la notion setups dépendants de la séquence de passage sur des machines parallèles. Les auteurs ont proposé une méthode basée sur la génération de colonnes pour résoudre le problème.

2.2.4 Méthodes approchées

Puisque le problème MCLS est NP-Difficile au sens fort [31], les chercheurs se sont intéressés aux méthodes approchées. L'une des méthodes les plus efficaces est la relaxation lagrangienne. En effet, en relaxant les contraintes de ressource du problème MCLS, on retombe sur plusieurs sous-problèmes de type ULS faciles à résoudre. Des heuristiques lagrangiennes sont utilisées pour trouver une borne supérieure au problème. Les autres approches sont des hybridations d'algorithmes exacts et d'heuristiques (ex. décomposition, heuristiques gloutonnes, méta-heuristiques...). Le tableau 2.5 rassemble les principales références utilisant des méthodes approchées pour les problèmes de lot-sizing à plusieurs références.

Approches	Références
Relaxation lagrangienne	[3], [4], [31], [48], [53], [62], [92], [93], [169], [171], [172], [173]
Heuristique à base de MIP	[35], [95], [129], [164], [166]
Heurisques : période par période, référence par référence, niveau par niveau	[2], [25], [39], [51], [56], [66], [98], [107], [118], [127], [197]
Agrégation/Désagrégation	[24], [67], [79], [128], [186]
Méta-heuristiques	[16], [46], [47], [63], [71], [81], [82], [83], [84], [89], [100], [102], [103], [104], [130], [140], [141], [142], [143], [158], [144], [168], [196]

TAB. 2.5 – Problèmes de lot-sizing à plusieurs références : Méthodes approchées

2.2.4.1 Relaxation lagrangienne

La relaxation lagrangienne est une méthode qui permet de trouver de bonnes bornes inférieures pour les problèmes de minimisation ou bornes supérieures pour des problèmes de maximisation. Elle permet également de trouver des solutions réalisables en appliquant des heuristiques lagrangiennes.

L'approche de la relaxation lagrangienne consiste à relaxer un sous-ensemble de contraintes tout en pénalisant leur violation dans la fonction objectif en leur associant un multiplicateur lagrangien. Un avantage de cette transformation est que pour tout vecteur de multiplicateurs, la solution optimale du problème relaxé est une borne inférieure de la solution optimale du problème initial (cas d'un problème de minimisation). Si le problème initial est un programme linéaire en variables mixtes, alors sa relaxation lagrangienne est au moins aussi bonne que sa relaxation linéaire. Le problème dual du problème relaxé consiste à déterminer la valeur des multiplicateurs qui maximiseront la fonction objectif du problème relaxé, et par conséquent la valeur de la borne inférieure. Dans la pratique, des méthodes itératives sont utilisées pour résoudre le problème dual. La méthode la plus utilisée est celle du sous-gradient. Pour une description détaillée de la méthode de relaxation lagrangienne, le lecteur peut se référer à [136].

La solution du problème relaxé est souvent exploitée afin de construire une solution réalisable au problème initial. Cette solution est obtenue en exécutant des heuristiques lagrangiennes qui se basent sur la solution fournie par la relaxation lagrangienne et les coûts duaux. Cette approche s'est avérée d'une grande efficacité pour les problèmes de lot-sizing à plusieurs références.

Thizy et Van Wassenhove [171] ont utilisé la relaxation lagrangienne afin de trouver une borne inférieure au problème MCLS. Le problème dual est résolu en utilisant la méthode du sous-gradient. Les auteurs ont construit une borne supérieure en fixant les variables de lancement à partir de la solution des sous-problèmes de Wagner et Whitin, et en résolvant des problèmes de flot à coût minimum afin de trouver les quantités à produire.

Afentakis *et al.* [4] ont résolu le problème multi-étapes sans contrainte de capacité en le reformulant avec la notion de d'echelon stock définie page 18. Afentakis et Gavish [3] ont montré des résultats expérimentaux sur des problèmes de lot-sizing à plusieurs étapes de production et à capacité finie. De façon similaire, ces résultats ont été obtenus en reformulant les problèmes en se basant sur la notion de d'echelon stock, et en les résolvant en utilisant la relaxation lagrangienne et des méthodes de programmation linéaire mixte.

Trigeiro *et al.* [172] sont les premiers à présenter un algorithme efficace qui prend en charge les besoins fixes (setups). Les auteurs ont utilisé la relaxation lagrangienne des contraintes de capacité pour obtenir une bonne borne inférieure. Une borne supérieure est obtenue en utilisant une heuristique lagrangienne. Celle-ci tente de déplacer les quantités produites vers les périodes futures puis vers les périodes passées afin d'éliminer les dépassements des capacités. Le processus s'interrompt si toutes les contraintes de capacité sont satisfaites, ou si les contraintes de capacité restent violées après deux essais de déplacement.

Chen et Thizy [31] ont traité le problème de lot-sizing à plusieurs références avec des contraintes de capacité, mais sans temps de setup. Les auteurs ont proposé plusieurs approches de relaxation lagrangienne ainsi qu'une formulation basée sur le problème du plus court chemin. Ils ont également effectué des comparaisons entre la programmation linéaire, la génération de colonnes et la relaxation lagrangienne utilisant la méthode du sous-gradient. La génération de colonnes a donné les meilleurs résultats.

Diaby *et al.* [48] ont appliqué plusieurs schémas de relaxation lagrangienne pour le MCLS avec des overtimes. Les overtimes sont des capacités supplémentaires avec un coût associé dans la fonction objectif. Ils ont résolu le problème dual en utilisant la méthode du sous-gradient. La première décomposition est obtenue en relaxant les contraintes de capacité. La seconde décomposition est effectuée en dualisant les contraintes de flux, les sous-problèmes sont des problèmes de sac-à-dos continus. Les auteurs ont proposé un algorithme de branch-and-bound spécialisé pour résoudre les sous-problèmes. Pour chaque relaxation, ils ont développé un algorithme de branch-and-bound où les bornes inférieures sont déterminées par relaxation lagrangienne.

Du Merle *et al.* [53] ont proposé une méthode de relaxation lagrangienne comparable à celle de Trigeiro *et al.* [172] en utilisant une méthode du point intérieur et des plans sécants au lieu d'utiliser la méthode classique du sous-gradient pour trouver de nouveaux coûts duaux.

Toledo et Armentano [173] ont traité le problème de lot-sizing à plusieurs références avec machines parallèles à capacité constante, ils ont utilisé une approche de relaxation lagrangienne afin de résoudre le problème.

Les bornes lagrangiennes sont souvent utilisées dans des méthodes de branch-and-bound. On peut notamment citer les travaux de Afentakis *et al.* [4], Fleischmann [62], Karmarkar *et al.* [92] et Karmarkar et Schrage [93]. Tempelmeier et Derstroff [169] ont proposé une méthode de relaxation lagrangienne pour le problème de lot-sizing à plusieurs étapes de production, cette méthode est basée sur la relaxation des contraintes de ressources et des contraintes de flux.

2.2.4.2 Autres heuristiques

Le temps d'exécution long ainsi que la mémoire requise pour une résolution exacte, rendent les algorithmes exacts inutilisables pour des problèmes réels. Plusieurs recherches ont été entreprises pour trouver une solution réalisable au problème MCLS. Les premières méthodes proposées sont des heuristiques juste à temps (JIT : Just-in-time). Ces heuristiques opèrent en traitant le problème période par période. Le parcours des périodes peut être fait vers le passé, vers le futur, vers le futur puis vers le passé, ou bien vers le passé puis vers le futur. En effet, le parcours vers le passé permet de créer un plan pour chaque période et d'anticiper ce qui ne peut être produit. Le parcours vers le futur permet de satisfaire les demandes en retard lorsque le backlog est autorisé, il permet également de lisser la production en éliminant le surplus de production. La méthode est basée sur un modèle de priorités et non sur un modèle de coûts. En effet, les références sont traitées par rapport à leurs priorités. Les références les plus prioritaires sont traitées en premier et les moins prioritaires en dernier. L'efficacité de ces méthodes réside dans le bon renseignement de ces priorités.

Plusieurs auteurs ont proposé des heuristiques hybrides mixant des approches de branch-and-bound à des heuristiques JIT. Ces méthodes sont appelées heuristiques à base de MIP. Contrairement à la méthode JIT, où les périodes sont traitées une par une, les heuristiques à base de MIP fonctionnent en traitant séquentiellement groupes de périodes par groupe de périodes. Parmi ces travaux on peut citer Clark [35], Kelly [95], Mercé et Fontan [129], Stadtler [164] et Suerie et Stadtler [166].

Les heuristiques proposées dans la littérature peuvent être classées en trois catégories : les heuristiques période par période, les heuristiques référence par référence, et les heuristique niveau par niveau.

Les heuristiques période par période fonctionnent de manière récursive. En effet, la solution d'un problème à t périodes est utilisée pour trouver une solution à un problème à $t+1$ périodes. Cette procédure est répétée jusqu'à ce que l'on obtienne une solution pour le problème à T périodes. De telles heuristiques ont été utilisées par Afentakis [2] pour le problème de lot-sizing à plusieurs étape de production. Elles ont égalment été appliquées au problème MCLS Dixon et Silver [51], Eisenhut [56], Lambrecht et Vanderveken [107] et Maes et Van Wassenhove [118]. L'heuristique doit s'assurer de la faisabilité de la solution des sous-problèmes par rapport aux contraintes de capacité.

Les heuristiques référence par référence fonctionnent par groupe de références. A chaque étape un groupe de références est planifié jusqu'à ce qu'un plan de production soit obtenu pour toutes les références. Kirca et Kökten [98] ont utilisé une telle méthode pour résoudre le problème MCLS.

Les méthodes niveau par niveau sont basées sur la décomposition d'un problème à plusieurs étapes de production en plusieurs sous-problèmes à un seul niveau de production. Cette approche a été utilisée par Blackburn et Millen [25], Coleman et McKnew [39] et McLaren [127] pour résoudre des problèmes de lot-sizing à plusieurs niveaux de production et sans contrainte de capacité. Pour le problème MCLS à plusieurs niveaux, Gabbay [66] a appliqué la méthode en présence de plusieurs contraintes de ressource. Zahorik *et al.* [197] ont également appliqué cette méthode pour le problème à une seule contrainte de capacité dans un système de production en série.

Plusieurs auteurs se sont intéressés à l'application des approches hiérarchiques (dites méthodes d'agrégation et de désagrégation) aux problèmes de la planification de la production. Nous pouvons citer les travaux de Bitran and Hax [24], Gfrerer et Zäpfel [67], Hetreux *et al.* [79], Mehra et *et al.* [128] et Vicens *et al.* [186]. Dans ces approches, la formulation monolithique est remplacée par une séquence de modèles et une hiérarchie de décisions à prendre. Les décisions agrégées sont prises en premiers, et sont ensuite imposées comme contraintes aux modèles plus détaillés. En retour, les décisions détaillées permettent d'évaluer la qualité des décisions agrégées. Les décisions en haut de la hiérarchie sont basées sur des modèles agrégés. Le succès de cette approche réside essentiellement sur l'uniformité entre l'agrégation et la désagrégation ainsi que l'interaction entre les modèles aux différents niveaux de la hiérarchie.

Plusieurs auteurs se sont intéressés à l'application des méta-heuristiques à différents problèmes de lot-sizing. Pour le problème MCLS on peut citer, Fleischmann et Meyr [63], Gopalakrishnan *et al.* [71], Hindi [81], Hung *et al.* [82], Hung *et al.* [84], Kohlmorgen *et al.* [100], Laguna [104], Meyr [130], Özdamar et Bozyel [144], Özdamar *et al.* [143] et Özdamar et Birbil [142]. Pour le MCLS en multi-niveau on peut citer, Barbarosoglu et Özdamar [16], Dellaert et Jeunet [46], Dellaert *et al.* [47], Hung et Chien [83], Kuik et Salomon [102], Kuik *et al.* [103], Özdamar et Barbarosoglu [140], Özdamar et Barbarosoglu [141], Salomon *et al.* [158], Tang [168] et Xie et Dong [196]. Pour un état de l'art sur l'application des méta-heuristiques aux problèmes de lot-sizing, le lecteur peut se référer à Jans et Degraeve [89].

2.3 Conclusion

Nous avons présenté dans ce chapitre un état de l'art non exhaustif des problèmes de lot-sizing. Nous avons axé cette étude sur les approches de résolution abordées dans ma thèse. En effet, une importance particulière a été accordée aux approches polyédriques, aux approches de programmation dynamique ainsi qu'aux approches heuristiques. Nous avons proposé une

classification des approches par type de problèmes (une seule référence, plusieurs références), puis par méthode de résolution (méthodes exactes, méthodes approchées). Toutefois, nous avons accordé plus d'attention aux problèmes de lot-sizing à un seul niveau de production, ainsi qu'aux problèmes à périodes longues. Cet état de l'art permet de situer nos approches par rapport à l'existant. En se basant sur le modèle MCLS décrit page 19, nous décrivons dans le chapitre suivant la structure de l'horizon de planification que nous utilisons, nous présentons également les principales caractéristiques des problèmes de lot-sizing auxquels nous nous sommes intéressés dans le cadre de cette thèse.

Chapitre 3

Modélisation des problèmes de lot-sizing MCLS

Introduction

Dans ce chapitre, nous allons décrire les principales caractéristiques des problèmes de lot-sizing auxquels nous nous sommes intéressés dans le cadre de cette thèse. Avant de décrire les modèles mathématiques, nous commençons par présenter la structure de l'horizon de planification. Rappelons que les modèles que nous proposons permettent de représenter des situations rencontrées dans des milieux industriels. Chaque particularité sera expliquée, modélisée et accompagnée d'un exemple issu d'une situation réelle. Nos modèles sont des extensions du modèle classique de lot-sizing à plusieurs références avec des contraintes de capacité et des temps de setup, noté MCLS tel décrit au chapitre 2, section 2.2.1. Les principales caractéristiques abordées sont : les ruptures sur la demande, les déficits sur le stock de sécurité, les groupes de références, plusieurs ressources, plusieurs gammes de production, la production minimale, le lancement minimum de production, la tailles de lot pour les références et les groupes de références. Chacune d'elle sera développée plus en détail dans ce qui suit.

3.1 Horizon de planification

La planification de la production s'effectue sur un horizon de planification discret. En effet, les clients passent leurs commandes pour une période donnée. En fonction du carnet de commandes ainsi établi et éventuellement complété par des prévisions de ventes, nous cherchons alors à déterminer les quantités à produire sur un horizon donné découpé en périodes. La saisonnalité de la demande et la limitation des capacités imposent une anticipation et un stockage éventuel des produits. Du point de vue de la granularité de la maille temporelle, la planification de production se fait le plus souvent sur un horizon de planification moyen terme (entre un an et deux ans), ce dernier peut être mixte, c'est-à-dire que le découpage de l'horizon se fait en semaines, en mois voir en trimestres (cf. figure 3.1). On peut remarquer que plus l'horizon est long plus les périodes sont longues, ceci peut induire une plus grande imprécision dans les données.

Dans une entreprise de manufacture, il existe principalement trois types de décisions. Les décisions prises à long terme (2 à 10 ans suivant le secteur d'activité) sont des décisions stratégiques, dont le raisonnement porte sur des grandes masses, des familles de références et

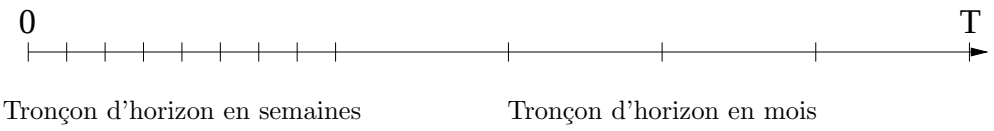


FIG. 3.1 – Horizon de planification mixte

non des références et qui visent l'établissement d'un PIC[1]. Les décisions prises à moyen terme (niveau mois) traduisent le PIC en exprimant les prévisions de production sur l'horizon retenu et produisent donc un PDP[2]. Les quantités mentionnées sont exprimées par référence. Celles prises à court terme (maille journalière) représentent les décisions au niveau des ateliers et visent à établir un ordonnancement.

Dans les problèmes de lot-sizing, l'horizon de planification est décomposé en plusieurs périodes discrètes. Tous les événements (demandes, productions, pertes...) se produisant durant une période donnée sont agrégés dans le temps comme si ils se produisaient tous en même temps. Dans notre étude, les décisions sont prises sur la base d'un horizon de planification mixte à moyen terme, ce qui va nous amener à produire un PDP. L'utilisation d'un horizon de planification mixte est justifiée par le besoin d'avoir des décisions précises au début de l'horizon, qui servirons de base à un ordonnancement, et des décisions moins précises sur le reste de l'horizon afin d'avoir une estimation des décisions futures.

En fonction du découpage de l'horizon de planification, il existe deux classes de problèmes de lot-sizing : modèles à période longues (big bucket), et modèles à périodes courtes (small bucket).

Les modèles big buckets, sont des modèles à horizon discret avec de longues périodes de temps. Le but d'une telle modélisation est de prendre des décisions globales sur la manière d'utiliser les ressources et comment affecter les produits aux usines. Très souvent, une représentation du temps plus détaillée n'est pas nécessaire, parce que les informations détaillées ne sont généralement pas disponibles, telles que les prévisions de ventes détaillées. De tels modèles sont employés pour résoudre des problèmes tactiques à moyen terme. La taille de la période varie entre une semaine et un mois sur un horizon de planification allant de quelques semaines à une année.

Les modèles small buckets, sont des modèles à horizon discret avec des périodes de temps plus courtes. Le but de tels modèles est de prendre des décisions détaillées au sujet de passage des produits sur les machines et les flux de matières premières dans les ateliers. Pour vérifier les contraintes de précédence entre les productions, une représentation de temps plus détaillée est nécessaire. De tels modèles sont utilisés pour tenter de résoudre des problèmes opérationnels à court terme. La taille de la période varie entre une heure et une journée sur un horizon de planification allant d'une journée à quelques semaines. Cette problématique est généralement une étape intermédiaire entre les problèmes de lot-sizing à période longue et les problèmes d'ordonnancement. Les modèles small buckets requièrent la représentation des séquences d'événements d'une période à une autre. Ces modèles sont employés quand le changement de setup sur les machines à un impact significatif sur les capacités utilisées ou sur les coûts de production. Pour ces modèles, le lancement de production d'un même produit est maintenu pour un nombre de périodes, ainsi le setup est encouru seulement quand un autre produit est lancé sur la machine. Généralement une seule référence est produite par période.

[1]PIC : Plan Industriel et Commercial.
[2]PDP : Plan Directeur de Production.

Quand la capacité ou le coût encouru durant le changement de setup ne dépend pas de l'ordre des références, mais seulement du changement de setup sur la machine, nous parlons de modèle à temps ou à coûts de startup. Quand le coût ou le temps de setup dépend de la séquence des produits, on parle de modèle à temps ou à coûts de changeover[3].

Les problèmes sur lesquels nous travaillons sont à horizon de planification discret avec de longues périodes de temps (Big Buckets).

3.2 Formulations mathématiques

Nous décrivons dans ce qui suit, les différents modèles mathématiques illustrant les particularités industrielles des problèmes rencontrés chez DynaSys. Ces modèles permettent de représenter des situations issues de milieux industriels. Chaque particularité sera expliquée, modélisée et accompagnée d'un exemple issu d'une situation réelle. Nous présentons tout d'abord le problème de lot-sizing à capacité finie, avec temps de setup et coûts de rupture sur la demande, noté MCLS2[4]. Nous poursuivrons par les particularités de nos problèmes à savoir, les déficits sur le stock de sécurité, la notion de groupes de références, le cas de plusieurs ressources et plusieurs gammes de production ainsi que la production minimale. Nous finirons par la présentation des notions de lancement minimum de production et des tailles de lot pour les références et les groupes de références.

3.2.1 Problème de lot-sizing avec des ruptures sur la demande (MCLS2)

Le problème de lot-sizing que nous traitons consiste à déterminer un plan de production d'un ensemble de N références, pour un horizon de planification constitué de T périodes longues (big buckets). Les capacités de production sont limitées et doivent être respectées. Ce plan doit également tenir compte d'un ensemble de contraintes additionnelles. En effet, le lancement de production d'une référence à une période donnée entraîne outre la consommation variable en ressources, une consommation dite fixe (temps de setup). La capacité des ressources est généralement exprimée en nombre d'heures disponibles dans la période. Ces ressources peuvent être matérielles ou humaines.

Le lancement de la production d'une référence à une période donnée induit un coût fixe ainsi qu'un coût variable unitaire.

La demande est exprimée à partir des commandes fermes complétées par les commandes prévisionnelles, à la période et pour chaque référence. Les demandes sont fermes à court terme, puis prévisionnelles à plus long terme.

Une particularité importante des problèmes de planification de production est la variation de la demande dans le temps. Ces variations poussent le décideur à anticiper ses demandes et à les fabriquer quelques périodes plus tôt. Afin d'amortir les coûts fixes, le décideur est également contraint de lancer des campagnes de production. Ces deux décisions ainsi que les capacités limitantes entraînent des stratégies de stockage. Ceci induit un coût de stockage unitaire pour chaque référence et à chaque période. Les références peuvent avoir une durée de vie limitée, celle-ci est exprimée en nombre de périodes. Cette contrainte limitera les anticipations de la production et par conséquent la production maximale.

[3] Les notions de startup et de changeover sont développées en Annexe 1, page 141

[4] MCLS2 est une abréviation en anglais de : Multi-item Capacitated Lot-sizing problem with Setup times and Shortage costs.

Une problématique souvent rencontrée dans l'industrie, est que la production soit un multiple d'une taille de lot fixée (cf. exemple 3.2.1). Dans les modèles classiques de lot-sizing, la variable qui représente les quantités produites est souvent continue, et ne prend pas en compte le fait que la production ne puisse être réalisée qu'en multiple d'une quantité donnée. Dans notre cas, cette problématique est traitée en résolvant successivement deux modèles mathématiques. Un premier modèle avec des variables de production continues, et un second modèle avec des variables de production discrètes. Les bornes inférieures et supérieures des variables de production du second modèle sont recalculées en se basant sur les résultats de la résolution du premier modèle. Nous considérons dans tous nos travaux que les variables de production sont continues, c'est-à-dire qu'elles ne tiennent pas compte des tailles de lot. Nous présenterons une modélisation des tailles de lot dans la section 3.2.7.

Exemple 3.2.1. *Dans le domaine de l'agroalimentaire, les cadences de production sont souvent données en (Heures/Kg), les productions en (Kg), et les ressources disponibles en (Heures). Or, les quantités sont regroupées par poids (en tonnes par exemple). Dans ce cas, les quantités produites doivent être en multiple de 1000. Donc, 1000 est la taille de lot.*

Dans la pratique, il arrive souvent que l'on n'admette pas de retards sur les commandes. En effet, il existe des clients qui ne tolèrent aucun retard sur les livraisons. Dans ce cas, les commandes sont annulées et on parle de perte partielle ou totale de la demande, c'est ce qu'on appelle la rupture sur la demande. Il existe essentiellement deux causes de ruptures, la première est la limitation des capacités disponibles. En effet, dans le cas où les capacités ne sont pas suffisantes pour satisfaire les demandes aux dates prévues, on observe des ruptures sur la demande. La seconde cause est économique. En effet, il arrive qu'il soit plus profitable d'avoir des ruptures sur les demandes plutôt que de les satisfaire en entier, même dans une situation où les capacités ne sont pas limitantes. Cette situation se présente quand la somme du coût de production et des coûts de stockage est supérieure au coût de rupture.

Nous allons montrer à travers un premier exemple (exemple 3.2.3, page 35)) qu'il est possible d'avoir des ruptures sur les demandes dans le cas où les capacités sont limitantes. Nous allons montrer à travers un second exemple (exemple 3.2.2, (page 34)) qu'il peut être rentable d'avoir des ruptures dans une situation surcapacitaire.

Une telle situation n'est pas sans conséquence pour le fabricant. En effet, le risque majeur est la perte du client. Une autre conséquence peut être une mauvaise réputation, qui peut avoir comme répercussion la perte de futurs clients et par conséquent une diminution des ventes. Ces problèmes poussent le décideur à fixer comme objectif principal la minimisation des ruptures sur la demande.

Dans une situation de monopole où le risque de perte du client est minime, les coûts de rupture peuvent représenter les coûts de vente du produit. Dans une situation où la concurrence est rude et le risque de perte du client est élevé, les coûts de rupture sur la demande sont des pénalités très élevées dans la fonction objectif en comparaison avec les autres coûts.

Les ruptures sur les demandes sont modélisées par des variables continues avec des pénalités unitaires dans la fonction objectif et une contrainte de borne.

La modélisation du problème MCLS2 se base sur les hypothèses suivantes :

- Tous les calculs se font à la fin de chaque période (Incrémentation des stocks, livraisons, pertes...) ;
- Les coûts ainsi que les besoins fixes sont comptabilisés à chaque période de production, même si la production est lancées à des périodes consécutives.

Afin de décrire le modèle mathématique du problème MCLS2, on utilise les notations suivantes :

Les paramètres :
d_{it} : Demande (commandes et prévisions) pour la référence i à la période t.
f_{it} : Besoin fixe en ressources de la référence i à la période t.
v_{it} : Besoin variable en ressources de la référence i à la période t.
c_t : Capacité disponible à la période t.
α_{it} : Coût de production unitaire variable de la référence i à la période t.
β_{it} : Coût fixe relatif à un lancement de la référence i à la période t.
γ_{it} : Coût unitaire de stockage pour la référence i à la période t.
σ_{it} : Période maximale pour laquelle on peut anticiper la production de la référence i à la période t.
φ_{it} : Coût unitaire de rupture sur la demande pour la référence i à la période t.

Les variables :
x_{it} : Quantité produite pour la référence i à la période t.
$y_{it} = \begin{cases} 1 \text{ s'il y a production de la référence } i \text{ à la période } t \text{ (i.e. si } x_{it} > 0). \\ 0 \text{ sinon} \end{cases}$
s_{it} : Valeur du stock en fin de période t pour la référence i.
r_{it} : Rupture sur la demande pour la référence i à la période t (inférieure à la demande).

En utilisant ces variables et paramètres, nous formulons le problème MCLS2 comme suit.

$$\min \sum_{i,t} \alpha_{it} x_{it} + \sum_{i,t} \beta_{it} y_{it} + \sum_{i,t} \gamma_{it} s_{it} + \sum_{i,t} \varphi_{it} r_{it} \tag{3.1}$$

s.c :

$$x_{it} + r_{it} - s_{it} + s_{i(t-1)} = d_{it}, \quad \forall i, \forall t \tag{3.2}$$
$$\sum_{i=1}^{N} v_{it} x_{it} + \sum_{i=1}^{N} f_{it} y_{it} \leq c_t, \quad \forall t \tag{3.3}$$
$$x_{it} \leq M y_{it}, \quad \forall i, \forall t \tag{3.4}$$
$$r_{it} \leq d_{it}, \quad \forall i, \forall t \tag{3.5}$$
$$x_{it}, s_{it}, r_{it} \geq 0, \quad \forall i, \forall t \tag{3.6}$$
$$y_{it} \in \{0, 1\}, \quad \forall i, \forall t \tag{3.7}$$

La fonction objectif (3.1) minimise le coût total induit par le plan de production (les coûts de production et de stockage, ainsi que les coûts fixes de lancement et les coûts de rupture sur la demande). La contrainte (3.2) exprime la conservation des flux à travers l'horizon. La contrainte (3.3) exprime le fait que le plan que nous souhaitons calculer doit être à capacité finie. En effet, pour la réalisation d'un plan, nous disposons d'une quantité de ressources qui sera consommée par la production d'une ou plusieurs références. La consommation totale doit rester inférieure à la capacité disponible. La contrainte (3.4) permet de modéliser la condition suivante : s'il y a lancement de production, alors la quantité produite ne devra pas dépasser

le majorant de la production M. Celui-ci représente le minimum entre la quantité maximale de la référence pouvant être produite et la demande sur le tronçon de l'horizon $[t, .., \sigma_{it}]$. Ainsi, $M = \min\left(\sum_{t'=t}^{\sigma_{it}} d_{it'}; ((c_t - f_{it})/v_{it})\right)$. La contrainte (3.5) exprime le fait que la rupture pour la référence i à la période t soit inférieure à la demande de la même référence à la même période. La contrainte (3.6) signifie que les variables x_{it}, s_{it} et r_{it} sont continues non négatives pour toute référence i, pour chaque période t. La dernière contrainte (3.7) exprime le fait que y_{it} est une variable binaire pour toute référence i, pour chaque période t.

Ce modèle est appelé également formulation agrégée, car la production d'une référence n'est définie que par sa période de production, contrairement à la formulation basée sur le problème de localisation d'entrepôts (Facility Location-based formulation, Krarup et Bilde [101]) où la production est définie par sa période de production et par sa période de consommation.

Il existe peu de références bibliographiques qui traitent le problème de lot-sizing avec des coûts de rupture sur la demande. Nous citons principalement les travaux suivants pour des problèmes polynomiaux. Récemment Sandbothe and Thompson [159] ont traité le problème de lot-sizing à une seule référence, avec des capacités constantes et des coûts de rupture sur la demande. Ils ont proposé un algorithme de programmation dynamique pour résoudre le problème en $O(T^3)$. Aksen *et al.* [8] ont proposé un algorithme de programmation dynamique pour résoudre le même problème mais sans contrainte de capacité en $O(T^2)$. Loparic *et al.* [112] ont traité le problème de lot-sizing à une seule référence, sans contrainte de capacité, avec des variable représentant les ventes au lieu de demandes fixes et des bornes inférieures sur les variables de stock, maximiser les ventes revient à minimiser les ruptures. Néanmoins, d'autres travaux ont été effectués pour pallier aux problèmes dont la capacité est limitée et la demande ne peut être satisfaite à chaque période. Dixon *et al.* [50] ont remédié au manque de capacité en rajoutant des overtimes, la capacité disponible est étendue en rajoutant une capacité supplémentaire à un certain coût (cf. Annexe 1, 142). Le problème de lot-sizing à plusieurs références avec des contraintes de capacité et des overtimes est traité par Diaby *et al.* [49] et Özdamar et Bozyel [144]. Une autre classe de modèle admet les retards sur la demande. Dans ce cas, la demande doit être satisfaite, mais la référence peut être produite plus tard avec des coûts supplémentaires, de tels modèles sont traités par Pochet et Wolsey [150] et Zangwill [198]. Dans ces deux modélisation, la demande doit être satisfaite et la quantité de ventes perdues pour chaque référence à chaque période n'est pas fournie. La seule information dont on dispose est la quantité de ressources manquantes à chaque période pour satisfaire toute la demande ou les demandes retardées.

Exemple 3.2.2. *A travers cet exemple, nous allons montrer qu'il peut être plus profitable de perdre des demandes plutôt que de les satisfaire.*

L'exemple du tableau 3.1 représente un problème de lot-sizing à 5 périodes, 1 référence et sans contrainte de capacité. La demande ainsi que les coûts unitaires de rupture sont variables dans le temps.

La solution optimale du problème de lot-sizing présentée dans le tableau 3.1 est obtenue en utilisant la modélisation mathématique (3.1)-(3.7) et le solveur de programmation linéaire en nombres entiers CPLEX 9.0. Les résultats sont présentés dans le tableau 3.2.

A partir des résultats présentés dans le tableau 3.2, il est intéressant de noter que les demandes aux périodes 1 et 3 sont perdues bien qu'il soit possible de les produire à des périodes

Périodes (t)	1	2	3	4	5
Coût unitaire de Production (α_t)	3	3	3	3	3
Coût de Setup (β_t)	12000	12000	12000	12000	12000
Coût unitaire de Stockage (γ_t)	1	1	1	1	1
Coût unitaire de rupture (φ_t)	5	7.5	3.5	6	6.5
Demandes (d_t)	1000	4000	800	2200	4100

TAB. 3.1 – Exemple lot-sizing avec ruptures

Périodes (t)	1	2	3	4	5
Demande (d_t)	1000	4000	800	2200	4100
Production (x_t)	0	6200	0	0	4100
Stock (s_t)	0	2200	2200	0	0
Rupture (r_t)	1000	0	800	0	0

TAB. 3.2 – Solution optimale du tableau 3.1

antérieures (il n'y a aucune limitation en ressources). Les demandes aux périodes 1 et 3 sont perdues parce que le prix de vente et la demande sont faibles à ces périodes, et l'anticipation de ces demandes n'est pas rentable pour amortir les coûts de production et de setup. On peut également noter que le stock n'est pas nul à ces périodes. En effet, ce stock est destiné à satisfaire la demande à des périodes ultérieures.

Exemple 3.2.3. *Nous allons reprendre l'exemple précédent en effectuant quelques modifications sur les données. Ainsi, les coûts de rupture ne représentent plus les coûts de vente, mais des pénalités très élevées. Les capacités et les anticipations maximales sont limitantes.*

A travers cet exemple, nous allons montrer qu'en rajoutant des contraintes de capacité et des anticipations maximales, on crée des ruptures.

L'exemple du tableau 3.3 représente un problème de lot-sizing à 5 périodes, 1 référence, une ressource limitante et des anticipations maximales fixées à 1. La demande ainsi que les coûts unitaires de production sont variables dans le temps.

Périodes (t)	1	2	3	4	5
Coût unitaire de Production (α_t)	3	5	3	3	2
Coût de Setup (β_t)	12000	12000	12000	12000	12000
Coût unitaire de Stockage (γ_t)	1	1	1	1	1
Coût unitaire de rupture (φ_t)	50000	50000	50000	50000	50000
Capacités (c_t)	300	300	300	300	300
Anticipation maximale (σ_t)	1	1	1	1	1
Demandes (d_t)	1000	4000	800	2800	6100

TAB. 3.3 – Exemple lot-sizing avec contraintes de capacité et ruptures

La solution optimale du problème de lot-sizing présentée dans le tableau 3.3 est obtenue en utilisant la modélisation mathématique présentée précédemment et un solveur de programmation linéaire en nombres entiers. Les résultats sont présentés dans le tableau 3.4.

Périodes (t)	1	2	3	4	5
Demande (d_t)	1000	4000	800	2800	6100
Production (x_t)	3000	2800	2800	3000	3000
Stock (s_t)	2000	800	2800	3000	0
Rupture (r_t)	0	0	0	0	100

TAB. 3.4 – Solution optimale du tableau 3.3

A partir des données du tableau 3.3, il est intéressant de noter que la somme de la demande sur tout l'horizon de planification est inférieure à la somme des capacités des ressources disponibles.

En analysant les résultats du tableau 3.4, on remarque que le plan de production illustre une stratégie de stockage. En effet, afin d'éviter d'avoir des ruptures, les demandes sont produites une période à l'avance jusqu'à saturation des ressources. On remarque également des ruptures à la dernière période. Ces ruptures sont dues à la limitation des capacités ainsi qu'à l'anticipation maximale qui est fixée à 1.

Une partie de la demande de la dernière période est perdue. En effet, les ressources sur les deux dernières périodes ne permettent de créer que 6000 unités, tandis que la demande à la dernière est de 6100. Ceci crée une rupture de 100 à la dernière période.

3.2.2 Modèle de lot-sizing avec prise en compte d'un stock de sécurité (MCLS4)

Il arrive couramment que l'un des objectifs du décideur soit d'avoir un profil de stocks supérieur à un seuil appelé *stock de sécurité*. Le stock de sécurité est une partie du stock qui n'est pas utilisée. Il permet de faire face à des événements imprévus (retard de livraison, augmentation de la consommation...). Le stock de sécurité peut être alimenté par des calculs de prévisions en se basant sur un historique de ventes. Une telle spécificité peut être modélisée en ajoutant une contrainte assurant le fait que le stock soit supérieur au stock de sécurité.

Le stock de sécurité est un objectif ou une cible à atteindre plutôt qu'une contrainte industrielle à respecter. Il peut arriver que l'on ne puisse pas atteindre ce stock de sécurité, dans ce cas on parle d'une situation de *déficit sur le stock de sécurité*. Ces déficits sont modélisés par des variables continues avec une pénalité unitaire dans la fonction objectif, une contrainte de bornes et une contrainte assurant que la somme du stock et du déficit sur le stock de sécurité soit supérieure au stock de sécurité. La pénalité unitaire des variables de déficit sur le stock de sécurité est inférieure à celle des variables de rupture sur la demande. En effet, l'objectif principal du décideur est de satisfaire la demande, la satisfaction du stock de sécurité vient en seconde position.

Le coût de déficit sur le stock de sécurité est supérieur au coût de stockage. La figure 3.2 représente la fonction de coût du stock. Quand le stock est au niveau du stock de sécurité, le coût est nul. La pente de la fonction en dessous du stock de sécurité est plus élevée que celle de la fonction au dessus du stock de sécurité.

Dans notre modélisation, nous allons proposer une autre représentation du stock de sécurité afin de l'incorporer dans la contrainte de conservation des flux à travers l'horizon. Ceci est réalisé en reformulant les variables de stock en trois parties. Une variable représentera le surstock par rapport au stock de sécurité, un paramètre fixera le stock de sécurité et une autre variable le déficit sur le stock de sécurité. Ainsi, au lieu d'avoir une variable de stock,

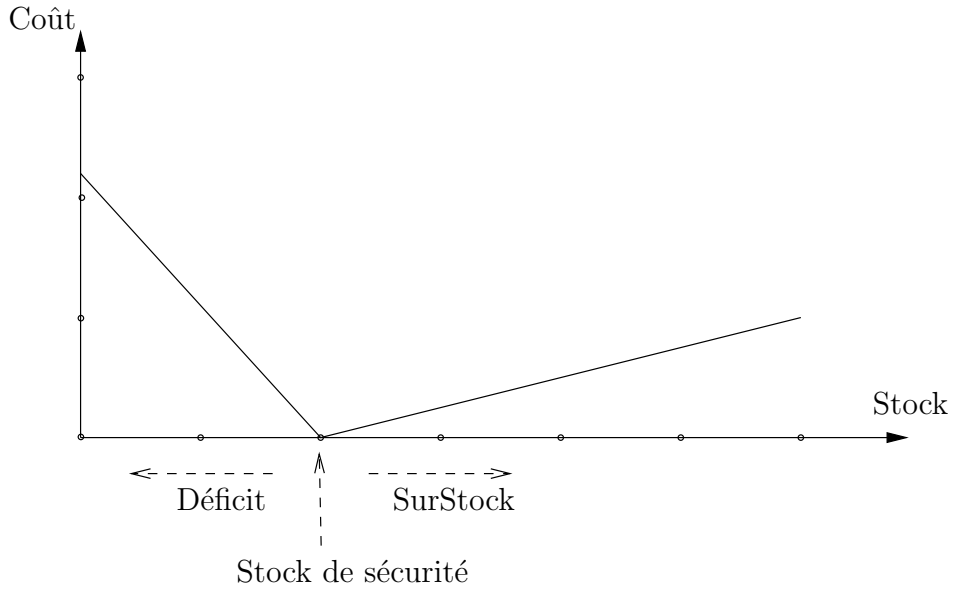


FIG. 3.2 – Coûts de stockage

nous aurons deux variables et un seuil. Soit s_{it} la variable représentant la valeur du stock de la référence i en fin de période t, I_{it}^+ et I_{it}^- les variables représentant respectivement le surstock et le déficit sur le stock de sécurité de la référence i en fin de période t, et L_{it} représentant le stock de sécurité de la référence i à la période t. La variable s_{it} peut donc être remplacée par $I_{it}^+ + L_{it} - I_{it}^-$ dans la contrainte de flux (cf. figure 3.3).

L'intérêt d'une telle modélisation est de pouvoir formuler le problème lot-sizing avec des ruptures et du déficit sur le stock de sécurité à une seule référence et sans contrainte de capacité, sous la forme d'un problème de flots à coût minimum avec des coûts fixes sur les arcs. Une telle formulation permet de caractériser les propriétés des solutions optimales et de développer un algorithme de programmation dynamique pour résoudre le problème sans contrainte de capacité à l'optimum.

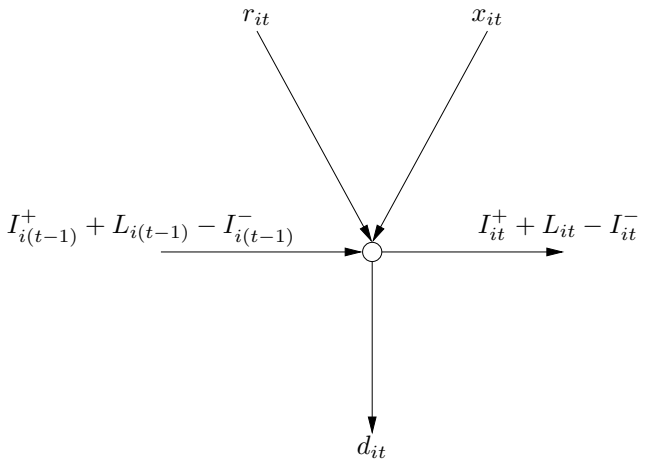


FIG. 3.3 – Reformulation du stock de sécurité

Outre les paramètres et les variables MCLS2 on considère :

Les paramètres :
L_{it} : Stock de sécurité de la référence i à la période t.
γ_{it}^{+} : Coût unitaire de stockage de la référence i à une période t.
γ_{it}^{-} : Coût unitaire de déficit sur le stock de sécurité de la référence i à la période t.

Les variables :
I_{it}^{+} : Valeur du surstock de la référence i à la fin de la période t (excédent de stock).
I_{it}^{-} : Valeur du déficit sur le stock de sécurité de la référence i à la fin de la période t (inférieur au stock de sécurité).

La contrainte (3.2) peut être modifiée en considérant la nouvelle formulation de la variable s_{it}. La différence entre les stocks de sécurité de la période t et de la période $t-1$ est considérée comme un besoin. celui-ci peut être positif ou négatif puisqu'il représente la variation du stock de sécurité entre la période $t-1$ et la période t. On note δ_{it} cette variation, $\delta_{it} = L_{it} - L_{i(t-1)}$.

En utilisant les variables et paramètres du problème MCLS2 ainsi que ceux présentés précédemment, nous formulons le nouveau problème comme suit, on note ce nouveau problème MCLS4[5].

$$\min \sum_{i,t} \alpha_{it} x_{it} + \sum_{i,t} \beta_{it} y_{it} + \sum_{i,t} \varphi_{it} r_{it} + \sum_{i,t} \gamma_{it}^{+} I_{it}^{+} + \sum_{i,t} \gamma_{it}^{-} I_{it}^{-} \tag{3.8}$$

s.c.

$$I_{i(t-1)}^{+} - I_{i(t-1)}^{-} + r_{it} + x_{it} = d_{it} + \delta_{it} + I_{it}^{+} - I_{it}^{-}, \quad \forall i, \forall t \tag{3.9}$$

$$\sum_{i=1}^{N} v_{it} x_{it} + \sum_{i=1}^{N} f_{it} y_{it} \leq c_t, \quad \forall t \tag{3.10}$$

$$x_{it} \leq M y_{it}, \quad \forall i, \forall t \tag{3.11}$$

$$r_{it} \leq d_{it}, \quad \forall i, \forall t \tag{3.12}$$

$$I_{it}^{-} \leq L_{it}, \quad \forall i, \forall t \tag{3.13}$$

$$x_{it}, r_{it}, I_{it}^{+}, I_{it}^{-} \geq 0, \quad \forall i, \forall t \tag{3.14}$$

$$y_{it} \in \{0, 1\}, \quad \forall i, \forall t \tag{3.15}$$

La fonction objectif (3.8) est une version modifiée de la fonction objectif du problème MCLS2. En effet, on tient compte des coûts de déficit sur le stock de sécurité. La contrainte (2.10) du problème MCLS2 est reformulée en remplaçant la variable de stock s_{it} par : $I_{it}^{+} + L_{it} - I_{it}^{-}$, et $L_{it} - L_{i(t-1)}$ par δ_{it} ce qui nous donne la contrainte (3.9). La contrainte (3.13) exprime le fait que le déficit sur le stock de sécurité de la référence i à la période t soit inférieure au stock de sécurité de la même référence à la même période. La contrainte (3.14) signifie que les variables x_{it}, I_{it}^{+} et I_{it}^{-} sont continues non négatives pour toute référence i, pour chaque période t. Les contraintes (3.10), (3.11), (3.12) et (3.15) sont exactement les

[5]MCLS4 est une abréviation en anglais de : Multi-item Capacitated Lot-sizing problem with Setup times, Shortage costs and Safety Stock deficit costs.

mêmes contraintes que (3.3), (3.4), (3.5) et (3.7) respectivement.

Rappelons que dans la littérature, le stock de sécurité est souvent traité comme une contrainte et non un objectif à atteindre. En effet, le stock de sécurité est fixé comme borne inférieure des variables de stock. Love [116] a étendu le problème de Wagner et Whitin en rajoutant des bornes inférieures et supérieures sur la production et sur le stock. L'auteur a proposé un algorithme de programmation dynamique en $O(T^3)$ pour résoudre le problème de lot-sizing avec des bornes inférieures et supérieures sur les stocks. Loparic *et al.* [112] ont considéré le stock de sécurité en imposant une borne inférieure sur les variables de stock.

3.2.3 Modèle de lot-sizing multi-groupes (MCLSG)

Il arrive souvent que l'on ait des références à deux ou plusieurs besoins fixes en ressources, un premier besoin fixe qui ne dépend que de la référence, et un autre besoin fixe commun à plusieurs références. Dans ce cas, les références ayant des besoins fixes en commun sont classées par groupes de références. Le lancement de la production d'une référence entraîne le lancement des groupes de références auxquels elle appartient. Les références sont donc classées par groupes de références. Une référence peut appartenir à un ou plusieurs groupes de références et un groupe de références peut donc contenir une ou plusieurs références.

Ceci se traduit par la prise en compte d'un besoin ou d'une consommation fixe des capacités en ressources disponibles dès qu'il y a lancement d'un *groupe*. Ceci entraîne également un coût de lancement pour le groupe (cf. exemple 3.2.4).

Cette situation peut être modélisée en rajoutant une dimension j pour les groupes et une variable binaire z_{jt} pour la détection de lancements de groupes, J représente le nombre de groupes.

Exemple 3.2.4. *Nous allons montrer à travers cet exemple comment des situations réelles peuvent être modélisées en utilisant des groupes de références.*

La production de parquets nécessite deux réglages, le premier réglage sert à préparer la machine pour effectuer un motif donné sur les parquets. Le second réglage sert à fixer son épaisseur. Une fois ces deux réglages effectués, une matière première est choisie pour fabriquer un type de parquet. Chaque matière première nécessite un temps de préparation distinct.

Dans ce cas, une référence est identifiée par un motif, une épaisseur et une matière première. Les références sont classées selon ces trois critères. Le réglage des deux machines (motif, épaisseur) permet de fabriquer tous les produits ayant la même épaisseur et le même motif en n'effectuant que des préparations concernant la matière première.

A partir du modèle MCLS4, il s'agira de modifier la fonction objectif, la contrainte de capacité et de rajouter une contrainte pour détecter les lancements de groupes.

On définit dans ce qui suit toutes les modifications apportées au problème MCLS4 afin de prendre en compte les groupes de références, on note ce nouveau problème MCLSG

Les paramètres :

g_{jt} : Besoin fixe en ressources, du groupe j à la période t.

ω_{jt} : Coût fixe relatif au lancement du groupe j à la période t.

$$a_{ij} = \begin{cases} 1 \text{ si la référence } i \text{ appartient au groupe } j. \\ 0 \text{ sinon} \end{cases}$$

Les variables :

$$z_{jt} = \begin{cases} 1 \text{ si le groupe } j \text{ est lancé à la période } t. \\ 0 \text{ sinon} \end{cases}$$

La fonction objectif :

$$\min \sum_{i,t} \alpha_{it} x_{it} + \sum_{i,t} \beta_{it} y_{it} + \sum_{i,t} \varphi_{it} r_{it} + \sum_{i,t} \gamma_{it}^{+} I_{it}^{+} + \sum_{i,t} \gamma_{it}^{-} I_{it}^{-} + \sum_{j,t} \omega_{jt} z_{jt} \tag{3.16}$$

Les contraintes :

La contrainte suivante remplace la contrainte (3.10) du problème MCLS4.

$$\sum_{i=1}^{N} v_{it} x_{it} + \sum_{i=1}^{N} f_{it} y_{it} + \sum_{j=1}^{J} g_{jt} z_{jt} \leq c_t, \ \forall t \tag{3.17}$$

Les deux contraintes suivantes modélisent le lancement de groupes :

$$z_{jt} \geq a_{ij} y_{it}, \quad \forall i, \forall j, \forall t \tag{3.18}$$

$$z_{jt} \in \{0, 1\}, \quad \forall j, \forall t \tag{3.19}$$

La fonction objectif (3.16) est celle du problème MCLS4 dans laquelle sont pris en compte les coûts de lancements de groupes. La contrainte (3.10) du problème MCLS4 est modifiée en rajoutant les consommations fixes des groupes en ressources, ce qui nous donne la contrainte (3.17). La contrainte (3.18) exprime le fait que s'il y a lancement d'au moins une des références $i \in j$ (c'est-à-dire $a_{ij} = 1$) alors il y a lancement du groupe j. La contrainte (3.19) exprime le fait que z_{jt} soit une variable binaire pour tout j,t. Les contraintes (3.9), (3.11), (3.12), (3.13), (3.14) et (3.15) du modèle MCLS4 restent inchangées.

Pour la modélisation d'une ligne d'embouteillage d'un fabriquant de boissons, Clark [35] a utilisé une modélisation proche de celle du problème MCLSG.

3.2.4 Modèle de lot-sizing multi-ressources (MCLSR)

La production d'une référence peut nécessiter l'utilisation d'une ou plusieurs ressources par période. Ceci peut être dû à la longueur des mailles temporelles ou au fonctionnement de plusieurs ressources en parallèle. Pour les clients DynaSys, une référence requiert fréquement deux ressources, une ressource matérielle et une ressource humaine (cf. figure 3.4). La fabrication de cette référence entraîne une consommation fixe et une consommation variable sur chaque ressource.

Pour modéliser cette situation, nous ajoutons la dimension *ressource* notée : r, avec R le nombre de ressources disponibles. Ainsi, les capacités disponibles et les paramètres de consommation (besoin variable, besoin fixe) seront liés à la dimension ressource. En reprenant

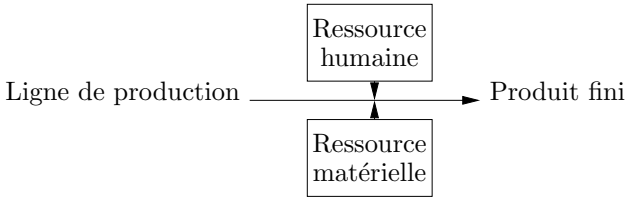


FIG. 3.4 – Deux ressources en parallèle

le problème MCLS4, il s'agira de modifier le besoin fixe, le besoin variable ainsi que les capacités disponibles en intégrant la dimension r, on note ce nouveau problème MCLSR.

Les paramètres :
f_{irt} : Besoin fixe en ressources, de la référence i sur la ressource r à une période t.
v_{irt} : Besoin variable en ressources, de la référence i sur la ressource r à une période t.
c_{rt} : Capacité disponible de la ressource r à la période t.

Les contraintes :
La contrainte (3.10) est reformulée comme suit :

$$\sum_{i=1}^{N} v_{irt}x_{it} + \sum_{i=1}^{N} f_{irt}y_{it} \leq c_{rt}, \ \forall r, \forall t \tag{3.20}$$

La fonction objectif du problème MCLS2 reste inchangée. En effet, les paramètres de coûts n'intègrent pas la dimension r.

Le problème de lot-sizing impliquant plusieurs ressources à été étudié par plusieurs auteurs. Nous pouvons citer les travaux de Katok *et al.* [94] pour le problème de lot-sizing à plusieurs niveaux de production, plusieurs machines et des temps de setup. Maes *et al.* [117] et Tempelmeier et Helber [170] ont considéré plusieurs ressources mais pas de temps de setup. Belvaux et Wolsey [18] ont développé un module d'optimisation *bc-prod* pour la résolution de problèmes de lot-sizing dans lesquels ils considèrent plusieurs ressources.

3.2.5 Modèle de lot-sizing multi-gammes (MCLSV)

Dans la pratique, la planification de la production ne vise pas uniquement la détermination des quantités à produire ou des quantités perdues, mais également le choix entre plusieurs modes de fabrication. En effet, on est souvent amené à choisir entre des modes alternatifs de production dès lors qu'interviennent plusieurs lignes de production (ou gammes), plusieurs vendeurs ou plusieurs sous-traitants. Chacune de ces alternatives a un coût de production et des consommations en ressources différents.

Par ailleurs, un fournisseur peut avoir plusieurs offres qui peuvent être utilisées pour la fabrication du produit fini. Ces offres peuvent différer dans leurs coûts de production et consommations en ressources et peuvent donc être modélisées par des gammes.

Les produits peuvent être fabriqués localement ou par un sous-traitant extérieur. En général, les coûts sont différents pour les sous-traitants. On peut remarquer que le choix entre

les fournisseurs ou la décision de faire appel ou non à ceux-ci, est le même que de choisir entre plusieurs lignes de production ou de sélectionner un produit parmi une liste de substituts.

Nous allons présenter un modèle qui permet de choisir une de ces alternatives tout en respectant les contraintes de capacité, et en minimisant la somme des coûts de production, de rupture, de déficit et de stockage.

Une manière de modéliser cette situation est de considérer un produit pour chaque gamme de production. Pour ce faire, on va considérer la nouvelle dimension *gamme* que l'on va noter v, Le nombre de gammes est V. Ainsi, les produits équivalents auront la même dimension référence, mais une dimension gamme différente. Les produits sont considérés comme distincts durant la phase de production, mais à leur entrée en stock, les quantités de tous les produits équivalents sont additionnées afin de satisfaire la demande.

Il existe deux types de lignes de production, celles avec des ressources séparées, et celles avec une ou plusieurs ressources communes.

Dans le premier cas, les produits équivalents ont des lignes de production totalement dissociées. La production d'un produit n'affecte pas directement la production des substituts. Cette situation est schématisée dans la figure 3.5 qui représente un exemple de deux lignes de production avec des ressources séparées.

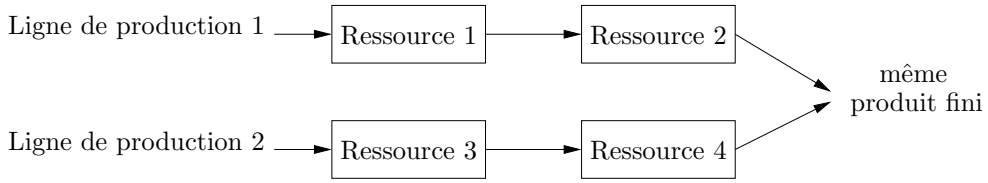


FIG. 3.5 – Deux lignes de production avec des ressources séparées

Le deuxième cas est plus complexe. En effet, les références équivalentes passent par une ou plusieurs ressources communes, appelées également ressources partagées. Cette situation est souvent rencontrée dans une même usine qui comporte plusieurs lignes de production, dont la ressource partagée est souvent la ressource humaine. Cette ressource est souvent limitante, elle constitue un goulot d'étranglement. Ainsi, la production d'un produit impacte directement la production des produits équivalents. Cette situation est schématisée dans la figure 3.6 représentant deux lignes de production qui partagent une même ressource.

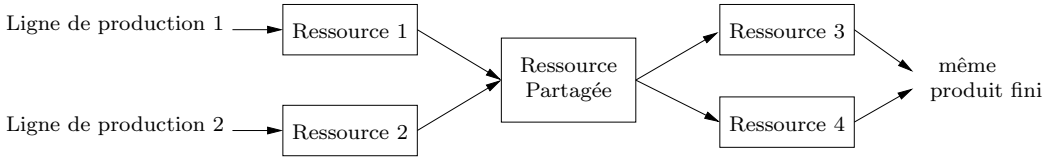


FIG. 3.6 – Deux lignes de production avec une ressource partagée

Les modifications apportées aux paramètres et aux variables du problème MCLSR portent essentiellement sur les paramètres et les variables de production.

Les données additionnelles requises pour la modélisation du problème, ainsi que les modifications apportées aux variables et paramètres des modèles précédents sont présentées dans ce qui suit,

Les paramètres :
f_{virt} : Besoin fixe en ressources de la référence i, pour la gamme v, sur la ressource r, à la période t.
v_{virt} : Besoin variable en ressources de la référence i, pour la gamme v, sur la ressource r, à la période t.
α_{vit} : Coût de production unitaire variable de la référence i, pour la gamme v, à la période t.
β_{vit} : Coût fixe relatif à un lancement de la référence i, pour la gamme v, à la période t.

Les variables :
x_{vit} : Quantité produite pour la référence i à la période t en utilisant la gamme v.

$$y_{vit} = \begin{cases} 1 \text{ s'il y a production de la référence } i \text{ à la période } t \\ \quad \text{en utilisant la gamme } v \text{ (i.e. si } x_{vit} > 0). \\ 0 \text{ sinon} \end{cases}$$

L'équation de conservation des flux à travers l'horizon sera formulée en remplaçant la production du produit fini par la somme des productions relatives à chaque gamme. Ainsi, la variable x_{it} dans la contrainte de flux (3.9) du problème MCLS4 est remplacée par la l'expression $\sum_{v=1}^{V} x_{vit}$.

En utilisant ces variables et paramètres, nous formulons le nouveau problème comme suit, on note ce nouveau problème MCLSV.

$$\min \sum_{v,i,t} \alpha_{vit} x_{vit} + \sum_{v,i,t} \beta_{vit} y_{vit} + \sum_{i,t} \varphi_{it} r_{it} + \sum_{i,t} \gamma_{it}^{+} I_{it}^{+} + \sum_{i,t} \gamma_{it}^{-} I_{it}^{-} \tag{3.21}$$

s.c :

$$I_{i(t-1)}^{+} - I_{i(t-1)}^{-} + r_{it} + \textstyle\sum_{v=1}^{V} x_{vit} = d_{it} + \delta_{it} + I_{it}^{+} - I_{it}^{-}, \quad \forall i, \forall t \tag{3.22}$$

$$\textstyle\sum_{v=1}^{V} \sum_{i=1}^{N} v_{virt} x_{vit} + \sum_{v=1}^{V} \sum_{i=1}^{N} f_{virt} y_{vit} \leq c_{rt}, \quad \forall r, \forall t \tag{3.23}$$

$$x_{vit} \leq M y_{vit}, \quad \forall v, \forall i, \forall t \tag{3.24}$$

$$r_{it} \leq d_{it}, \quad \forall i, \forall t \tag{3.25}$$

$$I_{it}^{-} \leq L_{it}, \quad \forall i, \forall t \tag{3.26}$$

$$r_{it}, I_{it}^{+}, I_{it}^{-} \geq 0, \quad \forall i, \forall t \tag{3.27}$$

$$x_{vit} \geq 0, \quad \forall v, \forall i, \forall t \tag{3.28}$$

$$y_{vit} \in \{0, 1\}, \quad \forall v, \forall i, \forall t \tag{3.29}$$

La fonction objectif (3.21) est une version de la fonction objectif (3.8) à laquelle est rajoutée la dimension gamme v aux variables de production. La contrainte (3.22) est aussi une version modifiée de la contrainte (3.9) où la variable x_{it} a été remplacée par l'expression $\sum_{v=1}^{V} x_{vit}$. La contrainte (3.23) est la même que la contrainte (3.20) excepté le fait que chaque couple référence-gamme est considéré comme un produit indépendant. La contrainte (3.24) est identique à la contrainte (3.11) avec une dimension gamme supplémentaire, $M = \min\left(\sum_{t'=t}^{\sigma_{it}} d_{it'}; \min_{r=1,\ldots,R}\left((c_{rt} - f_{virt})/v_{virt}\right)\right)$. La contrainte (3.27) signifie que la variable x_{vit} est continue non négative pour toute référence i, pour chaque période t, et pour toute

gamme v. La contrainte (3.28) signifie que les variables r_{it}, I_{it}^{+} et I_{it}^{-} sont continues non négatives pour toute référence i, pour chaque période t. La dernière contrainte (3.29) exprime le fait que y_{vit} est une variable binaire pour toute référence i, pour chaque période t et pour chaque gamme v.

La plupart des références bibliographiques qui traitent des problèmes de lot-sizing multi-gammes ne considèrent que des ressources séparées. Miller [131] a proposé des inégalités valides pour le problème MCLS avec des ressources séparées. Toledo et Armentano [173] ont utilisé une approche de relaxation lagrangienne pour résoudre le même problème sous des capacités constantes. Degraeve et Jans [88] ont proposé une méthode de décomposition de Dantzig et Wolfe basée sur la relaxation des contraintes de flux pour résoudre le problème MCLS à ressources séparées. Nous n'avons pas trouvé de références bibliographiques traitant du problème MCLS multi-gammes avec des ressources partagées.

La notion de gamme peut également apparaître dans un modèle avec groupes de référence. Ainsi, on aura un besoin fixe groupe pour chaque gamme v.

3.2.6 Modèles de lot-sizing avec contraintes de production minimale et de lancement minimum

La production minimale (ou plan minimum imposé) est un plan de production qu'il est impératif de produire même si la production de ce plan entraîne le non-respect des contraintes de production. Cette situation est rencontrée quand on fait une rotation d'horizon de planification. En effet, lorsque l'horizon de planification change, on définit un nouvel horizon de travail. Celui-ci peut coïncider avec des périodes où les ordres de fabrication ont déjà été lancés dans les ateliers. Dans ce cas, tout ce qui a été ordonné est figé (cf. figure 3.7). Ce type de problème peut être prétraité en soustrayant les ressources consommées par la production minimale aux capacités disponibles. La production minimale d'une référence i pour une gamme v à une période t est notée $p_{vit}^{\min}$.

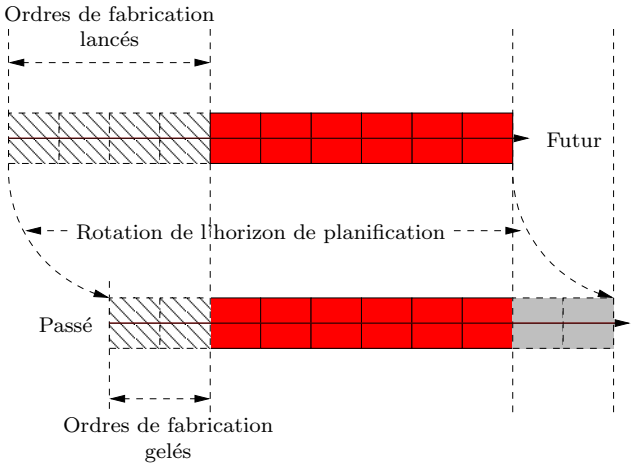


FIG. 3.7 – Rotation d'horizon de planification

Par ailleurs, pour des raisons économiques, et afin de ne pas faire de petites campagnes de production, le décideur peut être amené à fixer un seuil de production qu'il faut dépasser quand la fabrication de certaines références est lancée. Cette modélisation est souvent rencontrée en présence de grands coûts de lancement ou de grandes consommations fixes. Dans l'industrie textile, par exemple, lorsqu'un coloris est lancé sur une machine, on essaye de fabriquer des références de ce coloris en quantité importante. En effet, les temps d'attente pour le passage d'un coloris à un autre sont très importants. Le fait de fixer ce seuil minimum de production aide à amortir ces temps de non productivité.

Donc, lorsqu'il y a lancement de production d'une référence i pour une gamme v à une période t, il faut que la production soit au moins supérieure à une quantité minimale notée $l_{vit}^{\min}$.

Cette situation peut être modélisée en rajoutant une contrainte de seuil minimal pour la variable de production x_{vit} quand il y a lancement de production, c'est à dire, quand $y_{vit} = 1$. Cette formulation est donnée par la contrainte (3.30).

$$x_{vit} \geq l_{vit}^{\min} y_{vit} \tag{3.30}$$

Dans notre cas, la prise en compte des deux paramètres $l_{vit}^{\min}$ et $p_{vit}^{\min}$ est faite en reformulant la variable de production x_{vit} par l' expression suivante :

$$x_{vit} + \max\left\{0, l_{vit}^{\min} - p_{vit}^{\min}\right\} y_{vit},\ \forall v, i, t$$

On note :

$$l_{vit} = \max\left\{0, l_{vit}^{\min} - p_{vit}^{\min}\right\},\ \forall v, i, t$$

Dans le cas où $p_{vit}^{\min} > 0$, la variable y_{vit} ne va plus jouer le rôle d'une variable de lancement, mais le rôle d'une variable de détection de la production de la quantité l_{vit}. En effet, quand $p_{vit}^{\min} > 0$, la consommation fixe est retranchée aux capacités disponibles, et la variable de lancement de production y_{vit} sera fixée à 1. En revanche, dans le cas où le lancement minimum de production est supérieure à la production minimale (c'est-à-dire $l_{vit} > 0$), la variable de lancement de production y_{vit} est libre et jouera le rôle d'une variable de détection de lancement minimum de production. Les modifications apportées aux paramètres du problème MCLSV sont définies comme suit :

$$f'_{virt} = \begin{cases} f_{virt} & \text{si } p_{vit}^{\min} = 0. \\ 0 & \text{si } p_{vit}^{\min} > 0. \end{cases}$$

$$c'_{rt} = c_{rt} - \sum_{i=1}^{N} \sum_{v=1}^{V} (f_{virt} - f'_{virt}) - \sum_{i=1}^{N} \sum_{v=1}^{V} (p_{vit}^{\min} v_{virt}).$$

$$d'_{it} = d_{it} - \sum_{v=1}^{V} p_{vit}^{\min}$$

Le paramètre f'_{virt} représente le nouveau besoin fixe de la référence i pour la gamme v sur la ressource r à la période t. d'_{it} représente la nouvelle demande de la référence i à la période t. Le paramètre c'_{rt} représente la capacité disponible de la ressource r à la période t en prenant en considération la production minimale.

Il existe plusieurs références bibliographiques qui traitent des problèmes MCLS avec des contraintes de lancement minimum de production. Nous pouvons citer les travaux de Belvaux et Wolsey [18], Constantino [41], Mercé et Fontan [129] et Miller [131].

3.2.7 Prise en compte explicite de la taille des lots

Nous avons évoqué à la section 3.2.1, page 32 la notion de tailles de lot. Nous avons également présenté un exemple (exemple 3.2.1) et une manière de résoudre ce problème. Nous allons dans ce qui suit, présenter la modélisation mathématique de la notion de tailles de lot. La variable de production présentée dans les paragraphes précédents est continue, et ne prend pas en compte le fait que la demande soit un multiple d'une quantité fixée.

Afin de modéliser cette problématique, nous devons introduire un nouveau paramètre. La variable de production est remplacée par une nouvelle variable entière qui représente le nombre de lots produits. Nous présentons dans ce qui suit les modifications apportées au modèle MCLSV afin de prendre en charge la contrainte de taille de lot.

Les paramètres :

Q_{vit} : taille de lot de la référence i, pour la gamme v, à la période t.

Les variables :

x^Q_{vit} : Nombre de lots produits pour la référence i, sur la gamme v, à la période t.

La variable x_{vit} est remplacée par l'expression $Q_{vit}x^Q_{vit}$, et la contrainte $x_{vit} \geq 0$ est remplacée par la contrainte (3.31).

$$x^Q_{vit} \in \mathbb{N},\ \forall v, \forall i, \forall t \tag{3.31}$$

3.2.8 Modèle de lot-sizing avec prise en compte de la tailles de lot et du lancement minimum pour les groupes de références

Dans les paragraphes précédents, nous avons présenté les différentes spécificités du problème de lot-sizing à savoir les groupes de références, les contraintes de lancement minimum et de taille de lot. Dans ce qui suit, nous allons modéliser la notion de taille de lot et de lancement minimum pour les groupes de références.

Le lancement minimum pour les groupes est modélisé en considérant la production de toute les références appartenant à ce groupe, et en forçant cette production totale à être supérieure au lancement minimum groupe, si la production est lancée sur au moins une référence appartenant à ce groupe. Pour ce faire, on va rajouter le paramètre suivant :

l^G_{jt} : lancement minimum de production pour le groupe j à la période t.

En reprenant le problème MCLSV, il s'agira de rajouter la contrainte de lancement minimum groupe (3.32)

$$\sum_{v=1}^{V} \sum_{i=1}^{N} a_{ij} x_{vit} \geq l^G_{jt} z_{jt}, \quad \forall j, \forall t \tag{3.32}$$

Afin de modéliser la contrainte de taille de lots groupes, on rajoute un paramètre ainsi qu'une nouvelle variable entière représentant la production du groupe par multiple de taille de lot. Dans ce qui suit, nous présentons toutes les modifications apportées au problème MCLSV

afin de prendre en compte la taille de lots groupes.

Paramètres :
Q^G_{jt} : Taille de lot du groupe de références j à la période t.

Variables :
x^G_{jt} : Nombre de lots produits pour le groupe de références j à la période t.

La contrainte de taille de lots groupes peut être modélisée en rajoutant les contraintes (3.33) et (3.34).

$$\sum_{v=1}^{V} \sum_{i=1}^{N} a_{ij} x_{vit} = Q^G_{jt} x^G_{jt}, \quad \forall j, \forall t \tag{3.33}$$

$$x^G_{jt} \geq 0, \ \hat{x}^g_{it} \in \mathbb{N}, \quad \forall j, \forall t \tag{3.34}$$

3.3 Conclusion

Nous avons décrit dans ce chapitre de nouvelles modélisations de problèmes de lot-sizing rencontrées dans l'industrie. Nous avons proposé une modélisation du problème de lot-sizing classique MCLS en rajoutant les particularités de nos problèmes une par une. Nous avons introduit la notion de rupture sur la demande, puis la notion de déficit sur le stock de sécurité. Nous avons également modélisé nos problèmes en introduisant des contraintes de setup sur des groupes de références. Le problème a été modélisé pour le fonctionnement de plusieurs de ressources en parallèle. Nous avons introduit et modélisé la notion de gammes de production, ces dernière peuvent coupler plusieurs ressources et produire ainsi des goulots d'étranglement. Enfin, nous avons modélisé les contraintes de taille de lots, de lancement minimum de production et de production minimale pour les références et les groupes de références. D'autres extensions possibles du problème MCLS4 sont présentées en annexe.

Dans le chapitre suivant, nous proposons des inégalités valides pour le problème MCLS2 ainsi que des heuristiques de séparation polynomiales. Ces inégalités sont généralisées pour les cas, multi-groupes, multi-ressources et multi-gammes. Nous implémentons ces inégalités dans une approche de branch-and-cut afin de résoudre le problème MCLS2.

Chapitre 4

Approche de branch-and-cut

Introduction

Dans ce chapitre, nous traitons le problème de lot-sizing à plusieurs références, avec des contraintes de capacité, des coûts et des temps de setup, ainsi que des coûts de rupture sur la demande, noté MCLS2[1]. Le but de ce chapitre est de proposer une méthode basée sur le principe de branch-and-cut afin de résoudre le problème MCLS2. La motivation d'utiliser un algorithme de branch-and-cut pour résoudre le problème MCLS2 est de généraliser des résultats qui se sont avérés efficaces sur des problèmes de lot-sizing à plusieurs références sous contraintes de capacité et des temps de setup. Nos résultats sont intégrés à un logiciel APS[2].

Dans la section 4.1 nous rappelons les différentes modélisations mathématiques du problème traité. Dans la section 4.2 nous présentons la relaxation du problème MCLS2 sur une seule période. Dans les sections 4.3 et 4.4 nous présentons une généralisation des inégalités (l, S) pour le problème MCLS2 ainsi qu'une heuristique de séparation. Une généralisation des inégalités couvrantes et des inégalités couvrantes inverses ainsi que des lifting séquentiels de celles-ci sont présentés dans les sections 4.5, 4.6 et 4.7. Deux heuristiques de séparation de ces inégalités sont présentées aux sections 4.8 et 4.9. Des résultats expérimentaux sont décrits dans la section 4.10. Enfin, la section 4.11 est consacrée à la généralisation de ces inégalités valides pour différentes extensions du problème à savoir plusieurs groupes de références, plusieurs ressources et plusieurs gammes de production.

4.1 Modélisation mathématique du problème MCLS2

Nous rappelons dans ce qui suit la modélisation mathématique du problème de lot-sizing à plusieurs références, avec des contraintes de capacité, des coûts et des temps de setup, ainsi que des coûts de rupture sur la demande (MCLS2).

Il s'agit de déterminer un plan de production d'un ensemble de N références noté $\mathcal{I} = \{1, 2, \cdots, N\}$, pour un horizon de planification constitué de T périodes. Ce plan est à capacité finie et doit tenir compte d'un ensemble de contraintes additionnelles. En effet, le lancement de production d'une référence à une période donnée entraîne une consommation variable en ressources et une consommation fixe usuellement dénommée temps de setup.

[1] MCLS2 est une abréviation en anglais de : Multi-item Capacitated Lot-sizing problem with Setup times and Shortage costs.

[2] Advanced Planning and Scheduling.

Le problème MCLS2 a la particularité d'autoriser des ruptures sur la demande puisque nous traitons des problèmes avec des capacités limitantes. En effet, dans une situation où les capacités disponibles ne sont pas suffisantes pour satisfaire toute la demande et que les retards ne sont pas permis, on essaye de partager la capacité disponible entre les références tout en minimisant le coût total de rupture sur la demande. Ainsi, nous introduisons dans le modèle un coût de rupture unitaire pour chaque référence et à chaque période. Ces coûts doivent être considérés comme des pénalités en comparaison avec les autres coûts. Pour plus de détails sur la modélisation du problème MCLS2, nous invitons le lecteur à se reporter au chapitre 3 page 31.

Afin de satisfaire la demande d'une référence i à une période t, il est possible d'anticiper la production d'un certain nombre de périodes. Par conséquent, on définit le paramètre σ_{it} qui représente la dernière période à laquelle la production d'une référence i à la période t peut être consommée.

Le problème MCLS2 est de trouver un plan de production qui minimise la somme des coûts de production, de stockage ainsi que les coûts de rupture, tout en respectant les contraintes de stockage et de capacité. Nous rappelons dans ce qui suit, les paramètres et les variables du problème MCLS2 :

Les paramètres :
d_{it} : Demande (commandes et prévisions) pour la référence i à la période t.
f_{it} : Besoin fixe en ressources de la référence i à la période t.
v_{it} : Besoin variable en ressources de la référence i à la période t.
c_t : Capacité disponible à la période t.
σ_{it} : Période maximale pour laquelle on peut anticiper la production de la référence i à la période t.
α_{it} : Coût de production unitaire variable de la référence i à la période t.
β_{it} : Coût fixe relatif à un lancement de la référence i à la période t.
γ_{it} : Coût unitaire de stockage de la référence i à la période t.
φ_{it} : Coût unitaire de rupture sur la demande pour la référence i à la période t.

Les variables :
x_{it} : Quantité produite pour la référence i à la période t.
$y_{it} = \begin{cases} 1 \text{ s'il y a production de la référence } i \text{ à la période } t \text{ (i.e. si } x_{it} > 0). \\ 0 \text{ sinon} \end{cases}$
s_{it} : Valeur du stock de la référence i à la fin de la période t.
r_{it} : Rupture sur la demande pour la référence i à la période t (inférieure à la demande).

En utilisant ces variables et paramètres, nous formulons le problème MCLS2 comme suit :

$$\min \sum_{i,t} \alpha_{it} x_{it} + \sum_{i,t} \beta_{it} y_{it} + \sum_{i,t} \gamma_{it} s_{it} + \sum_{i,t} \varphi_{it} r_{it}. \tag{4.1}$$

s.c :

$$s_{i(t-1)} + x_{it} + r_{it} = d_{it} + s_{it}, \quad \forall i, \forall t \tag{4.2}$$
$$\sum_{i=1}^{N} v_{it}x_{it} + \sum_{i=1}^{N} f_{it}y_{it} \leq c_t, \quad \forall t \tag{4.3}$$
$$x_{it} \leq My_{it}, \quad \forall i, \forall t \tag{4.4}$$
$$r_{it} \leq d_{it}, \quad \forall i, \forall t \tag{4.5}$$
$$x_{it}, r_{it}, s_{it} \geq 0, \quad \forall i, \forall t \tag{4.6}$$
$$y_{it} \in \{0, 1\}, \quad \forall i, \forall t \tag{4.7}$$

La fonction objectif (4.1) minimise le coût total induit par le plan de production, à savoir les coûts de production et de stockage, ainsi que les coûts fixes de lancement et les coûts de rupture. La contrainte (4.2) exprime la conservation des flux à travers l'horizon. La contrainte (4.3) représente la contrainte de respect de la capacité. La contrainte (4.4) signifie que s'il y a lancement de production, alors la quantité produite ne devra pas dépasser le majorant de la production M. Celui-ci représente le minimum entre la quantité maximale de la référence pouvant être produite et la demande sur le tronçon de l'horizon $[t, .., \sigma_{it}]$, $M = \min\left\{\frac{c-f_{it}}{v_{it}}, \sum_{t'=t}^{\sigma_{it}} d_{it'}\right\}$. La contrainte (4.6) signifie que les variables x_{it}, r_{it} et s_{it} sont continues non négatives pour toute référence i, pour chaque période t. La dernière contrainte (4.7) exprime le fait que y_{it} est une variable binaire pour toute référence i, pour chaque période t.

Le modèle MCLS2 est appelé également formulation agrégée, car la production d'une référence n'est définie que par sa période de production, contrairement à la formulation basée sur le problème de localisation d'entrepôts [101] où la production est définie par sa période de production et par sa période de consommation.

Dans ce qui suit, nous proposons une formulation du problème MCLS2 basée sur le problème de localisation d'entrepôts. Cette formulation est notée : MCLS2$_{FL}$. En effet, la variable x_{it} peut être redéfinie plus finement, en considérant la période à laquelle cette production est réellement consommée. On pose $w_{itt'}$ la variable qui définit la quantité de la référence i produite à la période t ($t \neq 0$) et consommée à la période t', avec $t \leq t' \leq T$. La variable w_{i0t} représente le stock initial de la référence i au début de l'horizon qui sera consommé à la période t. On a alors,

$$x_{it} = \sum_{t'=t}^{T} w_{itt'}, \ \forall i, \forall t \tag{4.8}$$

La variable s_{it} est reformulée en utilisant la formule suivante :

$$s_{it} = \sum_{t'=0}^{t} \sum_{t''=t+1}^{T} w_{it't''}, \ \forall i, \forall t \tag{4.9}$$

On pose $\alpha'_{itt'} = \alpha_{it} + \gamma_{it} + \gamma_{i(t+1)}, \ldots, \gamma_{i(t'-1)}$ avec $t \leq t'$.

Le problème MCLS2$_{FL}$ est formulé comme suit :

$$\min \sum_{i,t} \sum_{t'=t}^{T} \alpha'_{itt'} w_{itt'} + \sum_{i,t} \beta_{it} y_{it} + \sum_{i,t} \varphi_{it} r_{it} \tag{4.10}$$

s.c :

$$\sum_{t'=1}^{t} w_{itt'} + r_{it} = d_{it}, \qquad \forall i, \forall t \tag{4.11}$$

$$\sum_{i=1}^{N} v_{it} \sum_{t'=t}^{T} w_{itt'} + \sum_{i=1}^{N} f_{it} y_{it} \leq c_t, \qquad \forall t \tag{4.12}$$

$$\sum_{t'=t}^{\sigma_{it}} w_{itt'} \leq M y_{it}, \qquad \forall i, \forall t \tag{4.13}$$

$$w_{itt'} \leq d_{it'} y_{it}, \qquad \forall i, \forall t \tag{4.14}$$

$$r_{it} \leq d_{it}, \qquad \forall i, \forall t \tag{4.15}$$

$$r_{it} \geq 0, \qquad \forall i, \forall t \tag{4.16}$$

$$w_{itt'} \geq 0, \qquad \forall i, \forall t, \forall t' \tag{4.17}$$

$$y_{it} \in \{0, 1\}, \qquad \forall i, \forall t \tag{4.18}$$

Les contraintes (4.11), (4.12) et (4.13) sont respectivement la reformulation de l'équation de flux, de la contrainte de respect de capacité et de la contrainte de production maximum en fonction des nouvelles variables $w_{itt'}$. La contrainte (4.14) exprime le fait que la production de la référence i à la période t pour satisfaire la demande à la période t' doit être inférieure ou égale à la demande de la référence i à la période t. La contrainte (4.16) signifie que la variable r_{it} est continue non négative pour toute référence i, pour chaque période t. La contrainte (4.17) exprime le fait que $w_{itt'}$ est une variable continue non négative pour toute référence i et pour chaque périodes t, t' avec $t' \geq t$. Les contraintes (4.15) et (4.18) sont exactement les mêmes que les contraintes (4.5), (4.7).

Toutes les inégalités valides développées dans ce chapitre sont basées sur le modèle agrégé MCLS2, d'une part, parce que le modèle agrégé est largement utilisé par les industriels ainsi que par DynaSys, et d'autre part, parce que nos résultats sont une généralisation d'autres résultats basés sur des modèles de lot-sizing agrégés. La majorité des tests sont effectuées sur le modèle agrégé. L'inconvénient du modèle désagrégé par rapport au modèle agrégé est le nombre de variables ainsi que la mémoire et le temps CPU requis pour résoudre les relaxations linéaires continues.

Les problèmes de planification de la production impliquant plusieurs références, des capacités limitantes et des temps de setup importants sont souvent rencontrés dans l'industrie. Trigeiro *et al.* [172] sont parmi les premiers à essayer de résoudre de tels problèmes. Les auteurs ont proposé une heuristique lagrangienne pour résoudre le problème de lot-sizing avec une seule ressource, plusieurs références et des temps de setup (MCLS) afin d'obtenir des solutions proches de l'optimum. Cependant, nous pouvons noter que pour les exemples avec des capacités serrées, ils n'ont pas trouvé de solution réalisable.

Belvaux et Wolsey [18], Leung *et al.* [119] et Pochet et Wolsey [149] ont proposé des méthodes exactes pour résoudre des problèmes MCLS. Ceci est fait en renforçant les bornes inférieures par des inégalités valides et en utilisant des solveurs de programmation linéaire en nombres entiers. Barany *et al.* [14] ont défini des inégalités valides pour le problème de lot-sizing à une seule référence et sans contrainte de capacité (ULS). Miller *et al.* [134] ont étudié la structure polyédrique de quelques problèmes de type MCLS. Nous pouvons également citer les travaux de Marchand et Wolsey [123] pour le problème de sac à dos continu qui peut apparaître comme une relaxation de plusieurs problèmes linéaires mixtes tel que le problème MCLS.

Dans le reste du chapitre, les inégalités valides pour l'ensemble des contraintes (4.2)-(4.7) seront dites valides pour MCLS2.

4.2 Relaxation du problème MCLS2 sur une seule période

En se basant sur la formulation du problème MCLS2 décrite dans la section 4.1, nous définissons un sous-modèle simplifié obtenu en considérant la relaxation sur une seule période de MCLS2. Ceci est particulièrement utile pour définir des inégalités valides pour le problème MCLS2. Le but de cette relaxation n'est pas de résoudre chaque période séparément en ne considérant que la demande de la période courante, mais de définir des inégalités valides pour le problème relaxé sur une seule période qui restent valides pour le problème initial en considérant une demande agrégée. Ceci est fait en autorisant des anticipations sur la demande.

Ce modèle est appelé la relaxation sur une seule période du problème MCLS2 avec un stock précédent [131]. Notre approche est similaire à celle utilisée par Constantino [41] et Miller [131] pour définir un ensemble d'inégalités valides pour le problème MCLS basé sur cette relaxation.

Dans cette relaxation, la production d'une référence à une période donnée peut satisfaire la demande d'un ensemble de périodes consécutives. Par conséquent, pour chaque période $t \leq T$ et pour chaque référence $i = 1,\ldots,N$ on utilise le paramètre σ_{it} défini précédemment. tel que : $\sigma_{it} = 1,\ldots,T$. Ceci nous permet de créer un modèle mathématique sur une seule période qui capture l'interaction entre d'une part les restrictions de capacités en t et d'autre part la demande, la production et les lancements de production entre la période t et σ_{it}, pour toute référence $i = 1,\ldots,N$.

Dans ce cas, le but est de définir des inégalités valides pour le problème MCLS2 en considérant des modèles simplifiés obtenus à partir de la relaxation sur une seule période avec un stock précédent.

On note : $\delta^i_{a,b} = \sum_{t=a}^{b} d_{it}$.

Une famille d'inégalités valides simples est donnée par la proposition suivante.

Proposition 1. *Les inégalités*

$$x_{it} + \sum_{t'=t}^{\sigma_{it}} r_{it'} + \left(s_{i,t-1} + \sum_{t'=t+1}^{\sigma_{it}} \delta^i_{t',\sigma_{it}} y_{it'} \right) \geq \delta^i_{t,\sigma_{it}},\ \forall i, \forall t \tag{4.19}$$

sont valides pour le MCLS2.

Preuve. Si on somme les équations (4.2) sur le tronçon d'horizon $[t, \ldots, \sigma_{it}]$, on obtient :

$$\sum_{t'=t}^{\sigma_{it}}(x_{it'} + r_{it'}) - s_{i,\sigma_{it}} + s_{i,t-1} = \sum_{t'=t}^{\sigma_{it}} d_{it'},\ \forall i, \forall t \tag{4.20}$$

La variable x_{it} peut être redéfinie plus finement, en considérant la période à laquelle cette production est réellement consommée. Cette reformulation est équivalente à celle du modèle MCLS2$_{FL}$. On pose $w_{itt'}$ la variable qui définit la quantité de la référence i produite à la période t ($t \neq 0$) et consommée à la période t', avec $t \leq t' \leq T$. La variable w_{i0t} représente le stock initial de la référence i au début de l'horizon qui sera consommé à la période t. On a alors,

$$x_{it} = \sum_{t'=t}^{T} w_{itt'},\ \forall i, \forall t \tag{4.21}$$

On peut également écrire :

$$s_{it} = \sum_{t'=0}^{t}\sum_{t''=t+1}^{T} w_{it't''},\ \forall i, \forall t \tag{4.22}$$

En remplaçant (4.21) et (4.22) dans (4.20), on aura pour tout $i = 1, \ldots, N$ et $t = 1, \ldots, T$:

$$s_{i,t-1} + x_{it} + \sum_{t'=t+1}^{\sigma_{it}}\sum_{t''=t'}^{T} w_{it't''} + \sum_{t'=t}^{\sigma_{it}} r_{it'} - \sum_{t'=0}^{\sigma_{it}}\sum_{t''=\sigma_{it}+1}^{T} w_{it't''} = \sum_{t'=t}^{\sigma_{it}} d_{it'} \tag{4.23}$$

De plus :

$$\sum_{t'=t+1}^{\sigma_{it}}\sum_{t''=t'}^{T} w_{it't''} = \sum_{t'=t+1}^{\sigma_{it}}\sum_{t''=t'}^{\sigma_{it}} w_{it't''} + \sum_{t'=t+1}^{\sigma_{it}}\sum_{t''=\sigma_{it}+1}^{T} w_{it't''} \tag{4.24}$$

et :

$$\sum_{t'=0}^{\sigma_{it}}\sum_{t''=\sigma_{it}+1}^{T} w_{it't''} = \sum_{t'=0}^{t}\sum_{t''=\sigma_{it}+1}^{T} w_{it't''} + \sum_{t'=t+1}^{\sigma_{it}}\sum_{t''=\sigma_{it}+1}^{T} w_{it't''} \tag{4.25}$$

En remplaçant (4.24) et (4.25) dans (4.23), on a pour tout $i = 1, \ldots, N$ et $t = 1, \ldots, T$:

$$s_{i,t-1} + x_{it} + \sum_{t'=t+1}^{\sigma_{it}}\sum_{t''=t'}^{\sigma_{it}} w_{it't''} - \sum_{t'=0}^{t}\sum_{t''=\sigma_{it}+1}^{T} w_{it't''} + \sum_{t'=t}^{\sigma_{it}} r_{it'} = \sum_{t'=t}^{\sigma_{it}} d_{it'} \tag{4.26}$$

D'après la définition de la variable $w_{it't''}$, on sait que :

1. $w_{it't''} \leq d_{it''} y_{it'}$

2. $w_{i0t} \leq d_{it}$
3. $\sum_{t'=0}^{t} \sum_{t''=\sigma_{it}+1}^{T} w_{it't''} = 0$

Par conséquent, de (4.26), on obtient pour tout $i = 1, \ldots, N$ et $t = 1, \ldots, T$:

$$s_{i,t-1} + x_{it} + \sum_{t'=t+1}^{\sigma_{it}} \sum_{t''=t'}^{\sigma_{it}} d_{it''} y_{it'} + \sum_{t'=t}^{\sigma_{it}} r_{it'} \geq \sum_{t'=t}^{\sigma_{it}} d_{it'}$$

De plus,

$$\sum_{t''=t'}^{\sigma_{it}} d_{it''} = \delta^i_{t',\sigma_{it}}$$

Finalement,

$$x_{it} + \sum_{t'=t}^{\sigma_{it}} r_{it'} + s_{i,t-1} + \sum_{t'=t+1}^{\sigma_{it}} \delta^i_{t',\sigma_{it}} y_{it'} \geq \delta^i_{t,\sigma_{it}}$$

□

Dans le reste du chapitre, on note SPMCLS2[3] la relaxation du problème MCLS2 sur une seule période où la contrainte (4.2) est remplacée par (4.19). Comme il a été mentionné précédemment, les inégalités valides pour l'ensemble de contraintes (4.3)-(4.19) seront dites valides pour SPMCLS2.

L'expression $s_{i,t-1} + \sum_{t'=t+1}^{\sigma_{it}} \delta^i_{t',\sigma_{it}} y_{it'}$ peut être considérée comme étant le stock sortant de la référence i à la période $t-1$ et noté $\widetilde{s}_{i,t-1}$. Ainsi, nous avons $\widetilde{s}_{i,t-1} = s_{i,t-1} + \sum_{t'=t+1}^{\sigma_{it}} \delta^i_{t',\sigma_{it}} y_{it'}$. D'une manière similaire, on note $\sum_{t'=t}^{\sigma_{it}} r_{it'}$ par $\widetilde{r}_{it}$ et $\delta^i_{t,\sigma_{it}}$ par $\widetilde{d}_{it}$. Les inégalités (4.19) sont équivalentes à :

$$x_{it} + \widetilde{r}_{it} + \widetilde{s}_{it} \geq \widetilde{d}_{it} \quad \forall i, \forall t \tag{4.27}$$

Puisque nous travaillons sur chaque période séparément pour le SPMCLS2 et étant donné que chaque période est traitée séparément, nous avons supprimé l'indice temporel dans l'expression précédente afin de faciliter la lecture des formulations mathématiques qui suivent.

Les inégalités (4.27) s'écrivent :

$$x_i + \widetilde{r}_i + \widetilde{s}_i \geq \widetilde{d}_i \quad \forall i \tag{4.28}$$

4.3 Inégalités (l, S) pour le problème SPMCLS2

Afin d'introduire les inégalités (l, S) pour le problème SPMCLS2, définissons tout d'abord le problème suivant noté MCLSP qui n'est qu'une version simplifiée du problème MSCL2 avec des v_{it} égales à 1 et aucune rupture sur les demandes n'est autorisée, les variables de ruptures

[3]SPMCLS2 est une abréviation en anglais de : Single Period relaxation of MLCS2.

r_{it} sont alors fixées à zéro. Ceci va nous permettre de rappeler la structure de quelques inégalités valides pour des relaxations du problème MCLSP.

$$\min \sum_{i=1}^{N} \sum_{t=1}^{T} \alpha_{it} x_{it} + \beta_{it} y_{it} + \gamma_{it} s_{it} \tag{4.29}$$

$$x_{it} - s_{it} + s_{i,t-1} = d_{it}, \qquad \forall i, \forall t \tag{4.30}$$
$$\sum_{i=1}^{N} x_{it} + \sum_{i=1}^{N} f_{it} y_{it} \leq c_t, \quad \forall t \tag{4.31}$$
$$x_{it}, s_{it} \geq 0, \ \forall i, \qquad \forall t \tag{4.32}$$
$$y_{it} \in \{0,1\}, \qquad \forall i, \forall t \tag{4.33}$$

On note :
- SPMCLSP la relaxation sur une seule période du problème MCLSP où la contrainte (4.30) est remplacée par $x_i + s_i \geq d_i$. On rappelle que l'indice temporel est supprimé ;
- ULS la relaxation sur une seule référence du problème MCLSP sans contrainte de capacité. Dans ce problème, la contrainte de capacité qui relie toutes les références est relaxée. Ainsi, chaque article est considéré séparément et l'indice des références est inutile.

Barany *et al.* [14, 15] ont prouvé qu'une description polyédrique complète de l'enveloppe convexe du problème ULS est donnée par des inégalités issues de la relaxation linéaire et d'une autre famille d'inégalités dite (l, S), elles sont données par :

$$\sum_{t \in S} x_t + \sum_{t \in \bar{S}} d_{tl} y_t \geq d_{1l}$$

où $l \in \{0, 1, \ldots, T\}$, $S \subset \{0, 1, \ldots, l\}$, $\bar{S} = \{0, 1, \ldots, l\} \setminus S$ et $d_{tt'} = \sum_{k=t}^{t'} d_k$. Les auteurs ont présenté des résultats expérimentaux intéressants pour le problème MCLSP en utilisant les inégalités (l, S) dans une méthode branch-and-cut.

Proposition 2. *Inégalités (l, S) pour le problème SPMCLSP, [132]*
Si $c \geq f_i + d_i$ alors les inégalités

$$s_i + d_i y_i \geq d_i \quad \forall i \tag{4.34}$$

définissent des facettes pour SPMCLSP.

Pochet et Wolsey [151] ont introduit les inégalités (k, l, S, I) pour le problème de lot-sizing à une seule référence et des capacités constantes sur tout l'horizon CCLS. Les auteurs ont montré que ces inégalités définissent des facettes non triviales de l'ensemble convexe des solutions réalisables. Les inégalités (k, l, S, I) peuvent être exprimées sous la forme générale suivante :

$$s_{k-1} + \sum_{t \in U} x_t + \sum_{t \in V} B_t y_t \geq B_0$$

pour tout k et l tel que $1 \leq k \leq l \leq T$, (U, V) est une partition de $[k, \ldots, l]$, $B_0 \in \mathbb{R}_+$ et $B_t \in \mathbb{R}_+$ $(t \in V)$. B_t est défini en respectant les demandes et les contraintes de capacité. Pour plus de détails, le lecteur peut se référer à [151].

Sans perte de généralité, on peut modifier les inégalités (4.19) afin d'avoir une structure similaire aux inégalités (k, l, S, I), en remplaçant $\delta^i_{t',\sigma_{it}} y_{it'}$ par $x_{it'}$ pour un sous-ensemble quelconque de $[t+1, \ldots, \sigma_{it}]$.

Proposition 3. *Etant donnée une partition (U, V) de l'ensemble $[t+1, \ldots, \sigma_{it}]$, les inégalités*

$$x_{it} + \sum_{t'=t}^{\sigma_{it}} r_{it'} + \left(s_{i,t-1} + \sum_{U} \delta^i_{t',\sigma_{it}} y_{it'} + \sum_{V} x_{it'} \right) \geq \delta^i_{t,\sigma_{it}}, \ \forall i, \forall t \tag{4.35}$$

sont valides pour MCLS2.

Preuve. La preuve est similaire à celle de la proposition 1. □

Les inégalités (4.35) sont appelées les inégalités (l, S) pour le problème MCLS2.

4.4 Heuristique de séparation des inégalités (l, S)

Dans cette section, nous présentons une heuristique de séparation afin de créer des inégalités (l, S) pour le problème MCLS2. En se basant sur la proposition 3, les inégalités (4.35) sont valides pour MCLS2.

L'idée de cette heuristique de séparation est de créer un ensemble $U \subset [t, \ldots, \sigma_{it}]$ pour toute référence i et pour toute période t dans le but de générer une inégalité valide (l, S) pour le problème MCLS2. On rajoute t' à l'ensemble U si $\delta^i_{t',\sigma_{it}} y_{it'} < x_{it'}$ ou à l'ensemble V sinon. Le principe de l'heuristique est illustré par l'algorithme 1 défini comme suit :

Algorithme 1 Heuristique de séparation pour les inégalités (l, S)

```
t ← 1, i ← 1
tant que (i ≤ N) faire
   tant que (t ≤ T) faire
      t' ← t + 1
      tant que (t' ≤ σ_it) faire
         si δ^i_{t',σ_it} y*_{it'} < x*_{it'} alors
            U ← U ∪ {t'}
         sinon
            V ← V ∪ {t'}
         finsi
         t' ← t' + 1
      fin tant que
      si (L'inégalité (4.35) basée sur S, U et T' est violée) alors
         Ajouter l'inégalité (4.35) au nœud courant.
      finsi
      t ← t + 1
   fin tant que
   i ← i + 1
fin tant que
```

La complexité de l'heuristique de séparation des inégalités (l, S) est $O(NT^2)$.

4.5 Inégalités couvrantes et couvrantes inverses pour SPMCLS2

Dans cette section, nous généralisons les résultats de Miller *et al.* [132] concernant *les inégalités couvrantes et les inégalités couvrantes inverses* (*cover and reverse cover inequalities*).

Défnition 1. *(Couverture)*
Un sous-ensemble de références S de $\mathcal{I}$ est dit "couverture" du problème SPMCLS2 si :

$$\lambda_S = \sum_{i \in S} \left(f_i + v_i \widetilde{d}_i \right) - c \geq 0 \tag{4.36}$$

λ_S exprime le déficit en terme de ressources lorsqu'est produite toute la demande des références de S. Si $\lambda_S > 0$ alors la demande sur l'ensemble des références de S est strictement supérieure à la capacité de production disponible.

Proposition 4. *(Inégalités couvrantes)*

L'inégalité

$$\sum_{i \in S} v_i \left(\widetilde{s}_i + \widetilde{r}_i \right) \geq \lambda_S + \sum_{i \in S} \max \left\{ -f_i, v_i \widetilde{d}_i - \lambda_S \right\} (1 - y_i) \tag{4.37}$$

est valide pour SPMCLS2.

Preuve. La preuve est similaire à celle présentée dans Miller *et al.* [132] en rajoutant les variables de ruptures $\widetilde{r}_i$ sur la demande ainsi que des besoins variables v_i. Plus précisément :

Les inégalités (4.28) peuvent s'écrire comme suit :

$$\widetilde{s}_i + \widetilde{r}_i \geq \widetilde{d}_i - x_i, \ \forall i$$

alors :

$$\sum_{i \in S} v_i \widetilde{s}_i + \sum_{i \in S} v_i \widetilde{r}_i \geq \sum_{i \in S} v_i \widetilde{d}_i - \sum_{i \in S} v_i x_i$$

Si toutes les références de l'ensemble S sont produites, i.e. : $y_i = 1 \ \forall i \in S$ alors : d'après (4.3) on a :

$$\sum_{i \in S} v_i x_i \leq c - \sum_{i \in S} f_i$$

D'où :

$$\sum_{i\in S} v_i\widetilde{s}_i + \sum_{i\in S} v_i\widetilde{r}_i \geq \sum_{i\in S} v_i\widetilde{d}_i - \left(c - \sum_{i\in S} f_i\right) = \sum_{i\in S}\left(v_i\widetilde{d}_i + f_i\right) - c$$

En remplaçant $\sum_{i\in S}\left(v_i\widetilde{d}_i + f_i\right) - c$ par λ_S on aura :

$$\sum_{i\in S} v_i\left(\widetilde{s}_i + \widetilde{r}_i\right) \geq \lambda_S \tag{4.38}$$

On définit un ensemble $S^0 = \{i \in S / y_i = 0\}$ qui représente les référence de S qui ne sont pas produites.

Si $\left|S^0\right| = 1$, on a exactement une référence $i' \in S$ tel que $y_{i'} = 0$.

D'après (4.38) on peut écrire :

$$\sum_{i\in S} v_i\left(\widetilde{s}_i + \widetilde{r}_i\right) \geq \lambda_S - f_{i'} \tag{4.39}$$

On sait que :

$$\sum_{i\in S} v_i\left(\widetilde{s}_i + \widetilde{r}_i\right) \geq v_{i'}\left(\widetilde{s}_{i'} + \widetilde{r}_{i'}\right) \geq v_{i'}\widetilde{d}_{i'} \tag{4.40}$$

D'après (4.39) et (4.40) on peut conclure que :

$$\sum_{i\in S} v_i\left(\widetilde{s}_i + \widetilde{r}_i\right) \geq \lambda_S + \max\left\{-f_{i'}, v_{i'}\widetilde{d}_{i'} - \lambda_S\right\} \tag{4.41}$$

Considérons maintenant le cas où $\left|S^0\right| > 1$. Le résultat (4.41) peut facilement être généralisé pour le cas $\left|S^0\right| > 1$ en traitant les références qui appartiennent à S^0 une par une. Par conséquent, nous obtenons :

$$\sum_{i\in S} v_i\left(\widetilde{s}_i + \widetilde{r}_i\right) \geq \lambda_S + \sum_{i\in S^0} \max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\} \tag{4.42}$$

L'inégalité (4.42) peut être généralisée pour l'ensemble S en introduisant le terme $(1 - y_i)$ afin de prendre en considération la production d'une référence, d'où l'inégalité :

$$\sum_{i\in S} v_i\left(\widetilde{s}_i + \widetilde{r}_i\right) \geq \lambda_S + \sum_{i\in S} \max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\}(1 - y_i)$$

□

Les inégalités précédentes peuvent être renforcées par une procédure de *lifting* décrite dans ce qui suit. Cette procédure consiste à renforcer les inégalités valides en rajoutant des références n'appartenant pas à la couverture S.

Proposition 5. *(Une seconde forme d'inégalités couvrantes)*
Étant donnés S une couverture de SPMCLS2, et un ordre des références $i \in S$ tel que $f_{[1]} + v_{[1]}\widetilde{d}_{[1]} \geq \cdots \geq f_{[|S|]} + v_{[|S|]}\widetilde{d}_{[|S|]}$. Soit $T = \mathcal{I} \setminus S$ et (T', T'') une partition quelconque de T. On pose $\mu_1 = f_{[1]} + v_{[1]}\widetilde{d}_{[1]} - \lambda_S$ et $f_{[2]} + v_{[2]}\widetilde{d}_{[2]} \geq \lambda_S$, l'inégalité

$$\sum_{i \in S} v_i \left(\widetilde{s}_i + \widetilde{r}_i\right) \geq \lambda_S + \sum_{i \in S} \max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\}(1 - y_i) + \frac{\lambda_S}{f_{[2]} + v_{[2]}\widetilde{d}_{[2]}} \sum_{i \in T'} \left(v_i x_i - (\mu_1 - f_i)\, y_i\right) \tag{4.43}$$

est valide pour SPMCLS2.

Preuve. Soit $(x^*, y^*, \widetilde{s}^*, \widetilde{r}^*)$ un point de l'enveloppe convexe de SPMCLS2. Soit $S^0 = \{i \in S : y_i^* = 0\}$, $S^1 = \{i \in S : y_i^* = 1\}$ et $\bar{T}' = \{i \in T' : y_i^* = 1\}$. On considère trois cas :

Si $\left|\bar{T}'\right| = 0$, alors l'inégalité (4.43) est valide, puisqu'on est dans le cas de la proposition 4.
Si $\left|\bar{T}'\right| = 1$, alors on suppose que $\bar{T}' = \{i'\}$; pour montrer que le point $(x^*, y^*, \widetilde{s}^*, \widetilde{r}^*)$ satisfait l'inégalité (4.43), il suffit de montrer que :

$$\begin{aligned}\sum_{i \in S} v_i \left(\widetilde{s}_i^* + \widetilde{r}_i^*\right) + \sum_{i \in S} \max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\} y_i^* \quad \geq \quad & \lambda_S + \sum_{i \in S} \max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\} + \\ & \frac{\lambda_S}{f_{[2]} + v_{[2]}\widetilde{d}_{[2]}} \left(v_{i'} x_{i'}^* - (\mu_1 - f_{i'})\right)\end{aligned} \tag{4.44}$$

On sait que :

$$\sum_{i \in S} v_i \left(\widetilde{s}_i^* + \widetilde{r}_i^*\right) + \sum_{i \in S} \max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\} y_i^* \geq \min_{x_{i'} = x_{i'}^*, y_{i'} = 1} \left\{\sum_{i \in S} v_i \left(\widetilde{s}_i + \widetilde{r}_i\right) + \sum_{i \in S} \max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\} y_i\right\}$$

On considère le problème de minimisation suivant :

$$\min_{x_{i'} = x_{i'}^*, y_{i'} = 1} \left\{\sum_{i \in S} v_i \left(\widetilde{s}_i + \widetilde{r}_i\right) + \sum_{i \in S} \max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\} y_i\right\} \tag{4.45}$$

Nous montrons que la solution optimale de ce problème a une valeur supérieure ou égale au membre droit de l'inégalité (4.44).

Soit $r_S = \left|\left\{i \in S : f_i + v_i\widetilde{d}_i > \lambda_S\right\}\right|$ et $A_a^b = \sum_a^b \left(f_{[i]} + v_{[i]}\widetilde{d}_{[i]}\right)$.

Si on définit :

$$\varphi_S(u) = \begin{cases} u, & if\ 0 \leq u \leq A_{r_S+1}^{|S|} \\ A_{r_S+1}^{|S|} + (r_S - j)\,\lambda_S, & if\ A_{j+1}^{|S|} \leq u \leq A_j^{|S|} - \lambda_S, j = 1, \ldots, r_S \\ A_{r_S+1}^{|S|} + (r_S - j)\,\lambda_S + \left(u - \left(A_j^{|S|} - \lambda_S\right)\right), & if\ A_j^{|S|} - \lambda_S \leq u \leq A_j^{|S|}, j = 2, \ldots, r_S \end{cases} \tag{4.46}$$

La solution optimale du problème de minimisation (4.45) est donnée par :

$$\min_{x_{i'} = x_{i'}^*, y_{i'} = 1} \left\{\sum_{i \in S} v_i \left(\widetilde{s}_i + \widetilde{r}_i\right) + \sum_{i \in S} \max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\} y_i\right\} = \sum_{i \in S} v_i\widetilde{d}_i + \varphi_S\left(c - \left(f_{i'} + v_{i'} x_{i'}^*\right)\right) \tag{4.47}$$

La démonstration n'est qu'une généralisation d'un résultat présenté par Miller *et al.* [132]. Une représentation de φ_S est donnée par la figure 4.1.

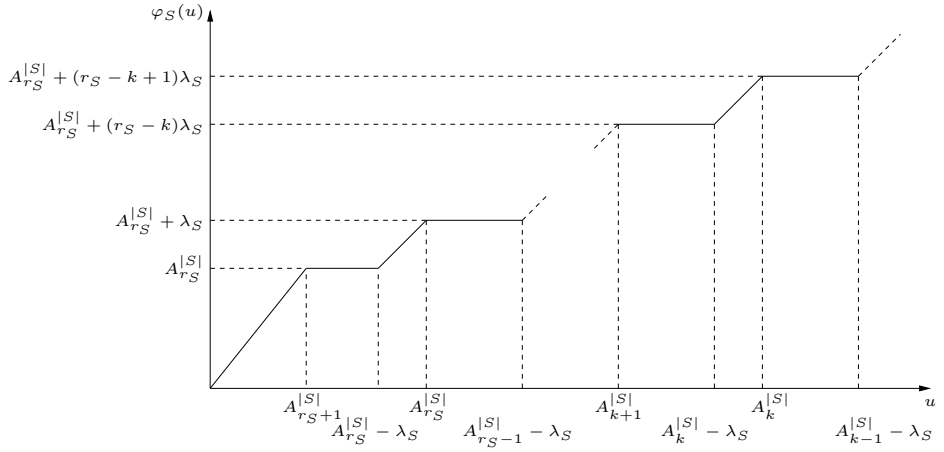


FIG. 4.1 – Fonction φ_S.

Maintenant, il reste à démontrer que :

$$\sum_{i\in S} v_i\widetilde{d}_i - \varphi_S\left(c - (f_{i'} + v_{i'}x^*_{i'})\right) \geq \lambda_S + \sum_{i\in S}\max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\} + \frac{\lambda_S}{f_{[2]} + v_{[2]}\widetilde{d}_{[2]}}\left(v_{i'}x^*_{i'} - (\mu_1 - f_{i'})\right) \tag{4.48}$$

Pour ce faire, nous utilisons la proporiété suivante,

$$\max_{0\leq x_i\leq\frac{c-f_i}{v_i}}\left\{\varphi_S\left(c - (f_i + v_ix_i)\right) + \frac{\lambda_S}{f_{[2]} + v_{[2]}\widetilde{d}_{[2]}}\left(v_ix_i - (\mu_1 - f_i)\right)\right\} = \sum_{i\in S\setminus[1]}\min\left\{f_i + v_i\widetilde{d}_i, \lambda_S\right\} \tag{4.49}$$

De plus, on a :

$$\begin{aligned}
\sum_{i\in S} v_i\widetilde{d}_i &= v_{[1]}\widetilde{d}_{[1]} + \sum_{i\in S\setminus[1]} v_i\widetilde{d}_i \\
\sum_{i\in S} v_i\widetilde{d}_i &= v_{[1]}\widetilde{d}_{[1]} + \sum_{i\in S\setminus[1]}\max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\} + \sum_{i\in S\setminus[1]}\min\left\{f_i + v_i\widetilde{d}_i, \lambda_S\right\}
\end{aligned}$$

En utilisant l'expression (4.49), on a :

$$\begin{aligned}
\sum_{i\in S} v_i\widetilde{d}_i &= v_{[1]}\widetilde{d}_{[1]} + \sum_{i\in S\setminus[1]}\max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\} + \\
&\qquad \max_{0\leq x_{i'}\leq\frac{c-f_{i'}}{v_{i'}}}\left\{\varphi_S\left(c - (f_{i'} + v_{i'}x_{i'})\right) + \frac{\lambda_S}{f_{[2]} + v_{[2]}\widetilde{d}_{[2]}}\left(v_{i'}x_{i'} - (\mu_1 - f_{i'})\right)\right\}
\end{aligned}$$

$$\begin{aligned}\sum_{i\in S} v_i\widetilde{d}_i &\geq v_{[1]}\widetilde{d}_{[1]} + \sum_{i\in S\setminus[1]} \max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\} + \\ &\quad \varphi_S\left(c - (f_{i'} + v_{i'}x^*_{i'})\right) + \frac{\lambda_S}{f_{[2]} + v_{[2]}\widetilde{d}_{[2]}}\left(v_{i'}x^*_{i'} - (\mu_1 - f_{i'})\right)\end{aligned}$$

$$\begin{aligned}\sum_{i\in S} v_i\widetilde{d}_i &\geq v_{[1]}\widetilde{d}_{[1]} - \lambda_S + \lambda_S + \sum_{i\in S\setminus[1]} \max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\} + \\ &\quad \varphi_S\left(c - (f_{i'} + v_{i'}x^*_{i'})\right) + \frac{\lambda_S}{f_{[2]} + v_{[2]}\widetilde{d}_{[2]}}\left(v_{i'}x^*_{i'} - (\mu_1 - f_{i'})\right) \qquad (4.50)\end{aligned}$$

Puisque $v_{[1]}\widetilde{d}_{[1]} - \lambda_S \geq -f_{[1]}$ (on sait que : $f_{[1]} + v_{[1]}\widetilde{d}_{[1]} \geq \lambda_S$), $v_{[1]}\widetilde{d}_{[1]} - \lambda_S$ peut être remplacé par $\max\left\{-f_{[1]}, v_{[1]}\widetilde{d}_{[1]} - \lambda_S\right\}$. En réécrivant l'expression (4.50), on obtient (4.48). D'où l'inégalité (4.44).

Si $\left|\bar{T}'\right| > 1$, alors l'expression (4.44) peut être facilement généralisée en traitant les références qui appartiennent à $\bar{T}'$ une par une. D'où l'inégalité (4.43).

□

Dans ce qui suit, nous décrivons une autre classe d'inégalités valides basées sur des ensembles de couvertures inverses.

Défnition 2. *(Couverture Inverse)*
Un sous-ensemble de références S de $\mathcal{I}$ est dit couverture inverse de SPMCLS2 si :

$$\mu_S = c - \sum_{i\in S}\left(f_i + v_i\widetilde{d}_i\right) \geq 0 \qquad (4.51)$$

Pour une couverture inverse S. μ_S exprime la capacité en ressources disponible lorsque toute la demande pour chaque référence appartenant à l'ensemble S est produite.

Proposition 6. *(Inégalités couvrantes inverses)*

Soit S une couverture inverse de SPMCLS2, soit $T = \mathcal{I} \setminus S$ et (T', T'') une partition quelconque de T. L'inégalité

$$\sum_{i\in S} v_i\left(\widetilde{s}_i + \widetilde{r}_i\right) \geq \left(\sum_{i\in S}\left(f_i + v_i\widetilde{d}_i\right)\right)\sum_{i\in T'} y_i - \sum_{i\in S} f_i\left(1 - y_i\right) - \sum_{i\in T'}\left(\left(c - f_i\right)y_i - v_i x_i\right) \qquad (4.52)$$

est valide pour SPMCLS2.

Preuve. La démonstration présentée est similaire à celle décrite dans Miller *et al.* [132] avec prise en compte additionnelle des variables de ruptures sur la demande $\widetilde{r}_i$ ainsi que des besoins variables v_i.

Soit $(x^*, y^*, \widetilde{s}^*, \widetilde{r}^*)$ un point quelconque de l'enveloppe convexe des solutions réalisables de SPMCLS2.
On doit considérer trois cas :
Si $y_i^* = 0$ pour tout $i \in T'$, alors l'inégalité est valide, puisque $\sum_{i\in S} v_i\,(\widetilde{s}_i^* + \widetilde{r}_i^*) \geq -\sum_{i\in S} f_i\,(1 - \widetilde{y}_i^*)$.

Soit $\bar{T}' = \left\{ j \in T' : y_j^* = 1 \right\}$

Si $\left|\bar{T}'\right| = 1$, supposons que $\bar{T}' = \{i'\}$

D'après (4.3) on a :

$$c - f_{i'} \geq \sum_{i\in S} (v_i x_i^* + f_i y_i^*) + v_{i'} x_{i'}^*$$

D'après (4.28) on a aussi :

$$x_i^* \geq \widetilde{d}_i - \widetilde{s}_i^* - \widetilde{r}_i^*$$

D'où :

$$c - f_{i'} \geq \sum_{i\in S} \left(v_i \left(\widetilde{d}_i - \widetilde{s}_i^* - \widetilde{r}_i^* \right) + f_i y_i^* \right) + v_{i'} x_{i'}^*$$

Ce qui donne :

$$\sum_{i\in S} v_i\,(\widetilde{s}_i^* + \widetilde{r}_i^*) \geq \sum_{i\in S} v_i \widetilde{d}_i + \sum_{i\in S} f_i y_i^* - ((c - f_{i'}) - v_{i'} x_{i'}^*)$$

L'inégalité

$$\sum_{i\in S} v_i\,(\widetilde{s}_i^* + \widetilde{r}_i^*) \geq \left(\sum_{i\in S} \left(f_i + v_i \widetilde{d}_i \right) \right) - \sum_{i\in S} f_i\,(1 - y_i^*) - ((c - f_{i'}) - v_{i'} x_{i'}^*) \qquad (4.53)$$

est donc valide pour SPMCLS2.

Si $\left|\bar{T}'\right| > 1$, l'inégalité (4.53) peut facilement être généralisée en traitant les références qui appartiennent à $\bar{T}'$ une par une, d'où l'inégalité (4.52).

□

4.6 Etude polyédrale

Dans cette section, nous présentons une étude polyédrale du problème SPMCLS2. Nous décrivons les points et les rayons extrêmes de l'enveloppe convexe des solutions réalisables du problème SPMCLS2 notée conv(SPMCLS2). Nous montrons que les inégalités couvrantes (4.37) définissent des facettes de conv(SPMCLS2) sous certaines conditions. (de façon similaire les inégalités couvrantes inverses (4.52) définissent des facettes de conv(SPMCLS2) sous certaines conditions).

Afin de montrer que l'inégalité valide (4.37) induit une facette du problème SPMCLS2, il suffit de montrer que l'intersection de l'hyperplan défini par l'inégalité (4.37) et conv(SPMCLS2) est de dimension dim(SPMCLS2)−1. Il suffira donc d'exhiber dim(SPMCLS2) points et rayons

extrêmes de conv(SPMCLS2), qui soient également dans l'hyperplan défini par (4.37), et qui soient linéairement indépendants. Nous allons décrire dans ce qui suit, les points extrêmes et les rayons extrêmes du problème SPMCLS2. Nous rappelons tout d'abord le modèle mathématique du problème SPMCLS2.

$$\min \sum_i \alpha_i x_i + \sum_i \beta_i y_i + \sum_i \gamma_i \widetilde{s}_i + \sum_i \varphi_i \widetilde{r}_i. \tag{4.54}$$

s.c :

$$\widetilde{s}_i + x_i + \widetilde{r}_i \geq \widetilde{d}_i, \qquad \forall i \tag{4.55}$$
$$\textstyle\sum_{i=1}^{N} v_i x_i + \sum_{i=1}^{N} f_i y_i \leq c, \tag{4.56}$$
$$x_i \leq \min\left\{\tfrac{c-f_i}{v_i}, \widetilde{d}_i\right\} y_i, \qquad \forall i \tag{4.57}$$
$$\widetilde{r}_i \leq \widetilde{d}_i, \qquad \forall i \tag{4.58}$$
$$x_i, \widetilde{r}_i, \widetilde{s}_i \geq 0, \qquad \forall i \tag{4.59}$$
$$y_i \in \{0,1\}, \qquad \forall i \tag{4.60}$$

4.6.1 Points et rayons extrêmes

Etant donné $(\bar{x}, \bar{y}, \bar{s}, \bar{r})$ un point extrême de conv(SPMCLS2). On pose, $Q = \{i \in \mathcal{I} : \bar{y}_i = 1\}$, $Q_u = \left\{i \in Q : \bar{x}_i = \widetilde{d}_i\right\}$, $Q_m = \left\{i \in Q : \bar{x}_i > 0, \bar{x}_i \neq \widetilde{d}_i\right\}$, $Q_l = \{i \in Q : \bar{x}_i = 0\}$. On définit pour $(\bar{x}_i, \bar{y}_i, \bar{s}_i, \bar{r}_i)$ les types suivants :

Type 1 : $\bar{x}_i = \widetilde{d}_i, \bar{y}_i = 1, \bar{s}_i = 0, \bar{r}_i = 0$.
Type 2 : $\bar{x}_i = 0, \bar{y}_i = 1, \bar{s}_i = \widetilde{d}_i, \bar{r}_i = 0$.
Type 3 : $\bar{x}_i = 0, \bar{y}_i = 1, \bar{s}_i = 0, \bar{r}_i = \widetilde{d}_i$.
Type 4 : $\bar{x}_i = 0, \bar{y}_i = 0, \bar{s}_i = \widetilde{d}_i, \bar{r}_i = 0$.
Type 5 : $\bar{x}_i = 0, \bar{y}_i = 0, \bar{s}_i = 0, \bar{r}_i = \widetilde{d}_i$.

Proposition 7. *Tout point extrême $(\bar{x}, \bar{y}, \bar{s}, \bar{r})$ de conv(SPMCLS2) est de la forme :*

- *$(\bar{x}_i, \bar{y}_i, \bar{s}_i, \bar{r}_i)$ est de type 1, $i \in Q_u$;*
- *$(\bar{x}_i, \bar{y}_i, \bar{s}_i, \bar{r}_i)$ est de type 2 ou de type 3, $i \in Q_l$;*
- *$(\bar{x}_i, \bar{y}_i, \bar{s}_i, \bar{r}_i)$ est de type 4 ou de type 5, $i \in \mathcal{I} \setminus Q$;*
- *$\bar{x}_{i'} = c - \sum_{i \in Q_u} \left(f_i + v_i \widetilde{d}_i\right) - \sum_{i \in Q_l} f_i - f_{i'}$, $\bar{y}_{i'} = 1$, $\bar{s}_{i'} = (\widetilde{d}_i - \bar{x}_{i'})^+$, $\bar{r}_{i'} = 0$ ou $\bar{x}_{i'} = c - \sum_{i \in Q_u} \left(f_i + v_i \widetilde{d}_i\right) - \sum_{i \in Q_l} f_i - f_{i'}$, $\bar{y}_{i'} = 1$, $\bar{s}_{i'} = 0$, $\bar{r}_{i'} = (\widetilde{d}_i - \bar{x}_{i'})^+$, $i' \in Q_m$.*

avec : $Q_m = \emptyset$ ou $|Q_m| = 1$.

Preuve. Tous les points listés sont extrêmes. La raison est la suivante :

Pour $i \in \mathcal{I} \setminus Q$, chacune des variables $\bar{x}_i$, $\bar{y}_i$, $\bar{s}_i$ et $\bar{r}_i$ est fixée à sa borne supérieure, à sa borne inférieure ou à la valeur maximum autorisée par la contrainte (4.55). Pour $i \in Q_m$, $\bar{y}_i$ est fixée à sa borne supérieure, $\bar{x}_i$ est fixée à la valeur maximale autorisée par la contrainte (4.56). $\bar{s}_i$ et $\bar{r}_i$ ont la valeur minimale autorisée par la contrainte (4.55) et leur borne inférieure. La même remarque peut être faite pour $i \in Q_u$ et $i \in Q_l$. Aucun de ces points ne peut être exprimé comme combinaison convexe d'autres points de conv(SPMCLS2).

Il faut maintenant monter que ces points sont les seuls points extrêmes de conv(SPMCLS2). Etant donné $(\bar{x}, \bar{y}, \bar{s}, \bar{r})$ un point extrême. Par définition, $i \in Q$ ou $i \in \mathcal{I} \setminus Q$ pour $i = 1, \ldots, N$.

Si $Q_m = \emptyset$ alors il est évident que le point $(\bar{x}, \bar{y}, \bar{s}, \bar{r})$ est de la forme de la proposition 7.

Si $Q_m \neq \emptyset$,

Supposons que $|Q_m| = 1$, soit $Q_m = \{i'\}$. Alors on doit avoir $(\bar{s}_i = (\widetilde{d}_i - \bar{x}_{i'})^+$ et $\bar{r}_i = 0)$ ou $(\bar{s}_i = 0$ et $\bar{r}_i = (\widetilde{d}_i - \bar{x}_{i'})^+)$ sinon $(\bar{x}, \bar{y}, \bar{s}, \bar{r})$ peut être exprimé comme combinaison convexe de deux autres points.

Supposons maintenant que $|Q_m| > 1$ pour un point $(\bar{x}, \bar{y}, \bar{s}, \bar{r})$ dans conv(SPMCLS2), alors il est facile de construire deux points tel que $(\bar{x}, \bar{y}, \bar{s}, \bar{r})$ est une combinaison convexe des deux.

On peut donc conclure que tous les points extrêmes de conv(SPMCLS2) ont la forme des points décrits à la proposition 7. □

Afin de décrire les rayons extrêmes du problème SPMCLS2, nous définissons tout d'abord le problème SPMCLS2^0 défini par les contraintes (4.61)-(4.66).

$$x_i + \widetilde{s}_i + \widetilde{r}_i \geq 0, \ \forall i \tag{4.61}$$

$$\sum_{i=1}^{N} v_i x_i + \sum_{i=1}^{N} f_i y_i \leq 0 \tag{4.62}$$

$$x_i \leq \min\left\{\frac{c - f_i}{v_i}, \widetilde{d}_i\right\} y_{it}, \ \forall i \tag{4.63}$$

$$\widetilde{r}_i \leq 0, \ \forall i \tag{4.64}$$

$$x_i, \widetilde{s}_i, \widetilde{r}_i \geq 0, \ \forall i \tag{4.65}$$

$$y_i \in \{0, 1\}, \ \forall i \tag{4.66}$$

Nous rappelons les définitions suivantes :

Défnition 3. *$(\hat{x}, \hat{y}, \hat{s}, \hat{r})$ est un rayon de conv(SPMCLS2) si $(\hat{x}, \hat{y}, \hat{s}, \hat{r}) \in SPMCLS2^0 \setminus \{0\}$.*

Défnition 4. *Un rayon $(\hat{x}, \hat{y}, \hat{s}, \hat{r})$ de conv(SPMCLS2) est dit rayon extrême s'il n'existe pas des rayons $(\hat{x}^1, \hat{y}^1, \hat{s}^1, \hat{r}^1), (\hat{x}^2, \hat{y}^2, \hat{s}^2, \hat{r}^2) \in SPMCLS2^0 \setminus \{0\}$, $(\hat{x}^1, \hat{y}^1, \hat{s}^1, \hat{r}^1) \neq \eta(\hat{x}^2, \hat{y}^2, \hat{s}^2, \hat{r}^2)$ avec $\eta \in \mathbb{R}_+$, tel que : $(\hat{x}, \hat{y}, \hat{s}, \hat{r}) = \frac{1}{2}(\hat{x}^1, \hat{y}^1, \hat{s}^1, \hat{r}^1) + \frac{1}{2}(\hat{x}^2, \hat{y}^2, \hat{s}^2, \hat{r}^2)$.*

Proposition 8. *Tous les rayons extrêmes $(\hat{x}, \hat{y}, \hat{s}, \hat{r})$ de conv(SPMCLS2) sont de la forme $\hat{x}_i = \hat{y}_i = 0, \hat{s}_i = 1, \hat{r}_i = 0$ pour un seul $i \in \mathcal{I}$ et $\hat{x}_j = \hat{y}_j = \hat{s}_j = \hat{r}_j = 0$, $j \neq i$.*

Preuve. A partir de (4.62), (4.65) et (4.66) on a : $\hat{x}_i = \hat{y}_i = 0$ pour tout $i \in \mathcal{I}$. A partir de (4.65) et (4.64) on a : $\hat{r}_i = 0$.

Les rayons extrêmes de conv(SPMCLS2) sont de la forme : $(0, 0, \hat{s}, 0)$.

Tous les rayons extrêmes de conv(SPMCLS2) sont donc de la forme $\hat{x}_i = \hat{y}_i = 0$, $\hat{s}_i = 1$ pour un seul $i \in \mathcal{I}$ et $\hat{x}_j = \hat{y}_j = \hat{s}_j = \hat{r}_j = 0$, $j \neq i$.

□

4.6.2 Propriétés des inégalités couvrantes

Nous allons montrer dans ce qui suit que les inégalités (4.37) définissent des facettes du problème SPMCLS2 sous certaines conditions.

Le problème SPMCLS2 est de pleine-dimension si $f_i < c$, $\forall i \in \mathcal{I}$, c'est à dire que dim(SPMCLS2)= $4N$. En effet, Les N rayons extrêmes $\hat{s}_i = 1$ sont linéairement indépendant. Il n'est pas très difficile d'exhiber $3N+1$ points de conv(SPMCLS2) linéairement indépendants entre eux et des N rayons extrêmes.

Proposition 9. *Etant donnés une couverture S de SPMCLS2 et un ordre des références $i \in S$ tel que $f_{[1]}+v_{[1]}\widetilde{d}_{[1]} \geq \cdots \geq f_{[|S|]}+v_{[|S|]}\widetilde{d}_{[|S|]}$. Soit $T = \mathcal{I} \setminus S$. On définit $\mu_1 = f_{[1]}+v_{[1]}\widetilde{d}_{[1]}-\lambda_S$. Si $|S| \geq 2$, $\lambda_S > 0$, $f_{[2]}+v_{[2]}\widetilde{d}_{[2]} \geq \lambda_S$, $f_i < \mu_1$, $i \in T$ et $f_i < c$, $\forall i \in \mathcal{I}$ alors l'inégalité (4.37) induit une facette de conv(SPMCLS2).*

Preuve. Il suffit alors de montrer que l'intersection de l'hyperplan défini par conv(SPMCLS2) et (4.37) est de dimension $4N-1$ soit dim(SPMCLS2)-1. Noter que les $|T|$ rayons avec $s_i = 1$, $i \in T$, sont des rayons extrêmes de conv(SPMCLS2) et sont également dans l'hyperplan défini par (4.37). Il suffit donc de trouver $4N-|T|$ points réalisables linéairement indépendants dans cet hyperplan, et qui soient également linéairement indépendants des $|T|$ rayons extrêmes $s_i = 1$. Supposons tout d'abord que $v_i\widetilde{d}_i + v_{[1]}\widetilde{d}_{[1]} \geq \lambda_S$ pour $i \in S \setminus [1]$. Considérons les $4N - |T|$ points suivants,

- Pour tout $i' \in S \setminus [1]$, ($|S|$ - 1 points)
  $x_{i'} = (\frac{v_{i'}\widetilde{d}_{i'}-\lambda_S}{v_{i'}})^+, y_{i'} = 1, s_{i'} = \widetilde{d}_{i'} - x_{i'}, r_{i'} = 0$
  $x_{[1]} = \widetilde{d}_{[1]} - (\frac{\lambda_S-v_{i'}\widetilde{d}_{i'}}{v_{[1]}})^+, y_{[1]} = 1, s_{[1]} = \widetilde{d}_{[1]} - x_{[1]}, r_{[1]} = 0$
  $x_i = \widetilde{d}_i, y_i = 1, s_i = 0, r_i = 0,\ i \in S \setminus \{i' \cup [1]\}$
  $x_i = y_i = 0, s_i = \widetilde{d}_i, r_i = 0,\ i \notin S$
- Pour [1], (1 point)
  $x_{[1]} = (\frac{v_{[1]}\widetilde{d}_{[1]}-\lambda_S}{v_{[1]}})^+, y_{[1]} = 1, s_{[1]} = \widetilde{d}_{[1]} - x_{[1]}, r_{[1]} = 0$
  $x_{[2]} = \widetilde{d}_{[2]} - (\frac{\lambda_S-v_{[1]}\widetilde{d}_{[1]}}{v_{[2]}})^+, y_{[2]} = 1, s_{[2]} = \widetilde{d}_{[2]} - x_{[2]}, r_{[2]} = 0$
  $x_i = \widetilde{d}_i, y_i = 1, s_i = 0, r_i = 0,\ i \in S \setminus \{[2] \cup [1]\}$
  $x_i = y_i = 0, s_i = \widetilde{d}_i, r_i = 0,\ i \notin S$
- Pour tout $i' \in S \setminus [1]$, ($|S|$ - 1 points)
  $x_{i'} = y_{i'} = 0, s_{i'} = \widetilde{d}_{i'}, r_{i'} = 0$
  $x_{[1]} = \widetilde{d}_{[1]} - (\frac{\lambda_S-f_{i'}-v_{i'}\widetilde{d}_{i'}}{v_{[1]}})^+, y_{[1]} = 1, s_{[1]} = \widetilde{d}_{[1]} - x_{[1]}, r_{[1]} = 0$
  $x_i = \widetilde{d}_i, y_i = 1, s_i = 0, r_i = 0,\ i \in S \setminus \{i' \cup [1]\}$
  $x_i = y_i = 0, s_i = \widetilde{d}_i, r_i = 0,\ i \notin S$
- Pour [1], (1 point)
  $x_{[1]} = y_{[1]} = 0, s_{[1]} = \widetilde{d}_{[1]}, r_{[1]} = 0$
  $x_i = \widetilde{d}_i, y_i = 1, s_i = 0, r_i = 0,\ i \in S \setminus \{[1]\}$
  $x_i = y_i = 0, s_i = \widetilde{d}_i, r_i = 0,\ i \notin S$
- Pour tout $i' \in S \setminus [1]$, ($|S|$ - 1 points)
  $x_{i'} = \widetilde{d}_{i'} + (\frac{v_{[1]}\widetilde{d}_{[1]}+f_{[1]}-\lambda_S}{v_{i'}})^+, y_{i'} = 1, s_{i'} = 0, r_{i'} = 0$
  $x_{[1]} = y_{[1]} = 0, s_{[1]} = \widetilde{d}_{[1]}, r_{[1]} = 0$
  $x_i = \widetilde{d}_i, y_i = 1, s_i = 0, r_i = 0,\ i \in S \setminus \{i' \cup [1]\}$
  $x_i = y_i = 0, s_i = \widetilde{d}_i, r_i = 0,\ i \notin S$
- Pour [1], (1 point)
  $x_{[1]} = \widetilde{d}_{[1]} + (\frac{v_{[2]}\widetilde{d}_{[2]}+f_{[2]}-\lambda_S}{v_{[1]}})^+, y_{[1]} = 1, s_{[1]} = 0, r_{[1]} = 0$

$x_{[2]} = y_{[2]} = 0, s_{[2]} = \widetilde{d}_{[2]}, r_{[2]} = 0$
$x_i = \widetilde{d}_i, y_i = 1, s_i = 0, r_i = 0,\ i \in S \setminus \{[2] \cup [1]\}$
$x_i = y_i = 0, s_i = \widetilde{d}_i, r_i = 0,\ i \notin S$

– Pour tout $i' \in S \setminus [1]$, ($|S|$ - 1 points)
$x_{i'} = \widetilde{d}_{i'} + (\frac{v_{[1]}\widetilde{d}_{[1]} + f_{[1]} - \lambda_S}{v_{i'}})^+, y_{i'} = 1, s_{i'} = 0, r_{i'} = 0$
$x_{[1]} = y_{[1]} = 0, s_{[1]} = 0, r_{[1]} = \widetilde{d}_{[1]}$
$x_i = \widetilde{d}_i, y_i = 1, s_i = 0, r_i = 0,\ i \in S \setminus \{i' \cup [1]\}$
$x_i = y_i = 0, s_i = \widetilde{d}_i, r_i = 0,\ i \notin S$

– Pour tout [1], (1 point)
$x_{[1]} = \widetilde{d}_{[1]} + (\frac{v_{[2]}\widetilde{d}_{[2]} + f_{[2]} - \lambda_S}{v_{[1]}})^+, y_{[1]} = 1, s_{[1]} = 0, r_{[1]} = 0$
$x_{[2]} = y_{[2]} = 0, s_{[2]} = 0, r_{[2]} = \widetilde{d}_{[2]}$
$x_i = \widetilde{d}_i, y_i = 1, s_i = 0, r_i = 0,\ i \in S \setminus \{[2] \cup [1]\}$
$x_i = y_i = 0, s_i = \widetilde{d}_i, r_i = 0,\ i \notin S$

– Pour tout $i' \in T$, ($|S|$ - 1 points)
$x_{i'} = 0, y_{i'} = 1, s_{i'} = \widetilde{d}_{i'}, r_{i'} = 0$
$x_{[1]} = y_{[1]} = 0, s_{[1]} = \widetilde{d}_{[1]}, r_{[1]} = 0$
$x_i = \widetilde{d}_i, y_i = 1, s_i = 0, r_i = 0,\ i \in S \setminus \{[1]\}$
$x_i = y_i = 0, s_i = \widetilde{d}_i, r_i = 0,\ i \notin S \cup \{i'\}$
– Pour tout $i' \in T$, ($|S|$ - 1 points)
$x_{i'} = (\frac{v_{[1]}\widetilde{d}_{[1]} + f_{[1]} - \lambda_S - f_{i'}}{v_{i'}})^+, y_{i'} = 1, s_{i'} = \widetilde{d}_{i'} - x_{i'}, r_{i'} = 0$
$x_{[1]} = y_{[1]} = 0, s_{[1]} = \widetilde{d}_{[1]}, r_{[1]} = 0$
$x_i = \widetilde{d}_i, y_i = 1, s_i = 0, r_i = 0,\ i \in S \setminus \{[1]\}$
$x_i = y_i = 0, s_i = \widetilde{d}_i, r_i = 0,\ i \notin S \cup \{i'\}$
– Pour tout $i' \in T$, ($|S|$ - 1 points)
$x_{i'} = (\frac{v_{[1]}\widetilde{d}_{[1]} + f_{[1]} - \lambda_S - f_{i'}}{v_{i'}})^+, y_{i'} = 1, s_{i'} = \widetilde{d}_{i'} - x_{i'}, r_{i'} = 0$
$x_{[1]} = y_{[1]} = 0, s_{[1]} = 0, r_{[1]} = \widetilde{d}_{[1]}$
$x_i = \widetilde{d}_i, y_i = 1, s_i = 0, r_i = 0,\ i \in S \setminus \{[1]\}$
$x_i = y_i = 0, s_i = \widetilde{d}_i, r_i = 0,\ i \notin S \cup \{i'\}$

Nous avons exhibé $4|S| + 3|T| = 4N - |T|$ points linéairement indépendants. On peut facilement vérifier que ces $4N - |T|$ points et les $|T|$ rayons extrêmes $s_i = 1$, $i \in T$, sont linéairement indépendants. Quand la condition $v_i\widetilde{d}_i + v_{[1]}\widetilde{d}_{[1]} \geq \lambda_S$ pour $i \in S \setminus [1]$ n'est pas vérifiée, une approche similaire peut être faite afin d'exhiber $4N - |T|$ points linéairement indépendants. On peut donc conclure que l'inégalité (4.37) induit une facette de conv(SPMCLS2).

□

De la même manière, nous pouvons montrer que les inégalités (4.52) définissent des facettes sous les mêmes conditions.

4.7 Lifting des inégalités couvrantes et des inégalités couvrantes inverses

Dans cette section, nous allons renforcer les inégalités valides couvrantes et couvrantes inverses en utilisant des fonctions superadditives pour une amélioration séquentielle. Le lifting séquentiel permet d'utiliser les variables n'appartenant pas à S et des fonctions de lifting superadditives afin de renforcer les inégalités valides définies précédemment. Les fonctions de lifting superadditives permettent de renforcer les inégalités valides en traitant tout un ensemble de variables en une seule étape. Pour plus de détails sur les fonctions superadditives le lecteur peut se reporter à [75] et [190].

Nous allons nous baser sur les résultats de Marchand et Wolsey [123] pour le problème de sac à dos continu, ainsi que les adaptations effectuées par Miller *et al.* [132] pour le problème SPMCLSP, afin d'améliorer les inégalités couvrantes et les inégalités couvrantes inverses pour le problème SPMCLS2.

4.7.1 Problème de sac-à-dos avec une variable continue

Définissons tout d'abord le problème suivant :

$$Y = \left\{ (y,s) \in \{0,1\}^n \times R^1_+ : \sum_{j \in \mathcal{J}} a_j y_j \leq b + s. \right\} \tag{4.67}$$

avec $\mathcal{J} = \{1,\ldots,n\}, a_j \in \mathbb{Z}_+, j \in \mathcal{J}$ et $b \in \mathbb{Z}_+$.
Soit $(\{j'\}, C, D)$ une paire couvrante de Y tel que :
- $C \cap D = \{j'\}, C \cup D = \mathcal{J}$;
- $\lambda_C = \sum_{j \in C} a_j - b > 0$;
- $a_{j'} > \lambda_C$.

Nous pouvons noter que $\mu_D = a_{j'} - \lambda_C = \sum_{j \in D} a_j - \left(\sum_{j \in \mathcal{J}} a_j - b \right) > 0$. C est donc une couverture et D une couverture inverse.

Nous allons rappeler maintenant les principaux résultats sur les inégalités couvrantes et les inégalités couvrantes inverses définies pour ce problème.

Proposition 10. *(Inégalités couvrantes continues, Marchand et Wolsey [123])*
Soit $(\{j'\}, C, D)$ une paire couvrante pour Y. On considère un ordre des éléments de C tel que $a_{[1]} \geq \cdots \geq a_{[r_C]}$ où r_C est le nombre d'éléments de C avec $a_j > \lambda_C$. Posons $A_0 = 0$ et $A_j = \sum_{p=1}^{j} a_{[p]}, j = 1,\ldots,r_C$. On considère :

$$\phi_C(u) = \begin{cases} (j-1)\lambda_C, & si\ A_{j-1} \leq u \leq A_j - \lambda_C, j = 1,\ldots,r_C \\ (j-1)\lambda_C + [u - (A_j - \lambda_C)], & si\ A_{j-1} - \lambda_C \leq u \leq A_j, j = 1,\ldots,r_C \\ (r_C - 1)\lambda_C + [u - (A_{r_C} - \lambda_C)], & si\ A_{r_C} - \lambda_C \leq u \end{cases} \tag{4.68}$$

L'inégalité

$$\sum_{j \in C} \min(\lambda_C, a_j)\, y_j + \sum_{D \setminus j'} \phi_C(a_j)\, y_j \leq \sum_{j \in C \setminus j'} \min(\lambda_C, a_j) + s \tag{4.69}$$

est valide pour Y et définit une facette de conv(Y).

Une représentation simple de ϕ_C est donnée par la figure 4.2.

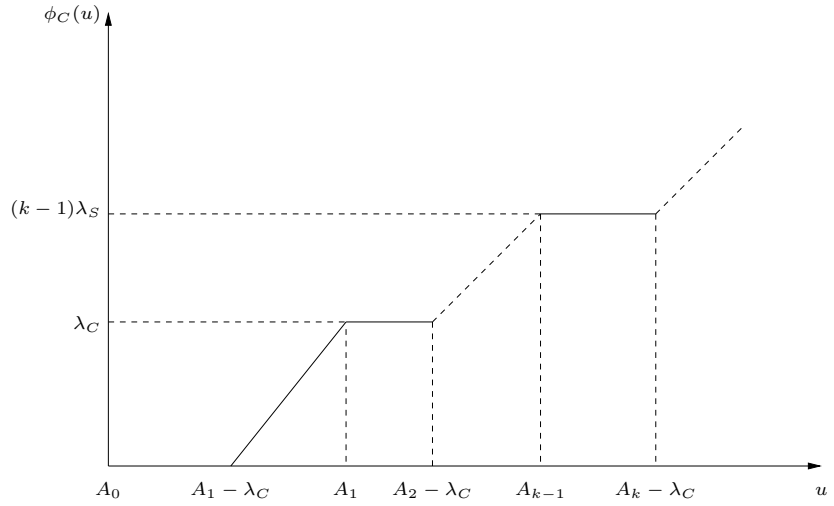


FIG. 4.2 – Fonction ϕ_C.

Proposition 11. *(Inégalités couvrantes inverses continues, Marchand et Wolsey [123])*
Soit $(\{j'\}, C, D)$ une paire couvrante de Y. On considère un ordre des éléments de D tel que $a_{[1]} \geq \cdots \geq a_{[r_D]}$ où r_D est le nombre d'éléments dans D avec $a_j > \mu_D$, où $\mu_D = a_{j'} - \lambda_C$. Soit $A_0 = 0$ et $A_j = \sum_{p=1}^{j} a_{[p]}, j = 1, \ldots, r_D$. On pose :

$$\psi_D(u) = \begin{cases} u - j\mu_D, & si\ A_j \leq u \leq A_{j+1} - \mu_D, j = 0, \ldots, r_D - 1 \\ A_j - j\mu_D, & si\ A_j - \mu_D \leq u \leq A_j, j = 1, \ldots, r_D - 1 \\ A_{r_D} - r_D\mu_D, & si\ A_{r_D} - \mu_D \leq u \end{cases} \tag{4.70}$$

L'inégalité

$$\sum_{j \in D} (a_j - \mu_D) y_j + \sum_{j \in C \setminus j'} \psi_D(a_j) y_j \leq \sum_{j \in C \setminus j'} \psi_D(a_j) + s \tag{4.71}$$

est valide pour Y et définit une facette pour conv(Y).

Une représentation simple de ψ_C est donnée par la figure 4.3.

4.7.2 Lifting des inégalités valides couvrantes du problème SPMCLS2

Dans ce qui suit, nous utilisons les résultats de Marchand et Wolsey [123] pour obtenir des inégalités valides plus fortes que (4.37). Rappelons que :

- S est une couverture pour le problème SPMCLS2 ;
- $T = \mathcal{I} \setminus S$;
- T', T'' est une partition de T ;

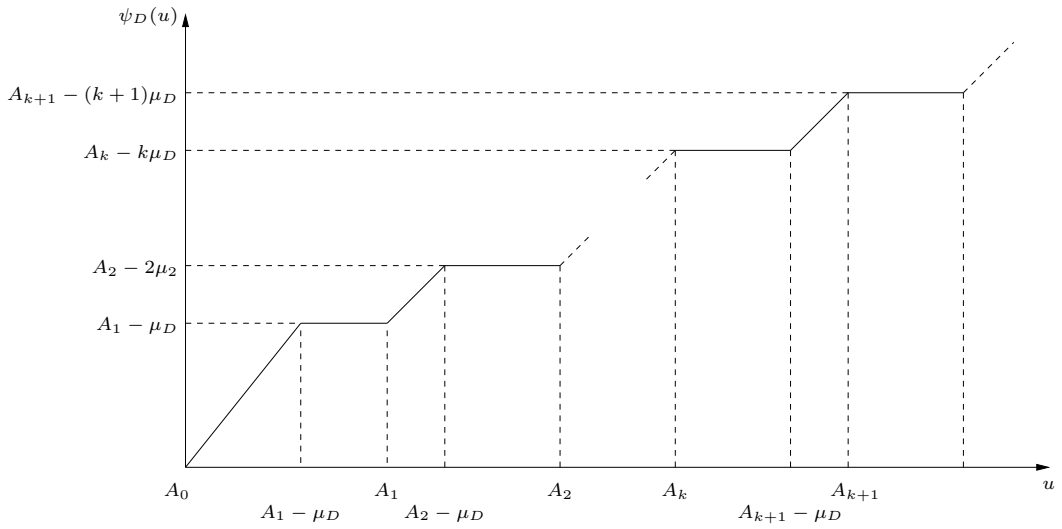


FIG. 4.3 – Fonction ψ_D.

– $U \subset T''$.

A partir des contraintes (4.3) et (4.28), on peut écrire :

$$\sum_{i \in S \cup U} \left(f_i y_i + v_i \widetilde{d}_i - v_i \widetilde{s}_i - v_i \widetilde{r}_i \right) \leq c$$

En ajoutant $\sum_{i \in S \cup U} v_i \widetilde{d}_i y_i$ aux membres de l'inéquation on aura :

$$\sum_{i \in S \cup U} v_i \widetilde{d}_i y_i + \sum_{i \in S \cup U} \left(f_i y_i + v_i \widetilde{d}_i - v_i \widetilde{s}_i - v_i \widetilde{r}_i \right) \leq c + \sum_{i \in S \cup U} v_i \widetilde{d}_i y_i$$

d'où :

$$\sum_{i \in S \cup U} \left(f_i + v_i \widetilde{d}_i \right) y_i \leq c + \sum_{i \in S \cup U} \left(v_i \widetilde{s}_i + v_i \widetilde{r}_i + v_i \widetilde{d}_i y_i - v_i \widetilde{d}_i \right) \tag{4.72}$$

si on pose $u = \sum_{i \in S \cup U} \left(v_i \widetilde{s}_i + v_i \widetilde{r}_i + v_i \widetilde{d}_i y_i - v_i \widetilde{d}_i \right)$, on aura :

$$\sum_{i \in S \cup U} \left(f_i + v_i \widetilde{d}_i \right) y_i \leq c + u \tag{4.73}$$

L'inéquation (4.73) peut être donc considérée comme une contrainte d'un problème de sac-à-dos continu. Ainsi, nous obtenons les propositions suivantes.

Proposition 12. *Etant donnés S une couverture de SPMCLS2 et U un sous-ensemble de $\mathcal{I} \setminus S$. L'inégalité (où ϕ_S est définie par (4.68))*

$$\sum_{i \in S \cup U} v_i \left(\widetilde{s}_i + \widetilde{r}_i \right) \geq \lambda_S + \sum_{i \in S} \max \left\{ -f_i, v_i \widetilde{d}_i - \lambda_S \right\} (1 - y_i) + \sum_{i \in U} \left(\phi_S \left(f_i + v_i \widetilde{d}_i \right) \right) y_i + \sum_{i \in U} v_i \widetilde{d}_i \left(1 - y_i \right) \tag{4.74}$$

est valide pour SPMCLS2.

Preuve. Soit $(\{j'\}, U \cup \{j'\}, S)$ une paire couvrante de SPMCLS2 tel que : $f_j + v_j\widetilde{d}_j > \lambda_S$. D'après la proposition 10, l'inégalité suivante est valide pour SPMCLS2.

$$\begin{aligned}\sum_{i\in S}\min\left(\lambda_S, f_i + v_i\widetilde{d}_i\right)y_i + \sum_{i\in U}\phi_S\left(f_i + v_i\widetilde{d}_i\right)y_i \quad &\leq \quad \sum_{i\in S\setminus\{j\}}\min\left(\lambda_S, f_i + v_i\widetilde{d}_i\right)\\ &\quad + \sum_{i\in S\cup U}\left(v_i\widetilde{s}_i + v_i\widetilde{r}_i + v_i\widetilde{d}_i y_i - v_i\widetilde{d}_i\right)\end{aligned}$$

Puisque $f_j + v_j\widetilde{d}_j > \lambda_S$, on a $\min(f_j + v_j\widetilde{d}_j, \lambda_S) = \lambda_S$. En ajoutant $\min(f_j + v_j\widetilde{d}_j, \lambda_S) - \lambda_S$ au membre droit de l'inégalité précédente, on obtient :

$$\begin{aligned}\sum_{i\in S}\min\left(\lambda_S, f_i + v_i\widetilde{d}_i\right)y_i + \sum_{i\in U}\phi_S\left(f_i + v_i\widetilde{d}_i\right)y_i \quad &\leq \quad \sum_{i\in S}\min\left(\lambda_S, f_i + v_i\widetilde{d}_i\right) - \lambda_S\\ &\quad + \sum_{i\in S\cup U}\left(v_i\widetilde{s}_i + v_i\widetilde{r}_i + v_i\widetilde{d}_i y_i - v_i\widetilde{d}_i\right)\end{aligned}$$

qui est valide pour SPMCLS2. On obtient l'inégalité (4.74) après simplifications.

□

Proposition 13. *Soit S une couverture de SPMCLS2. On considère un ordre $[1], \ldots, [|S|]$ tel que $f_{[1]} + v_{[1]}\widetilde{d}_{[1]} \geq \cdots \geq f_{[|S|]} + v_{[|S|]}\widetilde{d}_{[|S|]}$. Soit (U, T) une partition quelconque de $\mathcal{I}\setminus S$ et (T', T'') une partition quelconque de T. On définit $\mu_1 = f_{[1]} + v_{[1]}\widetilde{d}_{[1]} - \lambda_S$.*
Si $|S| \geq 2$ et $f_{[2]} + v_{[2]}\widetilde{d}_{[2]} \geq \lambda_S$, l'inégalité

$$\begin{aligned}\sum_{i\in S\cup U} v_i\left(\widetilde{s}_i + \widetilde{r}_i\right) \quad &\geq \quad \lambda_S + \sum_{i\in S}\max\left\{-f_i, v_i\widetilde{d}_i - \lambda_S\right\}(1 - y_i) + \sum_{i\in U}\phi_S\left(f_i + v_i\widetilde{d}_i\right)y_i\\ &\quad + \sum_{i\in U} v_i\widetilde{d}_i\,(1 - y_i) + \frac{\lambda_S}{f_{[2]} + v_{[2]}\widetilde{d}_{[2]}}\sum_{i\in T'}\left(v_i x_i - (\mu_1 - f_i)\,y_i\right) \qquad (4.75)\end{aligned}$$

est valide pour SPMCLS2.

Preuve. La preuve est similaire à celle de la proposition 5. □

4.7.3 Lifting des inégalités valides couvrantes inverses du problème SPMCLS2

De la même manière que précédemment, nous pouvons noter que l'inégalité (4.72) peut être considérée comme une contrainte d'un problème de sac à dos continu. Ainsi, nous obtenons les propostions suivantes.

Proposition 14. *Soit S une couverture inverse de SPMCLS2 et U un sous-ensemble de $\mathcal{I}\setminus S$. L'inégalité*

$$\sum_{i\in S\cup U} v_i\left(\widetilde{s}_i+\widetilde{r}_i\right) \geq \sum_{i\in S}\left(v_i\widetilde{d}_i-\psi_U\left(f_i+v_i\widetilde{d}_i\right)\right)(1-y_i)+\sum_{i\in U} v_i\widetilde{d}_i+\sum_{i\in U}\left(f_i-\mu_S\right)y_i \tag{4.76}$$

est valide pour SPMCLS2.

Preuve. D'après la proposition 11, l'inégalité

$$\sum_{i\in S}\psi_U\left(f_i+v_i\widetilde{d}_i\right)y_i+\sum_{i\in U}\left(f_i+v_i\widetilde{d}_i-\mu_S\right)y_i \leq \sum_{i\in S}\psi_U\left(f_i+v_i\widetilde{d}_i\right)+\sum_{i\in S\cup U} v_i\left(\widetilde{s}_i+\widetilde{r}_i+\widetilde{d}_iy_i-\widetilde{d}_i\right)$$

est valide pour SPMCLS2. On obtient l'inégalité (4.76) après simplifications.

□

Proposition 15. *Etant donnés S une couverture inverse de SPMCLS2 et $U \subseteq \mathcal{I}\setminus S$ tel que $f_i+v_i\widetilde{d}_i \geq \mu_S\ \forall i\in U$. On considère un ordre $[1],\ldots,[|U|]$ tel que $f_{[1]}+v_{[1]}\widetilde{d}_{[1]} \geq \cdots \geq f_{[|U|]}+v_{[|U|]}\widetilde{d}_{[|U|]}$. Soit (U,T) une partition de quelconque $\mathcal{I}\setminus S$ et (T',T'') une partition quelconque de T. L'inégalité (ψ_U est définie par (4.70))*

$$\begin{aligned}\sum_{i\in S\cup U} v_i\left(\widetilde{s}_i+\widetilde{r}_i\right) \;&\geq\; \sum_{i\in S}\left(v_i\widetilde{d}_i-\psi_U\left(f_i+v_i\widetilde{d}_i\right)\right)(1-y_i)+\sum_{i\in U}\left(f_i-\mu_S\right)y_i+\sum_{i\in U} v_i\widetilde{d}_i\\ &+\; \min_{i\in S}\left\{\frac{\psi_U\left(f_i+v_i\widetilde{d}_i\right)}{f_i+v_i\widetilde{d}_i}\right\}\sum_{i\in T'}\left(v_ix_i-\left(\mu_S-f_i\right)y_i\right)\end{aligned} \tag{4.77}$$

est valide pour SPMCLS2.

Preuve. La démonstration de cette proposition est similaire à celle de la proposition 4. □

4.8 Heuristique de séparation pour les inégalités couvrantes

Dans cette section, nous présentons une heuristique de séparation pour la création d'inégalités couvrantes pour le problème SPMCLS2, qui sont également valides pour le problème MCLS2. En effet, pour le problème précédent, on génère des inégalités couvrantes pour toute période de l'horizon de planification qui correspond à des inégalités couvrantes du problème SPMCLS2.

Afin de créer une inégalité couvrante pour le problème SPMCLS2, la première étape est de définir un ensemble couvrant S afin de définir λ_S et μ_1 (cf. (4.36) et proposition 5). La seconde étape est de parcourir tous les éléments $i\in\mathcal{I}\setminus S$ pour créer les ensembles U et T'.

Nous utilisons une méthode gloutonne pour la création de l'ensemble S. Pour ce faire, nous ordonnons les éléments $i\in\mathcal{I}$ selon un ordre décroissant par rapport à la valeur :

$$\max\left\{-f_i, v_i\widetilde{d}_i-\lambda_S\right\}(1-y_i^*)-v_i\left(\widetilde{s}_i^*+\widetilde{r}_i^*\right) \tag{4.78}$$

La formule (4.78) est obtenue à partir de l'inégalité (4.75) en ne considérant que les termes en relation avec S. La valeur de (4.78) représente la contribution de la violation de l'inégalité (4.75) par la référence $i \in S$ et dépend de λ_S. Cependant, λ_S n'est pas connue d'avance. Donc, nous estimons la valeur de λ_S. Pour ce faire, nous allons trier les références $i \in \mathcal{I}$ selon un ordre décroissant par rapport à leurs consommations en ressources en utilisant la formule (4.79). La formule (4.78) donnerait un meilleur ensemble S mais ne peut être utilisée puisque λ_S n'est pas connue, alors en pratique la formule (4.79) est utilisée.

$$\left(f_i + v_i \widetilde{d}_i\right) y_i^* \tag{4.79}$$

On remarque que la formule (4.79) représente la consommation en ressources de la référence i si la demande $\widetilde{d}_i$ est produite.

L'ensemble couvrant S est créé de manière gloutonne. On rajoute les éléments ordonnés selon la formule (4.79) jusqu'à l'obtention d'un ensemble couvrant.

Pour la création de l'ensemble U (respectivement T'), on parcourt les éléments $i \in \mathcal{I} \setminus S$ et on teste si la valeur de la somme des membres de l'inégalité (4.75) appartenant à U (respectivement à T') est positive. Si c'est le cas, on rajoute les éléments i à U (respectivement à T').

A partir des ensemble S, U et T', on crée une inégalité valide en se basant sur l'inégalité (4.75). Si la valeur de l'inégalité est positive on obtient une coupe.

Le principe précédemment décrit est traduit par l'algorithme 2.

On rappelle que l'évaluation de la fonction superadditive ϕ_S est en $O(N)$ pour chaque référence. L'heuristique de séparation pour la génération d'inégalités couvrantes pour le problème SPMCLS2 est en $O(N^2)$. Puisque chaque période est examinée séparément dans la création des inégalités valides pour le problème MCLS2, alors l'heuristique de séparation est en $O(N^2T)$ pour le problème précédent.

4.9 Heuristique de séparation pour les inégalités couvrantes inverses

L'idée de l'heuristique de séparation des inégalités couvrantes inverses est similaire à la précédente. La première étape consiste à créer un ensemble couvrant inverse S afin de définir μ_S (voir (4.51)). La seconde étape consiste à examiner tous les éléments $i \in \mathcal{I} \setminus S$ afin de créer les ensembles U et T'.

Nous utilisons une méthode gloutonne pour la création de l'ensemble S. Pour ce faire, nous ordonnons les éléments $i \in \mathcal{I}$ selon un ordre décroissant par rapport à la valeur :

$$\left(v_i \widetilde{d}_i - \psi_U \left(f_i + v_i \widetilde{d}_i\right)\right)(1 - y_i^*) - v_i\,(\widetilde{s}_i^* + \widetilde{r}_i^*) \tag{4.80}$$

De la même façon, la formule (4.80) est obtenue à partir de l'inégalité (4.77) en ne considérant que les termes en relation avec S. La valeur de (4.80) représente la contribution de la violation de l'inégalité (4.77) par la référence $i \in S$ et dépend de μ_S. Or, μ_S n'est

Algorithme 2 Heuristique de séparation des inégalités couvrantes

Ordonner les éléments de $\mathcal{I}$ dans l'ordre décroissant selon la formule (4.79).
$S \leftarrow \emptyset, U \leftarrow \emptyset, T' \leftarrow \emptyset, i' \leftarrow 0$
$i' \leftarrow \arg\min_{i=1,\ldots,N} \left\{ \sum_{k=1}^{i} \left(f_{[k]} + v_{[k]} \widetilde{d}_{[k]} \right) > c \right\}$
$S \leftarrow \{[1], \ldots, [i']\}$
$i_1 \leftarrow \arg\max_{i \in S} \left\{ f_i + v_i \widetilde{d}_i \right\}$
$i_2 \leftarrow \arg\max_{i \in S \setminus \{i_1\}} \left\{ f_i + v_i \widetilde{d}_i \right\}$
$\lambda_S \leftarrow \left(\sum_{k=1}^{i'} f_{[k]} + v_{[k]} \widetilde{d}_{[k]} \right) - c$
si $(|S| \geq 2)$ et $(f_{i_2} + v_{i_2} \widetilde{d}_{i_2} \geq \lambda_S)$ **alors**
    $\mu_1 \leftarrow \left\{ f_{i_1} + v_{i_1} \widetilde{d}_{i_1} \right\} - \lambda_S$
    $i \leftarrow i' + 1$
    tant que $(i \leq N)$ **faire**
        si $(\phi_S \left(f_{[i]} + v_{[i]} \widetilde{d}_{[i]} \right) y^*_{[i]} + v_{[i]} \widetilde{d}_{[i]} \left(1 - y^*_{[i]} \right) - v_{[i]} \left(\widetilde{s}^*_{[i]} + \widetilde{r}^*_{[i]} \right) > 0)$ **alors**
            $U \leftarrow U \cup \{[i]\}$
        sinon si $(v_{[i]} x^*_{[i]} > \left(\mu_1 - f_{[i]} \right) y^*_{[i]})$ **alors**
            $T' \leftarrow T' \cup \{[i]\}$
        finsi
        $i \leftarrow i + 1$
    fin tant que
finsi
si (L'inégalité (4.75) basée sur S, U et T' est violée) **alors**
    Ajouter l'inégalité (4.75) au nœud courant.
finsi

pas connue d'avance. Alors, on estime la valeur de μ_S. Pour ce faire, nous ordonnons les référence $i \in \mathcal{I}$ selon l'ordre décroissant par rapport à leur consommation en ressources en utilisant la formule (4.79). La formule (4.80) donnerait de meilleurs ensembles S mais ne peut être utilisée puisque μ_S n'est pas connue, alors en pratique la formule (4.79) est utilisée. Le principe de cette heuristique est illustré par l'algorithme 3.

On rappelle que l'évaluation de la fonction superadditive ψ_S est en $O(N)$ pour chaque référence. L'heuristique de séparation pour la génération d'inégalités couvrantes inverses pour le problème SPMCLS2 est en $O(N^2)$. Puisque chaque période est examinée séparément dans la création des inégalités valides pour le problème MCLS2, alors l'heuristique de séparation est en $O(N^2T)$ pour le problème précédent.

4.10 Implémentation et analyse des résultats

Dans cette section, nous présentons des résultats expérimentaux issus de l'utilisation des inégalités valides présentées précédemment. Nous rapportons des résultats dans le cadre de l'utilisation d'une méthode de cut-and-branch et de branch-and-cut.

La méthode cut-and-branch consiste à n'ajouter des coupes qu'au nœud racine de l'arbre de branch-and-bound dans le but d'améliorer la borne inférieure. La méthode branch-and-cut consiste à ajouter des coupes non seulement au premier nœud mais également à d'autres nœuds de l'arbre de recherche. Généralement, les coupes ne sont rajoutées qu'à un sous-

Algorithme 3 Heuristique de séparation des inégalités couvrantes inverses

Ordonner les éléments de $\mathcal{I}$ dans l'ordre décroissant selon la formule (4.79).
$S \leftarrow \emptyset$, $i' \leftarrow 1$
tant que $(i' \leq N)$ **faire**
  $S \leftarrow S \cup \{i'\}$
  $\mu_S = c - \sum_{j=1}^{i'} f_{[j]} + v_{[j]}\widetilde{d}_{[j]}$
  si $(\mu_S < 0)$ **alors**
    $T' \leftarrow \emptyset$, $U \leftarrow \emptyset$, $i \leftarrow i' + 1$
    tant que $(i \leq N)$ **faire**
      si $\left(\left(f_{[i]} - \mu_S\right) y^*_{[i]} + v_{[i]}\widetilde{d}_{[i]} - v_{[i]}\left(\widetilde{s}^*_{[i]} + \widetilde{r}^*_{[i]}\right) > 0\right)$ **alors**
        $U \leftarrow U \cup \{[i]\}$
      sinon si $\left(v_{[i]}x^*_{[i]} - \left(\mu_S - f_{[i]}\right) y^*_{[i]} > 0\right)$ **alors**
        $T' \leftarrow T' \cup \{[i]\}$
      finsi
      $i \leftarrow i + 1$
    fin tant que
    si (L'inégalité (4.77) basée sur S, U et T' est violée) **alors**
      Ajouter l'inégalité (4.77) au nœud courant.
      $i' \leftarrow N + 1$
    sinon
      $i' \leftarrow i' + 1$
    finsi
  sinon
    $i' \leftarrow N + 1$
  finsi
fin tant que

ensemble de nœuds de l'arbre de branch-and-bound afin de ne pas ralentir le temps CPU total en résolvant le problème.

Nos algorithmes sont implémentés en langage C++ et ils sont intégrés à un logiciel d'APS, n.SKEP. Nous utilisons la librairie de programmation linéaire mixte CPLEX 9.0 [110] pour la résolution des problèmes linéaires mixtes. CPLEX 9.0 fournit des fonctions de callback qui permettent à l'utilisateur d'implémenter son propre algorithme de branch-and-cut. Tous les tests ont été effectués sur un ordinateur personnel équipé d'un microprocesseur Pentium IV 2.66 GHz.

4.10.1 Instances de test

Nous avons effectué des tests sur une série d'instances étendues de la librairie de lot-sizing LOTSIZELIB [114], initialement décrites dans Trigeiro *et al.* [172]. Les instances de Trigeiro *et al.* sont notées Tr_{N-T}, où N est le nombre de références et T le nombre de périodes. Elles sont caractérisées par une consommation variable en ressources égale à 1, et des capacités suffisantes pour satisfaire la totalité de la demande sur tout l'horizon de planification. Elles sont également caractérisées par des coûts de setup importants, des petits besoins fixes en ressources (temps de setup) et pas de σ_{it}. On rappelle que σ_{it} représente la dernière période à laquelle peut être consommée la production de la référence i à la période t. Les caractéristiques

des instances de Trigeiro *et al.* [172] sont décrites dans le tableau 4.1.

Instance	N	T
tr_{6-15}	6	15
tr_{6-30}	6	30
tr_{12-15}	12	15
tr_{12-30}	12	30
tr_{24-15}	24	15
tr_{24-30}	24	30

TAB. 4.1 – Instances de Trigeiro *et al.* [172].

Puisque les instances de Trigeiro *et al.* sont caractérisées par des capacités suffisantes pour satisfaire toute la demande, nous avons effectué quelques modifications afin d'induire des ruptures sur la demande. Nous avons dérivé 24 nouvelles instances à partir des instances Tr_{N-T} en augmentant le besoin fixe en ressources (temps de setup), le besoin variable en ressources et en rajoutant des σ_{it}. Nous avons également généré des coûts de rupture sur la demande. Nous donnons plus de détails dans la suite.

Les nouvelles instances sont synthétisées en 4 classes de 6 instances chacune.
- Classe 1 : La première classe est obtenue en augmentant le besoin variable en ressources et en rajoutant des σ_{it}. Le besoin variable en ressources est multiplié par un coefficient $1 + \rho$ tel que $0 \leq \rho \leq 0.001 \times c_t$, c_t représente la capacité en ressources disponible à la période t. Les σ_{it} sont générés de telle sorte qu'on ne puisse pas anticiper la production pour plus de $\frac{1}{3}T$ périodes, T représente le nombre de périodes.
- Classe 2 : La seconde classe est obtenue en effectuant les mêmes modifications sur les besoins variables en ressources que la première classe. Les σ_{it} sont générés de telle sorte qu'on ne puisse pas anticiper la production pour plus de $\frac{2}{3}T$ périodes.
- Classe 3 : La troisième classe est basée sur la première classe. En effet, nous avons apporté des modifications sur les besoins fixes en ressources (temps de setup). Ils sont augmentés en les multipliant par un coefficient $1 + \tau$ tel que $\tau \approx 0.1 \times c_t$.
- Classe 4 : La dernière classe est obtenue en apportant les mêmes modifications sur les besoins fixes et besoins variables en ressources que la troisième classe. Les σ_{it} sont générés de telle sorte qu'on ne puisse pas anticiper la production pour plus de $\frac{2}{3}T$ périodes.

Les coûts de rupture sur la demande sont considérés comme des pénalités. Donc, φ_{it} sont fixés de façon à ce que $\varphi_{it} >> \max_{i',t'} \{\alpha_{i't'}; \beta_{i't'}; \gamma_{i't'}\}$. Les coûts φ_{it} ont la particularité de décroître dans le temps. En effet, les demandes au début de l'horizon correspondent à des ordres de fabrication réels et non à des prévisions, tandis que les demandes à la fin de l'horizon sont généralement des prévisions et sont donc moins pénalisées. Ces coûts sont générés de la même façon pour toutes les instances décrites précédemment.

4.10.2 Interprétation des résultats

Nous avons effectué une comparaison entre les deux méthodes suivantes :
- Un algorithme basé sur le branch-and-cut standard du solveur CPLEX que l'on note BC ;

– Un algorithme basé sur le branch-and-cut standard du solveur CPLEX incluant toutes les coupes présentées précédemment notée BC+.

Pour les deux algorithmes BC et BC+, nous avons utilisé le modèle agrégé défini dans la section 4.1 par l'ensemble de contraintes (4.1)-(4.7).

Deux types de coupes générale pour les problèmes linéaires mixtes sont activés pour toutes les méthodes. Il s'agit des coupes flow cover et des coupes Mixed Integer Rounding (MIR) du solveur CPLEX 9.0. Les coupes flow cover fonctionnent bien pour les problèmes de lot-sizing en raison de la structure de flots induite par les contraintes de flux (4.2). Pour une description des coupes flow cover, le lecteur peut se référer à Gomory [70], Nemhauser et Wolsey [138] et Wolsey [194]. Les coupes MIR sont principalement appliquées aux contraintes de capacité et aux contraintes de production. Pour plus de détails sur les inégalités MIR, le lecteur peut se référer à Padberg *et al.* [156] et Van Roy et Wolsey [179].

Pour tous les algorithmes, LB et UB représentent respectivement la borne inférieure et la borne supérieure à la fin de l'algorithme. NB_{Nodes} est le nombre de nœuds explorés dans l'arbre de branch-and-bound. U_{Cuts} est le nombre de coupes spécialisées ajoutées dans l'algorithme de branch-and-cut (inégalités couvrantes, inégalités couvrantes inverses et inégalités (l, S)). F_{Cuts} et MIR_{Cuts} représentent respectivement le nombre de coupes flow cover et MIR ajoutées par le solveur durant l'algorithme de branch-and-cut. La comparaison des algorithmes est basée sur les deux critères suivants : le premier appelé GAP est égal à $(UB - LB)/UB$, et le second est le temps CPU noté $Time$.

Au nœud racine de la méthode $BC+$, nous utilisons les algorithmes 1, 2 et 3 (voir les sections 4.4, 4.8 et 4.9) jusqu'à ce qu'il n'y ait plus aucune inégalité violée. La même procédure est poursuivie dans l'arbre de recherche.

La stratégie de branchement dans la méthode de branch-and-bound est une recherche *en profondeur d'abord* afin de trouver une solution réalisable. Une borne supérieure est obtenue quand la solution est entière ou par l'heuristique de programmation linéaire du solveur.

Le tableau 4.2 récapitule les résultats expérimentaux basés sur un critère d'arrêt de temps. Un temps d'exécution maximum de 600 secondes est alloué pour tous les algorithmes.

D'après le tableau 4.2, on peut facilement remarquer qu'utiliser les inégalités valides présentées dans ce chapitre améliore les performances de l'algorithme de branch-and-cut. Il est évident, que $BC+$ résout les instances de test plus efficacement que BC. Les inégalités valides qu'on a proposé sont intéressantes puisque toutes les bornes fournies par $BC+$ sont meilleures que celles fournies par BC.

On peut également noter que les bornes supérieures obtenues par $BC+$ sont meilleures que celles obtenues par BC. En effet, généralement une bonne borne inférieure implique de meilleures bornes supérieures. Une raison est qu'une meilleure borne inférieure peut être utilisée pour élaguer les nœuds dominés de l'arbre de recherche et permet à l'algorithme de branch-and-bound de visiter des branches plus intéressantes afin de trouver de meilleures solutions. Une autre raison est que quand les relaxations linéaires continues sont plus fortes, il est plus facile de trouver des solutions entières, parce que la solution linéaire continue est souvent entière ou bien parce que les heuristiques basées sur les programmes linéaires continus du solveur sont plus efficaces.

De plus, le nombre de nœuds explorés par la méthode BC est beaucoup plus élevé que celui exploré par $BC+$. Le rapport entre ces deux nombres varie entre 150% et 500% pour

N	T	Méthode	UB	LB	NB_{Nodes}	U_{Cuts}	MIR_{Cuts}	F_{Cuts}	GAP
Classe 1									
6	15	BC	4 038 212	3 979 268	123 900	0	309	254	1,46%
6	15	BC+	4 030 456	3 999 352	70 000	397	87	160	0,77%
6	30	BC	4 536 980	4 124 370	34 000	0	625	523	9,09%
6	30	BC+	4 492 288	4 271 466	10 300	877	100	200	4,92%
12	15	BC	7 669 660	7 495 023	62 600	0	616	464	2,28%
12	15	BC+	7 651 166	7 610 635	19 600	517	257	300	0,53%
12	30	BC	8 772 505	7 903 312	25 600	0	607	717	9,91%
12	30	BC+	8 609 716	8 345 637	1 500	2 063	64	364	3,07%
24	15	BC	14 117 000	13 780 000	73 600	0	370	600	2,39%
24	15	BC+	14 082 000	13 995 000	10 300	796	83	445	0,62%
24	30	BC	23 619 000	22 879 000	11 800	0	837	1 298	3,13%
24	30	BC+	23 558 000	23 180 000	300	4 048	134	807	1,60%
Classe 2									
6	15	BC	4 032 790	3 978 178	120 900	0	304	253	1,35%
6	15	BC+	4 034 967	3 987 919	47 400	248	157	2 001	1,17%
6	30	BC	4 530 576	4 120 298	34 700	0	637	499	9,06%
6	30	BC+	4 520 148	4 269 236	6 800	667	217	274	5,55%
12	15	BC	7 664 197	7 477 993	57 600	0	546	532	2,43%
12	15	BC+	7 666 557	7 528 738	14 500	439	201	393	1,80%
12	30	BC	8 793 271	7 899 450	22 800	0	630	693	10,16%
12	30	BC+	8 675 000	8 250 948	1 900	1 506	240	456	4,89%
24	15	BC	14 118 000	13 776 000	69 600	0	315	669	2,43%
24	15	BC+	14 112 000	13 850 000	6 800	731	133	619	1,86%
24	30	BC	23 634 000	22 875 000	13 200	0	742	1 335	3,21%
24	30	BC+	23 647 000	23 002 000	300	2 798	388	1 120	2,73%
Classe 3									
6	15	BC	5 300 603	5 167 025	74 000	0	347	216	2,52%
6	15	BC+	5 268 911	5 228 874	53 000	375	205	129	0,76%
6	30	BC	7 210 627	6 581 190	34 400	0	664	484	8,73%
6	30	BC+	7 138 995	6 798 534	14 000	539	372	251	4,77%
12	15	BC	12 345 000	11 632 000	5 700	0	692	358	5,77%
12	15	BC+	12 178 000	11 945 000	5 900	1 239	179	191	1,91%
12	30	BC	15 728 000	12 710 000	14 500	0	1 014	456	19,19%
12	30	BC+	15 641 000	14 081 000	3 700	2 160	74	153	9,97%
24	15	BC	26 330 000	24 822 000	10 700	0	850	445	5,73%
24	15	BC+	26 038 000	25 613 000	3 500	603	437	408	1,63%
24	30	BC	43 446 000	36 863 000	3 600	0	941	993	15,15%
24	30	BC+	43 476 000	39 202 000	10	3 559	55	292	9,83%
Classe 4									
6	15	BC	5 295 547	5 170 187	79 200	0	324	230	2,37%
6	15	BC+	5 312 724	5 204 658	48 710	25 386	190	160	2,03%
6	30	BC	7 202 452	6 581 498	32 800	0	674	462	8,62%
6	30	BC+	7 059 530	6 736 455	11 600	14 323	414	277	4,58%
12	15	BC	12 256 000	11 222 000	21 400	0	775	304	8,43%
12	15	BC+	12 348 667	11 724 138	6 221	8 332	269	255	5,06%
12	30	BC	15 697 000	12 520 000	18 000	0	932	492	20,24%
12	30	BC+	15 588 917	14 002 210	2 293	9 234	463	420	10,18%
24	15	BC	26 463 000	23 996 000	14 500	0	798	466	9,32%
24	15	BC+	26 452 668	25 207 181	3 180	6 268	427	380	4,71%
24	30	BC	43 364 000	35 593 000	4 600	0	959	926	17,92%
24	30	BC+	42 619 192	38 745 933	160	3 313	460	604	9,09%

TAB. 4.2 – Résultats expérimentaux : critère d'arrêt temps CPU

les instances de petite taille et entre 300% et 36000% pour les instances de plus grande taille. En effet, le fait de générer des coupes prend du temps et avoir plusieurs coupes ralentit les résolutions continues à chaque nœud. Ainsi, en activant les coupes que nous avons développées, le solveur génère un nombre de coupes générales inférieur à celui obtenu en désactivant nos coupes spécialisées. Puisque les coupes spécialisées sont générées avant les coupes générales du solveur, l'observation précédente peut être expliquée par le fait que les coupes spécialisées dominent une partie des coupes générales du solveur.

En visualisant le tableau 4.2, on remarque que les instances de la troisième et de la quatrième classe sont beaucoup plus difficiles que celles de la première et de la seconde classe. En effet, les problèmes de la troisième et de la quatrième classe sont caractérisés par un besoin fixe en ressources plus élevé que celui des problèmes de la première et de la seconde classe.

On peut également remarquer que les problèmes avec un σ_{it} faible ont un plus grand GAP que ceux avec un plus grand σ_{it}. En effet, en utilisant les algorithmes BC et $BC+$, les instances de classe 1 et 2 ont atteint un GAP plus petit que celui des instances de classe 2 et 4.

L'analyse générale des résultats expérimentaux montre qu'ajouter des inégalités valides au premier nœud améliore considérablement la borne inférieure. La moyenne d'amélioration de la borne inférieure au premier nœud de $BC+$ est de 80% pour la première classe, 53% pour la seconde classe, 73% pour la troisième classe et 48% pour la dernière classe. Ce rapport est le pourcentage obtenu entre la meilleure borne inférieure observée au premier nœud de BC et la meilleure trouvée à la fin de la méthode $BC+$.

Le tableau 4.3 montre le pourcentage des coupes générées par $BC+$ pour toutes les instances. $\%lS_{cuts}$, $\%C_{cuts}$ et $\%RC_{cuts}$ représentent respectivement le pourcentage des inégalités (l, S), des inégalités couvrantes et des inégalités couvrantes inverses générées par $BC+$. Un temps CPU limite de 600 secondes est alloué.

N	T	$\%lS_{cuts}$	$\%C_{cuts}$	$\%RC_{cuts}$	N	T	$\%lS_{cuts}$	$\%C_{cuts}$	$\%RC_{cuts}$
Class 1					Class 3				
12	15	4,08%	95,84%	0,08%	12	15	21,48%	77,20%	1,32%
12	30	14,62%	85,32%	0,05%	12	30	33,93%	64,89%	1,17%
24	15	13,16%	86,80%	0,04%	24	15	51,19%	34,24%	14,57%
24	30	43,96%	55,93%	0,11%	24	30	71,64%	21,44%	6,92%
6	15	18,78%	81,16%	0,06%	6	15	71,00%	27,43%	1,56%
6	30	71,16%	28,80%	0,04%	6	30	87,49%	10,18%	2,34%
Class 2					Class 4				
12	15	2,58%	97,35%	0,06%	12	15	9,30%	90,18%	0,52%
12	30	6,60%	93,30%	0,11%	12	30	15,69%	83,11%	1,20%
24	15	22,97%	77,03%	0,00%	24	15	45,11%	45,11%	9,78%
24	30	27,31%	72,68%	0,01%	24	30	59,13%	38,36%	2,51%
6	15	31,41%	68,56%	0,04%	6	15	69,78%	27,98%	2,23%
6	30	60,46%	39,49%	0,05%	6	30	80,05%	17,05%	2,90%

TAB. 4.3 – Résultats expérimentaux : pourcentage de coupes générées.

D'après le tableau 4.3, on peut noter que le pourcentage des coupes couvrantes inverses générées par $BC+$ est très petit. On peut également noter que $BC+$ a généré plus de coupes (l, S) que de coupes couvrantes pour les classes 1 et 2, excepté pour les instances à 24 références et 30 périodes. Pour les classes 3 et 4, $BC+$ a généré plus de coupes couvrantes que de coupes (l, S), excepté pour les instances à 6 références.

Nous avons également testé $BC+$ en utilisant chaque famille d'inégalités valides séparément (inégalités couvrantes, inégalités couvrantes inverses et inégalités (l, S)). Le tableau 4.4 résume les résultats expérimentaux sur les classes d'instances 1 et 3. GAP_{lS}, GAP_C et GAP_{RC} représentent respectivement, le GAP quand seules les inégalités (l, S), les inégalités couvrantes et les inégalités couvrantes inverses sont respectivement utilisées.

N	T	GAP_{lS}	GAP_C	GAP_{CR}
Class 1				
12	15	0,56%	2,45%	2,40%
12	30	4,93%	11,03%	10,02%
24	15	0,55%	2,50%	2,54%
24	30	1,87%	3,37%	3,24%
6	15	0,68%	1,40%	1,39%
6	30	5,33%	9,44%	9,10%
Class 3				
12	15	1,48%	3,16%	4,45%
12	30	11,17%	14,71%	19,40%
24	15	2,43%	4,28%	5,32%
24	30	10,65%	12,40%	14,63%
6	15	1,19%	2,20%	2,60%
6	30	4,88%	7,27%	7,71%

TAB. 4.4 – Résultats expérimentaux par famille de coupes.

Ces tests montrent que la famille des inégalités (l, S) est la plus efficace. La famille des inégalités couvrantes est moins efficace que la famille des inégalités (l, S). Utiliser la famille des inégalités couvrantes inverses toute seule n'est réellement pas efficace.

Afin de donner une comparaison appropriée concernant le nombre de coupes ajoutées et le nombre de nœuds explorés pour les deux algorithmes BC et $BC+$, nous avons résolu quelques problèmes jusqu'à atteindre un GAP minimum. Puisque nous n'avons pu résoudre aucun problème à l'optimum en un temps CPU raisonnable, nous avons utilisé un critère de temps CPU limite de 1800 secondes pour BC. Nous avons utilisé le GAP obtenu à la fin de BC comme un critère d'arrêt pour $BC+$. Nous avons également utilisé un temps CPU limite de 1800 secondes pour $BC+$. Le tableau 4.5 résume les résultats expérimentaux sur la classe d'instance 1 en utilisant le GAP comme critère d'arrêt.

A partir du tableau 4.5, on peut facilement remarquer que pour atteindre le même GAP, $BC+$ n'a pas besoin d'autant de nœuds et de temps CPU que la méthode BC. On peut noter également que sans brancher, $BC+$ atteint un GAP moins élevé que celui de BC pour

N	T	Méthode	NB_{Nodes}	U_{Cuts}	MIR_{Cuts}	F_{Cuts}	GAP	$Time$
Class 1								
6	15	BC	420 821	0	309	254	1,03%	1800
6	15	BC+	9 810	517	14	183	0,94%	53
6	30	BC	105 911	0	625	523	8,51%	1800
6	30	BC+	110	1924	64	364	7,65%	13
12	15	BC	174 601	0	617	464	2,18%	1800
12	15	BC+	40	783	33	377	2,05%	2
12	30	BC	75 836	0	747	808	9,60%	1800
12	30	BC+	0	4048	134	807	8,66%	35
24	15	BC	206 400	0	433	638	2,14%	1800
24	15	BC+	0	326	88	158	0,88%	4
24	30	BC	42 796	0	880	1 417	3,00%	1800
24	30	BC+	10	848	81	184	2,00%	175

TAB. 4.5 – Résultats expérimentaux : critère d'arrêt GAP.

l'instance avec 12 références et 30 périodes et pour celle avec 30 références et 24 périodes. Les coupes améliorent considérablement le GAP au nœud racine.

Plusieurs auteurs ([26], [150]) ont montré à partir des résultats expérimentaux qu'utiliser le modèle basé sur le problème de localisation d'entrepôts fournit une meilleure borne inférieure basée sur la relaxation linéaire continue, que celle obtenue par le modèle agrégé. Nous avons effectué des expérimentations en utilisant la formulation basée sur le problème de localisation d'entrepôts défini par l'ensemble de contraintes (4.10)-(4.18) (voir page 4.1).

On note BC_{FL} la méthode branch-and-cut utilisant cette formulation. Quelques résultats préliminaires corroborant l'observation précédente sont présentés dans le tableau 4.6. Un temps CPU maximum de 600 secondes est alloué à BC_{FL}.

A partir du tableau 4.6, on peut facilement noter qu'utiliser la formulation basée sur le problème de localisation d'entrepôts améliore considérablement les performances de la méthode BC. On peut également noter que les bornes supérieures obtenues par BC_{FL} sont meilleures que celles obtenues par BC et $BC+$. Les bornes inférieures de BC_{FL} et BC sont presque équivalentes.

On peut aussi noter que le nombre de coupes de type flow cover générées par BC_{FL} est inférieure au nombre de coupes de type flow cover générées par BC et $BC+$, mais le nombre de coupes MIR est plus important pour BC_{FL} que BC et $BC+$. Cette observation peut être expliquée par le fait que l'on perd la structure de flots dans BC_{FL} quand on utilise la formulation basée sur le problème de localisation d'entrepôts.

De plus, le nombre de nœuds explorés par la méthode BC est beaucoup plus grand que celui exploré par la méthode BC_{FL}. En effet, la formulation basée sur le problème de localisation d'entrepôts nécessite plus de variables et de contraintes que le modèle agrégé. Cette nouvelle formulation ralentit la résolution continue à chaque nœud de l'arbre de recherche. Des résultats expérimentaux nous ont montré que pour résoudre la relaxation linéaire continue de la formulation basée sur le problème de localisation d'entrepôts, nous avons besoin de deux fois plus de temps CPU que la relaxation linéaire continue de la formulation agrégée.

N	T	Méthode	UB	LB	NB_{Nodes}	MIR_{Cuts}	F_{Cuts}	GAP
Class 1								
6	15	BC_{FL}	4 026 868,52	4 010 676,67	38 608	573	62	0,40%
6	30	BC_{FL}	4 445 157,32	4 310 759,64	6 358	1 042	82	3,02%
12	15	BC_{FL}	7 646 515,50	7 616 239,97	11 702	1 145	84	0,40%
12	30	BC_{FL}	8 596 922,81	8 396 173,93	10 067	474	53	2,34%
24	15	BC_{FL}	14 066 432,40	14 029 609,17	30 155	126	127	0,26%
24	30	BC_{FL}	23 448 512,18	23 376 275,83	6 597	118	89	0,31%
Class 2								
6	15	BC_{FL}	4 029 454,12	4 008 242,33	40 480	572	56	0,53%
6	30	BC_{FL}	4 405 696,08	4 311 196,67	6 061	1 050	83	2,14%
12	15	BC_{FL}	7 643 808,26	7 617 456,19	11 110	1 139	84	0,34%
12	30	BC_{FL}	8 592 762,06	8 395 989,30	9 369	500	35	2,29%
24	15	BC_{FL}	14 050 681,24	14 030 093,27	24 885	157	129	0,15%
24	30	BC_{FL}	23 460 743,23	23 377 590,34	7 724	99	82	0,35%
Class 3								
6	15	BC_{FL}	5 283 361,02	5 221 642,21	15 626	495	52	1,17%
6	30	BC_{FL}	7 167 284,68	6 747 385,75	2 157	771	161	5,86%
12	15	BC_{FL}	12 217 281,93	11 884 973,27	2 091	845	122	2,72%
12	30	BC_{FL}	15 733 789,29	14 274 164,24	612	785	115	9,28%
24	15	BC_{FL}	26 418 930,71	25 529 298,16	3 043	586	171	3,37%
24	30	BC_{FL}	42 184 825,28	39 478 865,06	340	271	149	6,42%
Class 4								
6	15	BC_{FL}	5 269 002,47	5 226 125,26	16 421	493	53	0,81%
6	30	BC_{FL}	7 204 214,01	6 756 064,96	2 459	736	116	6,22%
12	15	BC_{FL}	12 161 284,65	11 873 157,64	2 278	852	73	2,37%
12	30	BC_{FL}	15 791 339,21	14 289 324,04	657	708	64	9,51%
24	15	BC_{FL}	25 906 049,18	25 382 177,04	2 857	448	94	2,02%
24	30	BC_{FL}	42 785 525,04	39 476 381,10	394	208	107	7,73%

TAB. 4.6 – Résultats expérimentaux utilisant la formulation basée sur le problème de localisation d'entrepôts.

Puisque nous n'avons pu résoudre aucun problème à l'optimum en un temps CPU raisonnable, nous avons apporté quelques modifications à trois instances de la classe 3 afin de créer des instances que l'on peut résoudre à l'optimum en temps CPU raisonnable. Cette modification est basée sur la réduction du nombre de périodes. Ces nouvelles instances sont définies par le nombre de références N et le nombre de périodes T. Le tableau 4.7 récapitule les résultats expérimentaux de la résolution à l'optimum des ces instances en utilisant les méthodes BC, $BC+$ et BC_{FL}. OPT représente la valeur de la fonction objectif.

N	T	Méthode	NB_{Nodes}	U_{Cuts}	MIR_{Cuts}	F_{Cuts}	OPT	$Time$
6	8	BC	6620	0	146	144	1637791	15,42
6	8	BC+	3256	341	14	41	1637791	14,16
6	8	BC_{FL}	446	0	294	21	1637791	7,07
12	5	BC	1958	0	140	194	2496916	5,66
12	5	BC+	1735	136	60	176	2496916	4,45
12	5	BC_{FL}	29	0	105	9	2496916	2,42
24	5	BC	10920	0	351	291	7898893	87,3
24	5	BC+	4473	200	226	239	7898893	37,95
24	5	BC_{FL}	701	0	196	43	7898893	11,79

TAB. 4.7 – Résultats expérimentaux : Résolution à l'optimum

A partir du tableau 4.7, nous pouvons noter que BC_{FL} trouve la solution optimale beaucoup plus rapidement que BC et BC+. BC_{FL} explore également beaucoup moins de nœuds que les deux autres méthodes. La méthode BC+ est deux fois plus rapide que BC pour la résolution de l'instance à 24 références. Pour les deux autres problèmes, BC+ est légèrement plus rapide que BC. On remarque également que la méthode BC+ explore beaucoup moins de nœuds que BC.

D'après les tableaux 4.6 et 4.7, on peut dire que BC_{FL} est une méthode prometteuse qui peut nous aider à améliorer l'algorithme de branch-and-cut pour résoudre des problèmes de planification de production. Il serait vraiment intéressant de généraliser les inégalités valides présentées dans ce chapitre à la formulation basée sur le problème de localisation d'entrepôts du problème MCLS2 et de les tester dans une procédure de branch-and-cut.

L'inconvénient du modèle désagrégé par rapport au modèle agrégé est le nombre de variables ainsi que la mémoire et le temps CPU requis pour résoudre les relaxations linéaires continues. Des résultats expérimentaux nous ont montré que pour résoudre la relaxation linéaire continue de la formulation désagrégée, nous avons besoin de beaucoup plus de temps CPU que la relaxation linéaire continue de la formulation agrégée. Nous allons montrer au chapitre suivant qu'en rajoutant la notion de stock de sécurité, le modèle agrégé donne de meilleurs résultats que le modèle désagrégé.

4.11 Généralisations des inégalités valides

Dans cette section, nous présentons une généralisation des inégalités (l, S), des inégalités couvrantes et des inégalités couvrantes inverses à des problématiques étendues du problème MCLS2.

4.11.1 Généralisation des résultats au problème MCLS2 multi-ressources

La production d'une référence peut nécessiter l'utilisation de plus d'une ressource par période. Pour plus de détails sur la modélisation du problème MCLS2 en multi-ressources, le lecteur peut se référer au chapitre 3 page 40.

La généralisation des résultats énoncés précédemment n'est pas en contradiction avec l'introduction de la dimension ressource. En effet, les inégalités couvrantes et inégalités couvrantes inverses ne portent que sur une seule ressource à la fois et les inégalités (l, s) ne dépendent pas des ressources. Les inégalités couvrantes ainsi que les inégalités couvrantes inverses seront générées pour chaque ressource séparément.

4.11.2 Généralisation des résultats au problème MCLS2 multi-groupes

Les références sont classées par groupes de références. Une référence peut appartenir à un ou plusieurs groupes et un groupe peut contenir une ou plusieurs références. Ceci se traduit par la prise en compte d'un besoin ou d'une consommation fixe des capacités disponibles dès qu'il y a lancement d'un *groupe*. Ceci entraîne également un coût de lancement pour le groupe. Ce problème peut être modélisé en rajoutant une dimension groupe j et une variable binaire z_{jt} pour la détection de lancements de groupes, J représente le nombre de groupes. Pour plus de détails se reporter au chapitre 3 page 39. En reprenant le problème MCLS2, il s'agit de modifier la fonction objectif, la contrainte de capacité et de rajouter une contrainte pour détecter les lancements de groupes.

On rappelle dans ce qui suit toutes les modifications apportées au problème MCLS2 afin de prendre en compte les groupes de références. On note MCLS2G le problème MCLS2 multi-groupes.

Les paramètres :

g_{jt} : Besoin fixe en ressources, du groupe j à la période t.

ω_{jt} : Coût fixe relatif au lancement du groupe j à la période t.

$$a_{ij} = \begin{cases} 1 \text{ si la référence } i \text{ appartient au groupe } j. \\ 0 \text{ sinon} \end{cases}$$

Les variables :

$$z_{jt} = \begin{cases} 1 \text{ si le groupe } j \text{ est lancé à la période } t. \\ 0 \text{ sinon} \end{cases}$$

La fonction objectif :

$$\min \sum_{i,t} \alpha_{it} x_{it} + \sum_{i,t} \beta_{it} y_{it} + \sum_{i,t} \varphi_{it} r_{it} + \sum_{i,t} \gamma_{it} s_{it} + \sum_{j,t} \omega_{jt} z_{jt} \tag{4.81}$$

Les contraintes :

La contrainte suivante remplace la contrainte (3.10) du problème MCLS2.

$$\sum_{i=1}^{N} v_{it}x_{it} + \sum_{i=1}^{N} f_{it}y_{it} + \sum_{j=1}^{J} g_{jt}z_{jt} \leq c_t, \ \forall t \tag{4.82}$$

Les deux contraintes suivantes modélisent le lancement de groupes :

$$z_{jt} \geq a_{ij}y_{it}, \quad \forall i, \forall j, \forall t \tag{4.83}$$
$$z_{jt} \in \{0,1\}, \quad \forall j, \forall t \tag{4.84}$$

Si les coûts $\omega_{jt} > 0$ pour tout j et t, la contrainte d'intégrité sur la variable z_{jt} peut être relaxée sans pour autant affecter la qualité des solutions trouvées. On note MCLS2G$_c$ le problème MCLS2G dans lequel on remplace la contrainte (4.84) par $0 \leq z_{gt} \leq 1$.

Proposition 16. *Toute solution optimale du problème MCLS2G$_c$ est une solution optimale du problème MCLS2G.*

Preuve. Soit $(\bar{x}', \bar{y}', \bar{z}', \bar{s}', \bar{r}')$ une solution optimale de MCLS2G$_c$.
- Si $\bar{y}'_{it} = 1$ alors d'après la contrainte (4.83) on aura : $\bar{z}'_{jt} = 1$ si $a_{ij} = 1$.
- Si $\bar{y}'_{it} = 0$ pour tout i tel que $a_{ij} = 1$, alors $\bar{z}'_{jt} = 0$. En effet, supposons que $\bar{y}'_{it} = 0$ pour tout i tel que $a_{ij} = 1$ et $\bar{z}'_{jt} > 0$ pour un j donné. Cette solution n'est pas optimale pour le problème MCLS2G$_c$ car si $\bar{z}'_{jt} = 0$ on aura une meilleure solution.

Donc, toute solution optimale de MCLS2G$_c$ est une solution optimale de MCLS2G. □

Afin de générer des inégalités couvrantes et couvrantes inverses pour le problème MCLS2G, nous procédons de la même manière que précédemment. On considère le problème SPM-CLS2G, la relaxation sur une seule période du problème MCLS2G. Pour alléger le modèle, nous allons supprimer l'indice temporel. L'idée est de construire un problème de sac à dos à partir des contraintes de capacité (4.82).

Etant donné $U \subset N \setminus \{S \cup T'\}$ (Tel que : S est une couverture pour le problème MCLS2G, $T = N \setminus S$ et (T', T'') une partition quelconque de T).

G_S est un sous-ensemble de groupes de G induit par le lancement des références de S.

A partir des contraintes (4.82) et (4.28) on peut écrire :

$$\sum_{i \in S \cup U} \left(f_i y_i + v_i \widetilde{d}_i - v_i \widetilde{s}_i - v_i \widetilde{r}_i \right) + \sum_{j \in G_S \cup G_U} (g_j z_j) \leq c$$

En rajoutant $\sum_{i \in S \cup U} v_i \widetilde{d}_i y_i$ aux membres de l'inéquation on aura :

$$\sum_{i \in S \cup U} v_i \widetilde{d}_i y_i + \sum_{i \in S \cup U} \left(f_i y_i + v_i \widetilde{d}_i - v_i \widetilde{s}_i - v_i \widetilde{r}_i \right) + \sum_{j \in G_S \cup G_U} (g_j z_j) \leq c + \sum_{i \in S \cup U} v_i \widetilde{d}_i y_i$$

d'où :

$$\sum_{i \in S \cup U} \left(f_i + v_i \widetilde{d}_i \right) y_i \leq c + \sum_{i \in S \cup U} \left(v_i \widetilde{s}_i + v_i \widetilde{r}_i + v_i \widetilde{d}_i y_i - v_i \widetilde{d}_i \right) - \sum_{j \in G_S \cup G_U} (g_j z_j)$$

si on pose :

$$u = \sum_{i \in S \cup U} \left(v_i \widetilde{s}_i + v_i \widetilde{r}_i + v_i \widetilde{d}_i y_i - v_i \widetilde{d}_i \right) - \sum_{j \in G_S \cup G_U} (g_j z_j)$$

on aura :

$$\sum_{i \in S \cup U} \left(f_i + v_i \widetilde{d}_i \right) y_i \leq c + u \tag{4.85}$$

L'inéquation (4.85) peut donc être considérée comme une contrainte d'un problème de sac-à-dos continu.

Proposition 17. *Étant donnée S une couverture de SPMCLS2G. On considère un ordre $[1], \ldots, [|S|]$ tel que : $f_{[1]} + v_{[1]} \widetilde{d}_{[1]} \geq \cdots \geq f_{[|S|]} + v_{[|S|]} \widetilde{d}_{[|S|]}$. Soit (T, U) une partition quelconque de $N \setminus S$ et (T', T'') une partition quelconque de T. Soit $\mu_1 = f_{[1]} + v_{[1]} \widetilde{d}_{[1]} - \lambda_S$.*
Si $|S| \geq 2$ et $f_{[2]} + v_{[2]} \widetilde{d}_{[2]} \geq \lambda_S$. L'inégalité

$$\begin{aligned} \sum_{i \in S \cup U} v_i \left(\widetilde{s}_i + \widetilde{r}_i \right) \quad \geq \quad & \lambda_S + \sum_{i \in S} \max \left\{ -f_i, v_i \widetilde{d}_i - \lambda_S \right\} (1 - y_i) + \sum_{i \in U} \left(\phi_S \left(f_i + v_i \widetilde{d}_i \right) \right) y_i \\ & + \sum_{i \in U} v_i \widetilde{d}_i \left(1 - y_i \right) + \sum_{j \in G_S \cup G_U} (g_j z_j) \end{aligned} \tag{4.86}$$

est valide pour SPMCLS2G.
Où ϕ_S est définie dans l'équation (4.68).

Preuve. La preuve est similaire à celle de la proposition 12. □

Proposition 18. *Étant donnés S une couverture inverse de SPMCLS2G et un ensemble $U \subset N \setminus S$ tel que : $f_i + v_i \widetilde{d}_i \geq \mu_S \ \forall i \in U$. On considère un ordre $[1], \ldots, [|U|]$ tel que : $f_{[1]} + v_{[1]} \widetilde{d}_{[1]} \geq \cdots \geq f_{[|U|]} + v_{[|U|]} \widetilde{d}_{[|U|]}$. Soit $T = N \setminus \{S \cup U\}$ et (T', T'') une partition quelconque de T.*
Alors, l'inégalité :

$$\begin{aligned} \sum_{i \in S \cup U} v_i \left(\widetilde{s}_i + \widetilde{r}_i \right) \quad \geq \quad & \sum_{i \in S} \left(v_i \widetilde{d}_i - \psi_U \left(f_i + v_i \widetilde{d}_i \right) \right) (1 - y_i) + \sum_{i \in U} v_i \widetilde{d}_i \\ & + \sum_{i \in U} \left(f_i - \mu_S \right) y_i + \sum_{j \in G_S \cup G_U} (g_j z_j) \end{aligned} \tag{4.87}$$

est valide pour SPMCLS2G.

Preuve. La preuve est similaire à celle de la proposition 14 □

4.11.3 Généralisation des résultats au problème MCLS2 multi-gammes

Dans la pratique, on est souvent amené à choisir entre plusieurs gammes de production. Une manière de modéliser cette situation est de considérer un produit pour chaque gamme

de production. Pour ce faire, on va considérer la nouvelle dimension *gamme* que l'on va noter par v. Ainsi, les produits équivalents auront la même dimension référence, mais une dimension gamme différente. Les produits sont considérés comme distincts durant la phase de production, mais à leur entrée en stock, les quantités de tous les produits équivalents sont additionnées afin de satisfaire la demande. Pour plus de détails sur la modélisation du problème MCLS2 en multi-gammes, nous invitons le lecteur à se reporter au chapitre 3 page 41.

Les modifications apportées aux paramètres et aux variables du problème MCLS2 portent essentiellement sur les paramètres et les variables de production.

Nous rappelons dans ce qui suit, les données additionnelles requises pour la modélisation du problème, ainsi que les modifications apportées aux variables et paramètres du problème MCLS2.

Les paramètres :
f_{vit} : Besoin fixe en ressources de la référence i, pour la gamme v à la période t.
v_{vit} : Besoin variable en ressources de la référence i, pour la gamme v, à la période t.
α_{vit} : Coût de production unitaire variable de la référence i, pour la gamme v, à la période t.
β_{vit} : Coût fixe relatif à un lancement de la référence i, pour la gamme v, à la période t.

Les variables :
x_{vit} : Quantité produite pour la référence i à la période t en utilisant la gamme v.

$$y_{vit} = \begin{cases} 1 \text{ s'il y a production de la référence } i \text{ à la période } t \\ \quad \text{en utilisant la gamme } v \text{ (i.e. si } x_{vit} > 0). \\ 0 \text{ sinon} \end{cases}$$

L'équation de conservation des flux à travers l'horizon est formulée en remplaçant la production du produit fini par la somme des productions relatives à chaque gamme. Ainsi, la variable x_{it} dans la contrainte de flux (4.2) du problème MCLS2 est remplacée par la l'expression $\sum_{v=1}^{V} x_{vit}$ (où V est le nombre de gammes).

En utilisant ces variables et paramètres, nous formulons le nouveau problème noté MCLS2V.

$$\min \sum_{v,i,t} \alpha_{vit} x_{vit} + \sum_{v,i,t} \beta_{vit} y_{vit} + \sum_{i,t} \varphi_{it} r_{it} + \sum_{i,t} \gamma it s_{it} \tag{4.88}$$

s.c :

$$s_{i(t-1)} + r_{it} + \textstyle\sum_{v=1}^{V} x_{vit} = d_{it} + s_{it}, \qquad \forall i, \forall t \tag{4.89}$$

$$\textstyle\sum_{v=1}^{V} \sum_{i=1}^{N} v_{vit} x_{vit} + \sum_{v=1}^{V} \sum_{i=1}^{N} f_{vit} y_{vit} \leq c_t, \qquad \forall t \tag{4.90}$$

$$x_{vit} \leq M y_{vit}, \qquad \forall v, \forall i, \forall t \tag{4.91}$$

$$r_{it} \leq d_{it}, \qquad \forall i, \forall t \tag{4.92}$$

$$r_{it}, s_{it} \geq 0, \qquad \forall i, \forall t \tag{4.93}$$

$$x_{vit} \geq 0, \qquad \forall v, \forall i, \forall t \tag{4.94}$$

$$y_{vit} \in \{0, 1\}, \qquad \forall v, \forall i, \forall t \tag{4.95}$$

les inégalités valides (l, S) du MCLS2V s'écrivent sous la forme suivante :

$$\sum_{v} x_{vit} + \left(\sum_{t'=t}^{\sigma_{it}} r_{it'}\right) + \left(s_{i(t-1)} + \sum_{t'=t+1}^{\sigma_{it}} \sum_{v} \delta_{t',\sigma_{it}} y_{vit'}\right) \geq \delta_{t,\sigma_{it}},\ \forall i, \forall t \tag{4.96}$$

L'expression $s_{i(t-1)} + \sum_{t'=t+1}^{\sigma_{it}} \sum_{v} \delta_{t',\sigma_{it}} y_{vit'}$ peut être notée $\widetilde{s}_{i,t-1}$. D'une manière similaire, on note $\sum_{t'=t}^{\sigma_{it}} r_{it'}$ par $\widetilde{r}_{it}$ et $\delta^{i}_{t,\sigma_{it}}$ par $\widetilde{d}_{it}$.

Notons SPMCLS2V la relaxation sur une seule période du problème MCLS2V. Puisque nous travaillons sur chaque période séparément dans le SPMCLS2V et étant donné que chaque période est traitée séparément, il peut être plus commode d'enlever l'indice temporel pour alléger les formules. Les inégalités (4.96) sont équivalentes à :

$$\sum_{v} x_{vi} + \widetilde{r}_i + \widetilde{s}_i \geq \widetilde{d}_i,\ \forall i \tag{4.97}$$

Nous construisons les inégalités couvrantes et les inégalités couvrantes inverses en se basant sur une couverture ou une couverture inverse. Cette couverture est un ensemble de références, il faut donc choisir une gamme par référence. Ce choix est un prétraitement. En effet, nous choisissons la gamme qui consomme le plus en produisant la totalité de la demande. Nous utilisons par exemple la formule suivante (la gamme choisie sera notée par l'indice v_i) :

$$v_i = \arg\min_{v \in V} \{(f_{vi} + v_{vi} d_i) y_{vi}\}$$

Afin de prendre en considération toutes les gammes, sans pour autant perdre la notion de couverture, nous choisissons des consommations variables et des consommations fixes identiques pour toutes les gammes.

Dans notre cas, nous utilisons le principe du pire cas, et nous choisissons les consommations minimales afin de garantir qu'une couverture avec de tels paramètres reste une couverture. On pose :

$$f_i^{min} = \min_{v \in V} f_{vi}$$

et

$$v_i^{min} = \min_{v \in V} v_{vi}$$

la couverture est définie par la formule suivante :

$$\lambda_S = \sum_{i \in S} \left(f_i^{min} + v_i^{min} d_i\right) - c \geq 0$$

– S est une couverture pour notre problème.
– $T = N \setminus S$.
– $U \subset T$.

A partir des contraintes de ressources (4.90), nous pouvons écrire :

$$\sum_i \sum_v (f_i^{min} y_{vi} + v_i^{min} x_{vi}) \leq c \tag{4.98}$$

A partir des contraintes (4.98) et de notre nouvelle inégalité (l,S) (4.97), on peut écrire :

$$\sum_{i \in S \cup U} \sum_{v \in V} (f_i^{min} y_{vi}) + \sum_{i \in S \cup U} \left(v_i^{min} d_i - v_i^{min} s_i - v_i^{min} r_i \right) \leq c$$

Donc :

$$\sum_{i \in S \cup U} f_i^{min} y_{v_i i} + \sum_{i \in S \cup U} \sum_{v \in V \setminus v_i} (f_i^{min} y_{vi}) + \sum_{i \in S \cup U} \left(v_i^{min} d_i - v_i^{min} s_i - v_i^{min} r_i \right) \leq c$$

On sait que :

$$\sum_{i \in S \cup U} \sum_{v \in V \setminus v_i} (f_i^{min} y_{vi}) > 0$$

D'où :

$$\sum_{i \in S \cup U} \left(f_i^{min} y_{v_i i} + v_i^{min} d_i - v_i^{min} s_i - v_i^{min} r_i \right) \leq c$$

En rajoutant $\sum_{i \in S \cup U} v_i^{min} d_i y_i$ aux membres de l'inéquation on obtient :

$$\sum_{i \in S \cup U} v_i^{min} d_i y_{v_i i} + \sum_{i \in S \cup U} \left(f_i^{min} y_{v_i i} + v_i^{min} d_i - v_i^{min} s_i - v_i^{min} r_i \right) \leq c + \sum_{i \in S \cup U} v_i^{min} d_i y_{v_i i}$$

d'où :

$$\sum_{i \in S \cup U} \left(f_i^{min} + v_i^{min} d_i \right) y_{v_i i} \leq c + \sum_{i \in S \cup U} v_i^{min} \left(d_i - s_i - r_i + d_i y_{v_i i} - d_i \right)$$

si on pose :

$$u = \sum_{i \in S \cup U} v_i^{min} \left(d_i - s_i - r_i + d_i y_{v_i i} - d_i \right)$$

on a :

$$\sum_{i \in S \cup U} \left(f_i^{min} + v_i^{min} d_i \right) y_{v_i i} \leq c + u \tag{4.99}$$

L'inéquation (4.99) peut être donc considérée comme une contrainte d'un problème de sac-à-dos continu.

On peut donc appliquer les résultats de Marchand et Wolsey [123] concerant le problème de sac-à-dos continu pour générer des inégaltés valides couvrantes et couvrantes inverses.

4.12 Conclusion

Nous avons proposé dans ce chapitre la modélisation d'un nouveau problème de lot-sizing à capacité finie avec des temps de setup et des coûts de rupture. Une approche polyédrique a donné lieu à des inégalités valides fortes. Les résultats expérimentaux montrent que l'utilisation de ces inégalités valides améliore d'une manière significative les algorithmes utilisés pour résoudre ce genre de problèmes. Nous avons également étudié la structure polyédrique de l'enveloppe convexe du problème traité, celle-ci nous a permis de montrer que les inégalités valides couvrantes induisent des facettes de l'enveloppe convexe des solutions réalisables sous certaines conditions. En utilisant la même approche, nous pouvons montrer que les inégalités couvrantes inverses définissent également des facettes sous des conditions similaires. Les inégalités valides présentées dans ce chapitre ont été généralisées pour tenir compte d'autres contraintes industrielles telles que, les groupes de références, le cas de plusieurs ressources en parallèle et celui de plusieurs gammes de production.

Les perspectives d'amélioration de ces résultats sont multiples. En effet, l'adaptation de ces inégalités valides pour la formulation utilisant le problème de localisation d'entrepôts est une voie prometteuse pour améliorer l'efficacité de l'approche.

Une autre perspective de nos travaux est de généraliser ces inégalités pour inclure les coûts de start-up. En effet, Van Hoesel *et al.* [177] ont généralisé les inégalités (l, S) en une nouvelle classe d'inégalités (l, R, S) afin de traiter les coûts de start-up pour les problèmes de lot-sizing à une seule référence et sans contrainte de capacité. Il serait intéressant de poursuivre ces travaux en généralisant les nouvelles inégalités proposées.

Il serait également intéressant d'utiliser cette approche conjointement avec une heuristique de décomposition de l'horizon de planification présentée au chapitre 6.

Nous avons proposé dans ce chapitre une méthode exacte pour le problème MCLS2. Dans le chapitre suivant, nous proposons une approche basée sur la relaxation lagrangienne afin de fournir une borne inférieure de bonne qualité au problème MCLS4. Nous proposons également un algorithme polynomial pour résoudre les sous-problèmes engendrés par la relaxation lagrangienne des contraintes de capacité.

Chapitre 5

Relaxation lagrangienne pour le problème MCLS4

Introduction

Depuis de longues années, les chercheurs se sont intéressés aux calculs de bornes inférieures pour les problèmes mathématiques d'optimisation difficiles (ex. programmes linéaires mixtes, programmation linéaire en nombres entiers). Les intérêts de tels calculs sont multiples. En effet, une bonne borne inférieure peut être utilisée :

- Dans des algorithmes de branch-and-bound. La borne inférieure est utilisée pour évaluer la qualité de la solution réalisable à chaque nœud de l'arbre de recherche. Elle permet également d'élaguer un nœud dès lors que sa borne inférieure est supérieure à la meilleure solution déjà obtenue. En général, les relaxations linéaires continues renforcées par des inégalités valides sont utilisées comme bornes inférieures dans les méthodes de branch-and-bound ;
- Pour évaluer la qualité d'une solution réalisable obtenue avec une heuristique (ex. heuristiques gloutonnes) ;
- Pour construire une solution réalisable. En effet, les bornes inférieures sont souvent issues d'heuristiques dont les solutions obtenues ne sont généralement pas réalisables. C'est le cas des méthodes basées sur la relaxation de contraintes telle que la relaxation lagrangienne.

Pour le calcul de bornes inférieures, on fait généralement appel à des méthodes dont la complexité est de préférence polynomiale. Les deux méthodes les plus utilisées sont la relaxation linéaire continue et la relaxation lagrangienne.

Dans ce chapitre, nous traitons le problème de lot-sizing à plusieurs références, avec des contraintes de capacité, des coûts et des temps de setup, des coûts de rupture sur la demande ainsi que des coûts de déficit sur le stock de sécurité, noté MCLS4[1]. Le but de ce chapitre est de présenter un algorithme qui fournit une borne inférieure au problème traité. Nous allons utiliser la relaxation lagrangienne afin de générer des bornes inférieures pour le problème MCLS4. Cette relaxation consiste à dualiser les contraintes de capacité. Nous proposons

[1]MCLS4 est une abréviation en anglais de : Multi-item Capacitated Lot-sizing problem with Setup times, Shortage costs and Safety Stock deficit costs.

également un algorithme polynomial pour résoudre les sous-problèmes engendrés par cette relaxation.

Dans la section 5.1, nous rappelons le modèle mathématique du problème traité ainsi que les différentes formulations. L'approche de résolution basée sur la relaxation lagrangienne est présentée dans la section 5.2. Dans la section 5.2.1, nous décrivons la relaxation des contraintes de capacité ainsi que les sous-problèmes engendrés par cette dernière. Dans la section 5.2.2, nous proposons un algorithme de programmation dynamique pour résoudre les sous-problèmes en un temps polynomial. La section 5.2.3 est consacrée à la description du schéma de la relaxation lagrangienne et à la résolution du problème dual. Enfin, des résultats expérimentaux sont décrits en section 5.3

5.1 Modélisation mathématique du problème MCLS4

Nous allons rappeler dans ce qui suit la modélisation mathématique du problème MCLS4. Il s'agit de déterminer un plan de production d'un ensemble de N références, pour un horizon de planification constitué de T périodes. Ce plan est à capacité finie et doit tenir compte d'un ensemble de contraintes additionnelles. En effet, le lancement de production d'une référence à une période donnée entraîne outre la consommation variable en ressources, une consommation dite fixe usuellement dénommée temps de setup. Les ressources disponibles pouvant ne pas être suffisantes pour satisfaire toute la demande, on autorise alors l'occurrence de *ruptures sur la demande*. Celles-ci sont modélisées par des variables continues avec une pénalité unitaire très élevée dans la fonction objectif, le but principal étant de satisfaire le client et donc d'avoir le minimum de ruptures sur la demande. Pour plus de détails sur la modélisation des ruptures sur la demande, nous invitons le lecteur à se reporter au chapitre 3 page 31.

Un autre objectif du décideur est d'avoir un profil de stocks supérieur à un seuil appelé *stock de sécurité*. Il peut arriver qu'on ne puisse pas atteindre ce seuil, dans ce cas on parle d'une situation de *déficit sur le stock de sécurité*. Ces déficits sont modélisés par des variables continues additionnées aux variables de stock avec une pénalité unitaire dans la fonction objectif et une contrainte supplémentaire assurant le fait que la somme du stock et du déficit sur le stock de sécurité soit supérieure au stock de sécurité. La pénalité unitaire des variables de déficit sur le stock de sécurité est inférieure à celle des variables de rupture sur la demande. En effet, l'objectif principal du décideur est de satisfaire la demande, la satisfaction du stock de sécurité vient en seconde position. La contrainte de stock de sécurité peut être reformulée afin de l'introduire dans la contrainte de conservation des flux à travers l'horizon. Pour plus de détails sur la modélisation stock de sécurité, nous invitons le lecteur à se reporter au chapitre 3 page 36.

Le critère d'optimisation consiste à générer un plan de production qui minimise la somme des coûts de production, des coûts de lancement, des coûts de stockage ainsi que des coûts de rupture sur la demande et de déficit sur le stock de sécurité, tout en respectant les contraintes de stockage et de capacité. Nous rappelons dans ce qui suit, les paramètres et les variables du problème MCLS4 :

Les paramètres :
d_{it} : Demande (commandes et prévisions) pour la référence i à la période t.
f_{it} : Besoin fixe en ressources de la référence i à la période t.
v_{it} : Besoin variable en ressources de la référence i à la période t.

c_t : Capacité disponible à la période t.
L_{it} : Stock de sécurité de la référence i à la période t.
α_{it} : Coût de production unitaire variable de la référence i à la période t.
β_{it} : Coût fixe relatif à un lancement de la référence i à la période t.
γ_{it}^{+} : Coût unitaire de stockage de la référence i à la période t.
γ_{it}^{-} : Coût unitaire de déficit sur le stock de sécurité de la référence i à la période t.
φ_{it} : Coût unitaire de rupture sur la demande pour la référence i à la période t.
On pose : $\delta_{it} = L_{it} - L_{i(t-1)}$.

Les variables :
x_{it} : Quantité produite pour la référence i à la période t.
$y_{it} = \begin{cases} 1 \text{ s'il y a production de la référence } i \text{ à la période } t \text{ (i.e. si } x_{it} > 0). \\ 0 \text{ sinon} \end{cases}$
I_{it}^{+} : Valeur du surstock de la référence i à la fin de la période t (Excédent de stock par rapport au stock de sécurité).
I_{it}^{-} : Valeur du déficit sur le stock de sécurité de la référence i à la fin de la période t (inférieure au stock de sécurité).
r_{it} : Rupture sur la demande pour la référence i à la période t (inférieure à la demande).

En utilisant ces variables et paramètres, nous formulons le problème MCLS4 comme suit :

$$\min \sum_{i,t} \alpha_{it} x_{it} + \sum_{i,t} \beta_{it} y_{it} + \sum_{i,t} \gamma_{it}^{+} I_{it}^{+} + \sum_{i,t} \gamma_{it}^{-} I_{it}^{-} + \sum_{i,t} \varphi_{it} r_{it} \tag{5.1}$$

s.c :

$$I_{i(t-1)}^{+} - I_{i(t-1)}^{-} + x_{it} + r_{it} = d_{it} + \delta_{it} + I_{it}^{+} - I_{it}^{-}, \quad \forall i, \forall t \tag{5.2}$$

$$\textstyle\sum_{i=1}^{N} v_{it} x_{it} + \sum_{i=1}^{N} f_{it} y_{it} \leq c_t, \quad \forall t \tag{5.3}$$

$$x_{it} \leq M y_{it}, \quad \forall i, \forall t \tag{5.4}$$

$$r_{it} \leq d_{it}, \quad \forall i, \forall t \tag{5.5}$$

$$I_{it}^{-} \leq L_{it}, \quad \forall i, \forall t \tag{5.6}$$

$$x_{it}, r_{it}, I_{it}^{+}, I_{it}^{-} \geq 0, \quad \forall i, \forall t \tag{5.7}$$

$$y_{it} \in \{0, 1\}, \quad \forall i, \forall t \tag{5.8}$$

La fonction objectif (5.1) minimise le coût total induit par le plan de production, à savoir les coûts de production et de stockage, ainsi que les coûts fixes de lancement, les coûts de rupture et les coûts de déficit sur le stock de sécurité. La contrainte (5.2) exprime la conservation des flux à travers l'horizon. La contrainte (5.3) représente la contrainte de respect de la capacité. La contrainte (5.4) exprime le fait que la production soit majorée par une production maximum. La contrainte (5.7) signifie que les variables x_{it}, r_{it}, I_{it}^{+} et I_{it}^{-} sont continues non négatives pour toute référence i, pour chaque période t. La dernière contrainte (5.8) exprime le fait que y_{it} est une variable binaire pour toute référence i, pour chaque période t.

Ce modèle est appelé également formulation agrégée, car la production d'une référence n'est définie que par sa période de production, contrairement à la formulation basée sur le

problème de localisation d'entrepôts (Facility Location-based formulation, Krarup et Bilde [101]) où la production est définie par sa période de production et par sa période de consommation.

Dans ce qui suit, nous proposons une formulation de problème MCLS4 basée sur le problème de localisation d'entrepôts. Cette formulation est notée : MCLS4$_{FL}$. En effet, la variable x_{it} peut être redéfinie plus finement, en considérant la période à laquelle cette production est réellement consommée. On pose $w_{itt'}$ la variable qui définit la quantité de la référence i produite à la période t ($t \neq 0$) et consommée à la période t', avec $t \leq t' \leq T$. La variable w_{i0t} représente le stock initial de la référence i au début de l'horizon qui sera consommé à la période t. On a alors,

$$x_{it} = \sum_{t'=t}^{T} w_{itt'}, \ \forall i, \forall t \tag{5.9}$$

La variable s_{it} représente le stock réel de la référence i à la période t, elle est égale à l'expression suivante : $I_{it}^{+} + L_{it} - I_{it}^{-}$. La variable s_{it} est reformulée en utilisant la formule suivante :

$$s_{it} = \sum_{t'=0}^{t} \sum_{t''=t+1}^{T} w_{it't''}, \ \forall i, \forall t \tag{5.10}$$

Afin d'obtenir la forumation de MCLS4$_{FL}$ basée sur le problème de localisation d'entrepôts, on remplace dans la contrainte (5.2), $I_{it}^{+} + L_{it} - I_{it}^{-}$ par s_{it} et on élimine les variables I_{it}^{+} du modèle MCLS4. On remplace également les deux variables x_{it} et s_{it} respectivement par les expressions (5.9) et (5.10).

On pose $\alpha'_{itt'} = \alpha_{it} + \gamma_{it}^{+} + \gamma_{i(t+1)}^{+}, \ldots, \gamma_{i(t'-1)}^{+}$ avec $t \leq t'$.

Le problème MCLS4$_{FL}$ est formulé comme suit :

$$\min \sum_{i,t} \sum_{t'=t}^{T} \alpha'_{itt'} w_{itt'} + \sum_{i,t} (\gamma_{it}^{-} + \gamma_{it}^{+}) I_{it}^{-} + \sum_{i,t} \varphi_{it} r_{it} - \sum_{i,t} L_{it} \tag{5.11}$$

s.c :

$$\sum_{t'=1}^{t} w_{itt'} + r_{it} = d_{it}, \qquad \forall i, \forall t \tag{5.12}$$
$$\sum_{i=1}^{N} v_{it} \sum_{t'=t}^{T} w_{itt'} + \sum_{i=1}^{N} f_{it} y_{it} \leq c_t, \qquad \forall t \tag{5.13}$$
$$w_{itt'} \leq d_{it'} y_{it}, \qquad \forall i, \forall t \tag{5.14}$$
$$r_{it} \leq d_{it}, \qquad \forall i, \forall t \tag{5.15}$$
$$I_{it}^{-} + \sum_{t'=0}^{t} \sum_{t''=t+1}^{T} w_{it't''} \geq L_{it}, \qquad \forall i, \forall t \tag{5.16}$$
$$I_{it}^{-} \leq L_{it}, \qquad \forall i, \forall t \tag{5.17}$$
$$r_{it}, I_{it}^{-} \geq 0, \qquad \forall i, \forall t \tag{5.18}$$
$$w_{itt'} \geq 0, \qquad \forall i, \forall t, \forall t' \tag{5.19}$$
$$y_{it} \in \{0, 1\}, \qquad \forall i, \forall t \tag{5.20}$$

Les contraintes (5.12) et (5.13) sont respectivement la reformulation de l'équation de flux et la contrainte de respect de capacité en fonction des nouvelles variables $w_{itt'}$. La contrainte (5.14) exprime le fait que la production de la référence i à la période t pour satisfaire la demande à la période t' doit être inférieure ou égale à la demande de la référence i à la période t. La contrainte (5.16) exprime le fait que la somme du stock disponible et du déficit sur le stock de sécurité soit supérieure au stock de sécurité pour toute référence i, pour chaque période t. La contrainte (5.18) signifie que les variables r_{it} et I_{it}^{-} sont continues non négatives pour toute référence i, pour chaque période t. La contrainte (5.19) exprime le fait que $w_{itt'}$ est une variable continue non négative pour toute référence i et pour chaque périodes t, t' avec $t' \geq t$. Les contraintes (5.15), (5.17) et (5.20) sont exactement les mêmes contraintes que (5.5), (5.6) et (5.8) respectivement.

5.2 Relaxation lagrangienne

Dans cette section, nous présentons une approche de relaxation lagrangienne basée sur la dualisation des contraintes de capacité. Cette approche permet de trouver une borne inférieure au problème MCLS4. Nous décrivons également les sous-problèmes induits par cette relaxation ainsi qu'un algorithme polynomial pour les résoudre. La méthode choisie pour résoudre le problème dual est le sous-gradient.

5.2.1 Relaxation lagrangienne des contraintes de capacité

L'idée générale derrière une approche de relaxation lagrangienne est de décomposer un problème de grande taille (problème de lot-sizing à plusieurs références et à capacité finie) en plusieurs sous-problèmes de plus petite taille (problème de lot-sizing à capacité infinie et une seule référence). Ceci est fait en relaxant les contraintes de capacité en utilisant des multiplicateurs lagrangiens (π_t) du modèle MCLS4 présenté page 93. Les contraintes (5.3) sont considérées comme des contraintes couplantes, puisqu'elles permettent de relier l'ensemble des références pour le calcul de la consommation globale en capacité.

Plusieurs auteurs ont appliqué la relaxation lagrangienne afin de trouver de bonnes bornes inférieures aux problèmes de lot-sizing à capacité finie pour le cas de plusieurs références. Nous pouvons citer les travaux de Chen et Thizy [31], Diaby *et al.* [48], Du Merle *et al.* [53], Thizy et Van Wassenhove [171] et Trigeiro *et al.* [172]. La plupart d'entre eux ont opté pour la

relaxation des contraintes de capacité. En effet, cette relaxation décompose le problème en N sous-problèmes de lot-sizing à une seule référence et sans contrainte de capacité. Pour un état de l'art plus détaillé sur l'utilisation de la relaxation lagrangienne pour les problèmes de lot-sizing, le lecteur peut se reporter au chapitre 2, section 2.2.4.1.

La relaxation des contraintes de capacité du problème MCLS4 décompose le problème en N problèmes MCLS4 à une seule référence et sans contrainte de capacité que l'on note ULS4[2]. Puisque nous traitons chaque référence séparément, nous allons ignorer dans la partie qui suit, l'indice des références i. La modélisation du problème ULS4 est présentée dans ce qui suit.

$$\min \sum_t \alpha_t x_t + \sum_t \beta_t y_t + \sum_t \gamma_t^+ I_t^+ + \sum_t \gamma_t^- I_t^- + \sum_t \varphi_t r_t + \sum_t \pi_t \left(v_t x_t + f_t y_t\right) \tag{5.21}$$

s.c :

$$I_{t-1}^+ - I_{t-1}^- + x_t + r_t = d_t + \delta_t + I_t^+ - I_t^-, \quad \forall t \tag{5.22}$$
$$x_t \leq M y_t, \quad \forall t \tag{5.23}$$
$$r_t \leq d_t, \quad \forall t \tag{5.24}$$
$$I_t^- \leq L_t, \quad \forall t \tag{5.25}$$
$$x_t, r_t, I_t^+, I_t^- \geq 0, \quad \forall t \tag{5.26}$$
$$y_t \in \{0,1\}, \quad \forall t \tag{5.27}$$

A notre connaissance, le problème ULS4 n'a jamais été traité dans la littérature. Le problème ULS4 est une extension du problème classique de lot-sizing ULS (le problème ULS est décrit au chapitre 2.1.1 page 10). En effet, en éliminant les variables r_t et I_t^- du modèle précédent on retombe sur le modèle ULS qui est résolu en $O(T \log T)$ ([5, 59, 188]). Si on élimine la variable de déficit sur stock de sécurité, on retombe sur le problème ULS avec des coûts de rupture sur la demande. Ce problème à été résolu en $O(T^2)$ par Aksen *et al.* [8].

Loparic *et al.* [112] ont considéré le stock de sécurité en imposant une borne inférieure aux variables de stock. Dans leurs cas, le déficit sur le stock de sécurité n'est pas autorisé, le stock de sécurité est une contrainte et non un objectif. Les auteurs ont également considéré des variables représentant les ventes au lieu de demandes fixes. La modélisation du problème de lot-sizing avec des variables de ventes est équivalente à la modélisation avec des variables de rupture sur la demande. En effet, maximiser les ventes revient à minimiser les ruptures sur les demandes. Les auteurs ont proposé une approche de programmation dynamique et de flots pour résoudre le problème.

L'originalité du problème ULS4 réside dans l'intégration conjointe des ruptures sur la demande et des déficits sur le stock de sécurité au problème ULS. Dans le paragraphe qui suit, nous présentons un algorithme polynomial pour résoudre le problème ULS4.

5.2.2 Résolution des sous-problèmes ULS4

Dans cette partie, nous proposons une présentation du problème ULS4 en un réseau à coûts fixes. Nous présentons également des propriétés de la solution optimale du problème ULS4.

[2]ULS4 est une abréviation en anglais de : Uncapacitated Lot-Sizing problem with Shortage costs and Safety Stock deficit costs.

Ces dernières vont nous permettre de proposer un algorithme récursif de programmation dynamique. Il résout le problème en un temps polynomial.

5.2.2.1 Propriétés de la solution optimale du problème ULS4

Dans ce qui suit, nous présentons le problème ULS4 sous forme d'un réseau à coûts fixes. En se basant sur cette représentation, nous fournissons quelques propriétés de la solution optimale.

Formulation du problème ULS4 en un réseau à coûts fixes La figure 5.1 représente la modélisation du problème ULS4 en un réseau à coûts fixes, il est appelé également problème de transbordement. Les sommets 1 à T représentent les différentes périodes définies dans le problème ULS4. Le sommet 0 représente la source principale dont la disponibilité est égale à D définie comme la somme des demandes et des différences entre les stocks de sécurité.
On a : $D = \sum_{t=1}^{T} d_t + \delta_t = \sum_{t=1}^{T} d_t + L_T - L_0$, avec $L_0 = 0$.

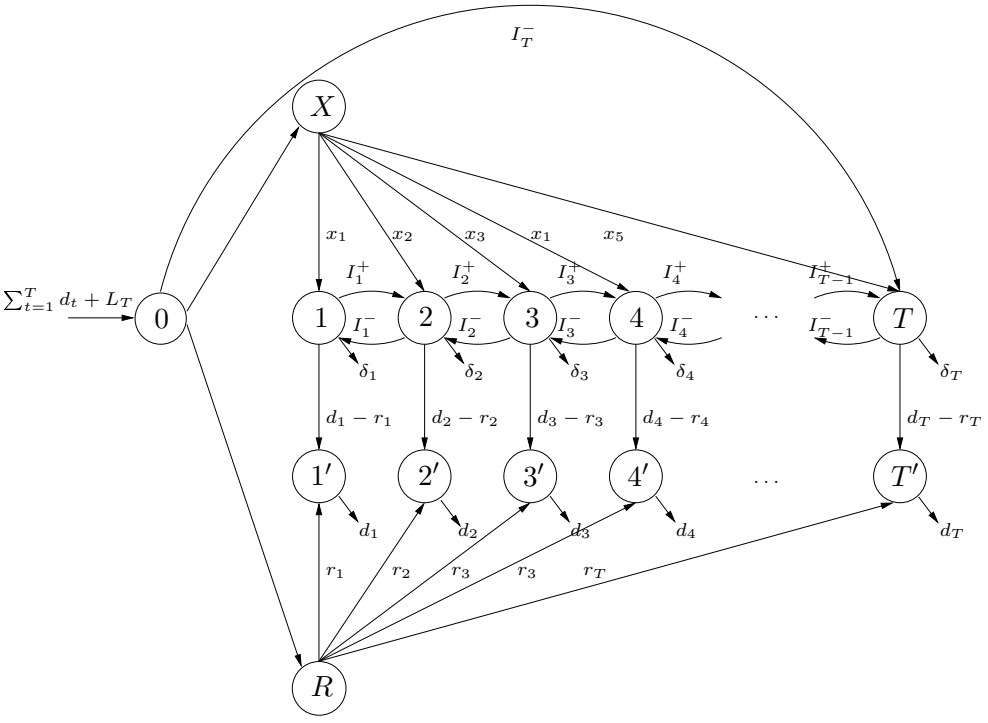


FIG. 5.1 – Représentation du problème ULS4 en réseau

X et R sont deux nœuds de transbordement qui transfèrent respectivement la production totale et la somme des ruptures sur la demande. Le reste du transbordement transite par le nœud T et représente le déficit sur le stock de sécurité maximum à la période T. Il est évident que la somme des transferts à X, R et T est égale à la quantité qui transite à travers le nœud 0. A chaque nœud $t \in \{1, 2, \ldots, T\}$ nous avons un besoin δ_t relative à la variation du stock de sécurité, $\delta_t = L_t - L_{t-1}$. δ_t peut être positive ou négative. Les nœuds $\{1', 2', \ldots, T'\}$ représentent les demandes d_t aux périodes $t \in \{1, \ldots, T\}$. A Chaque nœud $t \in \{1', 2', \ldots, T'\}$ nous avons une demande de d_t.

Les arcs de ce graphe sont définis comme suit :

- (X, t) sont des arcs de prodcution, $t \in \{1, 2, \ldots, T\}$. Ils représentent les quantités produites à chaque période t ;
- $(t, t+1)$ sont des arcs de stock, $t \in \{1, 2, \ldots, T-1\}$. Chaque arc représent le stock sortant à la période $t-1$ et entrant à la période t ;
- $(t, t-1)$ sont des arcs de déficit sur le stock de sécurité, $t \in \{2, \ldots, T\}$. Ils représentent le déficit sur le stock de sécurité de la période $t-1$. La capacité sur ces arcs est limitée. Elle est égale au stock de sécurité. En effet, le déficit sur le stock de sécurité de la période $t-1$ est limité par le niveau de stock de sécurité L_t de la même période ;
- (t, t') sont des arcs qui représentent la demande satisfaite à la période t, $t \in \{1, 2, \ldots, T\}$ et $t' \in \{1', 2', \ldots, T'\}$;
- (R, t') sont des arcs de rupture, $t' \in \{1', 2', \ldots, T'\}$. Il représentent les ruptures sur la demande pour chaque période.

Tous les coûts sur les arcs de notre réseau sont concaves. En effet, les coûts sur les arcs de type (t, t'), $(0, X)$ et $(0, R)$ ont des coûts nuls. Les arcs de type (R, t') ont un coût unitaire de φ_t. Les arcs de type $(t, t+1)$ et $(t, t-1)$ ont des coûts γ_t^+ et γ_t^- respectivement. Le coût sur les arcs de production (X, t) est égal à zéro si le flot est nul et $(\beta_t + \pi_t f_t + (\alpha_t + \pi_t v_t)x_t)$ si le flot x_t est positif.

Le réseau de la figure 5.1 représente un réseau à une seule source avec des coûts concaves. Trouver une solution optimale au problème ULS4 revient à trouver un flot extrême (cf. définition 7) à coût minimum sur le réseau de la figure 5.1.

Propriétés des solutions optimales Afin de proposer un algorithme de programmation dynamique pour résoudre le problème ULS4, nous présentons quelques propriétés des solutions optimales. Nous allons tout d'abord présenter quelques définitions.

Défnition 5. *Une période t est dite **point de production** si la production à cette période est supérieure à zéro ($x_t > 0$).*

Défnition 6. *Une période t est dite **point d'inventaire** si le surstock à cette période est nul et le déficit sur le stock de sécurité est nul ($I_t^+ = 0$ et $I_t^- = 0$) ou le surstock à cette période est nul et le déficit sur le stock de sécurité est égal au stock de sécurité ($I_t^+ = 0$ et $I_t^- = L_t$). Plus précisement, une période t est dite :*

- ***Point d'inventaire de type 1** si ($I_t^+ = 0$ et $I_t^- = 0$) ;*
- ***Point d'inventaire de type 2** si ($I_t^+ = 0$ et $I_t^- = L_t$).*

Défnition 7. *Un flot réaslisable $(\widehat{x}, \widehat{r}, \widehat{I^+}, \widehat{I^-})$ est dit flot extrême s'il n'existe pas de flots réalisables $(\widehat{x}_1, \widehat{r}_1, \widehat{I_1^+}, \widehat{I_1^-})$ et $(\widehat{x}_2, \widehat{r}_2, \widehat{I_2^+}, \widehat{I_2^-})$ tel que :*

$(\widehat{x}, \widehat{r}, \widehat{I^+}, \widehat{I^-}) = (\widehat{x}_1, \widehat{r}_1, \widehat{I_1^+}, \widehat{I_1^-}) + (\widehat{x}_2, \widehat{r}_2, \widehat{I_2^+}, \widehat{I_2^-})$

Les flots extrêmes correspondent aux solutions réalisables de base dans le tableau du simplexe [44, 199]. Il est bien connu [55, 199] que si la fonction objectif a un minimum global fini sur la région réalisable, alors il existe un flot extrême qui est également un flot optimal. Puisque l'ensemble des solutions réalisables a un nombre fini de flots extrêmes [44], la recherche de tous les flots extrêmes permet de trouver le flot optimal. Cependant, une telle recherche peut s'avérer extrêmement laborieuse. Dans qui suit, nous présentons quelques

caractéristiques des flots extrêmes du problème ULS4 qui vont nous permettre de faciliter cette recherche.

Lemme 1.
$I_t^+ I_t^- = 0$ *pour tout* $t \in 1, \ldots, T$.

Preuve. Ce résultat s'explique à partir du réseau de la figure 5.1, car on ne peut avoir un flot extrême avec $I_t^+ > 0$ et $I_t^- > 0$. En effet, dans une solution extrême il est impossible d'avoir un flot avec cette propriété. □

Proposition 19.
Les propositions suivantes sont équivalentes :
- $(\widehat{x}, \widehat{r}, \widehat{I}^+, \widehat{I}^-)$ *est un flot extrême pour le problème (P) ;*
- $(\widehat{x}, \widehat{r}, \widehat{I}^+, \widehat{I}^-)$ *est réalisable et entre chaque paire de points de production* (i, k) / $i < k$, *il existe un point d'inventaire* j / $(i \leq j < k)$.

Preuve. La preuve découle directement des caractéristiques d'un flot extrême dans un réseau cf. travaux de Zangwill [199].

En effet, supposons que l'on ait deux points de production i et k avec $i < k$ et qu'il n'existe pas de point d'inventaire entre ces deux points dans un flot extrême. Ceci implique que pour tous les sommets t compris entre i et k on ait soit ($I_t^+ > 0$ et $I_t^- = 0$) soit ($I_t^+ = 0$ et $0 < I_t^- < L_t$) (Lemme 1, Définition 6). Le réseau ainsi constitué contient un cycle où il n'existe aucune arête fixée à l'une de ses bornes. Or, la non existence d'un tel cycle est l'une des caractéristiques d'un flot extrême dans un réseau. Si un cycle ainsi constitué existe dans un flot extrême, alors ce flot peut facilement être exprimé en fonction de deux autres flots réalisables. Ceci prouve qu'un tel flot n'est pas extrême, ce qui contredit l'hypothèse que $(\widehat{x}, \widehat{r}, \widehat{I}^+, \widehat{I}^-)$ est un flot extrême.

□

La proposition suivante est équivalente à celles de la proposition 19 :
- $(\widehat{x}, \widehat{r}, \widehat{I}^+, \widehat{I}^-)$ est réalisable et entre chaque paire de points d'inventaire adjacents il existe au plus un point de production.

Les solutions optimales du problème ULS4 ont une structure particulière. En effet, les arêtes qui ne sont pas fixées à leurs bornes supérieures forment une structure d'arbres. Un exemple de la structure des solutions optimales du problème ULS4 est présenté dans la figure 5.2, les arêtes en gras représentent les arêtes fixées à leurs bornes supérieures. Nous présentons à travers l'exemple 5.2.1 la structure de la solution optimale d'un exemple numérique du problème ULS4.

Exemple 5.2.1. *A travers cet exemple, nous montrons la structure de la solution optimale d'un problème ULS4. L'exemple du tableau 5.1 représente les données d'un problème ULS4 à 5 périodes.*

La solution optimale du problème de lot-sizing présentée dans le tableau 5.1 est obtenue en utilisant la modélisation mathématique du problème ULS4 présentée précédemment et le solveur de programmation linéaire en nombres entiers CPLEX 9.0. Les résultats sont présentés dans le tableau 5.2 et le graphique 5.3.

A partir des résultats présentés dans le tableau 5.2, il est intéressant de noter qu'à la période 3, la demande n'est pas totalement perdue. Nous remarquons également que les

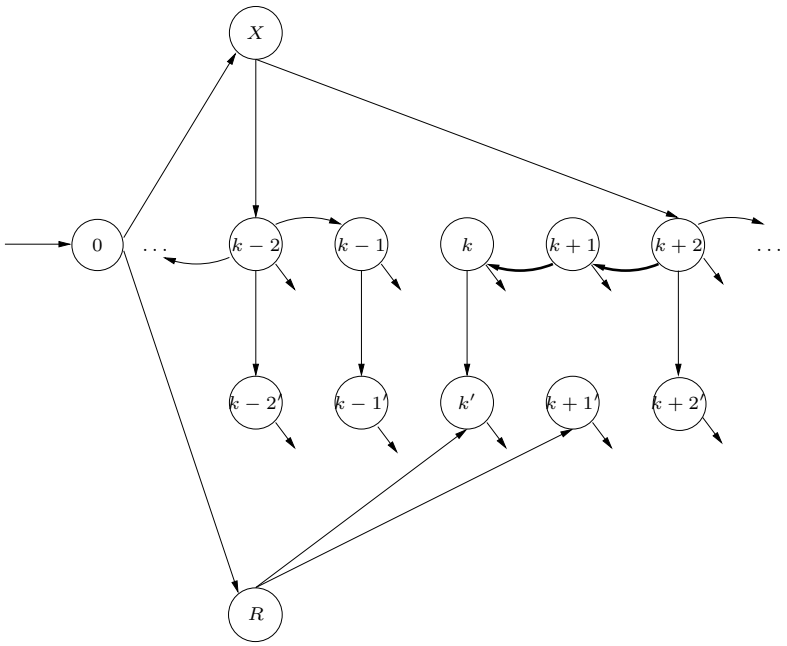


FIG. 5.2 – Structure de la solution optimale du problème ULS4

Périodes (t)	1	2	3	4	5
Demandes (d_t)	1 000	1 000	1 000	1 000	1 000
Stock de sécurité (L_t)	200	800	1 000	1 000	800
Coût de Setup (β_t)	200 000	250 000	250 000	100 000	250 000
Coût unitaire de Production (α_t)	70	70	70	70	70
Coût unitaire de Stockage (γ_t^+)	55	55	55	55	55
Coût unitaire de rupture (φ_t)	250	212.5	175	137.5	100
Coût unitaire de Déficit (γ_t^-)	150	127.5	105	82.5	60

TAB. 5.1 – Exemple ULS4 à 5 périodes

Périodes (t)	1	2	3	4	5
Demandes (d_t)	1 000	1 000	1 000	1 000	1 000
Production (x_t)	2 800	0	0	2 000	0
Rupture (r_t)	0	0	200	0	0
Stock de sécurité (L_t)	200	800	1 000	1 000	800
Excédent de stock (I_t^+)	1 600	0	0	0	0
Déficit sur le stock de sécurité (I_t^-)	0	0	1 000	0	800

TAB. 5.2 – Solution optimale du tableau 5.1

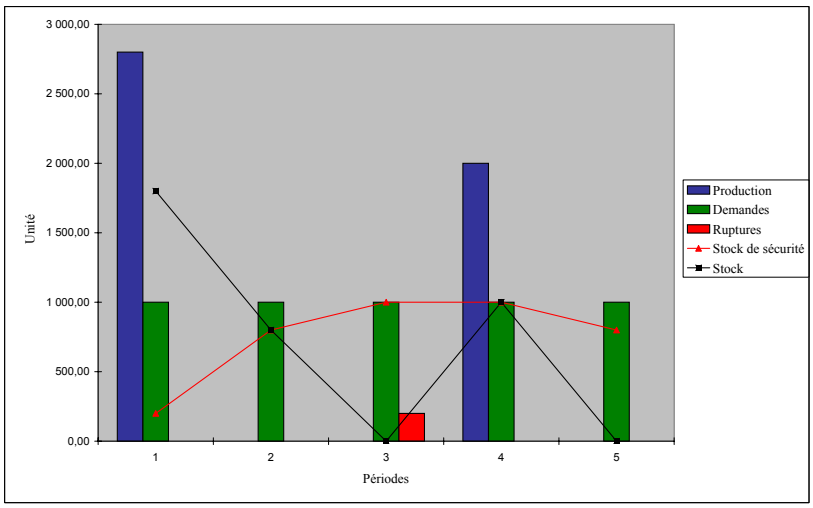


FIG. 5.3 – Solution optimale du tableau 5.1

périodes 2 et 3 sont des points d'inventaire de type 1 et de type 2 respectivement. Les périodes 1 et 4 sont des points de production. Les périodes 2 et 4 sont des points d'inventaire de type 1, et les périodes 3 et 5 sont des points d'inventaire de type 2. La figure 5.4 est la représentation de la solution optimale dans un réseau. Les arêtes en gras représentent les arêtes fixées à leurs bornes supérieures.

5.2.2.2 Algorithme de résolution du problème ULS4

En se basant sur les propriétés des solutions optimales (proposition 19), nous développons dans ce qui suit un algorithme de programmation dynamique afin de résoudre le problème ULS4 à l'optimum. Nous avons montré précédemment (proposition 19) qu'une solution réalisable est un flot extrême si et seulement si entre deux points de production il existe un point d'inventaire. D'une manière équivalente, entre deux points d'inventaire adjacents i, k avec $i < k$, il existe au plus une période de production j $(i < j \leq k)$. Il suffit alors de chercher tous les flots extrêmes qui satisfont ces propriétés et d'identifier le flot extrême de coût minimum. L'idée de l'algorithme de programmation dynamique proposé est de considérer tous les triplets (i, j, k) (avec $i < j \leq k$, i et k points d'inventaire, et j point de production) et de trouver la séquence de triplets (i, j, k) qui donne la meilleure valeur de la fonction objectif du problème ULS4. Donc, pour chaque triplet (i, j, k) on définit un sous-problème que l'on résout à l'optimum.

On définit alors le problème ULS4_{ijk}^{uv} qui n'est qu'un problème ULS4 restreint.

Les indices u et v caractérisent les points d'inventaire (de type 1 ou de type 2). L'indice u (resp. v) est fixé à 0 si la période i (resp. k) est un point d'inventaire de type 1. L'indice

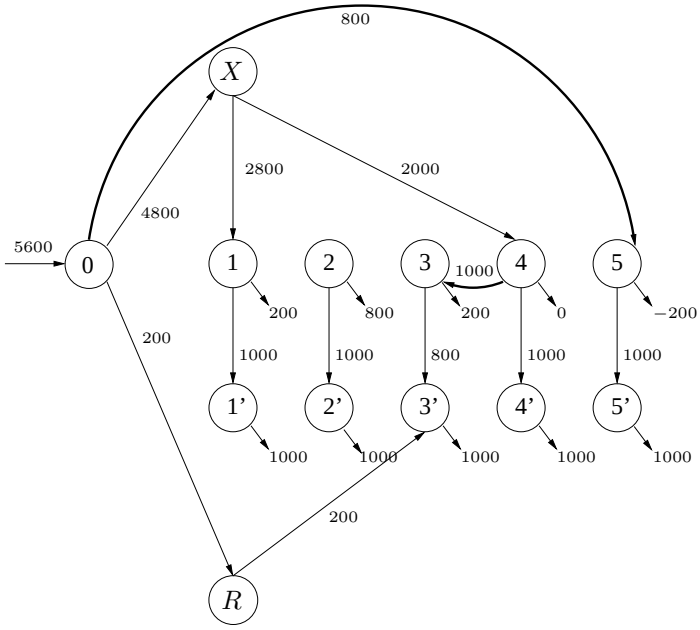


FIG. 5.4 – Représentation de la solution optimale du problème ULS4 dans un réseau

u (resp. v) est fixé à 1 si la période i (resp. k) est un point d'inventaire de type 2. $\{i, \ldots, k\}$ représente l'ensemble des entiers de i à k.

ULS4_{ijk}^{uv} est un problème défini sur l'intervalle des périodes $\{i, \ldots, k\}$. Il n'existe de la production qu'à la période j, la production aux périodes $t \in \{i, \ldots, k\} \setminus j$ est nulle $x_t = 0$. Les indices u et v caractérisent les points d'inventaire $i-1$ et k de la manière suivante,

- Si $u = 0$ alors $I_{i-1}^{+} = 0$ et $I_{i-1}^{-} = 0$ si $i > 1$;
- Si $u = 1$ alors $I_{i-1}^{+} = 0$ et $I_{i-1}^{-} = L_{i-1}$ si $i > 1$;
- Si $v = 0$ alors $I_{k}^{+} = 0$ et $I_{k}^{-} = 0$;
- Si $v = 1$ alors $I_{k}^{+} = 0$ et $I_{k}^{-} = L_k$.

Posons Z_{ijk}^{uv} la valeur de la fonction objectif du problème ULS4_{ijk}^{uv}.

Trouver la valeur de Z_{ijk}^{uv} revient à trouver la solution optimale au problème linéaire ULS4_{ijk}^{uv} défini par le programme linéaire suivant :

$$Z_{ijk}^{uv} = \beta_j + \pi_j f_j + \min \left((\alpha_j + \pi_j v_j)\, x_j + \sum_{t \in \{i,\ldots,k\}} (\gamma_t^{+} I_t^{+} + \gamma_t^{-} I_t^{-} + \varphi_t r_t) \right) \tag{5.28}$$

s.c :

$$I^+_{t-1} - I^-_{t-1} + r_t = d_t + \delta_t + I^+_t - I^-_t, \qquad \forall t \in \{i+1, \ldots, k\} \setminus j \tag{5.29}$$

$$I^+_{j-1} - I^-_{j-1} + x_j + r_j = d_j + \delta_j + I^+_j - I^-_j \tag{5.30}$$

$$x_j \leq M, \tag{5.31}$$

$$r_t \leq d_t, \qquad \forall t \in \{i, \ldots, k\} \tag{5.32}$$

$$I^-_t \leq L_t, \qquad \forall t \in \{i, \ldots, k\} \tag{5.33}$$

$$x_t, r_t, I^+_t, I^-_t \geq 0, \qquad \forall t \in \{i, \ldots, k\} \tag{5.34}$$

Le problème ULS4^{uv}_{ijk} peut être résolu en utilisant un algorithme de flot à coût minimum. Les meilleurs algorithmes de flot à coût minimum sont présentés dans le tableau 5.3 page 103.

Afin de trouver la valeur du plan de production optimal Z^{uv}_{ik} sur l'intervalle $i, \ldots, k$, on cherche le minimum entre toutes les politiques optimales pour les intervalles $i, \ldots, k$ avec j comme unique période de production ($i < j \leq k$) et la politique optimale pour l'intervalle $i, \ldots, k$ avec aucune période de production (que l'on note Z^{uv}_{i0k}).

$$Z^{uv}_{ik} = \min \left[Z^{uv}_{i0k}, \min_{i<j\leq k} Z^{uv}_{ijk} \right] \tag{5.35}$$

Une solution optimale de base de ULS4 correspond à la valeur minimale d'une séquence d'intervalles de points d'inventaire adjacents. Si la fonction F^u_k est définie comme étant le coût correspondant à la valeur minimale de séquences d'intervalles de points d'inventaire adjacents de la période 1 à la période k, F^u_k peut être calculé en utilisant la formule récursive suivante :

$$F^v_k = \min_{i\leq k; u=0,1} \left\{ F^u_{i-1} + Z^{uv}_{ik} \right\} \tag{5.36}$$

La valeur optimale du problème ULS4 est donnée par $F_T = \min_{v=0,1} F^v_T$.

L'algorithme de résolution du problème ULS4 est décrit par l'algorithme 4.

Complexité de l'algorithme de programmation dynamique Le problème ULS4^{uv}_{ijk} est résolu en utilisant un algorithme de flot à coût minimum. Les meilleurs algorithmes de flot à coût minimum sont présentés dans le tableau 5.3. U et C représentent respectivement la capacité maximum sur les arcs et le coût maximum sur les arcs. m et n représentent respectivement le nombre d'arcs et le nombre de nœuds du problème traité.

Auteurs	Complexité théorique
Goldberg and Tarjan [69]	$O(nm \log(n^2/m) \log(nC))$
Ahuja *et al.* [6]	$O(nm \log\log U \log(nC))$
Orlin [139]	$O(m \log n(m + n \log n)$

TAB. 5.3 – Complexité des algorithmes de flots à coût minimum

Algorithme 4 Algorithme de résolution du problème ULS4

$u \leftarrow 0$, $v \leftarrow 0$
pour tout $t \in \{1, \ldots, T\}$ et $u \in \{0, 1\}$ **faire**
  $F_t^u \leftarrow +\infty$, $F_0^u \leftarrow 0$
fin pour
pour $i = 1$ à $T - 1$ **faire**
  pour $k = i + 1$ à T **faire**
    répéter
      Calculer Z_{i0k}^{uv}
      $Z_{ik}^{uv} \leftarrow Z_{i0k}^{uv}$
      pour $j = i + 1$ à k **faire**
        Calculer Z_{ijk}^{uv}
        $Z_{ik}^{uv} \leftarrow \min\left[Z_{ik}^{uv}, Z_{ijk}^{uv}\right]$
      fin pour
      $F_k^v \leftarrow \min\left\{F_k^v, F_{i-1}^u + Z_{ik}^{uv}\right\}$
      $v \leftarrow 1 - v$
      si $v = 0$ **alors**
        $u \leftarrow 1 - u$
      finsi
    jusqu'à $u = 0$ et $v = 0$
  fin pour
fin pour
$F_T = \min_{u=0,1} F_T^u$

La méthode proposée par Goldberg and Tarjan [69] est une procédure de scaling des capacités. En effet, la complexité de leur algorithme dépend de U. Ahuja *et al.* [6] ont développé un algorithme de double scaling qui combine le scaling des coûts et des capacités. La complexité de cet algorithme dépend des deux paramètres U et C. L'algorithme proposé par Orlin [139] est fortement polynomial, il ne dépend pas des paramètres U et C. Pour plus de détails sur les meilleurs algorithmes de flot à coût minimum le lecteur peut se référer à [7].

Au pire cas, le nombre de sommets du graphe qui représente le problème Z_{ijk}^{uv} est de $2T+3$ et le nombre d'arcs est de $5T+1$. Donc $n = 2T+3$ et $m = 5T+1$. Le calcul de Z_{ijk}^{uv} s'effectue donc en $O(T^2 \log^2 T)$ (Orlin [139]). La complexité globale de l'algorithme de résolution du problème $ULS4$ est de $O(T^5 \log^2 T)$.

Cette complexité reste assez importante, mais elle prouve que le problème est solvable en temps polynomial. Plusieurs techniques peuvent être utilisées pour améliorer cette complexité. En effet, Wagelmans *et al.* [188] ont utilisé des techniques géométriques pour améliorer l'algorithme de Wagner et Whitin [189] en faisant passer son temps de calcul de $O(T^2)$ à $O(T \log T)$.

La complexité $O(T^5 \log^2 T)$ est réduite à $O(T^2)$ si le stock de sécurité n'est pas considéré. En effet, Aksen *et al.* [8] ont proposé un algorithme de programmation dynamique en $O(T^2)$ pour résoudre le problème classique de lot-sizing ULS avec des ruptures sur la demande noté ULS2. Ce problème satisfait aux conditions d'optimalité de Zangwill [199]. A chaque période t, nous avons deux possibilités, où bien la demande à cette période est totalement perdue, où bien satisfaite en totalité. Cette demande doit toujours être satisfaite par la production, le

stock ou perdue mais jamais par une combinaison de ces trois possibilités.

En relaxant la contrainte (5.33) du problème ULS4 et en éliminant les besoin δ_t, on retombe sur un problème de lot-sizing avec des ruptures sur la demande et du backlogging noté ULS2B[3]. En effet, quand la capacité sur les variables de stock de sécurité est relaxée, la demande peut être satisfaite à partir de la production de périodes futures. Ce problème satisfait également les conditions de Zangwill [199]. Ainsi, la demande d'une période donnée doit toujours être satisfaite par la production, le stock, le backlogging ou perdue mais jamais par une combinaison de ces quatre possibilités. En se basant sur ces conditions, il est facile de développer un algorithme de programmation dynamique en $O(T^3)$ pour résoudre le problème ULS2B à l'optimum. La figure 5.5 représente la structure en arbre de la solution optimale du problème ULS2B. Le principe de cet algorithme de programmation dynamique est basé sur les points de production. En effet, il suffit de chercher la séquence de points de production qui respecte les conditions de Zangwill et qui minimise le coût total. Ce parcours nécessite $O(T^2)$ opérations. Le calcul de coût pour chaque paire de points de production se fait au pire cas en $O(T)$. D'où la complexité de $O(T^3)$.

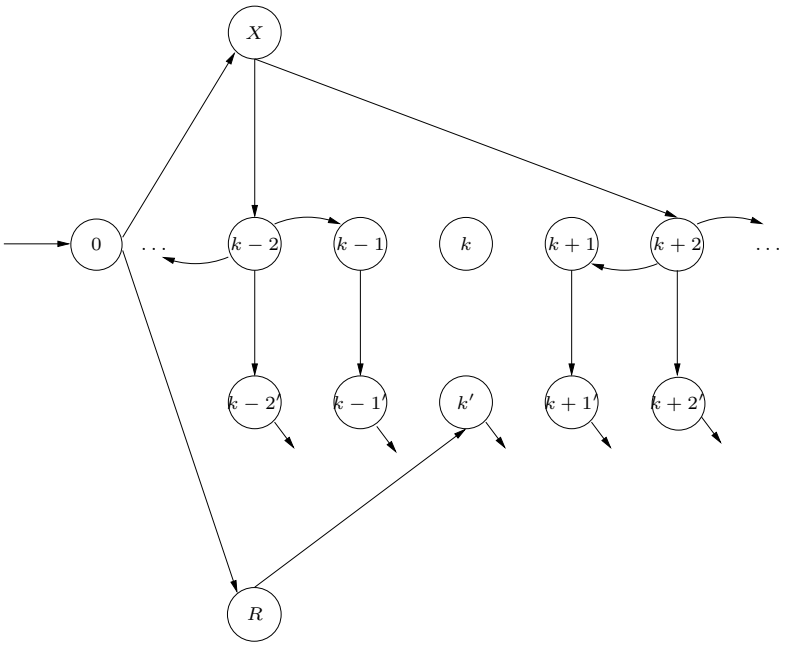


FIG. 5.5 – Structure en arbre de la solution optimale du problème ULS2B

5.2.3 Schéma de relaxation lagrangienne

Dans cette partie, nous décrivons le schéma de la relaxation lagrangienne ainsi que la résolution du problème dual obtenu après dualisation des contraintes de capacité.

[3]ULS2B est une abréviation en anglais de : Uncapacitated Lot-Sizing problem with Shortage and Backlogging costs.

5.2.3.1 Dualisation des contraintes de capacité

Dans notre étude, la relaxation lagrangienne sert à décomposer le problème MCLS4, qui est un problème d'optimisation combinatoire difficile en N sous-problèmes faciles à résoudre ULS4, en dualisant les contraintes difficiles couplantes à savoir les contraintes de capacité (5.3). Le but de cette relaxation est de fournir une borne inférieure au problème MCLS4. On rappelle que le problème MCLS4 est défini comme suit,

$$\begin{aligned} & \mathcal{Z} = \min \textstyle\sum_{i,t} \alpha_{it} x_{it} + \beta_{it} y_{it} + \gamma^{+}_{it} I^{+}_{it} + \gamma^{-}_{it} I^{-}_{it} + \varphi_{it} r_{it} \\ s.c. & \\ & (5.2), (5.3), (5.4), (5.5), (5.6), (5.7), (5.8) \end{aligned}$$

Les contraintes relaxées (5.3) sont dualisées dans la fonction objectif. En effet, à l'ensemble de contraintes (5.3), nous associons un vecteur $\pi = (\pi_1, \pi_2, \ldots, \pi_T)^T$ (avec $\pi_t \geq 0$ pour tout t) appelé *vecteur lagrangien*, les composantes π_t sont appelées *les multiplicateurs lagrangiens*. La *fonction lagrangienne associée* au problème MCLS4 est :

$$\begin{aligned} \mathcal{L}(x, y, r, I^{+}, I^{-}, \pi) = & \sum_{i,t} \left(\alpha_{it} x_{it} + \beta_{it} y_{it} + \gamma^{+}_{it} I^{+}_{it} + \gamma^{-}_{it} I^{-}_{it} + \varphi_{it} r_{it} \right) \\ + & \sum_{t} \pi_t \left(\sum_{i} (v_{it} x_{it} + f_{it} y_{it}) - c_t \right) \end{aligned} \tag{5.37}$$

On définit également l'ensemble de contraintes $\mathcal{S}$ tel que,

$$\mathcal{S} = \left\{ (x, y, r, I^{+}, I^{-}) \text{ vérfiant } (5.2), (5.4), (5.5), (5.6), (5.7), (5.8) \right\}$$

La fonction duale $\mathcal{W}(\pi)$ est définie pour tout $(\pi \geq 0)$ par :

$$\mathcal{W}(\pi) = \min_{(x,y,r,I^{+},I^{-}) \in \mathcal{S}} \mathcal{L}(x, y, r, I^{+}, I^{-}, \pi) \tag{5.38}$$

Pour tout $\pi \geq 0$, on définit le *problème lagrangien* par le programme linéaire suivant :

$$\begin{aligned} & \min \mathcal{L}(x, y, r, I^{+}, I^{-}, \pi) \\ & (x, y, r, I^{+}, I^{-}) \in \mathcal{S} \end{aligned}$$

$\mathcal{W}(\pi)$ définit une borne inférieure au problème MCLS4. En effet, $\mathcal{W}(\pi) \leq \mathcal{Z}$. Le but est donc de rechercher la valeur du vecteur π qui maximise la fonction $\mathcal{W}(\pi)$. Cette valeur est obtenue en résolvant le problème suivant, appelé problème *dual lagrangien*.

$$\begin{aligned} & \max_{u} \mathcal{W}(\pi) \\ s.c. & \\ & \pi \geq 0 \end{aligned}$$

5.2.3.2 Résolution du problème dual : Algorithme du sous-gradient

La méthode sous-gradient est une méthode itérative. A chaque étape k, une valeur de déplacement dans la direction du sous-gradient λ_k est choisie a priori afin d'améliorer la solution de l'étape précédente $\mathcal{W}(\pi^{k-1})$. La convergence de $\mathcal{W}(\pi^k)$ n'est pas monotone.

Polyak [153], a montré que la suite $\mathcal{W}(\pi^k)$ converge vers l'optimum $\mathcal{W}(\pi^*)$ sous les seules conditions : $\lambda_k \to 0$ quand $k \to +\infty$, alors : $\sum_k^{+\infty} \lambda_k \to +\infty$, mais on ne peut rien dire sur la vitesse de la convergence. D'autre part, la suite des valeurs $\mathcal{W}(\pi^k)$ obtenues n'est généralement pas monotone croissante. Si on choisit à chaque étape k des valeurs pour λ_k définies par :

$$\lambda_k = \nu_k \frac{\mathcal{W}(\pi^*) - \mathcal{W}(\pi^k)}{\|\Gamma(\pi^k)\|} \tag{5.39}$$

où le coefficient ν_k dit coefficient de relaxation vérifie $\epsilon < \nu_k \leq 2$, $\forall k$ ($\epsilon > 0$ fixé) alors la convergence est géométrique (Polyak [154]). $\Gamma(\pi^k)$ est la valeur du sous-gradient de $\mathcal{W}(\pi)$ à l'itération k. $\left\|\Gamma(\pi^k)\right\|$ est la norme de $\Gamma(\pi^k)$.

Ce résultat a un intérêt principalement théorique, puisqu'on ne connaît pas $\mathcal{W}(\pi^*)$. Mais si l'on remplace dans la formule précédente $\mathcal{W}(\pi^*)$ par une estimation par défaut $\underline{\mathcal{W}} \leq \mathcal{W}(\pi^*)$, Polyak [154] a montré que : soit la séquence $\mathcal{W}(\pi^k)$ converge vers $\underline{\mathcal{W}}$; soit on obtient en un nombre fini d'itérations un point π_k vérifiant $\underline{\mathcal{W}} \leq \mathcal{W}(\pi^k) \leq \mathcal{W}(\pi^*)$. Ceci se produit en particulier quand $\nu = 2$ pour tout k.

Held *et al.* [78] ont montré que l'on peut remplacer $\mathcal{W}(\pi^*)$ par une estimation par excès $\overline{\mathcal{W}} \geq \mathcal{W}(\pi^*)$ sans que la convergence de l'algorithme ne soit affectée. Dans ce cas, la condition $\lambda_k \to 0$ quand $k \to +\infty$ oblige à choisir $\nu_k \to 0$ quand $k \to +\infty$.

En pratique, on choisira souvent $\overline{\mathcal{W}}$ la valeur de $\mathcal{Z}$ correspondant à la meilleure solution du problème MCLS4 obtenue dans les étapes précédentes du calcul.

D'après les expériences de Held *et al.* [78], le choix de ν_k comme suite donne de bons résultats.

Prendre ν égal à 2 pendant $2T$ itérations, (où T est le nombre de variables duales) puis diviser par 2 la valeur de ν et le nombre d'itérations jusqu'à atteindre une limite inférieure à q (fixée à l'avance) du nombre d'itérations. Enfin, diviser par 2 la valeur de ν toutes les q itérations jusqu'à ce que les λ_k soient suffisamment petits $< \epsilon$ fixé (généralement $q \approx 5$).

L'algorithme du sous-gradient est défini par l'algorithme 5.
L'algorithme 5 s'arrête si l'une des conditions suivantes est vérifiée :

- La solution du problème relaxé est réalisable pour le problème initial ;
- Un nombre maximal d'itérations Q est atteint ;
- $\overline{\mathcal{W}} - LB < \epsilon_1,\ \epsilon_1 > 0$;
- Les conditions des expériences de Held *et al.* [78] sont vérifiées.

5.3 Résultats expérimentaux

Dans cette partie, nous présentons des résultats expérimentaux issus de l'application de la relaxation lagrangienne au problème MCLS4. Les bornes inférieures obtenues à la fin de la

Algorithme 5 Algorithme du sous-gradient

Initialiser les multiplicateurs lagrangiens $\pi^0 \geq 0$.
Définir une suite de nombres λ_k tel que : $\lambda_k \to 0$ quand $k \to +\infty$. (ex. la formule 5.39)
Initialiser la meilleure borne inférieure $LB = -\infty$.
$k \leftarrow 0$
tant que Aucun critère d'arrêt n'est atteint **faire**
  $\mathcal{W}(\pi^k) \leftarrow \mathcal{L}(x^k, y^k, r^k, I^{+k}, I^{-k}, \pi^k) = \min_{(x,y,r,I^+,I^-)\in\mathcal{S}} \mathcal{L}(x, y, r, I^+, I^-, \pi^k)$.
  si $LB < \mathcal{W}(\pi^k)$ **alors**
    $LB \leftarrow \mathcal{W}(\pi^k)$
  finsi
  pour $t = 1$ à T **faire**
    $\Gamma(\pi_t^k) \leftarrow \sum_i (v_{it}x_{it}^k + f_{it}y_{it}^k) - c_t$. ($\Gamma(\pi^k)$ est un sous-gradient de $\mathcal{W}$ en π^k).
  fin pour
  $\pi^{k+1} \leftarrow \max(0, \pi^k + \lambda_k\Gamma(\pi^k))$
  $k \leftarrow k + 1$
fin tant que

relaxation lagrangienne sont comparées à celles obtenues à la fin d'un algorithme de branch-and-bound.

Nos algorithmes sont implémentés en langage C++ et ils sont intégrés à le logiciel d'APS de la société DynaSys, n.SKEP. Tous les tests ont été effectués sur un ordinateur personnel équipé d'un microprocesseur Pentium IV 2.66 GHz.

5.3.1 Instances de test

Les instances des clients DynaSys contiennent généralement plusieurs groupes de références et plusieurs gammes de production, ces contraintes ne sont pas encore prises en compte par notre approche de relaxation lagrangienne. Nous avons donc effectué des tests sur une série d'instances étendues de la librairie de lot-sizing LOTSIZELIB [114], initialement décrites dans Trigeiro *et al.* [172]. Les instances de Trigeiro *et al.* sont notées Tr_{N-T}, où N est le nombre de références et T le nombre de périodes. Elles sont caractérisées par une consommation variable en ressources égale à 1, et assez de capacité pour satisfaire toute la demande sur l'horizon de planification. Elles sont également caractérisées par des coûts de setup importants, des besoins fixes en ressources peut signifiants et pas de stock de sécurité. Les caractéristiques des instances de Trigeiro *et al.* [172] sont décrites dans le tableau 5.4.

Instance	N	T
tr_{6-15}	6	15
tr_{6-30}	6	30
tr_{12-15}	12	15
tr_{12-30}	12	30
tr_{24-15}	24	15
tr_{24-30}	24	30

TAB. 5.4 – Instances de Trigeiro *et al.* [172].

Puisque les instances de Trigeiro *et al.* sont caractérisées par des capacités suffisantes pour satisfaire toute la demande, nous avons effectué quelques modifications pour induire des

ruptures sur la demande et des déficits sur le stock de sécurité. Nous avons dérivé 12 nouvelles instances issues des instances Tr_{N-T} en augmentant le besoin fixe en ressources (temps de setup), le besoin variable en ressources et en rajoutant un seuil pour le stock de sécurité. Dans ce qui suit, nous donnons plus de détails sur la création des nouvelles instances.

Les nouvelles instances sont synthétisées en 2 classes de 6 instances chacune.

– Classe A : La première classe est obtenue en augmentant le besoin variable en ressources et en rajoutant un niveau de stock de sécurité. Le besoin variable en ressource est multiplié par un coefficient $1+\rho$ tel que $0 \leq \rho \leq 0.001 \times c_t$, c_t représente la capacité en ressources disponible à la période t ;
Le niveau de stock de sécurité est basé sur une couverture exprimée en nombre périodes. A chaque période, le stock de sécurité est calculé en fonction des demandes futures et de cette couverture. Ainsi, le stock de sécurité est égal à la somme des demandes sur le tronçon de l'horizon qui commence à cette période et qui a la valeur de la couverture comme longueur.

– Classe B : La seconde classe est basée sur la première classe. En effet, nous avons effectué des modifications sur le besoin fixe en ressources (temps de setup). Ils sont augmentés en les multipliant par un coefficient $1+\tau$ tel que $\tau \approx 0.1 \times c_t$.

Les coûts de rupture sur la demande et les coûts de déficit sur le stock de sécurité sont considérés comme des pénalités. Donc, $\varphi_{it} >> \max_{i',t'} \left\{\alpha_{i't'}; \beta_{i't'}; \gamma^{+}_{i't'}; \gamma^{-}_{i't'}\right\}$ et $\gamma^{-}_{it} >> \max_{i',t'} \left\{\alpha_{i't'}; \beta_{i't'}; \gamma^{+}_{i't'}\right\}$. Les coûts φ_{it} et γ^{-}_{it} ont la particularité de décroître dans le temps. En effet, les demandes au début de l'horizon correspondent à des ordres de fabrication réels et non à des prévisions, tandis que les demandes à la fin de l'horizon sont généralement des prévisions et peuvent donc être moins pénalisées. Ces coûts sont générés de la même façon pour toutes les instances décrites précédemment.

5.3.2 Interprétation des résultats

Les sous-problèmes lagrangiens engendrés à chaque étape de la relaxation lagrangienne sont résolus à l'optimum en utilisant le solveur CPLEX 9.0 et le modèle mathématique agrégé présenté précédement (cf. page 93). Ce modèle à donné de meilleures performances que le modèle désagrégé. Cependant, il serait intéressant de faire des tests en utilisant un algorithme de flot à coût minimum dont la complexité est polynomiale.

Nous utilisons la méthode du sous-gradient pour mettre à jour les multiplicateurs lagrangiens. La formule utilisée pour calculer la valeur du déplacement λ_k est la suivante :

$$\lambda_k = \nu_k \frac{\mathcal{W}(\pi^*) - \mathcal{W}(\pi^k)}{\|\Gamma(\pi^k)\|}$$

Les paramètres de la relaxation lagrangienne sont fixés selon le schéma proposé par Held *et al.* [78]. Le pas de déplacement ν égal à 2 pendant $2T$ itérations, puis on divise par 2 la valeur de ν et le nombre d'itérations jusqu'à atteindre une valeur inférieure à q du nombre d'itérations, q est fixée à 5. Enfin, on divise par 2 la valeur de ν toutes les q itérations jusqu'à ce que les λ_k soient suffisamment petits $< \epsilon$ fixé.

La valeur de $\mathcal{W}(\pi^*)$ est remplacée par une estimation par excès $\overline{\mathcal{W}} \geq \mathcal{W}(\pi^*)$. Cette dernière est obtenue en utilisant des heuristiques basées sur la décomposition de l'horizon de

planification décrites au chapitre 6. Les temps d'exécution des heuristiques varient entre 1 seconde et 40 secondes.

L'algorithme de relaxation lagrangienne s'arrête si :

- Un nombre maximal d'itérations Q est atteint. On fixera Q à 50, 100 et 200 ;
- L'écart entre la borne inférieure et la borne supérieure est inférieur à 1%. Celui-ci est mesuré par le paramètre $GAP = \left(\overline{W} - LB\right) / \overline{W}$. LB est la valeur de la meilleure borne inférieure.

Le tableau 5.5 récapitule les résultats expérimentaux de l'application de la relaxation lagrangienne aux classes d'instances A et B. LB représente la valeur de la meilleure borne inférieure obtenue à la fin de la relaxation lagrangienne. $\overline{W}$ représente la valeur de la borne supérieure obtenue à la fin de l'exécution de l'heuristique de décomposition de l'horizon de planification. Le GAP mesure la qualité de la borne inférieure. $Time$ représente le temps d'exécution en secondes et Q le nombre d'itérations maximale de la relaxation lagrangienne.

A partir du tableau 5.5, on remarque qu'il y a une amélioration importante de la borne inférieure entre la 50ème itération et la 100ème itération de la relaxation lagrangienne. En revanche, l'amélioration entre la 100ème itération et la 200ème itération n'est pas très importante et les temps CPU sont élevés. Les graphiques 5.6 et 5.7 représentent l'amélioration du GAP en fonction du nombre d'itérations pour les classes d'instances A et B respectivement.

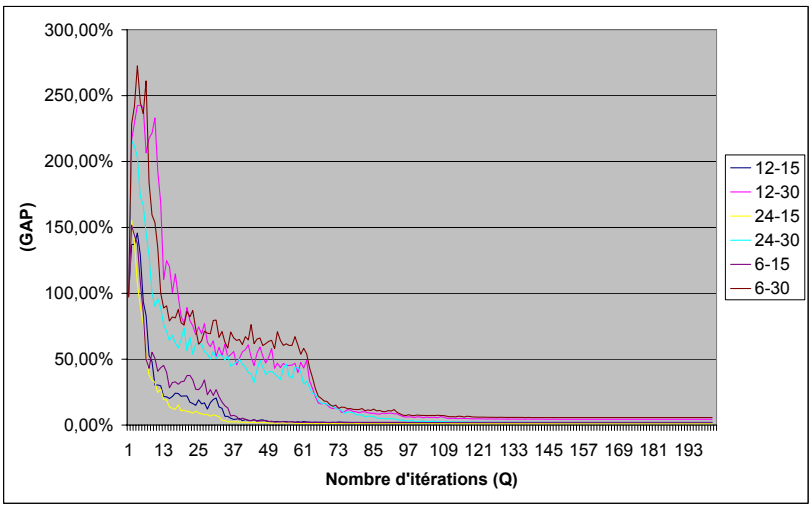


FIG. 5.6 – Relaxation lagrangienne : Classe d'instances A

Nous avons également mesuré la qualité des bornes inférieures obtenues par la relaxation lagrangienne en les comparant aux bornes inférieures obtenues en utilisant un logiciel d'optimisation CPLEX 9.0 pour les modèles monolithiques présentés à la page 93 et 94.

N	T	Q	$Time$	GAP	$\overline{\mathcal{W}}$	LB
Class A						
6	15	50	7,08	2,19%	4664906,20	4562625,44
6	30	50	13,31	58,04%	5455078,94	2288866,54
12	15	50	17,28	2,39%	9025396,59	8809326,78
12	30	50	49,64	43,07%	10563117,63	6013632,98
24	15	50	38,8	1,46%	16712435,78	16468972,69
24	30	50	93,97	32,40%	27275422,81	18437010,69
6	15	100	21,86	1,84%	4664906,20	4579192,26
6	30	100	40,39	7,25%	5455078,94	5059616,08
12	15	100	48,25	2,01%	9025396,59	8843801,44
12	30	100	172,71	5,74%	10563117,63	9956476,96
24	15	100	104,05	1,17%	16712435,78	16516646,54
24	30	100	261,51	2,78%	27275422,81	26517445,40
6	15	200	52,94	1,83%	4664906,20	4579315,51
6	30	200	208,84	5,67%	5455078,94	5145758,47
12	15	200	132,29	2,01%	9025396,59	8844038,91
12	30	200	538,7	4,39%	10563117,63	10099169,22
24	15	200	281,07	1,17%	16712435,78	16516985,35
24	30	200	664,48	1,97%	27275422,81	26739133,84
Class B						
6	15	50	9,28	4,55%	5858168,59	5591338,03
6	30	50	17,33	25,75%	7832065,14	5815403,96
12	15	50	21,45	5,36%	13292194,25	12579536,66
12	30	50	54,69	72,16%	16988946,09	4729878,40
24	15	50	47,19	4,76%	28655048,09	27292258,50
24	30	50	133,97	34,12%	45512020,24	29985269,51
6	15	100	26,7	3,29%	5858168,59	5665290,77
6	30	100	89,3	3,53%	7832065,14	7555426,57
12	15	100	69,34	4,08%	13292194,25	12749866,43
12	30	100	233,99	9,76%	16988946,09	15330246,23
24	15	100	134,49	3,73%	28655048,09	27587051,13
24	30	100	351,55	8,90%	45512020,24	41459351,49
6	15	200	71,27	3,29%	5858168,59	5665712,84
6	30	200	252,62	3,06%	7832065,14	7592239,01
12	15	200	132,29	4,08%	13292194,25	12750327,48
12	30	200	762,42	7,64%	16988946,09	15690378,35
24	15	200	366,74	3,72%	28655048,09	27588094,43
24	30	200	963,44	6,75%	45512020,24	42437756,99

TAB. 5.5 – Résultats expérimentaux : Relaxation Lagrangienne

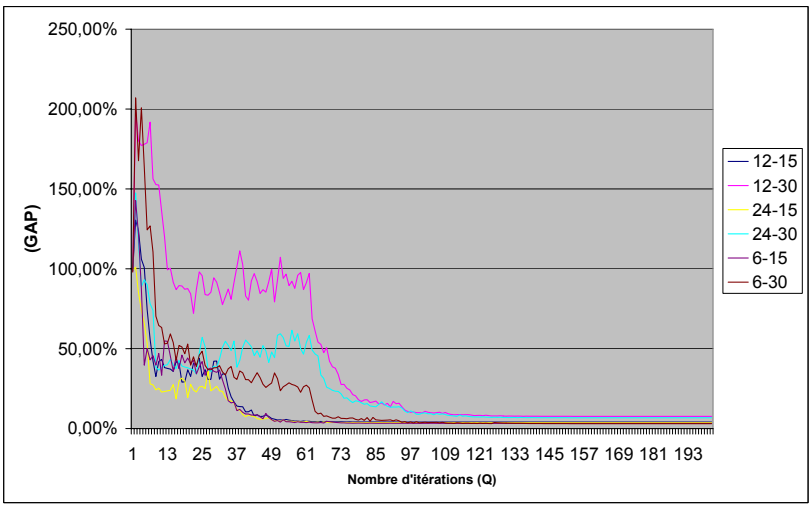


FIG. 5.7 – Relaxation lagrangienne : Classe d'instances B

Nous avons donc comparé la relaxation lagrangienne, notée : RL, aux deux méthodes suivantes :

- **BC**$_{AGG}$: Un algorithme basé sur le branch-and-bound standard de CPLEX 9.0 pour le modèle agrégé présenté à la page 93 ;
- **BC**$_{FL}$: Un algorithme basé sur le branch-and-bound standard de CPLEX 9.0 pour le modèle basé sur le problème de localisation d'entrepôts présenté à la page 94.

Pour les deux algorithmes, LB et UB représentent respectivement la borne inférieure et la borne supérieure à la fin de l'exécution des méthodes BC$_{AGG}$ et BC$_{FL}$. Pour la méthode RL, $UB = \overline{\mathcal{W}}$. Toutes les comparaisons sont effectuées sur les deux critères suivants : le premier est appelé GAP, il est égal à $(UB - LB)/UB$ et le second est le temps CPU noté $Time$. Les algorithmes BC$_{AGG}$ et BC$_{FL}$ s'arrêtent si un temps de résolution maximum de 900 secondes est atteint. Les temps CPU de la méthode RL sont obtenus après 100 itérations. N et T représentent respectivement, le nombre de références et le nombre de périodes. $Method$ représente le nom de la méthode. Le tableau 5.6 récapitule les résultats expérimentaux basés sur le critère d'arrêt présenté précédement.

A partir du tableau 5.6, on remarque que BC$_{AGG}$ donne de meilleurs résultats que BC$_{FL}$. On peut noter également que la relaxation lagrangienne avec 100 itérations donne généralement de meilleures bornes inférieures que BC$_{AGG}$ et BC$_{FL}$; les temps d'exécution sont également nettement meilleurs. En comparant les tableaux 5.6 et 5.5, on remarque que la relaxation lagrangienne avec 200 itérations donne les meilleurs résultats, mais les temps d'exécution sont excessifs.

En analysant le tableau 5.6, on remarque que les GAP fournis par l'utilisation conjointe de la relaxation lagrangienne à 100 itérations et l'heuristique de décomposition de l'horizon

N	T	$Method$	GAP	UB	LB	$Time$
Class A						
6	15	BC_{AGG}	0,75%	4641902,61	4607015,24	900
6	15	BC_{FL}	1,07%	4644016,19	4594442,28	900
6	15	RL	1,84%	4664906,20	4579192,26	21,86
6	30	BC_{AGG}	4,22%	5356554,65	5130361,42	900
6	30	BC_{FL}	9,14%	5458885,85	4960012,32	900
6	30	RL	7,25%	5455078,94	5059616,08	40,39
12	15	BC_{AGG}	1,35%	8915616,64	8795010,18	900
12	15	BC_{FL}	0,80%	8907659,58	8836323,56	900
12	15	RL	2,01%	9025396,59	8843801,44	48,25
12	30	BC_{AGG}	5,52%	10460541,12	9882927,72	900
12	30	BC_{FL}	9,22%	10662839,24	9679202,86	900
12	30	RL	5,74%	10563117,63	9956476,96	172,71
24	15	BC_{AGG}	1,25%	16599925,27	16392410,27	900
24	15	BC_{FL}	0,78%	16583359,31	16454521,41	900
24	15	RL	1,17%	16712435,78	16516646,54	104,05
24	30	BC_{AGG}	2,41%	27020066,09	26369067,12	900
24	30	BC_{FL}	6,75%	27518544,59	25660241,12	900
24	30	RL	2,78%	27275422,81	26517445,40	261,51
Class B						
6	15	BC_{AGG}	1,66%	5825939,77	5728964,48	900
6	15	BC_{FL}	2,63%	5844767,02	5691159,43	900
6	15	RL	3,29%	5858168,59	5665290,77	26,7
6	30	BC_{AGG}	4,01%	7814548,90	7501442,44	900
6	30	BC_{FL}	11,11%	8148507,49	7242896,66	900
6	30	RL	3,53%	7832065,14	7555426,57	89,3
12	15	BC_{AGG}	3,53%	13308806,95	12838374,09	900
12	15	BC_{FL}	4,46%	13380178,34	12782850,45	900
12	15	RL	4,08%	13292194,25	12749866,43	69,34
12	30	BC_{AGG}	14,87%	17044633,82	14510544,67	900
12	30	BC_{FL}	15,47%	18150679,46	15343139,86	900
12	30	RL	9,76%	16988946,09	15330246,23	233,99
24	15	BC_{AGG}	6,38%	28685166,06	26854630,17	900
24	15	BC_{FL}	4,09%	28695501,94	27522257,02	900
24	15	RL	3,73%	28655048,09	27587051,13	134,49
24	30	BC_{AGG}	12,94%	46005650,96	40053010,36	900
24	30	BC_{FL}	7,08%	45515974,88	42291417,13	900
24	30	RL	8,90%	45512020,24	41459351,49	351,55

TAB. 5.6 – Résultats expérimentaux : critère d'arrêt (900 sec)

de planification sont meilleurs que ceux fournis par les méthodes BC_{AGG} et BC_{FL} pour les instances de classes B. En revanche, BC_{AGG} donne de meilleurs GAP pour les instances de classe A. La méthode BC_{FL} donne de meilleurs GAP que la Relaxation lagrangienne pour les instances de classe A à 15 périodes, mais de moins bons GAP sur les instances à 30 périodes.

On remarque également que les instances de classe B sont plus difficiles que les instances de classe A. En effet, les instances de classe B ont un temps de setup fixe beaucoup plus élevé que ceux des instances de classe A.

A partir de cette analyse, nous pouvons conclure que la relaxation lagrangienne donne de bonnes bornes inférieures pour toutes les instances de tests. L'utilisation conjointe de la relaxation lagrangienne et l'heuristique de décomposition de l'horizon de planification donne de bons résultats pour les instances les plus difficiles. L'utilisation du modèle basé sur le problème de localisation d'entrepôts a montré ses limites. Puisqu'en ajoutant des variables de rupture sur la demande et de déficit sur le stock de sécurité, le modèle agrégé donne de meilleurs résultats.

La borne inférieure trouvée par la relaxation lagrangienne peut être utilisée dans une méthode de branch-and-bound. Une bonne borne inférieure permet d'avoir de meilleures bornes supérieures. En effet, une meilleure borne inférieure peut être utilisée pour élaguer les nœuds dominés de l'arbre de recherche et permet à l'algorithme de branch-and-bound de visiter des branches plus intéressante afin de trouver de meilleures solutions.

L'approche que nous avons proposée dans ce chapitre peut être généralisée afin de tenir compte d'autres contraintes industrielles telles que, les groupes de références et le multi-ressources. Les contraintes de lancement de groupes et les contraintes de capacité couplent plusieurs références. Une double relaxation de ces contraintes permet au problème de se décomposer en plusieurs sous-problèmes ULS4. Il serait intéressant de tester la qualité des bornes inférieures obtenues par ce schéma de relaxation lagrangienne. Notre approche peut également être étendue pour le cas multi-gammes. En effet, le but est de généraliser l'algorithme de programmation dynamique que nous avons proposé afin de considérer cette nouvelle contrainte. Les travaux de Toledo et Armentano [173] ont montré que l'algorithme de Wagner et Whitin peut être généralisé pour le problème ULS multi-gammes à ressources séparées.

5.4 Conclusion

Nous avons proposé dans ce chapitre la modélisation d'un nouveau problème de lot-sizing à capacité finie avec des temps de setup, des coûts de rupture sur la demande ainsi que des coûts de déficit sur le stock de sécurité. Afin de fournir une borne inférieure de bonne qualité au problème traité, une approche basée sur la relaxation lagrangienne des contraintes de capacité a été développée. Nous avons également proposé un algorithme polynomial pour résoudre les sous-problèmes induits par cette relaxation. Des résultats expérimentaux ont montré l'efficacité de cette approche. En effet, la relaxation lagrangienne a fourni des bornes inférieures de bonne qualité. Une perspective à ces recherches est d'implémenter un algorithme spécifique de flots afin d'améliorer les temps CPU pour la résolution des problèmes ULS4. Une autre perspective est d'essayer d'améliorer la complexité de l'algorithme de résolution des sous-problèmes induits par la relaxation lagrangienne en utilisant des méthodes géométriques telles utilisées par Wagelmans *et al.* [188]. En effet, des méthodes géométriques peuvent être utilisées pour effectuer une telle amélioration. Les bornes inférieures fournies par la relaxation

lagrangienne peuvent être utilisées dans une méthode de branch-and-bound afin de réduire la taille de l'arbre de recherche.

Une autre perspective à ces recherches est de relaxer d'autres ensembles de contraintes afin de traiter des sous-problèmes plus faciles à résoudre que le problème ULS4. En effet, en effectuant une double relaxation lagrangienne des contraintes de capacité et des contraintes de borne supérieure sur les variables de déficit sur le stock de sécurité, et en ignorant les δ_t, nous obtenons N sous-problèmes de lot-sizing à une seule référence, avec des coûts de ruptures et des coûts de retard (backlog). Nous avons montré à la page 105 que ce problème peut être résolu en utilisant un algorithme de programmation dynamique en $O(T^3)$. Nous pouvons également relaxer les contraintes de capacité et toutes les contraintes de stock de sécurité. Une telle relaxation engendre N sous-problèmes de lot-sizing à une seule référence et des coûts de rupture sur la demande pour lequel Aksen *et al.* [8] ont proposé un algorithme en $O(T^2)$ pour résoudre ce problème.

Le plan de production fourni par la relaxation lagrangienne n'est pas réalisable pour le problème MCLS4, les seules contraintes violées sont les contraintes de capacité. Afin de rendre cette solution réalisable, une heuristique lagrangienne basée sur le lissage et la suppression de productions peut être développée. Celle-ci peut être inspirée des travaux de Trigeiro *et al.* [172].

Nous avons développé dans les chapitres 4 et 5 des approches pour résoudre respectivement des problèmes MCLS2 et MLCS4 étendus. Dans le chapitre suivant, nous présentons une heuristique qui permet de fournir une solution réalisable à des problèmes dont la structure est générique. Cette heuristique est une hybridation d'une méthode de branch-and-bound et d'une approche de décomposition de l'horizon de planification.

Chapitre 6

Heuristiques de décomposition à base de MIP

Introduction

Le problème de lot-sizing à plusieurs références avec des contraintes de capacité et des temps de setup est NP-difficile au sens fort (Chen et Thizy [31]). Trouver une solution réalisable pour les instances réelles de grande taille demeure un challenge. Nous avons présenté aux chapitres précédents des méthodes efficaces qui permettent d'améliorer considérablement les bornes inférieures de quelques problèmes de lot-sizing. Mais dans le cas où toutes les caractéristiques des problèmes industriels des clients DynaSys sont prises en compte, de telles approches sont difficiles à généraliser et à implémenter. Pour plus de détails sur les problèmes traités dans le cadre de mes recherches, le lecteur peut se référer au chapitre 3. Le temps d'exécution long ainsi que la mémoire requise pour une résolution exacte, rendent les algorithmes exacts inutilisables pour des problèmes réels. Afin de trouver une solution réalisable de bonne qualité à des problèmes de lot-sizing de grande taille, les chercheurs se sont intéressés à des heuristiques de décomposition.

Les approches de décomposition fonctionnent généralement sur une ou plusieurs dimensions du problème. En effet, les heuristiques de décomposition pour le problème MCLS sont classées en deux catégories : les heuristiques *période par période* et les heuristiques *référence par référence*. Les heuristiques *période par période* fonctionnent souvent de manière récursive. La solution d'un problème à T périodes est utilisée pour trouver une solution à un problème à $T + 1$ périodes. Une solution réalisable du problème initial est obtenue lorsque toutes les périodes font partie du sous-problème traité. Une telle heuristique doit s'assurer de la faisabilité de la solution des sous-problèmes. De telles approches ont été étudiées par Afentakis [2], Dixon et Silver [51], Eisenhut [56], Lambrecht et Vanderveken [107] et Maes et Van Wassenhove [118]. Les heuristiques *référence par référence* fonctionnent par ensembles de références. A chaque étape, un ensemble de références est planifié jusqu'à ce qu'un plan de production soit obtenu pour toutes les références. Kirca et Kökten [98] ont utilisé une telle méthode pour résoudre le problème MCLS.

Récemment, plusieurs heuristiques dites à base de MIP ont été développées pour les problèmes linéaires mixtes. Fischetti et Lodi [61] ont proposé une méthode appelée *Local Branching*. Cette méthode est un branch-and-bound couplé à une recherche locale. Les voisinages sont obtenus en rajoutant des inégalités appelées *Local branching cuts* durant le par-

cours de l'arbre de recherche. Danna *et al.* [43] ont proposé une approche de recherche locale plus élaborée dans le branch-and-bound. En effet, durant le parcours de l'arbre de recherche, quelques variables sont fixées et un branch-and-bound est lancé sur les variables libres afin de trouver de meilleures solutions dans le voisinage de la solution en cours. D'autres auteurs se sont intéressés à des applications spécifiques des heuristiques à base de MIP aux problèmes de lot-sizing. Parmi ces travaux, nous pouvons citer, Clark [35], Kelly [95], Mercé et Fontan [129], Pochet et Van Vyve [148], Stadtler [164] et Suerie et Stadtler [166].

Dans ce chapitre, nous présentons une méthode basée sur une hybridation d'une méthode de branch-and-bound et une stratégie de décomposition de l'horizon de planification afin de fournir une solution réalisable au problème traité. En effet, le modèle que nous traitons dans ce chapitre, englobe toutes les caractéristiques décrites au chapitre 3. La solution obtenue peut également être utilisée comme borne supérieure dans une méthode de branch-and-bound. Nous présentons également des résultats expérimentaux sur des instances réelles.

Dans la section 6.1, nous rappelons la modélisation mathématique du problème de lot-sizing traité. Une stratégie de décomposition de l'horizon de planification est présentée dans la section 6.2. En effet, l'horizon de planification est partitionné en plusieurs sous-horizons pour lesquels une stratégie de relaxation ou de gel est appliquée. Dans la section 6.3, des heuristiques basées sur cette décomposition de l'horizon de planification sont présentées. Enfin, des résultats expérimentaux sur des instances réelles ainsi qu'une analyse de sensibilité des heuristiques seront présentés en section 6.4.

6.1 Modèle mathématique

Nous rappelons dans ce qui suit la modélisation mathématique du problème de lot-sizing traité dans ce chapitre, noté (P). Il s'agit de déterminer un plan de production d'un ensemble de N références, pour un horizon de planification constitué de T périodes. Ce plan est à capacité finie et doit tenir compte d'un ensemble de contraintes additionnelles. En effet, le lancement de production d'une référence à une période donnée entraîne outre la consommation variable en ressources, une consommation dite fixe (temps de setup).

L'objectif principal est de satisfaire le client, c'est-à-dire de minimiser le coût de *rupture sur la demande*. Un autre objectif du décideur est d'avoir un profil de stock supérieur à un seuil appelé *stock de sécurité*. Il peut arriver que l'on ne puisse pas atteindre ce seuil, dans ce cas on parle d'une situation de *déficit sur le stock de sécurité*. Le second objectif du décideur est donc de minimiser le coût de déficit sur le stock de sécurité. Pour plus de détails sur la modélisation des ruptures sur la demande et du déficit sur le stock de sécurité, nous invitons le lecteur à se reporter aux pages 31 et 36 respectivement.

Les références sont classées par groupes de références. Une référence peut appartenir à un ou plusieurs groupes de références et un groupe de références peut donc contenir une ou plusieurs références. Le lancement de la production d'une référence entraîne le lancement des groupes de références auxquels elle appartient. Ceci se traduit par la prise en compte :

- d'un besoin ou d'une consommation fixe des capacités disponibles dès qu'il y a lancement d'un *groupe* (on note J le nombre de groupes) ;
- d'un coût de lancement pour le groupe.

Pour plus de détails sur la modélisation du problème en multi-groupes, nous invitons le lecteur à se reporter à la page 39.

La production d'une référence peut nécessiter l'utilisation d'une ou plusieurs ressources par période. On note R le nombre de ressources disponibles. La fabrication d'une référence donnée peut se faire de plusieurs façons suivant la gamme choisie. On note V le nombre de gammes disponibles. Pour plus de détails sur la modélisation du problème (P) en multi-ressources et en multi-gammes, le lecteur peut se reporter à la pages 40 et 41.

Le plan de production doit également satisfaire les contraintes de lancement minimum de production. Se reporter à la page 44.

L'objectif de cette modélisation est de générer un plan de production qui minimise la somme des coûts de production, des coûts de lancement, des coûts de stockage ainsi que des coûts de rupture sur la demande et de déficit sur le stock de sécurité, tout en respectant les contraintes de stockage, les contraintes de capacité, ainsi que les contraintes de lancement minimum de production. Nous rappellons dans ce qui suit, les paramètres et les variables du problème (P) :

Les paramètres :
d_{it} : Demande (commandes et prévisions) pour la référence i à la période t.
L_{it} : Stock de sécurité de la référence i à la période t.
c_{rt} : Capacité disponible de la ressource r à la période t.
f_{virt} : Besoin fixe en ressources de la référence i, pour la gamme v, sur la ressource r, à la période t.
v_{virt} : Besoin variable en ressources de la référence i, pour la gamme v, sur la ressource r, à la période t.
g_{vjrt} : Besoin fixe en ressources, du groupe j, pour la gamme v, sur la ressource r, à la période t.
$a_{ij} = \begin{cases} 1 \text{ si la référence } i \text{ appartient au groupe } j. \\ 0 \text{ sinon} \end{cases}$
$l_{vit}^{\min}$: Lancement minimum de production de la référence i pour la gamme v à la période t.
α_{vit} : Coût de production unitaire variable de la référence i, pour la gamme v, à la période t.
β_{vit} : Coût fixe relatif à un lancement de la référence i, pour la gamme v, à la période t.
ω_{vjt} : Coût fixe relatif à un lancement du groupe j, pour la gamme v, à la période t.
γ_{it}^{+} : Coût unitaire de stockage de la référence i à une période t.
γ_{it}^{-} : Coût unitaire de déficit sur le stock de sécurité de la référence i à la période t.
φ_{it} : Coût unitaire de rupture sur la demande pour la référence i à la période t.
On pose : $\delta_{it} = L_{it} - L_{i(t-1)}$.

Les variables :
x_{vit} : Quantité produite pour la référence i à la période t en utilisant la gamme v.
$y_{vit} = \begin{cases} 1 \text{ s'il y a production de la référence } i \text{ à la période } t \text{ sur la gamme } v \text{ (si } x_{vit} > 0). \\ 0 \text{ sinon} \end{cases}$
$z_{vjt} = \begin{cases} 1 \text{ si le groupe } j \text{ est lancé à la période } t \text{ en utilisant la gamme } v. \\ 0 \text{ sinon} \end{cases}$
I_{it}^{+} : Valeur du surstock de la référence i à la fin de la période t (excédent de stock).
I_{it}^{-} : Valeur du déficit sur le stock de sécurité de la référence i à la fin de la période t.

En utilisant ces variables et paramètres, nous formulons le problème (P) comme suit.

$$\min \sum_{v,i,t} \alpha_{vit} x_{vit} + \sum_{v,i,t} \beta_{vit} y_{vit} + \sum_{v,j,t} \omega_{jt} z_{vjt} + \sum_{i,t} \varphi_{it} r_{it} + \sum_{i,t} \gamma^{+}_{it} I^{+}_{it} + \sum_{i,t} \gamma^{-}_{it} I^{-}_{it} \qquad (6.1)$$

s.c :

$$I^{+}_{i(t-1)} - I^{-}_{i(t-1)} + r_{it} + \textstyle\sum_{v=1}^{V} x_{vit} = d_{it} + \delta_{it} + I^{+}_{it} - I^{-}_{it}, \qquad \forall i, \forall t \qquad (6.2)$$

$$\textstyle\sum_{v=1}^{V} \sum_{i=1}^{N} (v_{virt} x_{vit} + f_{virt} y_{vit}) + \sum_{v=1}^{V} \sum_{j=1}^{J} g_{vjrt} z_{vjt} \leq c_{rt}, \qquad \forall r, \forall t \qquad (6.3)$$

$$x_{vit} \leq M y_{vit}, \qquad \forall v, \forall i, \forall t \qquad (6.4)$$

$$x_{vit} \geq l^{\min}_{vit} y_{vit}, \qquad \forall v, \forall i, \forall t \qquad (6.5)$$

$$y_{vit} \geq a_{ij} z_{vjt}, \qquad \forall v, \forall i, \forall j, \forall t \qquad (6.6)$$

$$r_{it} \leq d_{it}, \qquad \forall i, \forall t \qquad (6.7)$$

$$I^{-}_{it} \leq L_{it}, \qquad \forall i, \forall t \qquad (6.8)$$

$$r_{it}, I^{+}_{it}, I^{-}_{it} \geq 0, \qquad \forall i, \forall t \qquad (6.9)$$

$$x_{vit} \geq 0, \qquad \forall v, \forall i, \forall t \qquad (6.10)$$

$$y_{vit} \in \{0, 1\}, \qquad \forall v, \forall i, \forall t \qquad (6.11)$$

$$z_{vjt} \in \{0, 1\}, \qquad \forall v, \forall j, \forall t \qquad (6.12)$$

Ce modèle est une formulation dite *agrégée*. En effet, la production d'une référence n'est définie que par sa période de production, contrairement à la formulation *désagrégée* basée sur le problème de localisation d'entrepôts [101] où la production est définie par sa période de production et par sa période de consommation.

Trouver une bonne solution réalisable au modèle monolithique présenté à la section 6.1 peut s'avérer extrêmement laborieux, surtout si le volume d'information traité est très important. L'utilisation d'une méthode de type branch-and-bound pour résoudre tels problèmes est très courante dans l'industrie qui utilise des outils génériques tels que les logiciels d'APS, mais les temps d'exécution restent exorbitants. Dans ce cas, les méthodes de branch-and-bound ne peuvent être utilisées sans intégrer des heuristiques pour avoir des bornes supérieures de bonne qualité et des solutions réalisables. Il existe d'autres approches de résolution qui consistent à travailler sur des modèles de plus petite taille, pour lesquels il est possible de trouver des solutions proches de l'optimum. L'idée principale des heuristiques que nous présentons est donc de décomposer l'horizon de planification en plusieurs sections comme nous le décrivons par la suite.

6.2 Décomposition de l'horizon de planification

L'approche que nous proposons est une méthode itérative, elle résout à chaque étape un problème linéaire mixte réduit. Pour obtenir un tel problème linéaire mixte, nous réduisons le nombre de variables binaires et le nombre de contraintes. Cette réduction permet de diminuer la taille de l'arbre de recherche dans une méthode de branch-and-bound.

Pour effectuer une telle réduction, nous utilisons une approche de décomposition de l'horizon de planification. Ceci est réalisé en maintenant les contraintes d'intégrité des variables

binaires appartenant à un tronçon de l'horizon fixé, et en relaxant ces contraintes pour les variables n'appartenant pas à ce tronçon.

En fixant toutes les variables de lancement y_{vit} et z_{vjt} ainsi que les variables de production x_{vit} à une période t, on élimine les contraintes (6.3), (6.4), (6.5), (6.10), (6.11) et (6.12) de la même période. Ceci réduit également le nombre de contraintes. A une étape donnée de l'heuristique, les variables sont gelées selon les solutions obtenues aux étapes précédentes de la résolution.

De telles manipulations réduisent considérablement les temps de résolution des sous-problèmes. En effet, l'arbre de recherche est limité à un nombre réduit de variables binaires, et la taille du problème diminue puisque le nombre de contraintes et de variables est restreint.

Notre approche est donc une hybridation d'une méthode de branch-and-bound et d'une stratégie de décomposition de l'horizon de planification. A chaque étape de l'heuristique, l'horizon de planification est décomposé en trois parties (cf. figure 6.1) :

- une fenêtre gelée ;
- une fenêtre de décision ;
- une fenêtre d'approximation.

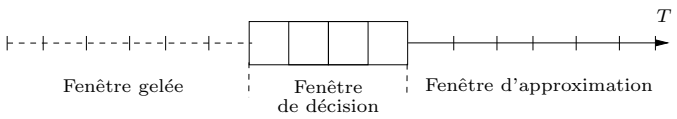


FIG. 6.1 – Décomposition de l'horizon de planification

Avant de décrire chaque fenêtre de temps de la stratégie de décomposition de l'horizon de planification, nous présentons tout d'abord quelques définitions.

Défnition 8. *Les variables de lancement de la production des références y_{vit} et les variables de lancement de la production des groupes de références z_{vjt} sont appelées* ***variables de setup****.*

Défnition 9. *Les variables x_{vit} sont appelées* ***variables de production****.*

Les différentes parties de l'horizon de planification sont décrites dans ce qui suit,

Fenêtre gelée : La fenêtre gelée est une partie de l'horizon où toutes les variables de setup y_{vit} et z_{vjt} sont fixées par des résolutions antérieures dans le cadre d'un processus itératif. Dans cette fenêtre, les variables de production x_{vit} peuvent être également gelées selon la méthode choisie. En effet, nous allons étudier une variante des heuristiques que nous proposons. Cette variante réside dans le gel des variables de production en même temps que les variables de setup.

Fenêtre de décision : La fenêtre de décision est un tronçon de l'horizon où les contraintes d'intégrité sur les variables de setup y_{vit} et z_{vjt} sont maintenues. Dans ce tronçon d'horizon, aucune modification n'est apportée au modèle mathématique (P) réduit à cette fenêtre.

Fenêtre d'approximation : La fenêtre d'approximation est une partie de l'horizon où les contraintes d'intégrité sur les variables de setup y_{vit} et z_{vjt} sont relaxées. En effet, en relaxant ces contraintes d'intégrité, le plan de production fourni par ce tronçon de l'horizon de

planification est approché. Il ne représente qu'une borne inférieure de la capacité nécessaire pour la satisfaction de la demande sur ce tronçon d'horizon.

L'idée de la fenêtre d'approximation est justifiée par des propriétés industrielles. En effet, les demandes du début de l'horizon de planification sont des commandes fermes sur des périodes courtes, e.g. jour ou semaine ; les décisions prises à ces périodes sont des ordres de fabrication qui vont servir de base à un ordonnancement au niveau atelier. En revanche, les demandes à la fin de l'horizon sont des prévisions sur des périodes plus longues, e.g. mois ou trimestre ; et ne sont pas relatives à des commandes passées par les clients, elles peuvent changer quand une nouvelle planification de la production est effectuée. L'impact de la relaxation des contraintes d'intégrité sur les variables de setup à la fin de l'horizon n'est donc pas aussi important que la relaxation de ces contraintes au début de l'horizon.

En se basant sur cette approche de décomposition, nous proposons deux heuristiques à base de MIP. Une première heuristique que nous dénommons Fix-and-Relax, et une seconde heuristique que nous dénommons Double-Fix-and-Relax. Nous présentons également deux variantes des deux heuristiques qui diffèrent dans la manière de geler les variables de production. Nous décrivons dans la section qui suit le principe des deux heuristiques.

6.3 Approches de résolution

Dans cette section, nous présentons une heuristique à base de MIP pour trouver une solution réalisable au modèle décrit à la section 6.1 en traitant des problèmes de plus petite taille, du point de vue du volume d'information et de la combinatoire. Ceci est réalisé en utilisant l'approche de décomposition de l'horizon de planification présentée à la section 6.2. La planification de la production se fait suivant plusieurs étapes successives durant lesquelles la fenêtre de décision est déplacée à travers l'horizon. La figure 6.2 schématise le principe du déplacement de la fenêtre de décision entre deux étapes k et $k+1$ de l'heuristique. Le principe des deux heuristiques est le même, la seule différence réside dans la manière de geler les variables de setup et les variables de production. Nous allons présenter dans ce qui suit, le principe des deux heuristiques Fix-and-Relax et Double-Fix-and-Relax.

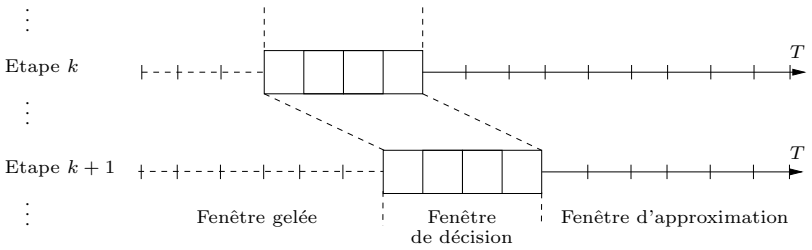


FIG. 6.2 – Principe de l'heuristique de décomposition

6.3.1 Fix-and-Relax

Le principe de l'heuristique Fix-and-Relax est de commencer par fixer une fenêtre de décision au début de l'horizon et une fenêtre d'approximation sur le reste de l'horizon. Ainsi, comme nous l'avons décrit à la section 6.2, des contraintes du problème (P) sont relaxées. Le

problème ainsi formulé est résolu en utilisant un algorithme de branch-and-bound. Dans une seconde étape, la fenêtre de décision est déplacée à travers l'horizon de planification (vers le futur) tout en gardant un chevauchement avec la fenêtre de décision précédente. A l'étape k de l'heuristique, les variables de setup appartenant au tronçon d'horizon qui précède la nouvelle fenêtre de décision sont gelées selon la solution obtenue à l'étape précédente de l'heuristique $k-1$. Le problème ainsi construit est résolu en utilisant un algorithme de branch-and-bound. Cette procédure est répétée jusqu'à atteindre la fin de l'horizon de planification. Le plan fourni par cette procédure est réalisable. En effet, la méthode Fix-and-Relax respecte toutes les contraintes d'intégrité.

L'heuristique Fix-and-Relax est contrôlée par quatre paramètres, deux paramètres portent sur la décomposition de l'horizon de planification et deux autres paramètres concernent le critère d'arrêt de la méthode de branch-and-bound. Les quatre paramètres sont décrits dans ce qui suit :

- σ_k : Taille de la fenêtre de décision à l'étape k de l'algorithme ;
- δ_k : Nombre de périodes chevauchantes entre les fenêtres de décision à la résolution $k-1$ et k, avec $\sigma_k > \delta_k \geq 0$;
- opt_k : Pourcentage d'optimalité minimum pour la résolution du sous-problème de l'étape k de l'algorithme ;
- $Time_k$: Le temps maximum de résolution à l'étape k de l'algorithme.

Avant de décrire l'algorithme, nous introduisons quelques notations. On note $P^k_{a_k b_k}$ le problème (P) résolu à l'étape k de l'heuristique avec $k \geq 1$. $[a_k, b_k]$ est la fenêtre de décision de l'étape k. Par convention, $[1, 0] = \emptyset$. Nous rappelons les propriétés suivantes :

- Les contraintes d'intégrité sur les variables de setup y_{vit} et z_{vjt} sont maintenues dans la fenêtre de décision définie par le tronçon d'horizon $[a_k, b_k]$;
- Les contraintes d'intégrité sur les variables de setup y_{vit} et z_{vjt} sont relaxées dans la fenêtre d'approximation définie par le tronçon d'horizon $[b_k + 1, T]$;
- Les variables de setup y_{vit} et z_{vjt} sont gelées dans le reste de l'horizon ($[1, a_k - 1]$) selon la solution obtenue à l'étape $k-1$ de l'heuristique.

K représente le nombre d'étapes maximum de la méthode Fix-and-Relax. Si σ_k et δ_k sont constants pour tout k (i.e. $\sigma_k = \sigma$ et $\delta_k = \delta$), alors $K = \lceil (T - \sigma)/(\sigma - \delta) \rceil + 1$. On peut noter que K est borné par T dans le pire cas.

Le principe de la méthode Fix-and-Relax est schématisé à la figure 6.3.

Le chevauchement entre la fenêtre de décision d'une étape k de l'algorithme et de l'étape qui la précède $k-1$ permet de remettre en cause les décisions prises à l'étape précédente. Cette remise en cause permet de proposer un meilleur plan de production, puisque la fenêtre d'approximation ne donne qu'une estimation par défaut de la capacité en ressources nécessaire. Une sous-estimation de la capacité nécessaire peut provoquer des ruptures sur la demande une fois les contraintes d'intégrité prises en compte. Or, en rajoutant ce chevauchement, une anticipation supplémentaire peut être faite afin d'éviter un manque de capacité éventuel et provoquer ainsi moins de ruptures. Nous allons montrer à travers l'exemple 6.3.1 une situation pour laquelle il est intéressant d'avoir des chevauchements entre les fenêtres de décision d'une itération k à une itération $k+1$.

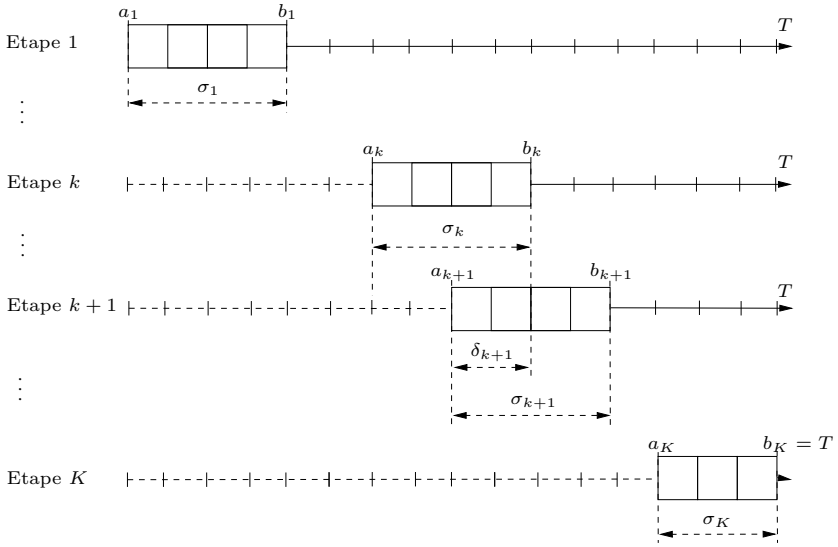


FIG. 6.3 – Méthode Fix-and-Relax

Nous pouvons observer qu'il n'existe pas de fenêtre gelée à la première étape de l'algorithme. Les sous-problèmes $P^k_{a_k b_k}$ sont résolus en utilisant un algorithme de branch-and-bound. L'algorithme Fix-and-Relax est décrit par l'algorithme 6 ci-dessous :

Algorithme 6 Algorithme Fix-and-Relax

1: $k \leftarrow 1$, $a_k \leftarrow 1$, $b_k \leftarrow \sigma_1$
2: **tant que** $b_k < T$ **faire**
3: Résoudre le problème $P^k_{a_k b_k}$
4: $k \leftarrow k+1$, $a_k \leftarrow b_k - \delta_k$, $b_k \leftarrow b_k + \sigma_k - \delta_k$
5: **si** $b_k > T$ **alors**
6: $b_k \leftarrow T$
7: **finsi**
8: **fin tant que**
9: Résoudre le problème $P^k_{a_k b_k}$

La solution approchée obtenue à la fin de l'algorithme 6 est la solution réalisable du problème (P) fournie par la méthode Fix-and-Relax.

Exemple 6.3.1. *Nous montrons à travers cet exemple une situation où il est important d'avoir des périodes chevauchantes entre les fenêtres de décision de périodes successives. Nous considérons le problème (P) avec des dimensions, des coûts et des données définies respectivement dans les tableaux 6.1, 6.2 et 6.3.*

V	N	G	R	T
1	1	1	1	5

TAB. 6.1 – Dimensions du problème (P)

t	1	2	3	4	5
φ_{it}	250	212,5	175	137,5	100
β_{vit}	20000	20000	20000	20000	30000
α_{vit}	10	10	10	10	10
γ_{it}^{+}	1	1	1	1	1

TAB. 6.2 – Coûts du problème (P)

t	1	2	3	4	5
d_{1t}	800	1000	0	1000	1000
f_{11t}	500	500	500	500	500
v_{11t}	1	1	1	1	1
c_{1t}	1500	1500	1500	2000	2000

TAB. 6.3 – Données du problème (P)

On exécute deux fois l'heuristique Fix-and-Relax avec $\sigma_k = 3$ pour tout k. La première exécution est paramétrée avec $\delta_k = 0$ pour tout k, tandis que la seconde est paramétrée avec $\delta_k = 1$ pour tout k. Les sous-problèmes $P^k_{a_k b_k c_k}$ sont résolus à l'optimum en utilisant la modélisation mathématique présentée précédement et le solveur de programmation linéaire en nombres entiers CPLEX 9.0. Les résultats sont présentés dans les tableaux 6.4 et 6.5.

t	1	2	3	4	5
x_{11t}	1000	1000	0	1500	0
y_{11t}	1	1	0	1	0
s_{1t}	200	200	200	700	0
r_{1t}	0	0	0	0	300

TAB. 6.4 – Résultats de l'heuristique Fix-and-Relax avec $\delta_k = 0$ pour tout k

On compare les résultats des deux exécutions en se basant sur les tableaux 6.4 et 6.5. On obtient avec le premier paramètrage, une rupture de 300 à la dernière période, tandis qu'avec le second, il n'y a pas de rupture. L'importance d'avoir un chevauchement entre les fenêtres de décision réside alors dans la possibilité de remettre en cause des décisions déjà prises. En effet, celles-ci sont biaisées par la relaxation des contraintes d'intégrité sur le tronçon de l'horizon qui suit la fenêtre de décision. Le chevauchement peut ainsi permettre une anticipation supplémentaire de la production.

6.3.2 Double-Fix-and-Relax

Le principe de l'heuristique Double-Fix-and-Relax est proche de celui de la méthode Fix-and-Relax. La seule différence réside dans le fait qu'initialement on fixe une fenêtre d'approximation de la taille de l'horizon de planification. Le problème ainsi formulé est résolu en utilisant la méthode du simplexe, puisque toutes les variables sont continues.

Dans une seconde étape, on fixe trois fenêtres : une fenêtre de décision au début de l'horizon de planification, une fenêtre d'approximation juste après la fenêtre de décision, et une fenêtre gelée sur toutes les périodes qui suivent la fenêtre d'approximation. Dans la fenêtre

t	1	2	3	4	5
x_{11t}	800	1000	500	1500	0
y_{11t}	1	1	1	1	0
s_{1t}	0	0	500	1000	0
r_{1t}	0	0	0	0	0

TAB. 6.5 – Résultats de l'heuristique Fix-and-Relax avec $\delta_k = 1$ pour tout k

gelée, les variables de setup ainsi que les variables de production sont fixées selon les résultats trouvés à la première étape. Le problème ainsi formulé est résolu en utilisant un algorithme de branch-and-bound.

Comme étapes suivantes, la fenêtre de décision, ainsi que la fenêtre d'approximation sont déplacées à travers l'horizon de planification (vers le futur) tout en gardant un chevauchement entre la fenêtre de décision de l'étape courante k et celle de l'étape précédente $k-1$. Les variables de setup appartenant au tronçon d'horizon qui précède la fenêtre de décision de l'étape k sont gelées selon la solution obtenue à l'étape précédente de l'algorithme $k-1$. Le problème ainsi construit est résolu en utilisant un algorithme de branch-and-bound. Cette procédure est répétée jusqu'à atteindre la fin de l'horizon de planification.

Le plan de production fourni à la fin de cette heuristique est réalisable. En effet, la méthode Double-Fix-and-Relax respecte toutes les contraintes d'intégrité.

Nous avons expliqué dans la section 6.2 page 122 la motivation d'avoir une fenêtre d'approximation à la fin de l'horizon de planification. L'idée de l'heuristique Double-Fix-and-Relax est de commencer par chercher une estimation par défaut du plan de production sur la totalité de l'horizon de planification en relaxant toutes les contraintes d'intégrité sur les variables de setup. Un tel plan capture l'interaction (si elle existe) entre les périodes du début et celles de la fin de l'horizon de planification. Une fois cette interaction estimée, le plan de production à la fin de l'horizon de planification est gelé afin de réduire la taille des problèmes traités. La production de ces périodes est dégelée au fur et à mesure de l'avancement de la fenêtre de décision. A chaque étape, la fenêtre de décision est suivie par une fenêtre d'approximation. La taille de la fenêtre d'approximation doit être suffisamment grande afin de corriger la première estimation par défaut et d'effectuer les anticipations nécessaires.

L'heuristique Double-Fix-and-Relax est contrôlée par cinq paramètres, trois paramètres portent sur la décomposition de l'horizon de planification et deux autres paramètres concernent le critère d'arrêt de la méthode de branch-and-bound. Les cinq paramètres sont décrits dans ce qui suit :

- σ_k : Taille de la fenêtre de décision à l'étape k de l'algorithme ;
- δ_k : Nombre de périodes chevauchantes entre les fenêtres de décision à la résolution $k-1$ et k ;
- ρ_k : Taille de la fenêtre d'approximation à l'étape k de l'algorithme ;
- opt_k : Pourcentage d'optimalité minimum pour la résolution du sous-problème de l'étape k de l'algorithme ;
- $Time_k$: Le temps maximum de résolution à l'étape k.

On note P^0 le problème (P) dans lequel toutes les contraintes d'intégrité sont relaxées. On note $P^k_{a_k b_k c_k}$ le problème résolu à chaque étape k de l'heuristique avec $k \geq 1$. par convention, si $\alpha > \beta$ pour la section $[\alpha, \beta]$ alors $[\alpha, \beta] = \emptyset$. Nous rappelons les caractéristique du problème $P^k_{a_k b_k c_k}$:

- Les contraintes d'intégrité sur les variables de setup y_{vit} et z_{vjt} sont maintenues dans la fenêtre de décision définie par le tronçon d'horizon $[a_k, b_k]$;
- Les contraintes d'intégrité sur les variables de setup y_{vit} et z_{vjt} sont relaxées dans la fenêtre d'approximation définie par le tronçon d'horizon $[b_k + 1, c_k]$;
- Les variables de setup y_{vit} et z_{vjt} sont gelées dans le tronçon d'horizon $[1, a_k - 1]$ selon la solution obtenue à l'étape $k - 1$ de l'heuristique ;
- Les variables de setup y_{vit} et z_{vjt} et les variables de prodcution x_{vit} sont gelées dans le tronçon d'horizon $[c_k + 1, T]$ selon la solution obtenue à l'étape $k - 1$ de l'heuristique.

K représente le nombre d'itération maximum de la méthode Double-Fix-and-Relax. On rappelle que K est en $O(T)$.

Le principe de la méthode Double-Fix-and-Relax est schématisé par la figure 6.4.

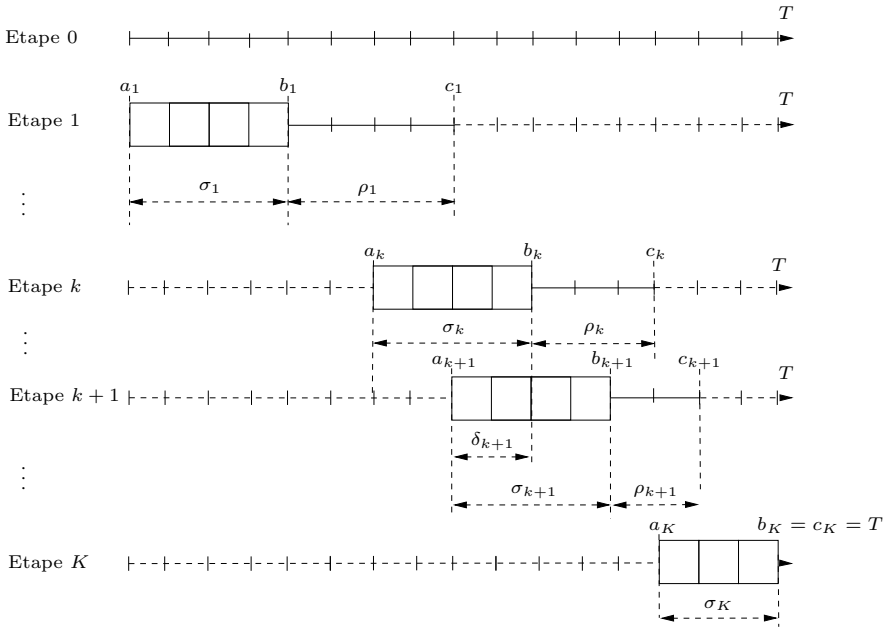


FIG. 6.4 – Méthode Double-Fix-and-Relax

Les sous-problèmes $P^k_{a_k b_k c_k}$ sont résolus en utilisant un algorithme de branch-and-bound. L'algorithme Double-Fix-and-Relax est décrit par l'algorithme 7 ci-dessous :

La solution approchée obtenue à la fin de l'algorithme 7 est la solution réalisable du problème (P) fournie par la méthode Double-Fix-and-Relax.

Algorithme 7 Algorithme Double-Fix-and-Relax

Résoudre le problème P^0
$k \leftarrow 1$, $a_k \leftarrow 1$, $b_k \leftarrow \sigma_1$, $c_k \leftarrow b + \rho_k$
tant que $b < T$ **faire**
  Résoudre le problème $P^k_{a_k b_k c_k}$
  $k \leftarrow k + 1$, $a_k \leftarrow b_k - \delta_k$, $b_k \leftarrow b_k + \sigma_k - \delta_k$, $c_k \leftarrow b_k + \rho_k$,
  si $c_k > T$ **alors**
    $c_k \leftarrow T$
  finsi
  si $b_k > T$ **alors**
    $b_k \leftarrow T$
  finsi
fin tant que
Résoudre le problème $P^k_{a_k b_k c_k}$

Une variante des deux méthodes présentées précédemment (voir section 6.3.1 et 6.3.2) est de geler les variables de production en même temps que les variables de setup. Une telle procédure réduit de façon considérable la taille des problèmes résolus à chaque étape de l'heuristique. En effet, en gelant les variables de production et les variables de setup, le nombre de contraintes est réduit considérablement, et par conséquent le temps de résolution l'est à chaque étape.

En fixant toutes les variables de setup y_{vit} et z_{vjt} ainsi que les variables de production x_{vit} dans la fenêtre gelée, on élimine du problème (P) toutes les contraintes (6.3), (6.4), (6.5), (6.10), (6.11) et (6.12) ayant des indices temporels dans la fenêtre gelée. En revanche, quand on ne gèle que les variables de setup on n'élimine du problème (P) que les contraintes (6.10), (6.11) et (6.12).

Un désavantage d'une telle variante est la réduction des anticipations sur la production. En effet, quand la production est gelée, aucune anticipation n'est possible. En revanche, si les variables de production sont libres, alors des anticipations supplémentaires sont possibles.

6.3.3 Eléments de comparaison

Un avantage de la méthode Double-Fix-and-Relax par rapport à la méthode Fix-and-Relax est de traiter moins de variables et moins de contraintes à chaque étape. En effet, le gel à la fin de l'horizon réduit la taille des sous-problèmes traités à chaque étape. Maintenir toutes les contraintes d'intégrité du problème (P) entraîne un modèle pour lequel il est très difficile de trouver une solution réalisable en utilisant une méthode de branch-and-bound.

Le principe des méthodes Fix-and-Relax et Double-Fix-and-Relax est de résoudre successivement plusieurs problèmes de type $P^k_{a_k b_k}$ et $P^k_{a_k b_k c_k}$ respectivement, ayant des périodes de chevauchement entre deux fenêtres de décision successives. Si on assume que : $\sigma = \sigma_k$ et $\delta = \delta_k$ pour tout k, alors au lieu de résoudre le problème (P) avec $VT(N + J)$ variables binaires et $(3N + R + 2VN + VJN)T$ contraintes, les méthode Fix-and-Relax et Double-Fix-and-Relax résolvent au maximum $(\lceil (T - \sigma)/(\sigma - \delta) \rceil + 1)$ problèmes avec $V(N + J)\sigma$ variables binaires.

6.4 Résultats expérimentaux

Dans cette partie, nous présentons des résultats expérimentaux issus de l'application des deux heuristiques au problème (P). Les solutions obtenues à la fin des deux heuristiques sont comparées à celles obtenues à la fin d'un algorithme de branch-and-bound. Nous présentons également une analyse de sensibilité concernant les paramètres des deux méthodes.

Nos algorithmes sont implémentés en langage C++ et ils sont intégrés à un logiciel d'APS, n.SKEP. Nous utilisons la librairie de programmation linéaire mixte CPLEX 9.0 [110] pour la résolution des problèmes linéaires mixtes. Tous les tests ont été effectués sur un ordinateur personnel équipé d'un microprocesseur Pentium IV 2.66 GHz.

6.4.1 Algorithmes et implémentation

Afin d'évaluer l'efficacité de nos algorithmes, nous comparons les deux méthodes Fix-and-Relax et Double-Fix-and-Relax à une résolution monolithique du problème initial (P). Le problème (P) ainsi que les sous-problèmes engendrés à chaque étape de l'heuristique sont résolus en utilisant le solveur de programmation linéaire en nombres entiers CPLEX 9.0 qui fournit un algorithme de branch-and-bound. Par ailleurs, nous pouvons utiliser avec CPLEX 9.0 plusieurs coupes standard afin de renforcer les relaxations linéaires continues à chaque nœud de l'arbre de recherche. Une autre variante des deux heuristiques est également étudiée, cette variante consiste à fixer les variables de production en même temps que les variables de setup.

Nous effectuons donc une étude comparative entre les cinq méthodes suivantes :

- **FR :** Un algorithme basé sur la méthode Fix-and-Relax et le branch-and-cut standard de CPLEX 9.0 pour résoudre les sous-problèmes $P^k_{a_k b_k}$;
- **FR1 :** Un algorithme basé sur la méthode Fix-and-Relax et le branch-and-cut standard de CPLEX 9.0 pour résoudre les sous-problèmes $P^k_{a_k b_k}$. Les variables de production sont fixées en même temps que les variables de setup ;
- **DFR :** Un algorithme basé sur la méthode Double-Fix-and-Relax et le branch-and-cut standard de CPLEX 9.0 pour résoudre les sous-problèmes $P^k_{a_k b_k c_k}$;
- **DFR1 :** Un algorithme basé sur la méthode Double-Fix-and-Relax et le branch-and-cut standard de CPLEX 9.0 pour résoudre les sous-problèmes $P^k_{a_k b_k c_k}$. Les variables de production sont fixées en même temps que les variables de setup ;
- **BC :** Un algorithme basé sur le branch-and-cut standard de CPLEX 9.0 pour une résolution monolithique du problème (P).

Pour les cinq algorithmes, nous utilisons le modèle agrégé défini à la section 6.1 par l'ensemble de contraintes (6.2)-(6.11).

6.4.2 Instances de test

Nous avons effectué des tests sur une série d'instances issues d'une situation réelle. En effet, les heuristiques que nous proposons dans ce chapitre sont implémentées dans le logiciel d'APS n.SKEP et sont opérationnelles chez des clients de DynaSys. Les instances traitées sont issues d'une problématique industrielle d'un fabriquant de pâtes. En effet, le fabriquant

dispose de plusieurs ressources et de plusieurs lignes de production alternatives, les références sont classées par familles de références et par poids. Il est également soumis à des contraintes de production minimale et des contraintes de lancement minimum de production. L'objectif principal du décideur est de minimiser les ruptures sur la demande ainsi que les déficits sur le stock de sécurité. Ces instances sont caractérisées par un grand nombre de groupes de références et par un besoin fixe en ressources (temps de setup) important. Les coûts de rupture sur la demande et les coûts de déficit sur le stock de sécurité sont considérés comme des pénalités. Ils ont la particularité de décroître dans le temps. En effet, les demandes au début de l'horizon correspondent à des ordres de fabrication réels et non à des prévisions, tandis que les demandes à la fin de l'horizon sont généralement des prévisions et peuvent donc être moins pénalisées. Les caractéristiques des instances traitées sont décrites dans le tableau 6.6.

Inst.	V	N	J	R	T	L_{min}	P_{min}
I1	3	264	144	21	19	Oui	Oui
I2	3	469	215	20	19	Oui	Oui
I3	3	292	144	20	30	Oui	Oui
I4	3	514	215	20	30	Oui	Oui

TAB. 6.6 – Instances de test

Les instances I1 et I2 sont caractérisées par des ressources qui ne sont pas très limitantes. L'instance I2 est caractérisée par des besoins fixes très élevés. Les instances I3 et I4 sont les plus difficiles car elles sont de plus grande taille, possèdent des contraintes de lancement minimum ainsi que des valeurs de besoins fixes en ressources élevées. Elle sont caractérisées également par des ressources très limitantes.

Afin d'évaluer l'efficacité des heuristiques, nous avons également effectué des tests sur les classes d'instances A et B du chapitre 5. Pour plus de détails sur ces instances, le lecteur peut se reporter à la page 108.

6.4.3 Résultats

Afin de comparer les algorithmes implémentés, nous utilisons les critères suivants : le premier est appelé GAP il est égal à $(UB - LB)/UB$, où UB et LB représentent respectivement les meilleures bornes supérieure et inférieure trouvées au cours de la recherche arborescente. Le second critère est le temps CPU noté Temps CPU. $\%Lost$ et $\%Fail$ représentent respectivement le pourcentage de rupture sur la demande et le pourcentage de déficit sur le stock de sécurité.

Les heuristiques que nous proposons ne fournissent qu'une borne supérieure au problème (P). Afin d'évaluer le pourcentage d'optimalité de ces deux méthodes, nous utilisons la borne inférieure trouvée à la fin de la méthode BC.

Deux types de coupes spécialisées dans les problèmes linéaires mixtes sont activés pour toutes les méthodes. Les coupes flow cover et les coupes Mixed Integer Rounding (MIR) du solveur CPLEX 9.0. Les coupes flow cover fonctionnent bien pour les problèmes de lot-sizing du fait de la structure de flots induite par les contraintes de flux (6.2). Pour une description des coupes flow cover, le lecteur peut se référer à Gomory [70], Nemhauser et Wolsey [138]

et Wolsey [194]. Les coupes MIR sont principalement appliquées aux contraintes de capacité et aux contraintes de production maximum. Pour plus de détails sur les inégalités MIR, le lecteur peut se référer à Padberg *et al.* [156] et Van Roy et Wolsey [179].

La stratégie de branchement dans la méthode de branch-and-bound est une recherche *en profondeur d'abord* afin de trouver une solution réalisable. Une borne supérieure est obtenue quand la solution est entière ou par l'heuristique de programmation linéaire du solveur.

Les méthodes Fix-and-Relax et Double-Fix-and-Relax sont contrôlées par plusieurs paramètres. Des tests empiriques ont montré que les paramètres suivants donnaient en général de bons résultats. Pour les méthodes de type Fix-and-Relax, on pose : $\sigma_k = 3$ pour tout k, $\delta_k = 1$ pour tout $k > 1$, $opt_k = 5\%$ pour tout k et un temps CPU limite de 300 secondes pour chaque étape. A chaque étape de l'heuristique k, l'algorithme de branch-and-bound s'arrête si la valeur $(UB_k - LB_k)/UB_k$ est inférieure opt_k ou si un temps CPU limite est atteint. LB_k et UB_k représentent respectivement la valeur de la borne inférieure et la valeur de la borne supérieure à la fin de l'algorithme de branch-and-bound de l'étape k de l'heuristique.

En plus de ces paramètres, nous définissons à chaque étape k de la méthode Double-Fix-and-Relax, $\rho_k = 4$ pour tout k. La méthode BC est contrôlée par le pourcentage d'optimalité minimum de 5% et un temps CPU limite de 3600 secondes. L'algorithme de branch-and-bound s'arrête si l'un des critères d'arrêt est atteint.

Le tableau 6.7 récapitule les résultats expérimentaux quand un temps CPU limite et un GAP minimum sont utilisés comme critère d'arrêt pour tous les algorithmes afin de résoudre les instances I1, I2, I3 et I4.

Method	*Temps CPU*	*GAP*	*UB*	*LB*	*%Lost*	*%Fail*
Instance I1						
BC	3600	25,44%	616057049,33	459347279,99	0,00%	0,86%
FR	1133	24,04%	604747858,07	459347279,99	0,01%	0,95%
FR1	1224	29,68%	653254932,91	459347279,99	0,03%	0,97%
DFR	365	27,62%	634629402,32	459347279,99	0,00%	0,88%
DFR1	521	30,99%	665577091,20	459347279,99	0,01%	1,12%
Instance I2						
BC	3600	8,41%	423228776,24	387628214,43	0,03%	0,91%
FR	200	15,80%	460348758,04	387628214,43	0,03%	1,50%
FR1	303	22,86%	502473221,20	387628214,43	0,04%	2,10%
DFR	135	11,64%	438691571,36	387628214,43	0,03%	1,49%
DFR1	186	14,49%	453295802,18	387628214,43	0,04%	1,30%
Instance I3						
BC	3600	72,98%	4422837201,10	1195061357,10	1,07%	10,15%
FR	3480	80,73%	6200212481,48	1195061357,10	1,32%	17,90%
FR1	3662	24,29%	1578441356,50	1195061357,10	0,16%	6,66%
DFR	629	24,44%	1581640954,47	1195061357,10	0,16%	8,10%
DFR1	1020	27,41%	1646320778,18	1195061357,10	0,19%	6,72%
Instance I4						
BC	3600	12,47%	6765340128,67	5921842374,92	3,04%	4,22%
FR	1770	22,52%	7642976846,79	5921842374,92	3,38%	6,62%
FR1	1567	22,78%	7668991587,14	5921842374,92	3,15%	6,29%
DFR	483	17,55%	7182226467,62	5921842374,92	3,32%	5,60%
DFR1	1020	28,12%	8238379889,67	5921842374,92	3,67%	5,98%

TAB. 6.7 – Résultats expérimentaux : Instances I1, I2, I3 et I4

D'après le tableau 6.7, il est clair que la méthode DFR est meilleure que les méthodes FR, FR1 et DFR1. En effet, le pourcentage d'optimalité, le temps CPU et le pourcentage de

rupture et de déficit sur le stock de sécurité sont meilleurs. On peut noter que DFR donne souvent des GAPs équivalents à ceux de BC pour les instances I1, I2 et I4. En revanche, les temps CPU sont nettement meilleurs. DFR et BC ont des pourcentages de rupture équivalents pour les instances I1, I2 et I4, mais BC donne moins de déficits sur le stock de sécurité. Pour l'instance I3, la méthode BC n'a pas réussi a trouver une bonne solution réalisable après 3600 secondes. La méthode DFR à trouvé une meilleure solution réalisable après 629 secondes, FR1 a trouvé une solution réalisable équivalente après 3662 secondes. Pour l'instance I1, FR a trouvé la meilleure solution réalisable mais au bout de 1224 secondes. A partir du tableau 6.7, nous pouvons conclure que la méthode DFR donne les meilleurs résultats rapport temps CPU, qualité de la solution.

Nous avons également effectué des tests sur les instances de classe A et de classe B. Pour les méthodes Fix-and-Relax, on pose : $\sigma_k = 3$ pour tout k, $\delta_k = 1$ pour tout $k > 1$, $opt_k = 1\%$ pour tout k et un temps CPU limite de 60 secondes pour chaque étape. En plus de ces paramètres, nous définissons à chaque k de la méthode Double-Fix-and-Relax, $\rho_k = 4$ pour tout k. La méthode BC est contrôlée par le pourcentage d'optimalité minimum de 1% et un temps CPU limite de 200 secondes. L'algorithme de branch-and-bound s'arrête si l'un des critères d'arrêt est atteint. N est le nombre de références et T le nombre de périodes.
Le tableau 6.8 résume les résultats expérimentaux quand un temps CPU limite et un GAP minimum sont utilisés comme critère d'arrêt pour les instances de class A et B.

A partir du tableau 6.8, on peut remarque que les problèmes de classe A sont beaucoup plus difficiles que les problèmes de classe B d'après les temps CPU et les GAP obtenus. En effet, les problèmes de classe B ont des besoins fixes (temps de setup) beaucoup plus grands que ceux des problèmes de classe A.

Les méthodes DFR et FR sont meilleures que les méthodes FR1 et DFR1 en analysant tous les critères. On peut également remarquer que les méthodes DFR et FR fournissent des GAPs proches de ceux fournis par la méthode BC pour les instances de classe A avec des temps CPU insignifiant. Les deux méthodes donnent de meilleurs GAPs que ceux obtenus par BC sur les instances de classe B avec des temps CPU sensiblement inférieurs. On peut facilement noter que geler le début de l'horizon implique des temps CPU et des GAPs plus élevés. DFR fourni de meilleurs GAPs et temps CPU que FR pour les instances de classe A, en revanche FR donne de meilleurs GAPs et temps CPU que DFR pour les instances de classe B. Généralement, FR a moins de déficit sur le stock de sécurité que DFR. Les méthodes DFR et FR fournissent les meilleurs résultats, les méthodes FR1 et DFR1 fournissent les plus mauvais résultats.

D'après les tableaux 6.7 et 6.8, nous pouvons conclue que geler la fin de l'horizon de planification aide à trouver de bons résultats. En revanche, ajouter une fenêtre de gel au début de l'horizon de planification ne permet d'avoir une amélioration significative de la qualité de la solution.

6.4.4 Analyse de sensibilité

La performance des méthodes Fix-and-Relax et Double-Fix-and-Relax dépend du choix des paramètres. Afin d'étudier ces méthodes en détails, nous avons analysé l'influence de la variation de ces paramètres sur les temps d'exécution et sur la qualité des solutions obtenues

N	T	$Method$	$Temps\ CPU$	GAP	UB	LB	$\%Lost$	$\%Deficit$
Class A								
6	15	BC	200	1,18%	4649063,73	4594355,62	27,05%	68,06%
6	15	FR	0,59	1,51%	4664906,20	4594355,62	27,26%	67,73%
6	15	FR1	0,64	2,50%	4712137,71	4594355,62	27,35%	66,45%
6	15	DFR	0,64	1,51%	4664906,20	4594355,62	27,26%	67,73%
6	15	DFR1	0,67	3,17%	4744841,17	4594355,62	27,96%	67,48%
6	30	BC	200	4,86%	5373962,39	5112526,07	13,93%	42,46%
6	30	FR	2,67	6,28%	5455078,94	5112526,07	14,06%	42,93%
6	30	FR1	2,7	7,10%	5503152,13	5112526,07	14,63%	39,36%
6	30	DFR	2,17	6,08%	5443339,47	5112526,07	14,22%	38,84%
6	30	DFR1	2,27	6,82%	5486538,01	5112526,07	14,64%	38,26%
12	15	BC	200	1,63%	8931582,99	8786329,68	24,03%	68,81%
12	15	FR	1,72	2,65%	9025396,59	8786329,68	24,37%	69,04%
12	15	FR1	1,64	3,31%	9087554,06	8786329,68	24,87%	69,45%
12	15	DFR	1,59	3,03%	9060634,23	8786329,68	24,58%	69,89%
12	15	DFR1	1,7	4,23%	9174185,95	8786329,68	25,20%	71,06%
12	30	BC	200	5,91%	10488446,95	9868131,34	13,28%	36,73%
12	30	FR	7,14	6,58%	10563117,63	9868131,34	13,39%	41,38%
12	30	FR1	8,3	8,66%	10803451,49	9868131,34	14,09%	37,15%
12	30	DFR	5,92	5,90%	10487023,27	9868131,34	13,15%	42,75%
12	30	DFR1	6,31	8,34%	10766018,53	9868131,34	13,95%	39,96%
24	15	BC	200	1,46%	16619159,95	16375783,15	24,00%	67,35%
24	15	FR	4,22	2,01%	16712435,78	16375783,15	24,28%	69,26%
24	15	FR1	3,72	3,57%	16981781,35	16375783,15	24,87%	68,81%
24	15	DFR	3,38	1,81%	16678253,35	16375783,15	24,25%	69,58%
24	15	DFR1	3,31	3,91%	17041520,56	16375783,15	25,03%	70,51%
24	30	BC	200	3,09%	27184143,95	26345431,03	19,02%	27,97%
24	30	FR	14,8	3,41%	27275422,81	26345431,03	18,50%	31,27%
24	30	FR1	17,19	4,42%	27563791,40	26345431,03	19,18%	27,86%
24	30	DFR	12,53	3,74%	27369808,34	26345431,03	18,74%	30,11%
24	30	DFR1	14,7	4,21%	27504422,93	26345431,03	19,11%	27,34%
Class B								
6	15	BC	200	2,27%	5830113,28	5698078,38	34,47%	59,13%
6	15	FR	0,69	2,73%	5858168,59	5698078,38	34,94%	62,79%
6	15	FR1	0,67	2,52%	5845495,92	5698078,38	34,67%	60,13%
6	15	DFR	0,75	2,39%	5837544,14	5698078,38	34,60%	59,49%
6	15	DFR1	0,64	2,52%	5845187,18	5698078,38	34,67%	60,17%
6	30	BC	200	5,14%	7887366,21	7482152,71	22,20%	39,53%
6	30	FR	1,86	4,47%	7832065,14	7482152,71	21,95%	40,24%
6	30	FR1	1,84	4,29%	7817366,30	7482152,71	21,94%	41,27%
6	30	DFR	1,75	4,47%	7832065,14	7482152,71	21,95%	40,24%
6	30	DFR1	1,53	4,25%	7814117,13	7482152,71	21,91%	41,75%
12	15	BC	200	4,34%	13376789,44	12796410,89	38,12%	60,67%
12	15	FR	3,58	3,73%	13292194,25	12796410,89	37,74%	60,98%
12	15	FR1	3,33	5,63%	13559460,73	12796410,89	39,05%	60,57%
12	15	DFR	4,17	3,88%	13312283,78	12796410,89	37,79%	61,00%
12	15	DFR1	3,11	5,07%	13479203,38	12796410,89	38,69%	59,68%
12	30	BC	200	16,18%	17242514,85	14453922,45	23,72%	42,12%
12	30	FR	20,34	14,92%	16988946,09	14453922,45	23,24%	42,86%
12	30	FR1	19,05	17,05%	17424705,06	14453922,45	24,69%	43,13%
12	30	DFR	24,09	15,03%	17009714,00	14453922,45	23,70%	46,02%
12	30	DFR1	18,81	16,62%	17334236,26	14453922,45	24,45%	42,15%
24	15	BC	200	7,55%	28920671,65	26737370,61	43,68%	62,30%
24	15	FR	10,58	6,69%	28655048,09	26737370,61	43,52%	60,50%
24	15	FR1	8,86	7,16%	28799376,04	26737370,61	43,99%	60,45%
24	15	DFR	17,84	5,68%	28348357,04	26737370,61	42,84%	60,81%
24	15	DFR1	7,2	6,49%	28593139,10	26737370,61	43,39%	60,94%
24	30	BC	200	14,84%	46725651,41	39792758,6	33,18%	44,23%
24	30	FR	45,33	12,57%	45512020,24	39792758,6	32,33%	43,86%
24	30	FR1	60,56	14,88%	46746268,85	39792758,6	33,66%	43,50%
24	30	DFR	56,28	12,72%	45591716,51	39792758,6	32,44%	46,02%
24	30	DFR1	98,69	15,87%	47297771,66	39792758,6	33,95%	43,65%

TAB. 6.8 – Résultats expérimentaux : Instances de classe A et B

par la méthode Double-Fix-and-Relax, puisque les résultats précédents ont montré que c'est la meilleure méthode. Une étude empirique a montré que de grandes valeurs de σ_k ne fournissaient pas de bons temps CPU. Nous avons remarqué que des valeurs de σ_k inférieures à 5 donnaient de bons compromis entre le GAP et le temps CPU. Les tests sont effectués sur les instances de la classe B.

Dans ce qui suit, on suppose que $\sigma_k = \sigma$, $\delta_k = \delta$, $Time_k = time$ et $opt_k = opt$ pour tout k. Le paramètre σ prend les valeur $1, 2, 3$ et 4. Le paramètre δ prend des valeurs entre $0, 1, \ldots, \sigma - 1$.

Les critères d'arrêt pour tous les algorithmes sont fixés à $opt = 1\%$ et un temps CPU limite de 60 secondes pour chaque étape. Nous avons également étudié l'effet de la variation des critères d'arrêt $time$ et opt. Le temps limite pour chaque étape $time$ prends les valeurs $1, 3, 5, 10, 20$ et 30 secondes avec $opt = 0$. Le paramètre opt prend les valeurs $1\%, 2\%, 3\%, 4\%, 5\%$ et 6% avec $time = 60$ secondes.

Les valeurs du GAP sont calculées en utilisant la borne supérieure trouvée par l'heuristique et la meilleure borne inférieure obtenue à la fin de la méthode BC. On alloue un temps CPU limite de 200 secondes pour la méthode BC. Des résultats expérimentaux sont résumés par les figures suivantes. Les figures 6.5 et 6.6 représentent respectivement la variation de la qualité de la solution calculée par GAP et Temps CPU par rapport à σ, avec δ variant entre 0 et $\sigma - 1$, GAP et Temps CPU représentent la moyenne de plusieurs tests en fonction de la variation de δ. Les figures 6.7 et 6.8 représentent respectivement la variation du GAP et du Temps CPU par rapport à δ, avec σ variant entre 1 et 4. GAP et Temps CPU représentent la moyenne de plusieurs tests en fonction de la variation de σ.

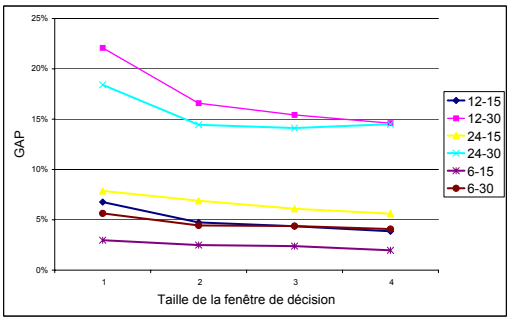


FIG. 6.5 – Variation du GAP en fonction de σ

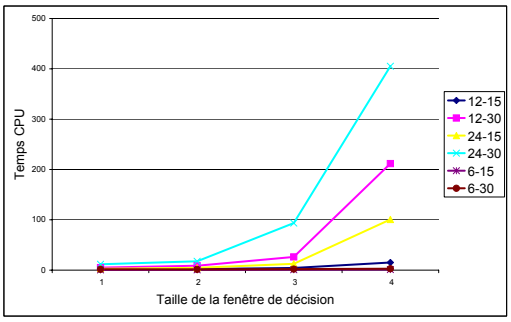


FIG. 6.6 – Variation du Temps CPU en fonction de σ

A partir des figures 6.5, 6.6, 6.7 et 6.8, nous pouvons remarquer que l'augmentation de σ et δ, accroître considérablement le temps CPU, mais permet une amélioration des GAPs induits. En effet, quand σ augmente, on remarque une nette diminution du GAP pour toutes les instances. Quand δ augmente, on remarque une amélioration du GAP mais pas pour toutes les instances. En effet, le GAP de l'instance 24-30 augmente entre $\delta = 1$ et $\delta = 3$.

Les figures 6.9 et 6.10 montrent respectivement la variation du GAP et du Temps CPU en fonction de opt. Les paramètres σ et δ sont fixés respectivement à 3 et 1 puisque l'analyse précédente à montré que ces calibrages donnaient le meilleur compromis entre le GAP et le temps CPU.

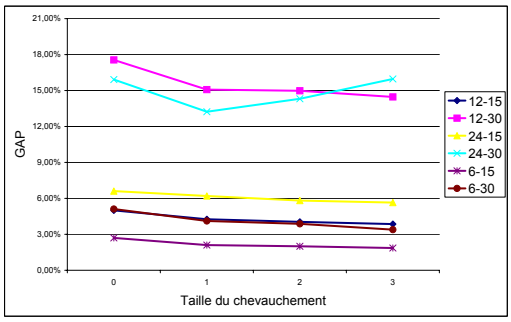


FIG. 6.7 – Variation du GAP en fonction de δ

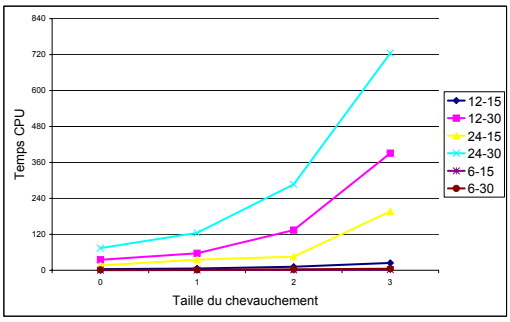


FIG. 6.8 – Variation du Temps CPU en fonction de δ

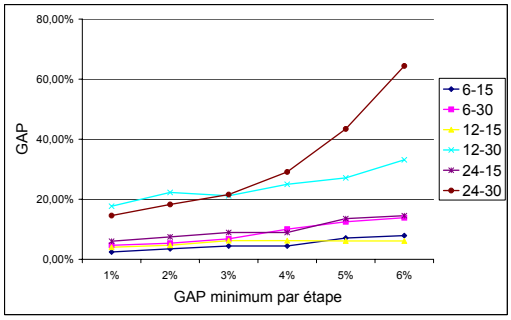


FIG. 6.9 – Variation du GAP en fonction de *opt*

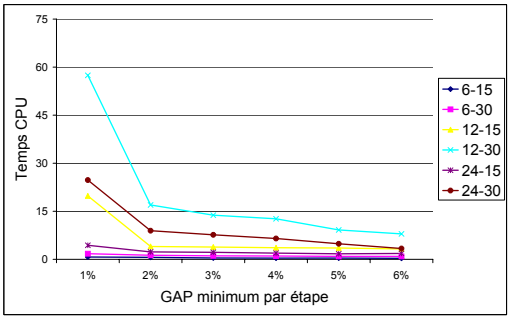


FIG. 6.10 – Variation du Temps CPU en fonction de *opt*

A partir des figures 6.9 et 6.10, nous pouvons remarquer que l'augmentation de *opt* réduit les temps d'exécution mais augmente le GAP considérablement.

Nous avons également étudié la variation du GAP en fonction du temps limite. Nous avons remarqué que le GAP décroît quand le temps limite accroît pour des valeurs inférieures à 5 secondes. Quand le temps limite est supérieur à 5 secondes, le GAP ne décroît pas considérablement.

Dans nos études expérimentales, nous avons supposé que les paramètres σ_k et δ_k sont constant. Il serait intéressant d'analyser l'effet de la variation de ces paramètres à travers l'exécution des heuristiques. Par exemple, prendre de grande valeurs de σ_k et δ_k au début de l'horizon de planification et réduire ces valeurs quand on déplace la fenêtre de décision vers le futur.

6.5 Conclusion

Nous avons proposé une nouvelle formulation mathématique d'un problème de lot-sizing à capacité finie impliquant, plusieurs références, des temps de setup, des coûts de rupture, des coûts de déficit sur le stock de sécurité, plusieurs groupes de références, plusieurs ressources, plusieurs gammes de production et des contraintes de production minimum. Ces contraintes sont issues de problématiques réelles de clients DynaSys. Afin de fournir une bonne borne supérieure dans des temps de calculs raisonnables, nous avons développé des heuristiques à base de MIP qui sont une hybridation d'une approche de décomposition de l'horizon de planification et d'une méthode branch-and-bound. Des résultats expérimentaux ont montré l'efficacité de cette approche sur des instances réelles. Le principe de cette heuristique peut être utilisé pour résoudre des problèmes de lot-sizing plus compliqués. Il peut également être adapté pour d'autres problèmes (ex. problèmes d'ordonnancement). Ces méthodes peuvent être améliorées en utilisant une meilleure approximation des variables de setup sur la fin de l'horizon de planification. Ces heuristiques peuvent également être utilisées conjointement avec une approche polyédrique afin de résoudre les sous-problèmes à l'optimum. Les solutions réalisables fournies par ces heuristiques peuvent être utilisées comme borne supérieure dans une méthode de branch-and-bound afin de l'accélérer.

Chapitre 7

Conclusions et perspectives de recherches

Dans cette thèse, nous avons analysé une variété de problèmes de lot-sizing rencontrés chez des clients de la société DynaSys. Nous avons exploré plusieurs approches afin d'identifier les particularités des problèmes de lot-sizing traités par rapport à la littérature, de formuler ces particularités et de proposer des méthodes pour résoudre ces problèmes. Les contributions de nos travaux de recherche sont multiples.

Nous avons tout d'abord présenté un état de l'art sur les problèmes de lot-sizing. Pour une meilleure présentation de celui-ci, nous avons proposé une classification des approches par type de problèmes : une seule référence, plusieurs références, puis par méthode de résolution : méthodes exactes, méthodes approchées. Nous avons accordé une attention particulière aux travaux utilisant des approches de résolution proches de celles abordées dans cette thèse, à savoir, les approches polyédriques, les approches de programmation dynamique ainsi que les approches heuristiques. Nous avons également accordé plus d'attention aux problèmes de lot-sizing à un seul niveau de production, à périodes longues, avec des contraintes de capacité, ainsi que ceux avec des coûts de rupture sur la demande.

Dans une seconde étape, nous avons présenté des modélisations mathématiques des problèmes rencontrés en milieu industriel. Nous avons introduit de nouvelles notions qui, à notre connaissance, ont très peu été traitées dans la littérature, telles que les coûts de rupture sur la demande, les coûts de déficit sur le stock de sécurité et la présence de plusieurs gammes de production.

Nous avons proposé trois classes d'inégalités valides pour le problème de lot-sizing à capacité finie avec plusieurs références, des temps de setup, ainsi que des coûts de rupture sur la demande (MCLS2). Ces inégalités définissent des facettes pour le problème (MCLS2) sous certaines conditions. Des heuristiques polynomiales ont été développées pour la séparation de ces inégalités. Celles-ci ont été utilisées dans une méthode branch-and-cut pour résoudre le problème (MCLS2). Les tests effectués sur des instances académiques sont très prometteurs.

Nous avons également proposé un algorithme de programmation dynamique pour la résolution du problème de lot-sizing à une seule référence sans contrainte de capacité, avec des coûts de rupture sur la demande et de déficit sur le stock de sécurité (ULS4). Cet algorithme est polynomial, il consiste à résoudre $O(T^3)$ problèmes de flots à coût minimum. Ce résultat nous a encouragé à utiliser la relaxation lagrangienne afin de fournir une borne inférieure pour le problème de lot-sizing à capacité finie impliquant plusieurs références, des

temps de setup, ainsi que des coûts de rupture sur la demande et de déficit sur le stock de sécurité (MCLS4). Cette méthode a fourni des bornes inférieures de bonne qualité comparées à celles trouvées par le solveur CPLEX 9.0. En utilisant CPLEX 9.0, La formulation agrégée du problème MCLS4 à donné de meilleurs résultats que la formulation désagrégée.

Dans le but de proposer une solution réalisable dans des temps raisonnables pour des problèmes de lot-sizing incluant toutes les particularités des jeux de données issus de situations industrielles, nous avons développé des heuristiques dites à base de MIP. Ces heuristiques découlent d'une hybridation d'une approche de décomposition de l'horizon de planification et d'une méthode de branch-and-bound. Celles-ci ont montré leur efficacité sur des instances réelles.

Tous ces développements sont intégrés au logiciel n.SKEP de la société DynaSys et sont installés chez plusieurs de ses clients. L'utilisateur dispose d'une interface graphique lui permettant de paramétrer les différentes méthodes (cf. Annexe 2).

Il existe plusieurs perspectives à nos travaux de recherches. En effet, nous avons prouvé qu'il est possible de résoudre le problème ULS4 en temps polynomial, mais la complexité de l'algorithme proposé reste assez élevée. Un premier axe de recherche serait d'essayer d'améliorer la complexité de cet algorithme pour permettre une meilleure résolution par relaxation lagrangienne. Une autre amélioration consiste à implémenter un algorithme de flot à coût minimum afin d'accélérer la résolution du problème ULS4.

Le plan de production fourni par la relaxation lagrangienne n'est pas réalisable pour le problème MCLS4, les seules contraintes violées sont les contraintes de capacité. En se basant sur ce plan de production, une heuristique de lissage peut être développée afin de rendre cette solution réalisable. Cette heuristique peut être développée en se basant sur celle proposée par Trigeiro *et al.* [172]. Ils ont proposé une heuristique lagrangienne afin de résoudre le problème MCLS présenté au chapitre 2 page 19. Les auteurs ont considéré également la relaxation des contraintes de capacité. L'avantage que peut avoir notre approche par rapport à celle de Trigeiro *et al.* est la possibilité d'ajuster le plan de production afin de trouver une solution réalisable à chaque étape de la relaxation lagrangienne. En effet, les problèmes traités par Trigeiro *et al.* sont caractérisées par des capacités suffisantes pour satisfaire toute la demande, ajuster le plan en supprimant des quantités produites n'est pas toléré. En revanche, nos instances sont de nature sous-capacitaire, ajuster le plan en supprimant des productions permet d'avoir des solutions réalisables. Ainsi, une heuristique lagrangienne peut toujours être utilisée afin de rendre la solution réalisable.

L'approche de relaxation lagrangienne utilisée pour le problème MCLS4 peut être généralisée afin de considérer des contraintes industrielles telles que, les groupes de références et le multi-ressources. Les contraintes de lancement de groupes et les contraintes de capacité sont considérées comme des contraintes couplantes, puisqu'elles permettent de relier l'ensemble des références pour le calcul de la consommation globale en capacité et le lancement des groupes. La double relaxation de ces contraintes décompose le problème en plusieurs problèmes ULS4. Il serait intéressant de tester la qualité des bornes inférieures obtenues par la relaxation lagrangienne de ces contraintes. Une autre perspective est de généraliser l'algorithme de programmation dynamique que nous avons proposé au chapitre 5 afin de considérer plusieurs gammes de production.

Il serait également intéressant d'utiliser les bornes inférieures obtenues par la relaxation lagrangienne dans un algorithme de branch-and-bound.

Un autre axe de recherche consiste à généraliser les inégalités valides proposées dans cette thèse au modèle MCLS2 basé sur le problème de localisation d'entrepôts. En effet, cette formulation a montré sa supériorité à la formulation dite agrégée du problème MCLS2. Une autre perspective de nos travaux est de généraliser ces inégalités pour inclure les coûts de start-up. En effet, Van Hoesel *et al.* [177] ont généralisé les inégalités (l, S) en une nouvelle classe d'inégalités (l, R, S) afin de traiter les coûts de start-up pour les problèmes de lot-sizing à une seule référence et sans contrainte de capacité. Il serait intéressant de généraliser ces travaux pour le problème de lot-sizing avec des contraintes de capacité, des temps de start-up et des coûts de rupture sur la demande.

Les inégalités valides que nous avons développées pour le problème MCLS2 n'ont pas encore été testées pour les cas : multi-groupes, multi-ressources et multi-gammes. Il serait intéressant d'implémenter ces inégalités dans une approche de branch-and-cut afin de tester leur efficacité.

Une autre perspective à ces travaux est d'utiliser une approche d'optimisation multicritère. En effet, la fonction objectif est une somme de plusieurs composantes de coûts. Les coûts de rupture et de déficit sur le stock de sécurité sont élevés en comparaison aux autres coûts. Il serait intéressant de caracteriser l'ensemble des solutions efficaces ou non dominées par une analyse multicritère de ces deux coûts. Il serait également intéressant de recourir à des techniques de résolution de programmes linéaires en nombres entiers multicritères.

Les heuristiques de décomposition de l'horizon de planification peuvent être améliorées en utilisant une meilleure approximation des variables de setup sur la fin de l'horizon de planification. Il serait également intéressant d'utiliser cette approche conjointement avec une méthode de coupes afin de trouver de meilleures solutions aux sous-problèmes.

Les planificateurs cherchent souvent à équilibrer les ruptures sur la demande. Les solutions obtenues par un solveur de programmation linéaire en nombres entiers sont souvent non équilibrées. En effet, en présence de références équivalentes (en termes de coûts et de consommations en ressource), la programmation linéaire n'équilibre pas les ruptures entre ces références. L'utilisation d'un objectif quadratique pour les ruptures permet d'effectuer un tel équilibrage.

Les overtimes sont des capacités supplémentaires avec un coût associé dans la fonction objectif. Ces derniers peuvent remplacer les variables de rupture sur la demande. Le plan de production obtenu en considérant des overtimes viole les contraintes de capacité si les ressources disponibles ne sont pas suffisantes pour satisfaire toute la demande. Une heuristique de lissage peut être utilisée afin de rendre cette solution réalisable tout en équilibrant les ruptures entre les références.

Les méthodes hiérarchiques, dites également approches d'agrégation/désagrégation, consistent à remplacer la formulation monolithique par une séquence de modèles et une hiérarchie de décisions à prendre. Les décisions agrégées sont prises en premier, et sont ensuite imposées comme contraintes aux modèles plus détaillés. En retour, les décisions détaillées permettent d'évaluer la qualité des décisions agrégées. Les décisions en haut de la hiérarchie sont basées sur des modèles agrégés. Le succès de cette approche réside essentiellement sur l'uniformité entre l'agrégation et la désagrégation ainsi que l'interaction entre les modèles aux différents niveaux de la hiérarchie. Au niveau agrégé, les références sont regroupées par groupes, les références appartenant au même groupe sont classées par famille. Chaque famille est assimilée à une référence agrégée, pour laquelle on calcule des coûts et des consommations en ressource. Afin de classer les références par famille on peut utiliser des techniques de classification. Au niveau détaillé, on dispose d'un modèle relatif à chaque famille de références. Les

décisions prises au niveau agrégées (plan de production) sont imposées comme contraintes (contrainte de ressources) aux modèles du niveau détaillé. Cette approche a été initiée à DynaSys, il serait intéressant de poursuivre ces travaux.

Le dernier axe de recherche consiste à aborder les problèmes de lot-sizing à plusieurs niveaux de production. La notion d'*echelon stock*, introduite par Clark et Scart [34], peut être utilisée afin de développer des inégalités valides ou une approche de relaxation lagrangienne pour ce type de problèmes.

Annexes

Annexe 1 : Quelques extensions du problème MCLS

Nous présentons dans cette section les principales extensions du problème MCLS rencontrées dans la littérature.

Les startups

Le coût de start-up (start-up cost) et le temps de start-up (start-up time) sont encourus pour une référence i à une période t, quand il y a un lancement de production de cette référence à la période t mais pas à la période $t-1$. En effet, quand les périodes sont courtes, et que les temps et coûts de changement sont importants, il est évident de vérifier les changements de situation pour minimiser les coûts de lancement.

Afin d'introduire les start-ups dans le modèle MCLS, on définit une variable binaire supplémentaire u_{it} qui est égale à 1 quand il y a un start-up à une période t. Ceci conduit à rajouter les contraintes suivantes au problème MCLS :

$$y_{it} - y_{i(t-1)} \leq u_{it} \leq y_{it}, \quad \forall i, \forall t \tag{7.1}$$

$$u_{it} \in \{0,1\}, \quad \forall i, \forall t \tag{7.2}$$

Les modifications apportées au problème MCLS sont, l'ajout des coûts de start-up dans la fonction objectif, et des temps de start-up dans les contraintes de ressource.

Les changeovers

Dans quelques environnements de production, les changements ne dépendent pas seulement de la référence qui doit être produite, mais également de tout ce qui s'est passé sur toutes les références à la période précédente. En effet, le passage de la production d'une référence à une autre référence peut entraîner un temps et un coût de changement qui diffère d'un couple de références à un autre. Cette notion est appelée : changeover.

Exemple 1 *Un exemple souvent rencontré dans l'industrie est celui du coloris. Le passage d'un coloris à un autre dépend vraiment du couple de coloris. Ainsi, le réglage d'une machine pour le passage du coloris bleu au blanc est plus lent et plus coûteux que le passage du coloris bleu au noir. En effet, pour le passage du bleu au blanc, la machine doit être nettoyée à fond, alors que pour le passage du bleu au noir, un tel nettoyage n'est pas nécessaire.*

$w_{ii't}$ est une variable binaire qui sert à détecter les changeovers, elle est égale à 1, si les références i et i' sont lancées respectivement aux périodes t et $t-1$. Le changeover est formulé en rajoutant les contraintes suivantes au problème MCLS.

$$w_{ii't} \geq y_{it} + y_{i'(t-1)} - 1, \quad i = 1, \ldots, N, i' = 1, \ldots, N, t = 1, \ldots, T \tag{7.3}$$

$$w_{ii't} \in \{0, 1\}, \quad i = 1, \ldots, N, i' = 1, \ldots, N, t = 1, \ldots, T \tag{7.4}$$

On apporte les mêmes modifications que celles apportées au problème avec le start-up en rajoutant au problème MCLS, les coûts de changeover dans la fonction objectif, ainsi que les temps de changeover dans les contraintes de capacité.

L'ajout du changeover au problème MCLS est souvent suivi de la contrainte de small buckets (périodes courtes), où une seule référence a le droit d'être lancée. Cette contrainte est modélisée en utilisant la contrainte suivante :

$$\sum_{i=1}^{N} y_{it} \leq 1, \ \forall t \tag{7.5}$$

Les overtimes

Il est souvent réducteur d'imposer une contrainte de capacité forte à chaque période et pour chaque ressource. En effet, il est fréquemment possible d'ajouter de la capacité. C'est ce que l'on appelle les overtimes.

Exemple 2 *Un exemple très fréquent est celui de l'augmentation des ressources humaines. En effet, cette augmentation est considérée comme des heures supplémentaires. Son ajout coûte alors un prix plus élevé que celui des ressources humaines disponibles.*

Cette modélisation est faite en rajoutant une variable continue o_t aux ressources disponibles à la période t (c_t). La variable o_t est pénalisée dans la fonction objectif, cette pénalité est relative au coût des ressources supplémentaires utilisées.

Le backlogging

Le backlogging est la possibilité d'avoir du retard sur les commandes. Ce retard est pénalisé puisqu'il peut avoir un impact négatif sur la satisfaction des clients. Ceci arrive quand l'usine n'a pas assez de capacité en ressources pour satisfaire le client à temps. Cette problématique est modélisée en permettant à la production d'une période donnée de satisfaire des demandes exprimées à des périodes précédentes. On aura donc deux flux de stock, un flux allant du début vers la fin de l'horizon (flux des stocks), et un autre allant de la fin vers le début de l'horizon (flux des retards). Pour modéliser le flux des retards, on introduit une nouvelle variable positive ou nulle b_{it}, pour chaque référence i et période t, représentant le retard de la demande à la fin de la période t. Ainsi, b_{it} est la demande des périodes $1, \ldots, t$ qui sera livrée en retard (à des périodes dans $t+1, \ldots, T$. Une formulation classique du backlogging est faite en remplaçant la contrainte de flux (2.10) du problème MCLS par la contrainte (7.6). Les variables b_{it} sont pénalisées dans la fonction objectif.

$$x_{it} - s_{it} + b_{it} + s_{i(t-1)} - b_{i(t-1)} = d_{it},\ \forall i, \forall t \tag{7.6}$$

Le multi-niveau

Dans le cas où on a une structure de produits en multi-niveaux, on a une restriction des flux de matières. En effet, les produits peuvent être en entrée d'un niveau de production, comme ils peuvent être en sortie d'autres niveaux de production. Ce qui crée des contraintes de précédences entre les produits. Ces restrictions sont modélisées en apportant quelques modifications à la contrainte de conservation des flux à travers l'horizon.

Le problème de lot-sizing à plusieurs niveaux de production peut être vu comme un seul modèle monolithique contenant un modèle MCLS pour les produits finis et un modèle ULS pour chaque produit intermédiaire et matière première. Le but d'une telle modélisation est d'optimiser simultanément les productions et les achats de toutes les références, afin de satisfaire les demandes externes (les demandes indépendantes) provenant des clients, et les demandes interne (les demandes dépendantes) provenant de la production d'autres produits. La dépendance entre les produits est modélisée à travers la définition d'une structure de produits, appelée nomenclature (notée : BOM[1]). La figure 7.1 présente une nomenclature. Les sommets représentent les produits et les arêtes les liens entre les produits. Chaque arc (i, j) est pondéré par une valeur représentant la quantité du produit i nécessaire à la fabrication d'une unité du produit j. l'exemple représente la nomenclature d'un produit fini 1, deux produits intermédiaires $2, 3$ et trois matières premières $4, 5, 6$.

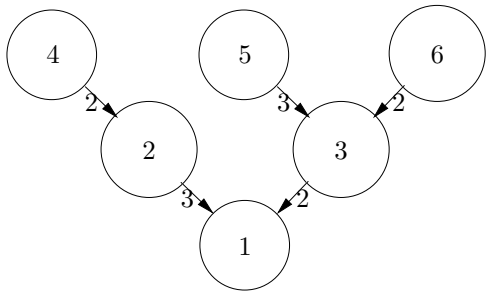


FIG. 7.1 – Exemple de nomenclature à deux niveaux

Afin de modéliser le problème en multi-niveaux, nous allons introduire un nouveau paramètre b_{ij}, b_{ij} représente la quantité de la référence i nécessaire à la fabrication d'une unité de la référence j. Ce paramètre est utilisé pour calculer la demande dépendante (d_{it} représente la demande indépendante). Nous allons également définir $\mathcal{L}(i)$, l'ensemble de toutes les références successeurs de la référence i ($\mathcal{L}(i) = \emptyset$ pour les produits finis). Pour chaque référence i, on défini par τ_i le temps d'attente pour la production ou la livraison de la référence i. En reprenant le problème MCLS il s'agit de remplacer la contrainte de flux (2.10) par la contrainte (7.7).

$$x_{i(t-\tau_i)} - s_{it} + s_{i(t-1)} = d_{it} + \sum_{j \in \mathcal{L}(i)} b_{ij} x_{jt},\ \forall i, \forall t \tag{7.7}$$

[1]BOM : Bill Of Materials.

La contrainte 7.7 exprime la conservation des flux à travers l'horizon de planification pour chaque référence i à chaque période t. La quantité produite est à la période $t-\tau_i$. La demande à satisfaire est à la période t. cette dernière est la somme de la demande indépendante d_{it} et la demande dépendante $\sum_{j\in\mathcal{L}(i)} b_{ij}x_{jt}$.

Annexe 2 : Développement logiciel

Les travaux de ma thèse s'inscrivent dans le cadre du développement du logiciel n.SKEP de la société DynaSys. DynaSys est un éditeur de logiciels qui propose une solution globale pour l'optimisation de la chaîne logistique. Cette dernière s'appelle n.SKEP.

Nous présentons dans cette annexe une description rapide du logiciel n.SKEP ainsi qu'un aperçu sur l'intégration des algorithmes développés dans le cadre de mes travaux de recherche.

n.SKEP gère les flux d'information qui circulent entre les clients et les fournisseurs. Il dispose également de plusieurs modules qui permettent d'optimiser et de simuler les flux physiques qui circulent des fournisseurs vers les clients.

n.SKEP dispose d'une arborescence d'objets appelée *générateur hiérarchique*. Le générateur hiérarchique permet : la modélisation des problèmes, l'administration de l'application ainsi la personnalisation des affichages (cf. figure 7.2). A travers cette arborescence, plusieurs entités peuvent être modélisées, telles que les entités de production, de prévision, de distribution...

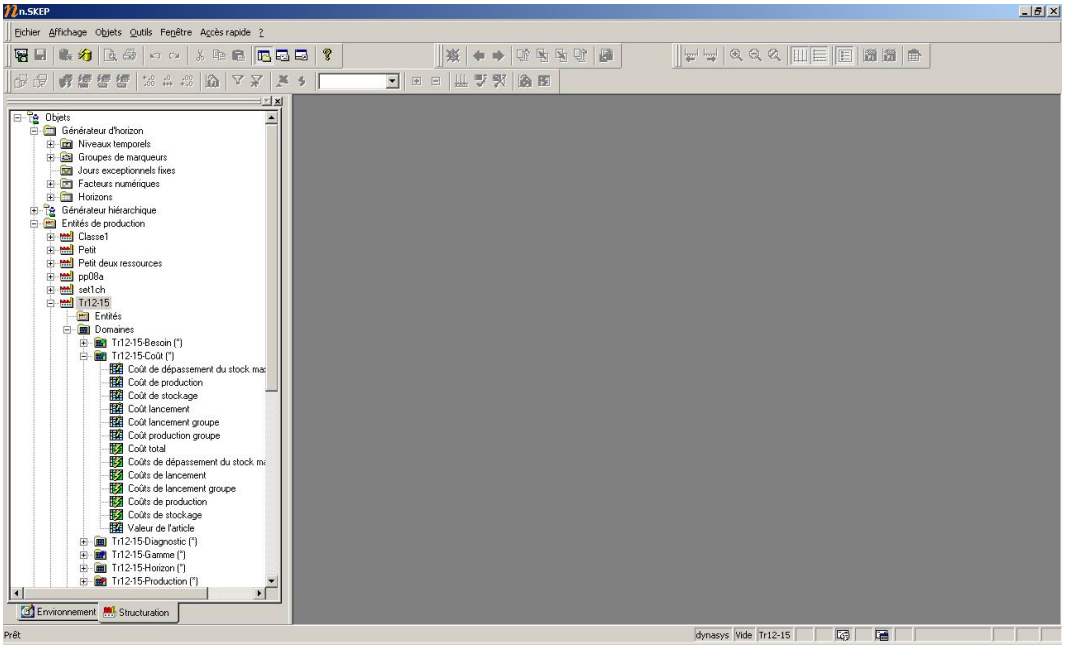


Fig. 7.2 – n.SKEP : Générateur Hiérarchique

Mes travaux de recherche portent essentiellement sur les entités de production. Une entité de production se compose de plusieurs domaines à partir desquels des données peuvent être saisies ou consultées. L'exécution des algorithmes se fait à l'aide de liens dits d'*alimentation* puisqu'ils permettent d'alimenter des paramètres tel que le plan de production (cf. figure 7.3).

Le lien d'alimentation *Optimizer* permet d'optimiser le plan de production en se basant sur une fenêtre de paramétrage. On a vu au chapitre 6 que Les heuristiques à base de MIP sont contrôlées par plusieurs paramètres. Ces paramètres sont renseignés à partir de cette fenêtre de paramétrage. Ainsi, l'utilisation de ces heuristique est activée si la case à cocher *Option rapide* est activée. Une fois cette option activée, nous pouvons accéder à toutes les options des heuristiques décrites au chapitre 6 (cf. figure 7.4). Durant l'exécution des algorithmes, l'utilisateur ne voit que l'état d'avancement de la résolution sur une fenêtre de progression (cf. figure 7.5).

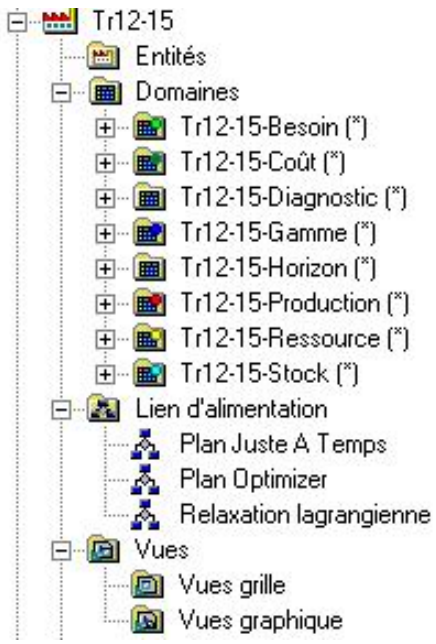


FIG. 7.3 – n.SKEP : Entité de production

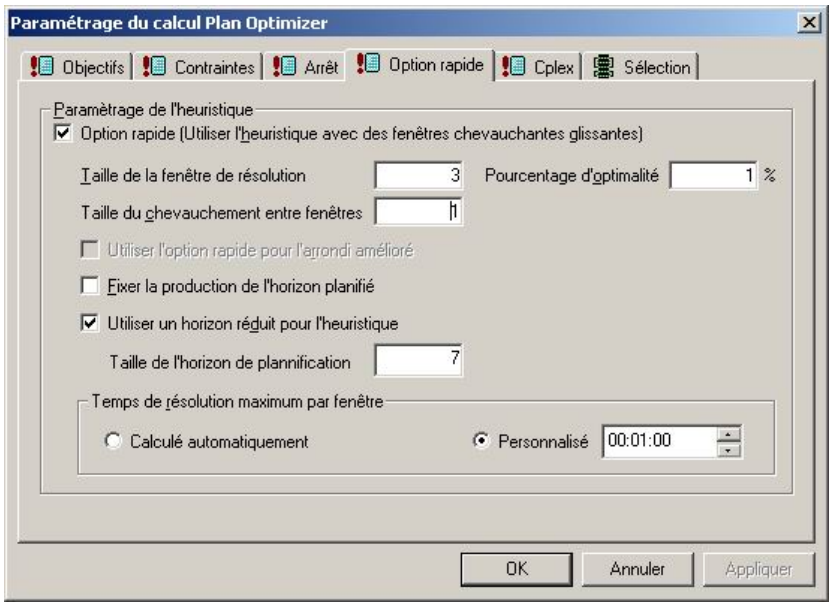


FIG. 7.4 – n.SKEP : Fenêtre de paramétrage (Option rapide)

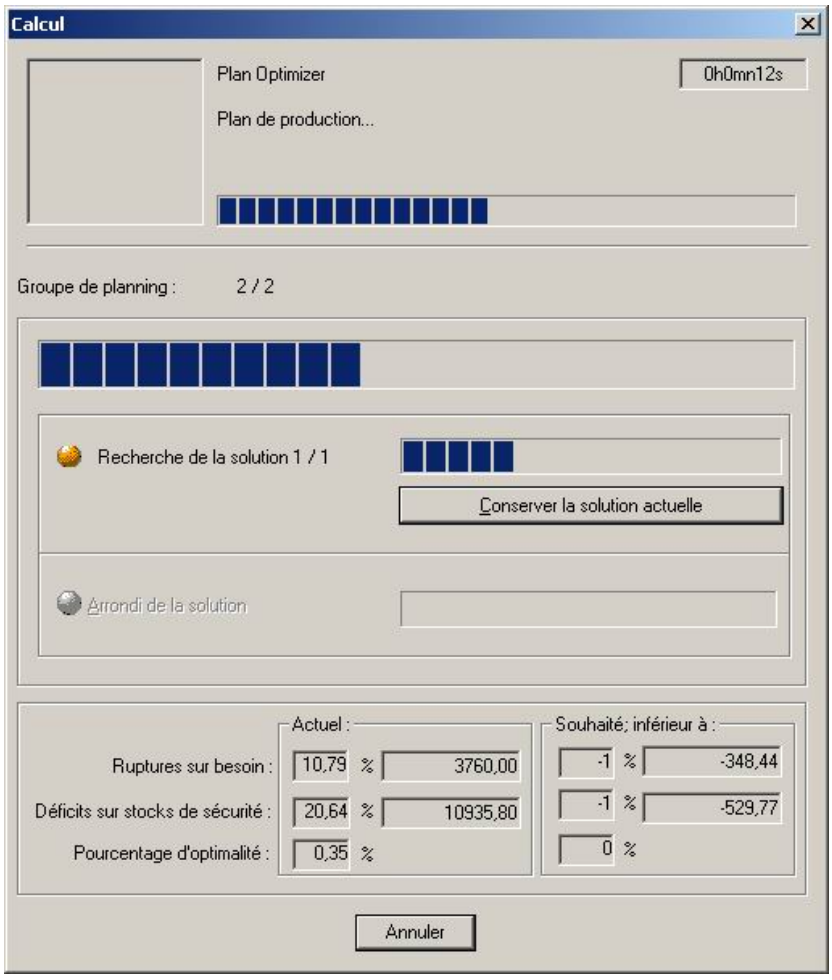
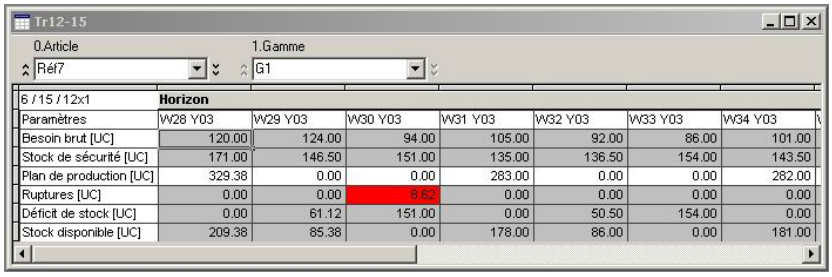


FIG. 7.5 – n.SKEP : Fenêtre de progression

Une fois le calcul fini, l'utilisateur peut consulter ses résultats en visualisant des grilles numériques ou des vues graphiques. (cf. figures 7.6, 7.7 et 7.8).



6 / 15 / 12x1	Horizon						
Paramètres	W28 Y03	W29 Y03	W30 Y03	W31 Y03	W32 Y03	W33 Y03	W34 Y03
Besoin brut [UC]	120.00	124.00	94.00	105.00	92.00	86.00	101.00
Stock de sécurité [UC]	171.00	146.50	151.00	135.00	136.50	154.00	143.50
Plan de production [UC]	329.38	0.00	0.00	283.00	0.00	0.00	282.00
Ruptures [UC]	0.00	0.00	8.62	0.00	0.00	0.00	0.00
Déficit de stock [UC]	0.00	61.12	151.00	0.00	50.50	154.00	0.00
Stock disponible [UC]	209.38	85.38	0.00	178.00	86.00	0.00	181.00

FIG. 7.6 – n.SKEP : Grille numérique

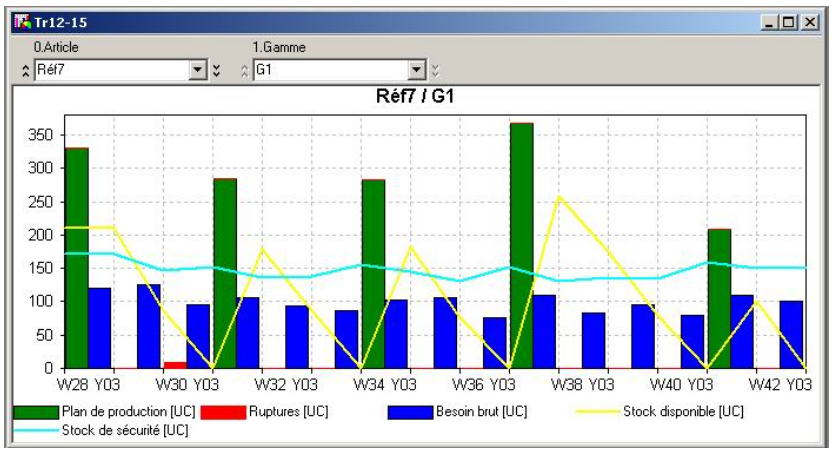


FIG. 7.7 – n.SKEP : Vue graphique

Il existe également un lien d'alimentation consacré à la relaxation lagrangienne des contraintes de capacité présentée au chapitre 5. Les inégalités valides présentées au chapitre 4 sont activées par des options avancées de la base des registres.

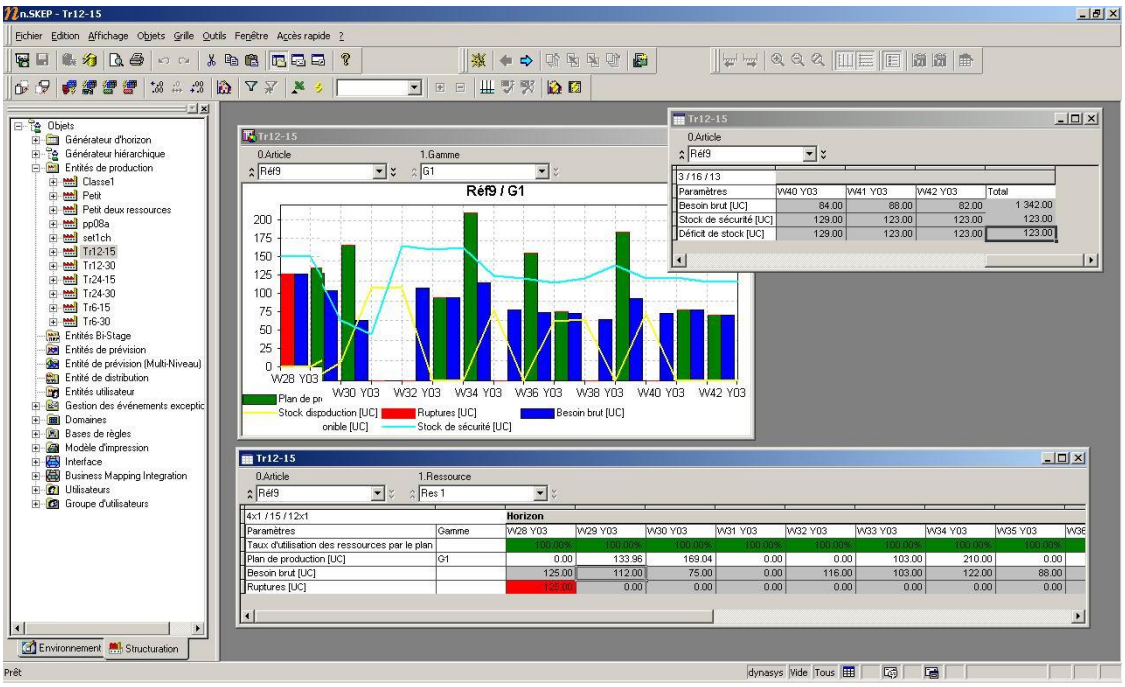


FIG. 7.8 – n.SKEP : Différentes vues

Bibliographie

[1] K. Aardal, Y. Pochet, and L.A. Wolsey. Capacitated facility location : Valid inequalities and facets. *Mathematics of Operations Research*, 20 :562–582, 1995.

[2] P. Afentakis. A parallel heuristic algorithm for lot-sizing in muli-stage production systems. *IIE Transactions*, 34 :34–42, 1987.

[3] P. Afentakis and B. Gavish. Optimal lot-sizing algorithms for complex product structures. *Operations Research*, 34 :237–249, 1986.

[4] P. Afentakis, B. Gavish, and U. Karmarkar. Computationally efficient optimal solutions to the lot-sizing problem in multistage assembly systems. *Management Science*, 30 :222–239, 1984.

[5] A. Aggarwal and J.K. Park. Improved algorithms for economic lot size problems. *Operations Research*, 41 :549–571, 1993.

[6] R.K. Ahuja, A.V. Goldberg, J.B. Orlin, and R.E. Tarjan. Finding minimum-cost flows by double scaling. *Mathematical Programming*, 53 :243–266, 1992.

[7] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows : Theory, Algorithms, and Applications*. Prentice Hall, New Jersey, 1993.

[8] D. Aksen, K. Altinkemer, and S. Chand. The single-item lot-sizing problem with immediate lost sales. *European Journal of Operational Research*, 147 (3) :558–566, 2003.

[9] A. Atamtürk and S. Küçükyavuz. Lot sizing with inventory bounds and fixed costs : Polyhedral study and computation. *Operations Research*, 53 :711–730, 2005.

[10] A. Atamtürk and J. C. Mun oz. A study of the lot–sizing polytope. *Mathematical Programming*, 99 :443–465, 2004.

[11] S. Axsäter. Performance bounds for lot sizing heuristics. *Management Science*, 31(5) :634–640, 1985.

[12] H.C. Bahl. Column generation based algorithm for multi-item scheduling. *IIE Transactions*, 15(2) :136–141, 1983.

[13] K.R. Baker, P. Dixon, M.J. Magazine, and E.A. Silver. An algorithm for the dynamic lot-size problem with time-varying production capacity constraints. *Management Science*, 24 (16) :1710–1720, 1978.

[14] I. Barany, T. Van Roy, and L.A. Wolsey. Strong formulations for multi-item capacitated lot-sizing. *Management Science*, 30 (10) :1255–1261, 1984.

[15] I. Barany, T. Van Roy, and L.A. Wolsey. Uncapacitated lot-sizing : the convex hull of solutions. *Mathematical Programming Study*, 22 :32–43, 1984.

[16] G. Barbarosoglu and L. Özdamar. Analysis of solution space-dependent performance of simulated annealing : the case of the multi-level capacitated lot sizing problem. *Computers and Operations Research*, 27 (9) :895–903, 2000.

[17] R.E. Bellman. *Dynamic programming*. Princeton University Press : Princeton, 1957.

[18] G. Belvaux and L.A. Wolsey. bc-prod : A specialized branch-and-cut system for lot-sizing problems. *Management Science*, 46 :724–738, 2000.

[19] G. Belvaux and L.A. Wolsey. Modelling practical lot-sizing problems as mixed integer programs. *Management Science*, 47 :993–1007, 2001.

[20] W.L. Berry. Lot sizing procedures for requirements planning systems : a framework for analysis. *Production and Inventory Management*, 13 :19–34, 1972.

[21] G. Bitran, T. L. Magnanti, and H.H. Yanasse. Approximation methods for the uncapacitated dynamic lot-size problem. *Management Science*, 30 :1121–1140, 1984.

[22] G. Bitran and H. Matsuo. The multi-item capacitated lot size problem : Error bounds of manne's formulations. *Management Science*, 32 (3) :350–359, 1986.

[23] G. Bitran and H.H. Yanasse. Computational complexity of the capacitated lot size problem. *Management Science*, 28 (10) :1174–1186, 1982.

[24] G.R. Bitran and A.C. Hax. On the design of hierarchical production planning systems. *Decision science*, 8 :28–55, 1977.

[25] J.D. Blackburn and R.A. Millen. Improved heuristics for multi-stage requirements planning systems. *Management Science*, 28 (1) :44–56, 1982.

[26] N. Brahimi. *Planification de la production : modèles et algorithmes pour les problèmes de dimensionnement de lots*. PhD thesis, Université de Nantes, 2004.

[27] N. Brahimi, S. Dauzère-Pérès, N.M. Najid, and A. Nordli. Single item lot sizing problems. *European Journal of Operational Research*, 168 (1) :1–16, 2006.

[28] D. Cattrysse, J. Maes, and L.N. Van Wassenhove. Set partitioning and column generation heuristics for capacitated dynamic lotsizing. *European Journal of Operational Research*, 46 (1) :38–47, 1990.

[29] H. Chen, D. Hearn, and C. Lee. A new dynamic programming method for the single item capacitated dynamic lot size model. *Journal of Global Optimization*, 4 :285–300, 1994.

[30] W.-H Chen and J.M. Thizy. An efficient column generation algorithm for the multi-item capacitated lot-sizing problem. Technical Report 02-52, University of Ottawa, 2002.

[31] W.H. Chen and J.M. Thizy. Analysis of relaxations for the multi-item capacitated lot-sizing problem. *Annals of Operations Research*, 26 :29–72, 1990.

[32] C.S. Chung, J. Flynn, and C.H. Lin. An effective algorithm for the capacitated single item lot size problem. *European Journal of Operational Research*, 75 (2) :427–440, 1994.

[33] C.S. Chung and M. Lin. An $o(t^2)$ algorithm for the ni/g/ni/nd capacitated single item lot size problem. *Management Science*, 34 :420–426, 1988.

[34] A.J. Clark and H. Scarf. Optimal policies for a multi-echelon inventory problem. *Management Science*, 6 (4) :475–490, 1960.

[35] A.R. Clark. Hybrid heuristics for planning lot setups and sizes. *Computers & Industrial Engineering*, 45 (4) :545–562, 2003.

[36] M. Cohen. Joint pricing and ordering policy for exponentially decaying inventory with known demand. *Naval Research Quarterly*, 24 :257–268, 1977.

[37] B.J. Coleman. A further analysis of variable demand lot-sizing techniques. *Production and Inventory Management Journal*, Third Quarter :19–24, 1992.

[38] B.J. Coleman and M.A. McKnew. A technique for order placement and sizing. *Journal of Purchasing and Materials Management*, 26 (2) :32–40, 1990.

[39] B.J. Coleman and M.A. Mcknew. An improved heuristic for multilevel lot sizing material requirements planning. *Decision Sciences*, 22 :136–156, 1991.

[40] M. Constantino. A cutting plane approach to capacitated lot-sizing with start-up costs. *Mathematical Programming*, 75 :353–376, 1996.

[41] M. Constantino. Lower bounds in lot-sizing models : A polyhedral study. *Mathematics of Operations Research*, 23(1) :101–118, 1998.

[42] W.J. Crowder, E.L. Johnson, and M.W. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31 :803–834, 1983.

[43] E. Danna, E. Rothberg, and C. Le Papa. Exploring relaxation induced neighborhoods to improve mip solutions. *Mathematical Programming*, 102 (1) :71–90, 2005.

[44] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press : Princeton, 1963.

[45] S. Dauzère-Pérès, N. Brahimi, N.M. Najid, and A. Nordli. Uncapacitated lot-sizing problems with time windows. Technical report, Ecole des Mines de Saint-Etienne, 2005.

[46] N. Dellaert and J. Jeunet. Solving large unconstrained multilevel lot-sizing problems using a hybrid genetic algorithm. *International Journal of Production Research*, 38 (5) :1083–1099, 2000.

[47] N. Dellaert, J. Jeunet, and N. Jonard. A genetic algorithm to solve the general multi-level lot-sizing problem with time-varying costs. *International Journal of Production Economics*, 68 :241–257, 2000.

[48] M. Diaby, H.C. Bahl, M.H. Karwan, and S. Zionts. Capacitated lot-sizing and scheduling by lagrangean relaxation. *European Journal of Operational Research*, 59 (3) :444–458, 1992.

[49] M. Diaby, H.C. Bahl, M.H. Karwan, and S. Zionts. A lagrangean relaxation approach for very large scale capacitated lot-sizing. *Management Science*, 38 :1329–1340, 1992.

[50] P.S. Dixon, M.D. Elder, G.K. Rand, and E.A. Silver. A heuristic algorithm for determining lot sizes of an item subject to regular and overtime production capacities. *Journal of Operations Management*, 3 :121–130, 1983.

[51] P.S. Dixon and E.A. Silver. A heuristic solution procedure for the multi-item, single-level, limited capacity, lotsizing problem. *J. of Operations Management*, 2 :23–39, 1981.

[52] A. Drexl and A. Kimms. Lot sizing and scheduling - survey and extensions. *European Journal of Operational Research*, 99 (2) :221–235, 1997.

[53] O. du Merle, J.-L. Goffin, C. Trouiller, and J.-P. Vial. A lagrangian relaxation of the capacitated multi-item lot sizing problem solved with an interior point cutting plane algorithm. Technical report, GERAD - Faculty of Management, McGill University, Montreal, Canada, 1997. Technical Report.

[54] B.P. Dzielinski and R.E. Gomory. Optimal programming of lotsizes, inventories and labor allocations. *Management Science*, 11 (9) :874–890, 1965.

[55] H.G. Eggleston. Complexity. In Cambridge University Press, editor, *Cambridge tracts in mathematics and mathematical physics*, number 47, 1963.

[56] P.S. Eisenhut. A dynamic lotsizing algorithm with capacity constraints. *AIIE Transactions*, 7(2) :170–176, 1975.

[57] G.D. Eppen and R.K. Martin. Solving multi-item capacitated lot-sizing problems using variable redefinition. *Operations Research*, 35 (6) :832–848, 1987.

[58] S.S. Erenguc. Multiproduct dynamic lot-sizing model with coordinated replenishments. *Naval Research Logistics*, 35(1) :1–22, 1988.

[59] A. Federgruen and M. Tzur. A simple forward algorithm to solve general dynamic lot sizing models with n periods in $o(n \log n)$ or o(n) time. *Management Science*, 37 :909–925, 1991.

[60] A. Federgruen and M. Tzur. The dynamic lot-sizing model with backlogging : A simple $o(n \log n)$ algorithm and minimal forecast horizon procedure. *Naval Research Logistics*, 40 :459–478, 1993.

[61] M. Fischetti and Andrea Lodi. Local branching. *Mathematical Programming*, 98 (3) :23–47, 2003.

[62] B. Fleischmann. The discrete lot-sizing and scheduling problem. *European Journal of Operational Research*, 44(3) :337–348, 1990.

[63] B. Fleischmann and H. Meyr. The general lotsizing and scheduling problem. *OR Spektrum*, 19(1) :11–21, 1997.

[64] M. Florian and M. Klein. Deterministic production planning with concave costs and capacity constraints. *Management Science*, 18 (1) :12–20, 1971.

[65] M. Florian, J.K. Lenstra, and A.H.G. Rinnoy Kan. Deterministic production planning : algorithms and complexity. *Management Science*, 26(7) :669–679, 1980.

[66] H. Gabbay. Multi-stage production planning. *Management science*, 25(11) :1138–1148, 1979.

[67] H. Gfrerer and G. Zäpfel. Hierarchical model for production planning in the case of incertain demand. *European Journal of Operational Research*, 86(1) :142–161, 1995.

[68] P. Ghare and G. Schrader. A model for exponentially decaying inventories. *Journal of Industrial Engineering*, 14 :238–243, 1963.

[69] A.V. Goldberg and R.E. Tarjan. Solving minimum cost flow by successive approximation. *Mathematics of Operations Research*, 15 :430–466, 1990.

[70] R. E. Gomory. An algorithm for the mixed integer problem. Technical report, The Rand Corporation, 1960.

[71] M. Gopalakrishnan, K. Ding, J.-M. Bourjolly, and S. Mohan. A tabu-search heuristic for the capacitated lot-sizing problem with set-up carryover. *Management Science*, 47 (6) :857–863, 2001.

[72] T. Gorham. Dynamic order quantities. *Production and Inventory Management*, 10 :75–81, 1968.

[73] G.K. Groff. A lot sizing rule for time-phased component demand. *Production and Inventory Management*, 20 :47–53, 1979.

[74] M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32 :1195–1220, 1984.

[75] Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh. Sequence independent lifting in mixed integer programming. *Journal of Combinatorial Optimization*, 4 :109–129, 2000.

[76] K. Haase and A. Kimms. Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities. *International Journal of Production Economics*, 66(2) :159–169, 2000.

[77] J.R. Hardin, G.L. Nemhauser, and M.W.P. Savelsbergh. Analysis of bounds for a capacitated single-item lot-sizing problem. *Computers and Operations Research*, To appear :–, 2005.

[78] M. Held, P. Wolfe, and H.D. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6 :62–88, 1974.

[79] G. hetreux, C. Mercé, and G. Fontan. Planification de la production : robustesse des décisions en contexte incertain. In *2ème Congrès International Franco-Québecois*, Albi, France, 1997.

[80] R.M. Hill. Note : Dynamic lot-sizing for a finite rate input process. *Naval Research Logistics*, 44 :221–228, 1997.

[81] K.S. Hindi. Solving the clsp by a tabu search heuristic. *Journal of the Operational Research Society*, 47 :151–161, 1996.

[82] Y.F. Hung, C.P. Chen, C.C. Shih, and M.H. Hung. Using tabu search with ranking candidate list to solve production planning problems with setups. *Computers and Industrial Engineering*, 45(5) :615–634, 2003.

[83] Y.F. Hung and K.L. Chien. A multi-class multi-level capacitated lot sizing model. *Journal of the Operational Research Society*, 51(11) :1309–1318, 2000.

[84] Y.F. Hung, C.C. Shih, and C.P. Chen. Evolutionary algorithm for production planning problems with setup decisions. *Journal of the Operational Research Society*, 50(8) :857–866, 1999.

[85] F.R. Jacobs and B.M. Khumawala. Simplified procedure for optimal single-level lot sizing. *Production and Inventory Management*, 28(3) :39–43, 1987.

[86] R. Jagannathan and M.R. Rao. A class of deterministic planning problems. *Management science*, 19(11) :1295–1300, 1973.

[87] K. Jain and E.A. Silver. Lot sizing for a product subject to obsolescence or perishability. *European Journal of Operations Research*, 75 :287–295, 1994.

[88] R. Jans and Z. Degraeve. Improved lower bounds for the capacitated lot sizing problem with set up times. *Operations Research Letters*, 32(2) :185–195, 2004.

[89] R. Jans and Z. Degraeve. Meta-heuristics for dynamic lot sizing : A review and comparison of solution approaches. *European Journal of Operational Research*, To appear :–, 2005.

[90] S. Kang, K. Malik, and L.J. Thomas. Lotsizing and scheduling on parellel machines with sequence-dependent setup costs. *Management Science*, 45 (2) :273–289, 1999.

[91] B. Karimi, S.M.T. Fatemi Ghomi, and J.M. Wilson. The capacitated lot sizing problem : a review of models and algorithms. *Omega*, 31(5) :365–378, 2003.

[92] U.S. Karmarkar and S. Kekre. The dynamic lot-sizing problem with startup and reservation costs. *Operations Research*, 35 :389–398, 1987.

[93] U.S. Karmarkar and L. Schrage. The deterministic dynamic product cycling problem. *Operations Research*, 33 :326–345, 1985.

[94] E. Katok, H.S. Lewis, and T.P. Harrison. Lot sizing in general assembly systems with setup costs, setup times and multiple constrained resources. *Management science*, 44(6) :859–877, 1998.

[95] J.D. Kelly. Chronological decomposition heuristic for scheduling : A divide & conquer method. *AIChE Journal*, 48(12) :2995–2999, 2002.

[96] Ö. Kirca. An efficient algorithm for the capacitated single item dynamic lot size problem. *European Journal of Operations Research*, 45(1) :15–24, 1990.

[97] Ö. Kirca. A primal-dual algorithm for the dynamic lotsizing problem with joint set-up costs. *Naval Research Logistics*, 42 :791–806, 1995.

[98] Ö. Kirca and M. Kökten. A new heuristic approach for the multi-item dynamic lot sizing problem. *European Journal of Operations Research*, 75(2) :332–341, 1994.

[99] P.R. Kleindorfer and E.F.P. Newson. A lower bounding structure for lot-size scheduling problems. *Operations Research*, 23 (2) :299–311, 1975.

[100] U. Kohlmorgen, H. Schmeck, and K. Haase. Experience with fine-grained parallel genetic algorithms. *Annals of Operations Research*, 90 :203–219, 1999.

[101] J. Krarup and O. Bilde. Plant location, set covering and economic lot sizes : An o(mn) algorithm for structured problems, in optimierung bei graphentheoretischen and ganzzahligen probleme. *L. COLLATZ et al. (eds), Birkhauser Verlag, Basel*, pages 155–180, 1977.

[102] R. Kuik and M. Salomon. Multi-level lot-sizing problem : Evaluation of a simulated annealing heuristic. *European Journal of Operational Research*, 45(1) :25–37, 1990.

[103] R. Kuik, M. Salomon, L.N. Van Wassenhove, and J. Maes. Linear programming, simulated annealing and tabu search heuristics for lotsizing in bottleneck assembly systems. *IIE Transactions*, 25 (1) :62–72, 1993.

[104] M. Laguna. A heuristic for production scheduling and inventory control in the presence of sequence-dependent setup times. *IIE Transactions*, 31(2) :125–134, 1999.

[105] A. Lambert and H. LUSS. Production planning with time-dependent capacity bounds. *European Journal of Operations Research*, 9(3) :275–280, 1982.

[106] M. Lambrecht and J. Vander Eecken. A capacity constrained single-facility dynamic lot-size model. *European Journal of Operational Research*, 2(2) :132–136, 1978.

[107] M. Lambrecht and J. Vanderveken. Heuristic procedures for the single operation, multi-item loading problem. *AIIE Transactions*, 11 (4) :319–326, 1979.

[108] L.S. Lasdon and R.C. Terjung. An efficient algorithm for multi-item scheduling. *Operations research*, 19(4) :946–969, 1971.

[109] C.Y. Lee, S. Çetinkaya, and A.P.M. Wagelmans. A dynamic lot-sizing model with demand time windows. *Management science*, 47 (10) :1384–1395, 2001.

[110] CPLEX Callable Library. Ilog sa. http ://www.ilog.com, 2005.

[111] M. Loparic, H. Marchand, and L.A. Wolsey. Dynamic knapsack sets and capacitated lot-sizing. *Mathematical Programming*, 95(1) :53–69, 2003.

[112] M. Loparic, Y. Pochet, and L.A. Wolsey. The uncapacitated lot-sizing problem with sales and safety stocks. *Mathematical Programming*, 89(3) :487–504, 2001.

[113] V. Lotfi and Y.S. Yoon. An algorithm for the single-item capacitated lot-sizing problem with concave production and holding costs. *Journal of Operational Research Society*, 45 (8) :934–941, 1994.

[114] LOTSIZELIB. http ://www.core.ucl.ac.be :16080/wolsey/lotsizel.htm, 1999.

[115] S.F. Love. Bounded production and inventory models with piecewise concave costs. *Management Science*, 20(3) :313–318, 1972.

[116] S.F. Love. A facilities in series inventory model with nested schedules. *Management Science*, 18(5) :327–338, 1972.

[117] J. Maes, J.O. McClain, and L.N. Van Wassenhove. Multilevel capacitated lotsizing complexity and lp-based heuristics. *European Journal of Operational Research*, 53(2) :131–148, 1991.

[118] J. Maes and L. N. Van Wassenhove. A simple heurisitc for the multi-item single level capacitated lotsizing problem. *Operations Research Letters*, 4(6) :265–273, 1986.

[119] J.M.Y. Leung T.L. Magnanti and R. Vachani. Facets and algorithms for capacitated lot-sizing. *Mathematical Programming*, 45 :331–359, 1989.

[120] T.L. Magnanti and R. Vachani. A strong cutting plane algorithm for production scheduling with changeover costs. *Operations Research*, 38 (3) :456–473, 1990.

[121] A.S. Manne. Programming of economic lot-sizes. *Management Science*, 4 (2) :115–135, 1958.

[122] H. Marchand. *A polyhedral study of the mixed knapsack set and its use to solve mixed integer programs.* PhD thesis, Université Catholique de Louvain - Center for Operations Research and Economics, 1998.

[123] H. Marchand and L.A. Wolsey. The 0-1 knapsack problem with a single continuous variable. *Mathematical Programming*, 85(1) :15–33, 1999.

[124] A. Martel and A. Gascon. Dynamic lot-sizing with price changes and dependent holding costs. *European Journal of Operations Research*, 111 (1) :114–128, 1998.

[125] R.K. Martin. Generating alternative mixed-integer programming models using variable redefinition. *Operations Research*, 35(6) :820–831, 1987.

[126] J.J. De Matteis and A.G. Mendoze. An economic lot sizing technique. *IBM System Journal*, 7(5) :30–46, 1968.

[127] B.J. McLaren. Multi-level lot sizing procedures in a material requirements planning environnemnt. Discussion Paper 64, Division of research, School of Business, Indiana University, 1976.

[128] A. Mehra, I. Minis, and J. M. Proth. Hierarchical production planning for complex manufacturing systems. *Advances in Engineering Software*, 26(3) :209–218, 1996.

[129] C. Mercé and G. Fontan. Mip-based heuristics for capacitated lotsizing problems. *International Journal of Production Economics*, 85 :97–111, 2003.

[130] H. Meyr. Simultaneous lotsizing and scheduling by combining local search with dual reoptimization. *European Journal of Operational Research*, 120(2) :311–326, 2000.

[131] A. J. Miller. *Polyhedral Approaches to Capacitated Lot-Sizing Problems*. PhD thesis, Georgia Institute of Technology, 1999.

[132] A. J. Miller, G.L. Nemhauser, and M.W.P. Savelsbergh. On the polyhedral structure of a multi-item production planning model with setup times. *Mathematical Programming*, 94 :375–405, 2003.

[133] A.J. Miller, G.L. Nemhauser, and M.W.P. Savelsbergh. On the capacitated lot-sizing and continuous 0-1 knapsack polyhedra. *European Journal of Operational Research*, 125(2) :298–315, 2000.

[134] A.J. Miller, G.L. Nemhauser, and M.W.P. Savelsbergh. Solving multi-item capacitated lot-sizing problems with setup times by branch-and-cut. Technical Report 0039, Université Catholique de Louvain - Center for Operations Research and Economics, 2000.

[135] A.J. Miller, G.L. Nemhauser, and M.W.P. Savelsbergh. A multi-item production planning model with setup times : Algorithms, reformulations, and polyhedral characterization for a special case. *Mathematical Programming*, 95(1) :71–90, 2003.

[136] M. Minoux. *Programmation Mathématique*. Dunod, France, 1983.

[137] S. Nahmias and S. WANG. A heuristic lot-size reorder point model for decaying inventories. *Management science*, 25 :90–97, 1979.

[138] G. L. Nemhauser and L. A. Wolsey. A recursive procedure for generating all cuts for 0-1 mixed integer programs. *Mathematical Programming*, 46 :379–390, 1990.

[139] J.B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2) :338–350, 1993.

[140] L. Özdamar and G. Barbarosoglu. Hybrid heuristics for the multi-stage capacitated lot sizing and loading problem. *Journal of the Operational Research Society*, 50(8) :810–825, 1999.

[141] L. Özdamar and G. Barbarosoglu. An integrated lagrangean relaxation. simulated annealing approach to the multi-level multi-item capacitated lot sizing problem. *International Journal of Production Economics*, 68(3) :319–331, 2000.

[142] L. Özdamar and S.I. Birbil. Hybrid heuristics for the capacitated lot sizing and loading problem with setup times and overtime decisions. *European Journal of Operational Research*, 110(3) :525–547, 1998.

[143] L. Özdamar, S.I. Birbil, and M.C. Portmann. Technical note : New results for the capacitated lot sizing problem with overtime decisions and setup times. *Production Planning and Control*, 13(1) :2–10, 2002.

[144] L. Özdamar and M.A. Bozyel. The capacitated lot sizing problem with overtime decisions and setup times. *IIE Transactions*, 32 :1043–1057, 2000.

[145] M.W. Padberg and G. Rinaldi. Optimization of a 532 city symmetric traveling salesman problem by branch and cut. *Operations Research Letters*, 6(1) :1–7, 1987.

[146] O. Pereira and L.A Wolsey. On the wagner-within lot-sizing polyhedron. *Mathematics of Operations Research*, 26 (3) :591–600, 2002.

[147] Y. Pochet. Valid inequalities and separation for capacitated economic lot sizing. *Opereration Research Letters*, 7(3) :109–115, 1988.

[148] Y. Pochet and M. Van Vyve. A general heuristic for production planning problems. *INFORMS Journal on Computing*, 16 :316–327, 2004.

[149] Y. Pochet and L. A. Wolsey. Solving multi-item lot-sizing problems using strong cutting planes. *Management Science*, 37(1) :53–67, 1991.

[150] Y. Pochet and L.A. Wolsey. Lot-size models with backlogging : strong reformulations and cutting planes. *Mathematical Programming*, 40 :317–335, 1988.

[151] Y. Pochet and L.A. Wolsey. Lot-sizing with constant batches : Formulation and valid inequalities. *Mathematics of Operations Research*, 18 :767–785, 1993.

[152] Y. Pochet and L.A. Wolsey. Polyhedra for lot-sizing with wagner-whitin costs. *Mathematical Programming*, 67 :297–323, 1994.

[153] B.T. Polyak. Minimization of unsmooth functionals. *Soviet Mathematics*, 8 :593–597, 1967.

[154] B.T. Polyak. Minimization of unsmooth functionals. *USSR Computational Mathematics and Mathematical Physics*, 9 :14–29, 1969.

[155] E.P. Robinson and L. Gao. A dual ascent procedure for multiprodut dynamic demand coordinated replenishment with backlogging. *Management science*, 42(11) :1556–1564, 1996.

[156] M.W. Padberg T.J Van Roy and L.A Wolsey. Valid linear inequalities for fixed charge problems. *Operations Research*, 33 :842–861, 1985.

[157] M. Salomon, L.G. Kroon, R. Kuik, and L.N. Van Wassenhove. Some extensions of the discrete lotsizing and scheduling problem. *Management Science*, 37 :801–812, 1991.

[158] M. Salomon, R. Kuik, and L.N. Van Wassenhove. Statistical search methods for lotsizing problems. *Annals of Operations Research*, 41 :453–468, 1993.

[159] R.A. Sandbothe and G.L. Thompson. A forward algorithm for the capacitated lot size model with stockouts. *Operations Research*, 38 (3) :474–486, 1990.

[160] Y. Shah. An order level lot-size inventory model for deteriorating items. *AIIE Transactions*, 9 :190–197, 1977.

[161] D.X. Shaw and A.P. Wagelmans. An algorithm for single-item capacitated economic lot sizing with piecewise linear production costs and general holding costs. *Management Science*, 44 (6) :831–838, 1998.

[162] E.A. Silver and H.C. Meal. A heuristic for selecting lot size quantities for the case of a deterministic time-varying demand rate and discrete opportunities for replenishment. *Production and Inventory Management*, 2nd Quarter :64–74, 1973.

[163] N.C. Simpson. Questioning the relative virtues of the dynamic lot sizing rules. *Computers and Operations Research*, 28 :899–914, 2001.

[164] H. Stadtler. Multilevel lot sizing with setup times and multiple constrained resources : Internally rolling schedules with lot-sizing windows. *Operations Research*, 51(3) :487–502, 2003.

[165] A.T. Staggemeier and A.R. Clark. A survey of lot sizing and scheduling models. Technical report, The 23rd Annual Symposium of the Brazilian Operational Research Society, 2001.

[166] C. Suerie and H. Stadtler. The capacitated lot-sizing problem with linked lot sizes. *Management Science*, 49 :1039–1054, 2003.

[167] P.R. Tadikamalla. An eoq model for items with gamma distributed deterioration. *AIIE Transactions*, 10 :100–103, 1978.

[168] O. Tang. Simulated annealing in lot sizing problems. *International Journal of Production Economics*, 88 :173–181, 2004.

[169] H. Tempelmeier and M. Derstroff. A lagrangean based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. *Management Science*, 42 :738–757, 1987.

[170] H. Tempelmeier and S. Helber. A heuristic for dynamic multi-item multi-level capacitated lotsizing for general product structures. *European Journal of Operational Research*, 75(2) :296–311, 1994.

[171] J.M. Thizy and L.N. Van Wassenhove. Lagrangean relaxation for the multi-item capacitated lot-sizing problem : A heuristic implementation. *IIE Transactions*, 17(4) :308–313, 1985.

[172] W. Trigeiro L.J. Thomas and J.O. McLain. Capacitated lotsizing with setup times. *Management Science*, 35 :353–366, 1989.

[173] F.M.B. Toledo and V.A. Armentano. A lagrangian-based heuristic for the capacitated lot-sizing problem in parallel machines. *European Journal of Operational Research*, To appear :–, 2005.

[174] R. Vachani. Performance of heuristics for the uncapacitated lot-size problem. *Naval Research Logistics*, 39 :801–813, 1992.

[175] W. van den Heuvel and A.P.M. Wagelmans. An efficient dynamic programming algorithm for a special case of the capacitated lot-sizing problem. *Computers and Operations Research*, To appear :–, 2005.

[176] C.P.M. van Hoesel. *Models and algorithms for single-item lot sizing problems.* PhD thesis, Erasmus Universiteit, Rotterdam, 1991.

[177] C.P.M. van Hoesel, A.P.M.Wagelmans, and L.A.Wolsey. Polyhedral characterization of the economic lot-sizing problem with start-up costs. *Journal of Discrete Mathematics*, 7(1) :141–151, 1994.

[178] C.P.M. van Hoesel and A.P.M. Wagelmans. An $o(t^3)$ algorithm for the economic lotsizing problem with constant capacities. *Management Science*, 42(1) :142–150, 1996.

[179] T.J. van Roy and L.A. Wolsey. Valid inequalities for mixed 0-1 programs. *Discrete Applied Mathematics*, 14(2) :199–213, 1986.

[180] M. van Vyve. Algorithms for single item constant capacity lotsizing problems. Technical Report 0007, Université Catholique de Louvain - Center for Operations Research and Economics, 2003.

[181] M. van Vyve and F. Ortega. Lot-sizing with fixed charges on stocks : the convex hull. *Discrete Optimization*, 1 :189–203, 2004.

[182] G.J.J. van Zyl. *Inventory Control for Perishable Commodities*. PhD thesis, University of North Carolina, 1964.

[183] F. Vanderbeck. Lot-sizing with start-up times. *Management Science*, 44 :1409–1425, 1998.

[184] A. F. Veinott. *Optimal Ordering, Issuing and Disposal of Inventory with Known Demand*. PhD thesis, Columbia University, 1960. Unpublished Ph.D. Dissertation.

[185] A. F. Veinott. Unpublished class notes for the program in operations research at stanford university. -, 1963.

[186] E. Vicens, M. E. Alemany, C. Andrés, and J. J. Guarch. A design and application methodology for hierarchical production planning decision support systems in an enterprise integration context. *International Journal of Production Economics*, 74 :5–20, 2001.

[187] S. Voß and D.L. Woodruff. *Introduction to computational optimization models for production planning in a supply chain*. Springer, Germany, 2003.

[188] A. Wagelmans, C.P.M. Van Hoesel, and A. Kolen. Economic lot sizing : an $o(n \log n)$ that runs in linear time in the wagner-whitin case. *Operations Research*, 40 :S145–S156, 1992.

[189] H. M. Wagner and T. M. Whitin. A dynamic version of the economic lot size model. *Management Science*, 5 (1) :89–96, 1958.

[190] L.A. Wolsey. Valid inequalities and superadditivity for 0/1 integer programs. *Mathematics of Operations Research*, 2 :66–77, 1977.

[191] L.A. Wolsey. Uncapacitated lot-sizing problems with start-up costs. *Operations Research*, 37(5) :741–747, 1989.

[192] L.A. Wolsey. Progress with single-item lot-sizing. *European Journal of Operational Research*, 86(3) :395–401, 1995.

[193] L.A. Wolsey. Mip modelling of changeovers in production planning and scheduling problems. *European Journal of Operational Research*, 99(1) :154–165, 1997.

[194] L.A. Wolsey. *Integer Programming*. New York : Wiley, 1998.

[195] L.A. Wolsey. Solving multi-item lot-sizing problems with an mip solver using classification and reformulation. *Management Science*, 48(12) :1587–1602, 2002.

[196] J. Xie and J. Dong. A heuristic genetic algorithm to general capacitated lot-sizing problems. *Computers and Mathematics with Applications*, 44 :263–276, 2002.

[197] A. Zahorik, L.J. Thomas, and W. Trigeiro. Network programming models for production scheduling in multi-stage, multi-item capacitated systems. *Management science*, 30(3) :308–325, 1984.

[198] W.I. Zangwill. A deterministic multi-period production scheduling model with backlogging. *Management Science*, 13 :105–119, 1966.

[199] W.I. Zangwill. Minimum concave cost flows in certain networks. *Management Science*, 14 :429–450, 1968.

[200] W.I. Zangwill. A backlogging model and a multi-echelon model of a dynamic economic lot size production system - a network approach. *Management Science*, 15 (9) :506–527, 1969.

Résumé

Le problème de la planification de la production dans une entreprise de manufacture peut être défini comme étant un ensemble de décisions, qui doivent être prises à court et à moyen terme afin de convertir des matières premières en produits finis pour satisfaire la demande à moindre coût. Le problème de lot-sizing prend une place importante dans cet ensemble de décisions. En effet, il permet de décider des quantités à produire sur chaque ligne de production pour un horizon de planification discret. Dans les applications industrielles, il existe plusieurs facteurs qui peuvent compliquer un tel ensemble de décision. En effet, la présence de plusieurs références pouvant consommer d'une même ressource limitante rend le problème plus complexe. Ceci peut provoquer des ruptures sur la demande des lors que la capacité en ressource disponible n'est pas suffisante pour satisfaire toute la demande. Le stock de sécurité est également une contrainte qui peut compliquer le problème puis que c'est souvent un objectif ou une cible à atteindre plutôt qu'une contrainte industrielle à respecter.

Dans cette thèse, nous traitons essentiellement des problèmes de lot-sizing à capacité finie, impliquant plusieurs références, des coûts de setup, des coûts de rupture sur la demande ainsi que des coûts de déficit sur le stock de sécurité. Nous proposons un modèle mathématique incluant ces nouvelles contraintes. Trois approches sont abordées, nous proposons un algorithme polynomial pour résoudre le problème à une seule référence et sans contrainte de capacité. En se basant sur cet algorithme, nous développons une méthode de relaxation lagrangienne des contraintes de capacité afin de trouver une borne inférieure au problème traité. La seconde approche est un algorithme de branch-and-cut basé sur une étude polyédrique du problème de lot-sizing à capacité finie, impliquant plusieurs références, des coûts de setup et des coûts de rupture sur la demande. La dernière approche est une heuristique dont le but est de traiter des problèmes plus complexes et de plus grande taille, c'est une hybridation d'une approche de décomposition de l'horizon de planification et d'une méthode de branch-and-bound. Différents tests sont effectués afin de montrer l'efficacité et les limites de chaque approche.

Mots clés : Planification de la production, lot-sizing, inégalités valides, programmation dynamique, relaxation lagrangienne, heuristiques, branch-and-cut, programmation linéaire mixte, modélisation.

Abstract

The production planning problem in manufacturing environments can be defined as a set of decisions that must be taken in short and medium term in order to convert row materials into end products in order to satisfy the demand with a minimum cost. The lot-sizing problem takes an important place in the set of decisions to be made. In fact, it allows to decide which quantity to produce over each production line on a discrete horizon. In industrial applications, there are several factors that can complicate such a decision set. The presence of several items that can consume the same restricted resource makes the problem more complex. Indeed, this can create shortage on demand when we are in lack of capacity to produce the total demand. Safety stock is also a complicating constraint since it is generally an objective or a target to reach rather that an industrial constraint to respect.

In this thesis, we deal mainly with multi-item capacitated lot-sizing problems with setup times, shortages on demand and safety stock deficit costs. We propose a new mathematical model that includes these new constraints. Three approaches are considered. We propose a polynomial algorithm to solve the single-item uncapacitated version of the problem. Based on this algorithm, we develop a lagrangean relaxation of the capacity constraints to find a lower bound to the main problem. The second approach is a branch-and-cut algorithm. It is based on a polyhedral study of the multi-item capacitated lot-sizing problem with setup times and shortage on demand costs. The last approach is a MIP-based heuristic. The goal is to find a feasible solution for more complex problems with a bigger size. This method is an hybridization of a branch-and-bound method and a strategy of horizon decomposition. Experimental results showing the effectiveness and the limit of each approach are presented.

Keywords : Production planning, lot-sizing, valid inequalities, dynamic programming, lagrangean relaxation, heuristic, branch-and-cut, mixed integer programming, models.