SOMMAIRE

LISTE DES FIGURES

Liste des tableaux

Liste des algorithmes

Liste des abréviations

ATSP Asymmetric Traveling Salesman Problem. 90, 91, 97

B2B Business To Business. 5, 106, *Glossaire :* B2B

B2C Business To Consumer. 5, *Glossaire :* B2C

CBN Calcul des besoins nets. 6

CLSD Capacitated Lot-Sizing with Sequence Dependent setups. 33, 111

CLSD-PM Capacitated Lot-sizing with Sequence Dependent setups on parallel machines. 30, 34

CLSP Capacitated Lot-Sizing Problem. 22–25, 27, 28, 30, 33

CLSSD-PM Capacitated lot-sizing problem with lost sales, Safety stock and Sequence Dependent setup times on Parallel Machines. 35, 85, 88, 99, 100, 102, 107, 109, 114

CPU Central Processing Unit. 53, 75, 94

DLC Date Limite de Consommation. 9, 10

DLUI Date Limite D'Utilisation Interne. xvii, 10

DLUO Date Limite d'Utilisation Optimale. 9, 10

DRP Distribution Resource Planning. 6

ERP Enterprise Ressource Planning. 5

F&O Fix-and-Optimize. 27–29, 32, 65, 66, 80, 109, 110

FIFO First In, First Out. 26, 36

IAA Industrie Agroalimentaire. 2

LB Lower Bound (Borne inférieure). 64

MES Manufacturing Execution System. 5

MIP Mixed Integer Program. 20, 24, 53, 61, 94

MTZ Miller-Tucker-Zemlim. 31, 42

PDP Programme Directeur de Production. 5–9, 109, 111

R&F Relax-and-Fix. 26, 27, 32, 61–67, 76, 80, 109

RFFO Relax-and-Fix et Fix-and-Optimize. 66, 76, 80

S&OP Sales And Operations Planning. xvii, 5, 6

SCF Single Commodity Flow. 42

SCP Supply Chain Planning. 5

SD Sequence Dependent. 67

SI Sequence Independent. 67

TBO Time Between Orders. 33

TSP Traveling Salesman Problem. 31, 41, 42

UB Upper Bound (Borne supérieure). 64

USILSP Uncapacitated Single Item Lot Sizing Problem. 19

ZIO Zero Inventory Ordering. 19

GLOSSAIRE

B2B : Business To Business (B2B) est un terme désignant les activités d'une entreprise visant d'autres entreprise. 5

B2C : Business To Consumer (B2C) est un terme désignant les activités d'une entreprise visant les consommateurs. 5

Branch-and-Bound : méthode de résolution basée sur une énumération des solutions admissibles pour un problème donné. 20, 21, 24, 25

Branch-and-Cut : méthode de résolution basée sur un Branch-and-Bound dans lequel nous ajoutons dans inégalités valides pour couper les solutions relâchés. 24, 35, 41, 46, 48, 53, 56, 58, 59, 75, 94

Branch-and-Price : méthode de résolution basée sur un Branch-and-Bound dans lequel nous appliquons une génération de colonnes à chaque noeud de l'arbre d'énumération. 25

Cannibalisation : terme utilisé pour désigner le fait qu'un nouveau produit va provoquer une diminution de la vente des autres produits. 5

Capacité allouée : temps de production qui est définie par lignes dans le S&OP comme étant disponible pour la planification de l'atelier. 13

Capacité dure : temps de production, supérieure à la capacité allouée, qui ne peut pas être dépassé par le planificateur. 13

Dégagement : terme dans l'agroalimentaire pour désigner le fait de jeter, vendre à perdre ou donner des aliments qui ne sont plus acceptable pour la vente à cause du dépassement de la Date Limite D'Utilisation Interne (DLUI). 2, 6, 10

Gap : terme anglais définissant un écart de la solution obtenue avec une borne inférieure exprimé en pourcentage. 53

INTRODUCTION

De nos jours, de nombreuses entreprises sont bien conscientes que maîtriser leur chaîne logistique (*Supply Chain* en anglais) est crucial pour augmenter leurs profits et rester compétitif. Cette maîtrise des flux physiques et des coûts qui leur sont associés passe par une bonne gestion des étapes allant de la fourniture des matières premières jusqu'au consommateur final en passant par la production. Ceci est particulièrement visible ces dernières années où l'on observe des pénuries de certaines matières premières qui amène à des ruptures de stock de masse [1]. De même, l'étape de distribution, dans notre économie mondialisée, amène à de nouveaux enjeux techniques où un ralentissement de la distribution peut avoir des conséquences au niveau mondial. La Figure 0.1 donne l'exemple d'une chaîne logistique type et des étapes la composant. La nature séquentielle de ces étapes implique qu'une défaillance au niveau d'un acteur peut impacter la performance de l'ensemble de la chaîne logistique. Cela peut se produire par exemple lors d'un défaut de flux physique mais aussi lors d'un défaut de flux d'information. Pour mieux maîtriser ces différentes opérations, les entreprises essayent donc de s'adapter et se mettent à utiliser différents outils informatiques pour échanger les informations de façon plus rapide, fiable et sécurisée.

Dans ce manuscrit, nous concentrons nos recherches sur l'étape de production et de stockage dans la chaîne logistique (étapes 2 et 3 sur la Figure 0.1). Planifier la production consiste à déterminer quand, comment et en quelle quantité produire. Plus précisément, construire un plan de production revient à déterminer la production d'un ensemble de produits distincts sur un horizon de temps futur en tenant compte des ressources (limitées) disponibles et de façon à satisfaire les demandes sur cet horizon. Cette tâche est donc particulièrement importante pour les industriels afin de mettre en adéquation leur production avec une demande fluctuante, en évitant les dérives susceptibles d'entraîner des ruptures ou, à l'inverse, un excès de stock. Ainsi il peut être difficile de construire un plan permettant à chaque instant d'éviter les ruptures de stock ou à l'inverse de surstocker. De plus, la concurrence amène constamment les industriels à réduire leurs marges ce qui augmente encore davantage les pertes en cas de mauvaise gestion de la production.

1. En 2021, nous avons observé par exemple qu'une pénurie de certains composants électroniques peut entraîner un ralentissement important de la production de véhicules ([75])

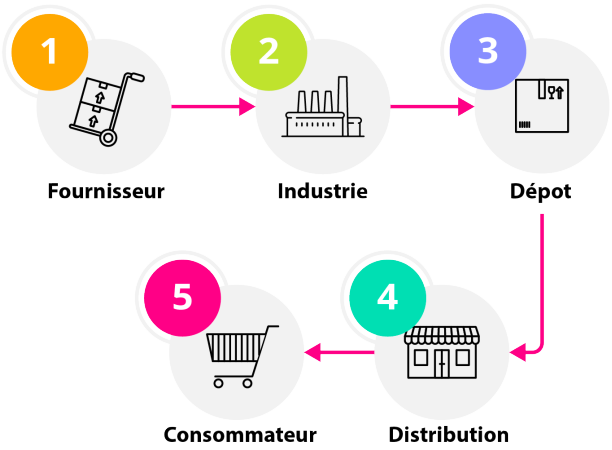


FIGURE 0.1 – Exemple d'une chaîne logistique d'approvisionnement ([90])

Au delà des coûts, les nouveaux enjeux liés à la limitation des ressources planétaires encouragent aussi à mesurer la performance de leur politique de production sur la réduction du gaspillage ainsi que sur la réduction de la consommation énergétique. Dans l'industrie agroalimentaire (IAA), ceci est d'autant plus vrai lorsque l'on sait que l'une des problématiques majeures de ce secteur concerne le gaspillage alimentaire. Ce gaspillage, au niveau des industriels, est souvent dû à une mauvaise gestion des stocks ou à une mauvaise anticipation de la demande future dans la construction du plan de production. Il est en effet courant que la demande de certains produits fluctue grandement dans le temps [2] alors même que les produits peuvent avoir des dates de péremption relativement courtes. En général, dans l'IAA, deux critères sont utilisés pour déterminer si un plan de production est de bonne qualité : le taux de service qui correspond au pourcentage de la demande assurée par la production et le taux de *dégagement* [3] désignant le pourcentage de produits non distribués. Cette vision assez simplifiée vient du fait qu'il est en réalité difficile de prendre en compte l'ensemble des informations venant du plan de production. Les industriels se base alors uniquement sur ces deux critères pour évaluer la qualité de leur plan de production. L'utilisation d'outils d'optimisations trouve alors une place im-

2. Certains produits ont une forte demande uniquement pendant une période de temps court, par exemple à Noël, il faut alors prévoir la demande pour assurer la satisfaction client et réduire le gaspillage.

3. Dans l'ensemble du manuscrit, les mots décrient dans le glossaire sont écrits en italique lors de leurs premières apparitions

portante dans ce secteur en permettant d'avoir une vision plus juste des différents critères pouvant affecter la qualité du plan de production. De nos jours, beaucoup d'industriels continuent de définir les étapes de leur chaînes logistiques sans outils d'optimisation dédiés. La marge de progression de ces derniers est donc importante et l'utilisation de ces outils peut à terme permettre d'améliorer considérablement les rendements et jouer un rôle crucial dans les défis environnementaux. Nous allons dans la suite de ce manuscrit, présenter des méthodes et algorithmes d'aide à la décision pour améliorer les plans de production dans le secteur de l'IAA.

Ce manuscrit est organisée en 6 chapitres. Le chapitre 1 présente le contexte plus précis de ma thèse dans le cadre de la collaboration avec VIF. On y présente notamment les problématiques rencontrées par les industriels de l'agroalimentaire, ainsi que les solutions actuellement proposées par VIF pour les accompagner dans l'amélioration de leurs chaînes logistiques. Le chapitre 2 présente un état de l'art des problèmes de lot-sizing. En particulier, nous présenterons plusieurs extensions de ce problème et les méthodes de résolution utilisées. Dans le chapitre 3, nous proposons plusieurs modélisations mathématiques ainsi qu'une génération d'instances. Nous adaptons également un ensemble d'inégalités valides classiques au problème que nous traitons. Dans la dernière partie de ce chapitre, nous présentons des premiers résultats de nos modélisations avec l'utilisation d'un solveur de programmation linéaire en nombres entiers. Nous montrons cependant que les résultats ne permettent pas d'envisager une application industrielle directe. De ce fait, dans le chapitre 4, nous introduisons une heuristique constructive ainsi qu'une heuristique de recherche locale. Nous proposons de plus une nouvelle approche de pré-traitement basée sur de l'apprentissage machine. Cette approche, que nous appelons *clustering*, permet de réduire significativement la taille du problème en agrégeant certaines informations. Le chapitre 5 se concentre sur des approches alternatives envisagées au cours de la thèse. Finalement dans le chapitre 6, nous proposons des modélisations de certaines problématiques additionnelles rencontrées chez quelques clients.

Chapitre 1

Contexte et enjeu de l'étude

1.1 Présentation de VIF

Ma thèse a été réalisée en collaboration avec VIF, une entreprise spécialisée dans la création de logiciels dédiés à l'industrie agroalimentaire. VIF compte à ce jour plus de 500 clients couvrant le secteur de l'agroalimentaire. Ces entreprises, que l'on nommera producteurs dans la suite, travaillent soit en *Business To Business (B2B)* soit en *Business To Consumer (B2C)*, ce qui amène à des pratiques différentes et variées. Plusieurs suites logicielles sont commercialisées par VIF pour y répondre dont les suites Enterprise Ressource Planning (ERP), Manufacturing Execution System (MES) et Supply Chain Planning (SCP). Mon travail s'intègre dans la suite SCP dans laquelle des logiciels de planification sont développés. Les solutions proposées dans la suite SCP sont :

Sales And Operations Planning (S&OP) : Outil de simulation qui permet aux entreprises d'élaborer des plans stratégiques sur des périodes longues. Entre autres, les décisions relatives au positionnement sur de nouveaux marchés, la définition d'une estimation des capacités de production, la mise en place de nouvelles lignes de production ou encore de la simulation de différents scénarios de long terme sont possible avec cet outil.

Prévision des ventes : Outil qui permet d'avoir une vision à long terme des ventes pour mieux anticiper les décisions. Les modèles de prévision se basent généralement sur des modèles statistiques ou de machine learning prenant en compte les historiques de ventes et les différents événements susceptibles de les influencer, comme les promotions ou la *cannibalisation* très présentes dans l'agroalimentaire.

Planification : L'offre planification chez VIF propose plusieurs modules imbriqués :

Programme Directeur de Production (PDP) : cet outil permet au planificateur d'élaborer et manipuler le plan de production qu'il construit à l'aide de tableaux et d'outils visuels. Le planificateur peut ainsi construire son plan pour différents produits finis sur plusieurs jours/semaines en fonction de la demande tout en

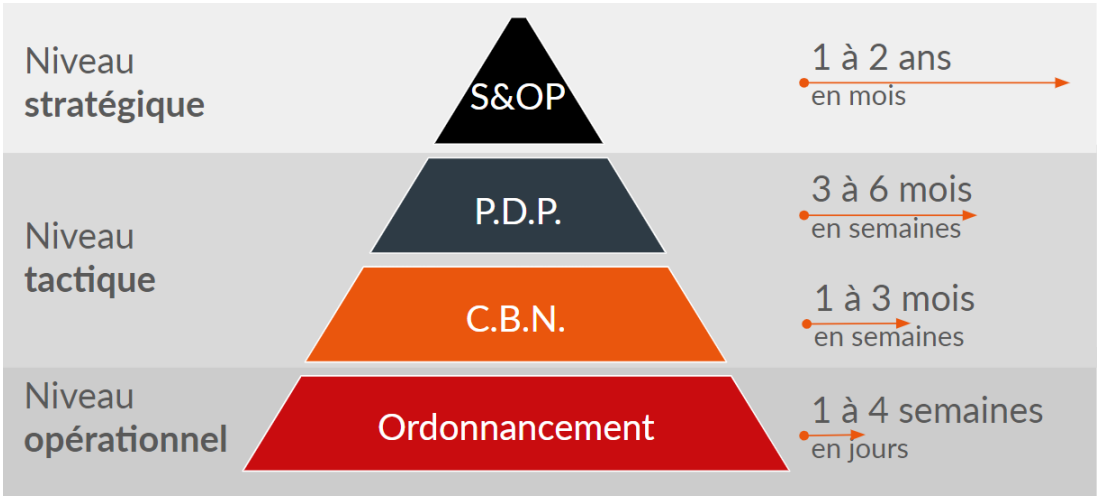


FIGURE 1.1 – Niveau de décision des outils de la Supply Chain VIF

définissant les capacités nécessaires et le stock prévisionnel sur cette même période de temps.

Distribution Resource Planning (DRP) : ce module de planification permet de gérer les flux entre les sites de distribution et de production.

Calcul des besoins nets (CBN) : ce module permet de planifier les approvisionnements des matières premières et de s'assurer de leur disponibilité.

Ordonnancement : ce module est imbriqué dans l'offre planification chez VIF. Il est le dernier maillon et offre une vision plus fine de la production. À partir des objectifs de production définis dans le PDP, le but de l'ordonnancement est de définir les ordres de fabrication sur les différentes lignes de production.

La figure 1.1 montre les différents niveaux de décision du PDP et du S&OP. À ce stade, l'offre Planification possède plusieurs interfaces graphiques pour permettre à l'utilisateur d'avoir une vision globale des éléments à planifier et de mettre en lien les différents modules entre eux. Néanmoins, la demande d'outils d'aide à la décision dans ce domaine est en forte croissance et motive l'ajout de modules d'optimisation. En effet, l'agroalimentaire est un domaine où il est primordial de bien anticiper la variation de la demande client et maîtriser le stock pour éviter d'avoir des baisses de rentabilité, car au-delà des coûts de stockage ou de rupture présents dans beaucoup d'industries, la périssabilité des produits ajoute un risque de dégagement très élevé. Mes travaux de thèse portent donc sur l'étude et l'intégration d'algorithmes d'optimisation dans le module PDP de l'offre Planification. Jusqu'à présent, VIF propose des outils de lissage simple de la production

pour aider les planificateurs à construire leur plan, mais aucune réelle méthode d'optimisation n'est encore intégrée dans le module actuel. L'enjeu principal de ma thèse est donc de développer un outil d'aide à la décision basé sur des techniques mathématiques dédiées qui permette d'améliorer de façon significative les plans de production actuels, pour un pilotage quotidien plus efficace et robuste. L'objectif pour VIF est de déployer le résultat de mes travaux dans le module planification proposé à leurs clients afin d'automatiser la conception de politiques de production et de gestion des stocks performantes. L'une des contraintes majeures de ce projet réside dans la rapidité d'exécution de l'outil, dont les sorties doivent pouvoir être modifiées et paramétrées par les planificateurs a posteriori, par exemple en ajoutant des contraintes additionnelles à une solution existante. Ce besoin s'explique (entre autre) par l'incertitude qui caractérise les demandes de dates lointaines, qui peut amener l'utilisateur à modifier dynamiquement les données d'entrée du programme via la mise à jour des estimations issues du module de prévision associé.

Nous allons nous intéresser dans ce chapitre aux différentes problématiques de planification de production que peuvent rencontrer les producteurs, clients de VIF.

1.2 Description du problème

Comme toute entreprise, les industriels agroalimentaires cherchent constamment à améliorer leurs prises de décision pour limiter certains coûts récurrents dans leurs processus. C'est notamment le cas en production, où garder des produits en stock, surestimer la demande en produisant en trop grande quantité ou à l'inverse avoir une rupture de stock (ce qui engendre un mauvais taux de service) sont les principaux problèmes que les planificateurs veulent éviter. Le PDP a pour but de définir un plan de production sur un horizon de temps fini, découpé en plusieurs mailles temporelles appelées périodes. Ces périodes peuvent correspondre à des jours ou des semaines suivant les pratiques des producteurs et les délais de production. Le plan construit vise à satisfaire la demande de leur(s) client(s) pour toutes les références d'articles sur l'horizon de planification. Il faut pour cela déterminer, à chaque période, quelles références produire et en quelle quantité. Les articles ainsi produits peuvent également être stockés pour satisfaire une demande ultérieure dans les limites de la périssabilité. Dans l'agroalimentaire, on retrouve deux types de produit :

— les produits dit « frais » qui ont une date de péremption courte.

— les produits dit « secs » qui peuvent être stockés sur de longues périodes.

Ainsi, les producteurs de produits frais ont tendance à travailler avec un horizon de temps

de quelques semaines, découpé en jours, tandis que les autres définissent plutôt des plans de plusieurs mois, découpés en semaines.

L'enjeu est alors de planifier les différentes opérations de production sur l'horizon de temps considéré de la meilleure façon, en respectant les contraintes et en minimisant les coûts globaux. L'objectif de mon travail pour VIF est de développer un moteur d'optimisation suffisamment générique et paramétrable pour répondre à tous les producteurs, afin de les aider à définir des plans de production plus efficaces et obtenus plus rapidement que de façon manuelle dans l'outil. Les plans ainsi générés doivent bien sûr respecter les habitudes et les contraintes métiers, afin d'être utilisables en pratique et de nous permettre une comparaison sincère sur les critères de performance retenus.

Dans la suite de ce chapitre, nous exposons les caractéristiques de planification rencontrées dans l'agroalimentaire (qui peuvent aussi se retrouver dans d'autres domaines) en se référant au cas des producteurs utilisant les logiciels proposés par VIF. Certaines problématiques présentées, marquées par le symbole $\star$, sont marginales et spécifiques à quelques producteurs : elles seront traitées uniquement dans le chapitre 6 comme des extensions des modèles proposés au coeur de de la thèse.

1.2.1 Caractéristiques des articles

La demande en multi-références implique de planifier la production de plusieurs références d'articles. Les demandes pour ces différentes références sont disjointes et ne peuvent pas être substituées les unes aux autres, c'est-à-dire qu'une demande pour un article d'une référence particulière ne peut être satisfaite avec un article d'une autre référence. Dans la suite on s'intéressera donc à un problème comportant plusieurs références d'article dont la demande à satisfaire est définie sur chacune des mailles de l'horizon de temps. Ainsi, la demande d'un article correspond à la quantité à fournir pour une période spécifique. Elle peut être décomposée en une demande certaine et une demande estimée. La demande qui est certaine vient d'une part des commandes clients en portefeuille déjà enregistrées et des besoins provenant d'autres sites. La part incertaine de la demande peut venir de la prévision des ventes qui ne tient pas compte de facteurs extérieurs comme les promotions appliquées par les enseignes ou les promotions d'articles similaires concurrents qui peuvent influencer provisoirement la demande d'une référence donnée à la hausse ou à la baisse. Au niveau du PDP, les incertitudes ne sont pas prises en compte directement : nous nous plaçons donc dans un cadre où toutes les demandes sont considérées comme déterministes. Néanmoins, cela implique de redéfinir le plan de production régulièrement en fonction des connaissances sur les ventes futures qui ajusteront la demande dans le

PDP.

La rupture de stock correspond à une situation dans laquelle la quantité d'articles disponible est insuffisante pour satisfaire la demande correspondante sur une période. Cette situation peut notamment se produire lorsque la demande est très forte par rapport aux moyens de production et que l'accumulation de stock en anticipation ne permet pas au producteur d'honorer l'ensemble de ses demandes. On distingue alors deux cas en fonction de la politique choisie, des produits et des exigences vis-à-vis des clients :

- **Les ventes perdues** qui impliquent que la demande non satisfaite est perdue définitivement. Ces ventes perdues entraînent donc une perte de marge pour le producteur, mais aussi des pénalités appliquées par les clients (comme la grande distribution) qui achètent les produits. C'est le cas le plus courant dans l'agroalimentaire.
- **La mise en attente** * qui est le cas des industriels travaillant en B2B, pour lesquels il est possible de reporter une partie de la demande pour la satisfaire à une date ultérieure. Cette demande repoussée doit être distribuée avant une date limite.

Le stock cible correspond à une partie du stock qui est gardée pour servir des demandes futures. En pratique, il est fixé par les planificateurs comme la somme de demandes prévues sur un certain nombre de périodes futures. Ce nombre de périodes est appelé la couverture cible et est calculé de façon empirique par les planificateurs. Pour simplifier leur gestion, il est courant que les planificateurs cherchent à se rapprocher de ce stock cible. Cette volonté vient du fait qu'il peut y avoir une incertitude sur les ventes, mais aussi des pénuries sur les matières premières qui freinent la production à des moments où la demande est haute. Le stock cible permet ainsi de produire en avance une demande, évitant qu'elle ne puisse être satisfaite en cas de d'arrêts de ligne non planifiés. Le stock cible, qui est aussi appelé stock de sécurité, permet donc de limiter les risques liés à des événements imprévus qui pourrait limiter la production.

Example 1.2.1. *En semaine S_1, pour une référence donnée, si la couverture cible est de 2 et les demandes de S_2 et S_3 respectivement de 300 UVC et de 600 UVC, alors le stock cible en fin de semaine S_1 est de 900 UVC.*

La périssabilité d'un produit se caractérise par le temps qui sépare la production de la consommation. Les produits frais ont une Date Limite de Consommation (DLC) relativement proche de leur date de fabrication, alors que les produits secs peuvent être consommés dans un laps de temps plus long et ont généralement uniquement une Date

Limite d'Utilisation Optimale (DLUO) [1]. Dans les deux cas, le planificateur doit distribuer les articles produits en respectant une date appelée la date limite d'utilisation interne que l'on notera dans la suite DLUI. Elle dépend de la date de production et correspond à la dernière date où un produit peut sortir de l'usine pour être distribué aux revendeurs. Cette date est donc inférieure à la date de péremption (DLC ou DLUO) des produits, étant donné qu'il faut d'une part acheminer les produits et d'autre part laisser le temps aux revendeurs d'écouler le stock sans dépasser la date de péremption. Tous les articles produits présentent donc une DLUI avant laquelle ils doivent être distribués (c'est-à-dire sortir du stock), qui dépend de leur date de fabrication et de leur durée d'utilisation interne. Les planificateurs définissent ce qu'on appelle la couverture maximale qui correspond au nombre de périodes pour lesquelles la production peut être utilisée pour satisfaire une demande sans dépasser la DLUI. En cas de dépassement de cette date, les articles concernés ne pourront pas être utilisés pour satisfaire la demande. Dans l'agroalimentaire, on utilise le terme « dégagement » pour désigner ces articles qui sont soit vendus à perte (à des organismes d'aide alimentaire par exemple), soit jetés.

Le stock maximal est ainsi défini par les planificateurs comme étant la quantité de stock à ne pas dépasser pour permettre son utilisation avant sa DLUI et éviter les dégagements. Produire au-delà de cette limite reviendrait en effet à perdre ce qui est produit. Cette pratique vient d'un manque de vision future sur les ventes qui oblige les industriels à fixer une limite sur le stock et donc sur la production. De manière équivalente au stock cible, le stock maximal vient de la couverture maximale correspondant à un nombre de périodes.

Example 1.2.2. *En semaine S_1, pour une référence donnée, si la couverture maximale est de 6 semaines et les demandes cumulées des semaines S_2 à S_7 de 3000 UVC, alors le stock maximale sera de 3000 UVC car toute quantité produite au delà ne pourra, a priori, pas être utilisée pour satisfaire la demande et impliquera donc des dégagements.*

La visualisation du plan de production dans l'outil PDP chez VIF permet de paramétrer les différents éléments décrits plus tôt selon le besoin. La figure 1.2 montre un exemple d'une référence fictive sur un horizon de temps où l'on voit apparaître la demande, la production, le stock et le stock cible.

1. En pratique les articles avec cette mention peuvent être consommés pendant plusieurs années mais le goût peut être altéré.

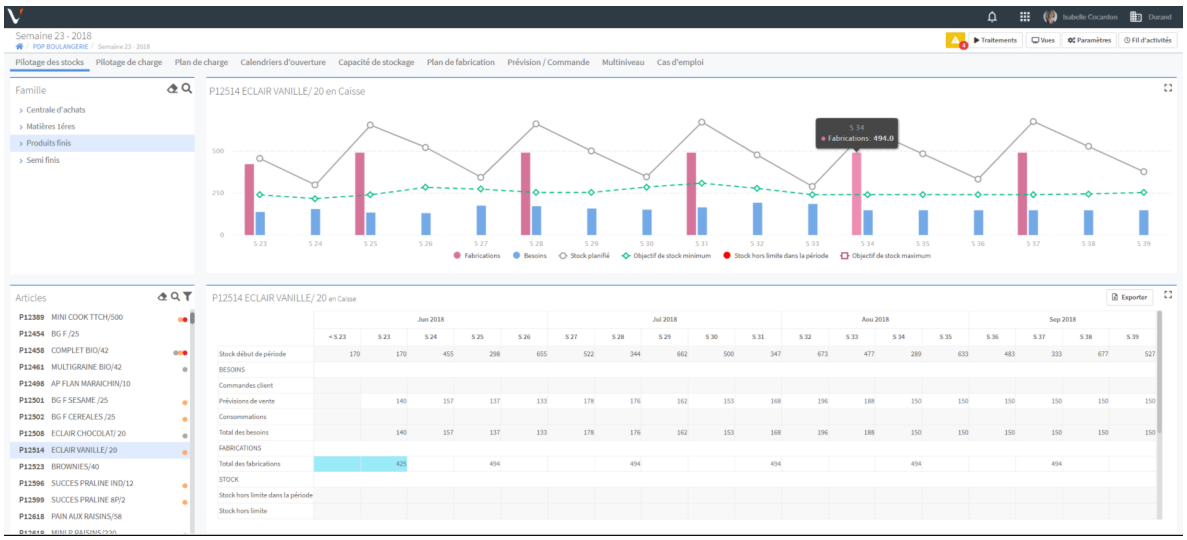


FIGURE 1.2 – Exemple d'affichage du plan de production d'une référence

1.2.2 Contraintes de production sur certaines références

Le lancement minimum d'une référence impose que la production d'un type d'article soit supérieure à une quantité minimum. Cette mesure vise à limiter les petites productions pour des raisons économiques ou des raisons physiques liées à l'atelier.

La production par taille de lot prédéfinie ⋆ contraint la quantité totale en sortie d'atelier à être un multiple d'une quantité de base, aussi appelée batch. Cela permet notamment d'assembler les articles par lots de tailles prédéterminées pour des questions de stockage et/ou de transport.

Example 1.2.3. *Dans un atelier, on peut avoir des cadences en UVC/Heures mais les quantités sont regroupées par colis. On veut alors produire de façon à avoir un multiple de colis. Si un colis contient 20 UVC alors le lot a une taille de 20.*

Le délai de rétention ⋆ s'applique dans certains cas où l'on a besoin de bloquer une partie des unités produites pendant une ou plusieurs périodes pour une référence d'article, par exemple pour des contrôles sanitaires, avant de les rendre disponibles dans le stock. Pendant leur rétention, les unités bloquées ne sont pas comptabilisées dans le stock et ne peuvent pas satisfaire des demandes. Ce délai est constant, propre à chaque référence et s'applique systématiquement à tous les articles concernés.

Les matières premières ⋆ (farine, œufs, sucre, ...) ou les semi-finis (pâte à gâteau par exemple) nécessaires à la fabrication des produits finis présentent souvent des contraintes de disponibilité, dues à leur périssabilité. En effet, ces produits sont acheminés dans les usines par lots (souvent des lots relativement importants comme des citernes). On retrouve alors deux types de dynamiques de flux différentes :

— **Le flux tiré** : le planning de production tire les besoins en semi-finis et matières premières. Le planificateur doit alors décider de l'arrivée des lots dans son planning. Les matières premières doivent être consommées durant la période d'arrivée et ne peuvent pas être stockées.

— **Le flux poussé** : certaines matières premières peuvent avoir des contraintes de réception (lait, viande, légumes) imposées et il faut alors produire en ayant connaissance de ces disponibilités. Les matières premières peuvent être stockées plusieurs périodes (correspondant à la durée d'utilisation de ces produits).

Il est alors nécessaire de prévoir la production en fonction de la disponibilité de ces matières premières.

1.2.3 Caractéristiques des lignes de production

Plusieurs lignes de production sont souvent utilisées au sein d'un même atelier. Cela implique d'assigner la production de chaque référence d'article aux différentes lignes disponibles. Une ligne donnée ne peut pas forcément traiter tous les types de produits (par exemple une ligne de production ne peut pas faire du conditionnement).De plus, la cadence de production d'une référence spécifique dépend de la ligne sur laquelle elle est traitée/produite. On distingue deux types de données possibles pour représenter les cadences des différentes références, en fonction de la méthode de calcul utilisée par les industriels :

– **La cadence nominale** correspond à la cadence théorique de production d'une référence. Cette cadence n'est en pratique pas respectée à chaque instant, car des imprévus peuvent faire ralentir la production comme des pannes, un manque de main-d'œuvre, des maintenances de machine, etc.

– **La cadence moyenne** ⋆ est calculée sur l'intégralité du temps d'ouverture, qui intègre le temps de production, mais également des temps de nettoyage et de réglage des machines induits par le lancement d'une nouvelle production ou le passage d'une référence à une autre sur la ligne considérée. Cette cadence est donc moins précise, car elle ne reflète pas exactement la vitesse de production effective de la machine.

La cadence nominale est principalement utilisée par les industriels matures qui ont des données relativement précises. Dans ce cas, le planificateur devra considérer explicitement les temps pour passer de la production d'un type d'article à un autre. En revanche, certains industriels n'ont pas suffisamment d'informations, par exemple sur les temps de nettoyage des machines, et utilisent donc des cadences moyennes pour estimer les temps de mobilisation des machines.

La capacité de production désigne pour une ligne donnée le temps disponible pour la production sur une période. En pratique, la capacité de production de chaque ligne est limitée par les horaires d'ouvertures des lignes, mais aussi par les horaires de travail des équipes définis lors du S&OP. Cette capacité est utilisée pour la fabrication des articles : ainsi, chaque article mobilise une partie du temps de production disponible sur la ligne considérée en fonction de sa cadence de production, ainsi que les éventuels réglages et opérations préalables sur celle-ci.

Example 1.2.4. *Un atelier travaillant en deux huit (2 équipes de 8 heures par jour) aura donc une capacité de 16 heures théorique par jours. Un produit ayant une cadence de 50 UVC/H pourra être produit dans cette configuration à hauteur de 800 UVC par jour.*

On appelle *capacité allouée* l'estimation du temps de travail nécessaire pour produire la demande qui a été prévue en amont par le S&OP. Cependant, cette capacité doit pouvoir être dépassée pour ajuster la production face à une demande élevée. Cela peut être le cas lorsque la demande d'un produit a été sous-estimée lors des décisions stratégiques. En pratique, ce dépassement reste faible et représente des heures supplémentaires et non une réorganisation des équipes. On parle alors de tolérance de dépassement sur la capacité planifiée. Par opposition, on appelle *capacité dure* la capacité planifiée à laquelle on ajoute la tolérance. Cette capacité ne peut pas être dépassée car elle correspond soit à des limites physiques sur le temps de travail (24h si on travaille en 3×8) soit à des obligations légales sur le temps de travail ou d'ouverture de l'atelier.

L'outil PDP propose une visualisation de la charge comme la figure 1.3 le montre sur un exemple fictif. Nous pouvons constater sur cet exemple que le temps de production peut dépasser la capacité allouée (ligne jaune) dans certains cas. Nous voyons aussi que la production dépasse largement la capacité dure sur la première période. Le planificateur voit ainsi rapidement que le plan qu'il veut mettre en place n'est pas réalisable, et qu'il doit déplacer une partie de cette production sur une autre période et/ou machine.

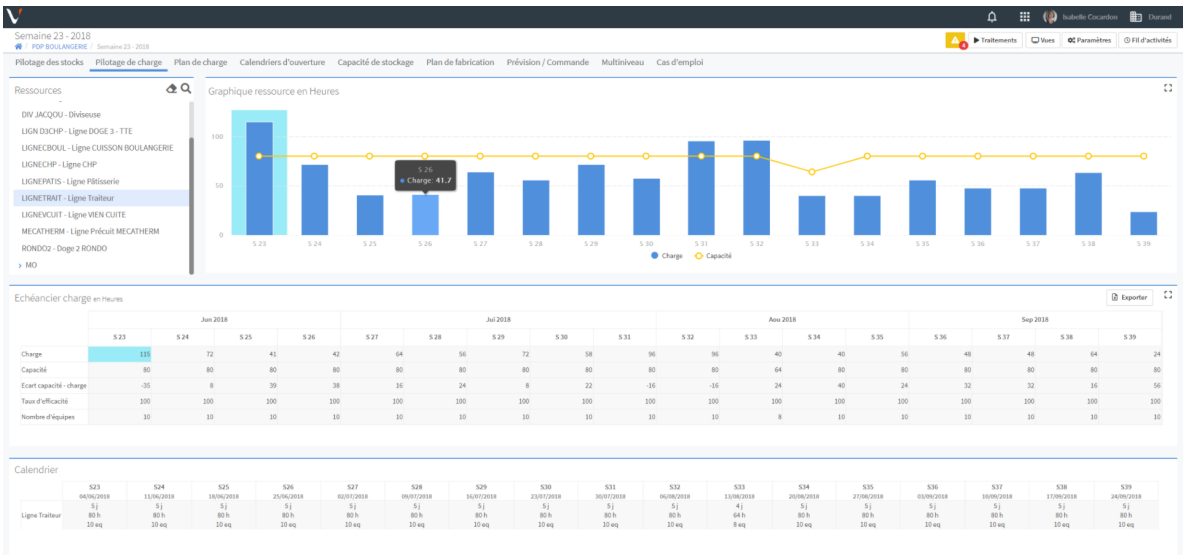


FIGURE 1.3 – Exemple d'affichage du pilotage de charge d'une ligne de production

Le temps de changement correspond à un temps nécessaire pour lancer la production d'une référence, qui nécessite une ou plusieurs opérations (nettoyages, réglages, montée en température des fours, etc.) à effectuer. Souvent, le délai pour passer de la production d'un type d'article A à un autre B est fonction des caractéristiques de la paire (A, B) : on parle alors de temps de changement dépendant de la séquence. Ce dernier n'est pas nécessairement symétrique, c'est-à-dire que le temps requis pour une paire (A,B) peut être différent de celui qui s'applique pour la paire (B,A). Comme expliqué précédemment, ce temps est masqué dans le cas des cadences moyennes. Dans le cas des cadences nominales, le temps net de production et les temps de changement doivent donc être pris en compte séparément dans le temps de mobilisation des lignes de production pour la construction du PDP.

Example 1.2.5. *Un exemple classique dans l'agroalimentaire est celui des allergènes. Le passage d'un produit contenant des allergènes à un autre sans allergènes nécessite un nettoyage très long tandis que la transition entre deux produits dépourvus d'allergènes demande un temps de nettoyage moindre.*

Il est admis dans le cas où la production s'arrête (par exemple en fin de journée ou le dimanche) qu'une équipe est mobilisée en fin de journée pour remettre les lignes de production dans un état neutre après la fin de la production. De même, une équipe peut être mobilisée en début de journée pour les réglages des machines. Le temps de changement nécessaire pour ce procédé n'est pas comptabilisé dans la capacité allouée et n'est donc pas

pris en compte. C'est ce qui est pratiqué par la grande majorité des producteurs clients de VIF.

Le regroupement d'articles * par famille est une pratique commune des planificateurs pour simplifier la construction de leur plan de production sans prendre en compte les temps de changement directement. Sur une durée fixée qui dépend généralement des heures de travail des équipes, on limite la production à des articles d'une unique famille. Cette pratique permet de ne pas considérer les temps de réglages directement en s'assurant d'enchaîner la production d'articles similaires mais reste approximative sur le temps total de production décidé au préalable.

Example 1.2.6. *Dans l'agroalimentaire, on peut retrouver par exemple la famille Cookie qui peut regrouper les cookies chocolat blanc et les cookies chocolat noisette. Dans ce cas, les planificateurs peuvent prévoir une durée fixe dans laquelle seuls les articles de cette famille peuvent être produit. Pour une équipe de travail de 8h, on peut ainsi considérer un temps de production net de seulement 7h et prendre ainsi en compte les temps de réglage de manière implicite. Une autre solution appliquée par les planificateurs consiste à utiliser les cadences moyennes qui intègrent les temps de réglage.*

La durée fixée pour un regroupement correspond en général à une équipe de travail ou une demi-équipe. Ce temps dépend à la fois de la famille d'article, mais aussi de la ligne choisie pour les produire. Les temps de changement entre articles sont considérés comme nuls au sein d'une famille d'articles similaires. De même, on ne considère pas de temps de changement pour passer de la production d'un groupe d'articles d'une famille à une autre, ceux-ci étant directement approximés par la réduction de la plage horaire disponible ou moyennés dans les cadences considérées.

1.3 Deux cas d'étude chez VIF

Les caractéristiques spécifiques à la planification de production dans l'agroalimentaire listées jusqu'à présent se retrouvent chez la plupart des clients de VIF. Cependant, comme nous l'avons énoncé en 1.2, on peut dissocier ces derniers en deux sous-groupes : les fabriquants de produits secs et les fabriquants de produits frais. Nous allons donc détailler les caractéristiques propres aux productions de chacune de ces grandes catégories d'articles qui justifie que notre étude distingue deux classes de problèmes.

Producteurs fabriquant des produits secs ⋆ : La date d'utilisation interne relativement longue des produits secs permet d'en stocker de grandes quantités, afin d'anticiper des demandes à plus long terme tout en conservant un risque de dégagement limité. Ainsi, l'horizon de temps est généralement relativement long (plusieurs mois) avec des mailles de l'ordre de la semaine. Les planificateurs ont donc une vision moins fine de leur plan de production, ce qui leur permet de faire des regroupements dans lesquels il n'y a aucun temps de changement, en les compensant avec des cadences moyennes.

Producteurs fabriquant des produits frais : Les produits frais ont la particularité de ne pas pouvoir être stockés sur de longues périodes et doivent donc être distribués rapidement. La problématique se situe alors à la frontière entre la planification de production et l'ordonnancement où l'objectif est d'une part de définir un plan de production détaillé sur un horizon de temps et d'autre part de proposer une organisation fine de la production par période. Les planificateurs travaillent avec des mailles jours dans lesquelles il est préférable de lancer des petites productions pour répondre à une demande à court terme. Il est alors important d'avoir des informations précises sur les lignes de production sur cette échelle de temps. Les planificateurs utilisent donc des cadences nominales pour estimer les temps de production et prennent en compte les temps de changement dans la capacité des lignes.

Problématique	Cas Frais	Cas Sec
Horizon de temps	plusieurs semaines découpées en jours	plusieurs mois découpés en semaines
Caractéristiques des articles	produit avec DLC courte donc peu d'anticipation de la production et de la demande	produit avec DLUO donc possibilité de stocker pour anticiper la demande
Caractéristiques des lignes	cadence nominale, prise en compte des temps de changement	cadence moyenne, regroupement par famille sans temps de changement

Tableau 1.1 – Tableau récapitulatif des différences entre les deux problèmes

On voit ainsi apparaître deux problèmes distincts, qui partagent la plupart des caractéristiques de planification énoncées précédemment, mais qui se différencient par la nature des articles produits. Dans la suite, nous nous focalisons sur le problème industriel de

production de produits frais qui constituera l'objet central de notre étude. En effet, les autres applications introduites plus haut peuvent généralement être interprétées comme des cas particuliers de ce problème plus complexe. On peut ainsi envisager de les résoudre en s'appuyant sur les méthodes développées dans les chapitres suivants sans modifier fondamentalement leurs mécanismes. Dans le chapitre 6, nous montrerons comment passer de ce problème central au cas des producteurs de produits secs et nous présenterons des modélisations de l'ensemble des caractéristiques identifiées précédemment.

Chapitre 2

État de l'art

Dans ce chapitre, nous présentons un état de l'art des problèmes de planification de la production. Le problème de *lot-sizing* est étudié depuis de nombreuses années et les extensions et méthodes développées sont variées. Le problème de lot-sizing consiste à déterminer le plan de production d'un ou plusieurs articles sur un horizon de temps fini et divisé en T périodes. Ce problème a été formellement décrit par Wagner et Whitin [114] et depuis, il a été largement étudié et enrichi pour s'approcher des problématiques réelles. L'objectif de ce chapitre est de présenter de façon non exhaustive les principales extensions et méthodes de résolution pour ce problème, afin de donner au lecteur une vision globale des contributions scientifiques en lien avec les travaux effectués au cours de cette thèse. L'organisation des sections permettra de proposer une vision globale du problème et de l'évolution des contributions scientifiques.

2.1 Lot-sizing sans capacité

Le problème de lot-sizing sans capacité à une référence, habituellement noté USILSP [1] a été largement étudié depuis sa formalisation par Wagner et Whitin en 1958. Le problème consiste à déterminer la production d'une référence en chaque période. Cette production complétant le stock de l'atelier peut soit servir la demande, soit être disponible pour les périodes futures. La figure 2.1 présente une illustration schématique du problème de lot-sizing. Ces auteurs ont proposé un algorithme qui réduit le problème à la recherche d'un plus court chemin dans un graphe grâce à la propriété de dominance *Zero Inventory Ordering (ZIO)*. Cette propriété indique qu'il existe un plan de production optimal dans lequel une production est effectuée uniquement si le stock est nul. L'espace de recherche est ainsi réduit à $T(T+1)/2$ possibilités. Les auteurs développent un programme dynamique de complexité $O(T^2)$. Par la suite, plusieurs auteurs ont développé des méthodes pour réduire cette complexité grâce à des propriétés structurelles du problème (Wagelmans *et al.* [113], Aggarwal et Park [7]) ou de l'algorithme de résolution (Federgruen et Tzur [41]).

1. Uncapacitated Single Item Lot Sizing Problem

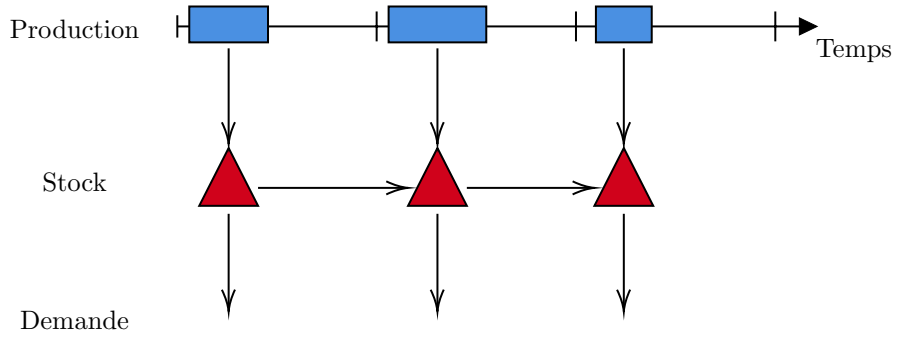


FIGURE 2.1 – Représentation du problème de lot-sizing

Dans tous ces cas, la complexité finale est $O(T \log T)$ et $O(T)$ lorsque les coûts respectent certaines propriétés dites "de Wagner-Whitin". Une autre approche possible est d'utiliser une formulation comme un Programme Linéaire en Nombre Entier, Mixed Integer Program (MIP) en anglais, et de s'appuyer sur une résolution par *Branch-and-Bound* qui consiste à trouver une solution optimale par une énumération intelligente des solutions. Le problème USILSP peut être formalisé sous forme de plusieurs modélisations MIP. Nous en présentons une qui est la plus naturelle pour ce problème. Cette modélisation s'appuie sur plusieurs notations : soit $t = 1, \ldots, T$ l'indice des périodes, nous introduisons d_t la demande en période t de la référence. Produire et stocker implique des coûts à prendre en compte dans la fonction objectif à minimiser. À chaque unité produite et stockée, un coût dit unitaire est appliqué. Nous notons p_t le coût unitaire de production en période t et h_t son coût de stockage. De plus, lancer une production ajoute un coût fixe supplémentaire, qui ne dépend pas de la quantité produite. Ce coût est noté sc_t pour chaque période t. Le modèle MIP doit permettre de déterminer au sein de quelle période produire et en quelle quantité. Nous modélisons ces différentes décisions au moyen des variables suivantes :

— x_t : variables linéaires indiquant la quantité produite en période t.

— y_t : variables binaires égale à 1 si la décision de produire est prise en période t.

— I_t : variables linéaires indiquant le stock en fin de période t.

Le modèle mathématique du USILSP s'écrit alors comme suit :

$$\min \sum_{t=1}^{T} p_t x_t + sc_t y_t + h_t I_t \tag{2.1}$$

$$\text{s.c. } x_t + I_{t-1} - d_t = I_t \quad \forall t = 1, \ldots, T \tag{2.2}$$

$$x_t \leq M_t y_t \quad \forall t = 1, \ldots, T \tag{2.3}$$

$$x_t, I_t \geq 0 \quad \forall t = 1, \ldots, T \tag{2.4}$$

$$y_t \in \{0, 1\} \quad \forall t = 1, \ldots, T \tag{2.5}$$

La fonction objectif (2.1) représente la somme des coûts décrits précédemment. Les contraintes (2.2) correspondent à la conservation des flux physiques sur l'horizon. Elles indiquent que le stock en fin de période t est égal au stock en période $t-1$ auquel on ajoute la différence entre ce qui a été produit et ce qui a été utilisé pour satisfaire la demande d_t. Les contraintes (2.3) indiquent qu'il peut y avoir une production en période t uniquement si la décision de produire a été prise à cette période (grâce à y_t). Dans ce cas, la production est majorée par M_t que l'on peut par exemple fixer à $M_t = \sum_{u=t}^{T} d_u$ pour ne pas limiter la production dans un plan optimal. Finalement les contraintes (2.4) et (2.5) définissent le domaine des variables. Cette formulation est dite agrégée car les variables de production de chaque référence sont définies uniquement par la période de production. Des formulations désagrégées sont évoquées et présentées dans la suite de ce manuscrit. À partir de cette modélisation, le problème peut être résolu grâce à des méthodes de résolution exactes telles que celles présentées dans Wolsey [115] (Branch-and-Bound, programmation dynamique, etc). Le problème de lot-sizing à une référence a reçu beaucoup d'attention dans la littérature et est bien résolu pour de nombreuses extensions que nous présenterons par la suite. Le lecteur peut se référer à Brahimi *et al.* [23] pour un état de l'art approfondi du problème et de ces extensions avec capacité finie sur la production. Plus récemment les mêmes auteurs ont publié une mise à jour de l'état de l'art ([22]) qui inclut les nouvelles contributions depuis cette publication. Bien que ce problème ne permette pas directement de modéliser des problèmes industriels complexes, son étude permet de mettre en évidence des propriétés structurelles qui peuvent être réinvesties pour la résolution de problèmes plus complexes. Ces travaux permettent également d'intégrer de nouveaux aspects dans les problèmes de planification, tels que des considérations environnementales, des incertitudes sur la demande ou sur les ressources de production, des approches inspirées de l'économie circulaire, etc.

Dans le cas de problèmes industriels réels, il arrive fréquemment qu'un atelier soit chargé de la production de plusieurs références. Ce constat a motivé la définition du problème de lot-sizing multi-référence, qui (dans sa version sans contrainte de capacité) peut être modélisé de manière équivalente en introduisant un ensemble N de références et en ajoutant un indice $i = 1, \ldots, N$ sur chaque notation et variable pour désigner les références. À partir de la formulation précédente, nous obtenons le modèle en multi-

référence suivant :

$$\min \sum_{t=1}^{T}\sum_{i=1}^{N}(p_t^i x_t^i + h_t^i I_t^i) + sc_t y_t \tag{2.6}$$

$$\text{s.c. } x_t^i + I_{t-1}^i - d_t^i = I_t^i \quad \forall i = 1,\dots,N,\ \forall t = 1,\dots,T \tag{2.7}$$

$$x_t^i \leq M_t^i y_t \quad \forall i = 1,\dots,N,\ \forall t = 1,\dots,T \tag{2.8}$$

$$x_t^i, I_t^i \geq 0 \quad \forall i = 1,\dots,N,\ \forall t = 1,\dots,T \tag{2.9}$$

$$y_t^i \in \{0,1\} \quad \forall i = 1,\dots,N,\ \forall t = 1,\dots,T \tag{2.10}$$

Dans la suite nous nous concentrons sur le problème en multi-référence pour décrire et modéliser les extensions classiques du lot-sizing et notamment le Capacitated Lot-Sizing Problem (CLSP)) qui reçoit une attention particulière depuis de nombreuses années.

2.2 Lot-sizing avec capacité

Le CLSP est un problème central du domaine de la planification de la production. Il enrichit les modèles précédents en leur ajoutant une limitation de la capacité de production ce qui est le cas dans la quasi-totalité des problèmes industriels réels. Florian et Klein [47] sont les premiers à introduire ce type de contraintes dans le cas à une référence. En introduisant C_t la capacité en période t, le CLSP en multi-référence peut se modéliser comme suit :

$$\min \sum_{t=1}^{T}\sum_{i=1}^{N} p_t^i x_t^i + sc_t^i y_t^i + h_t^i I_t^i \tag{2.11}$$

$$\text{s.c. } \sum_{i=1}^{N} x_t^i \leq C_t \quad \forall t = 1,\dots,T \tag{2.12}$$

$$x_t^i + I_{t-1}^i - d_t^i = I_t^i \quad \forall i = 1,\dots,N,\ \forall t = 1,\dots,T \tag{2.13}$$

$$x_t^i \leq M^i y_t^i \quad \forall i = 1,\dots,N,\ \forall t = 1,\dots,T \tag{2.14}$$

$$x_t^i, I_t^i \geq 0 \quad \forall i = 1,\dots,N,\ \forall t = 1,\dots,T \tag{2.15}$$

$$y_t^i \in \{0,1\} \quad \forall i = 1,\dots,N,\ \forall t = 1,\dots,T \tag{2.16}$$

La contrainte (2.12) indique que la somme de la production des références ne doit pas dépasser la capacité en chaque période. Ce problème a été prouvé comme étant NP-difficile pour le cas à une référence par Bitran et Yanasse [21] et NP-difficile au sens fort pour le cas à plusieurs références ([28]). L'ajout de la capacité engendre donc une

complexité importante pour le problème. Dans le cas particulier d'une capacité constante ($C_t = C$ pour tout t) à une référence, Florian et Klein [47] introduisent une nouvelle propriété de dominance qui leur permet de développer un algorithme de programmation dynamique de complexité $O(T^4)$, basé sur une décomposition de l'horizon en sous-plans de production. Par la suite, van Hoesel et Wagelmans [110] proposent une amélioration de cette procédure pour atteindre une complexité $O(T^3)$. En revanche lorsqu'on considère un problème à plusieurs références, le problème reste NP-difficile au sens fort même avec une capacité constante ([28, 72]).

Dans un cas plus général que celui présenté plus haut, la contrainte de capacité peut modéliser une limite sur le temps de production disponible durant une période et également intégrer les temps de réglage en plus du temps de production proportionnel au nombre d'unités fabriquées. Le problème qui en résulte est appelé CLSP avec *setup times*. Le capacité ne fait alors plus référence à une limite de production mais à une limite de temps de production et la contrainte (2.12) indique que la somme des temps de production unitaire et des temps de setup doit être inférieure à la capacité pour chaque période :

$$\sum_{i=1}^{N} \lambda_t^i y_t^i + \tau_t^i x_t^i \leq C_t \quad \forall t = 1, \ldots, T \tag{2.17}$$

avec τ_t^i et λ_t^i respectivement les temps de production unitaire et le temps de *setup* de la référence i en période t. Karimi *et al.* [65] présentent un état de l'art des modèles mathématiques pour le CLSP. Cette modélisation du CLSP, souvent appelée "Big bucket" ou "Large Bucket", s'oppose à la modélisation en "small bucket". Un problème CLSP small bucket suppose qu'au sein de chaque période de temps un unique article peut être produit. Plusieurs modèles en small bucket existent comme le *Continuous Setup Lot-Sizing Problem* (CSLP) ou le *Discrete Lot-sizing and Scheduling problem* (DLSP). Pour plus d'information sur cette typologie de problème, Drexl et Kimms [39] et Jans et Degraeve [62] proposent des modélisations mathématiques pour des problèmes en small bucket. Dans la suite, nous nous focaliserons sur les modélisations en "Big bucket".

Les formulations agrégées présentées pour le moment sont les plus naturelles pour exprimer les problèmes de lot-sizing. Plusieurs auteurs se sont intéressés à d'autres formulations plus compactes pour modéliser ce problème. Eppen et Martin [40] ont présenté une modélisation mathématique qui se base sur le problème de plus court chemin. Elle permet d'avoir une formulation plus compacte et d'obtenir de meilleures bornes inférieures en règle générale. Dans la même idée, Krarup et Bilde [67] ont proposé une formulation basée sur la localisation de dépôt, souvent appelée formulation *Facility location* ou formu-

lation *désagrégée*. Ces dernières peuvent également être adaptées pour prendre en compte une capacité de production.

Dans la suite, nous présentons les principales méthodes de résolution pour le CLSP. La plupart des méthodes utilisées se basent sur des formulations MIP. De plus, Barany *et al.* [17] ont présenté les inégalités valides (l, S) pour le CLSP. Ajouter des inégalités valides permet de réduire l'espace de recherche en obtenant une meilleure borne inférieure et donc d'accélérer la résolution des modèles mathématiques. Ces inégalités valides sont ajoutées par une méthode de plans sécants (*cutting plane method* en anglais) dans un Branch-and-Bound. L'association de ces deux méthodes se nomme *Branch-and-Cut*. Un algorithme de séparation en $O(NT^2)$ est utilisé pour identifier les inégalités (l, S) les plus pertinentes à ajouter sous forme de plan sécant. Dans Barany *et al.* [16], les mêmes auteurs ont montré que ces inégalités valides définissent l'enveloppe convexe du problème USILSP. Plusieurs auteurs se sont intéressés aux méthodes de Branch-and-Cut et à l'ajout d'inégalités valides ([8, 31, 87]). Nous présentons des adaptations de certaines d'entre elles aux problèmes que nous traitons dans le chapitre suivant.

Le problème étant NP-difficile au sens fort, de nombreux auteurs se sont intéressés à l'utilisation d'heuristiques pour obtenir des solutions réalisables. Maes et Van Wassenhove [78] présentent une classification des heuristiques pour le CLSP. Plusieurs heuristiques se basent sur des modèles mathématiques comme les *LP-based* heuristiques présentées dans Alfieri *et al.* [10]. Trigeiro [107] présente pour sa part une approche basée sur une relaxation Lagrangienne, qui consiste à pénaliser directement dans la fonction objectif la violation d'une ou plusieurs contrainte(s) difficile(s) du problème. Dans le CLSP, relâcher la contrainte de capacité permet de réduire le problème à un sous-ensemble de problèmes USILSP ce qui rend le problème plus simple à résoudre. Trigeiro [107] a utilisé cette approche pour construire une heuristique basée sur les coûts duaux. Plusieurs auteurs ont également présenté des heuristiques basées sur des relaxations Lagrangiennes ([3, 94, 102]). Une autre approche connue, basée sur les modélisations MIP, est la décomposition de Dantzig-Wolfe. Cette approche consiste à transformer le problème de base en le décomposant en deux : d'un côté un problème dit "maître", et de l'autre des sous-problèmes qui viennent l'alimenter. Cette reformulation a pour particularité d'engendrer un nombre exponentiel de variables dans le problème maître de par sa structure. Ainsi comme il n'est pas possible de les ajouter toutes en amont, une génération de colonnes est utilisée pour ajouter les variables les plus prometteuses au problème maître par la résolution successive de sous-problèmes. Pour une utilisation efficace d'une génération de colonnes, il est important que le problème maître et le sous-problème puissent être résolus

rapidement puisqu'ils seront utilisés itérativement afin d'améliorer la solution en ajoutant de nouvelles variables. Pour plus de détails sur cette approche dans un cadre général, Vanderbeck et Savelsbergh [112] font une description détaillée de la méthode. Cette approche est étudiée dans Manne [79] pour le CLSP. Plus récemment, Degraeve et Jans [36] ont présenté une décomposition de Dantzig-Wolfe et un Branch-and-Price pour résoudre le CLSP. Le *Branch-and-Price* consiste à résoudre le problème par la combinaison de la méthode de Branch-and-Bound et d'une génération de colonnes appliquée à chaque noeud de l'arbre d'énumération.

Bien que celles-ci restent minoritaires par rapport aux méthodes exactes évoquées plus haut, on trouve également dans la littérature de nombreuses approches de résolution basées sur des heuristiques. Les algorithmes génétiques et le recuit simulé sont les exemples les plus courant de méta-heuristique appliquées à des problèmes de lot-sizing complexes. Par exemple, Gourgand *et al.* [53] proposent une résolution du CLSP avec une fonction bi-objectif qui se base sur un recuit simulé. Pour une vision plus générale des méthodes de résolution employées pour le CLSP, le lecteur peut se référer à Jans et Degraeve [61], qui présentent une comparaison de plusieurs métaheuristiques pour le problème, ou Buschkühl *et al.* [24], qui donnent une liste détaillée des différentes types de techniques existantes dont les heuristiques constructives, les méta-heuristiques et les approches exactes.

Une extension classique du CLSP est d'ajouter la possibilité de dépasser la capacité au prix d'un coût supplémentaire. On utilise le terme anglophone *overtime* pour désigner le temps d'utilisation au-delà de la capacité. Il est possible de modéliser les *overtimes* simplement en ajoutant une nouvelle variable O_t indiquant la production qui dépasse la capacité et un nouveau paramètre de coûts o_t pour pénaliser le dépassement dans la fonction objectif. La contrainte (2.17) peut être remplacer par :

$$\sum_{i=1}^{N} \lambda_t^i y_t^i + \tau_t^i x_t^i \leq C_t + O_t \quad \forall t = 1, \ldots, T \tag{2.18}$$

Parmi les références de la littérature qui intègrent des *overtimes* à leur modélisation, on peut notamment citer [121, 120, 30, 69].

Il existe encore de nombreuses autres extensions au CLSP, telles que le lot-sizing problem multi-niveaux ou les *demand time windows* proposées dans Belvaux et Wolsey [18] et Jans et Degraeve [62], mais dans la suite nous nous concentrons principalement sur celles qui présentent un lien direct avec le problème industriel que nous souhaitons modéliser.

2.3 Gestion de la rupture de la demande

Prendre en compte les ruptures de stock dans la modélisation de problème de lot-sizing permet de s'approcher de problématiques réelles. Dans la littérature, nous retrouvons deux façons de gérer les ruptures.

2.3.1 Les ventes perdues

Les ventes perdues ou *lost sales* (en anglais) correspondent à la possibilité de ne pas satisfaire la totalité de la demande d'un ou plusieurs articles dans certaines périodes. Nous modélisons cette décision par une nouvelle variable L_t^i qui représente le volume de ventes perdues pour la référence i en période t. D'autre part, toute unité de la demande correspondante non satisfaite engendre un coût l_t^i. On peut formuler ce problème en se basant sur (2.11)-(2.16), où on intègre les ventes perdues à la contrainte d'équilibrage du stock (2.13) :

$$x_t^i + L_t^i + I_{t-1}^i - d_t^i = I_t^i \quad \forall i = 1, \dots, N, \forall t = 1, \dots, T \tag{2.19}$$

On impose également que les ventes perdues soient positives mais inférieures à la demande en ajoutant à l'ensemble des contraintes les inégalités

$$0 \leq L_t^i \leq d_t^i \quad \forall i = 1, \dots, N, \forall t = 1, \dots, T \tag{2.20}$$

Enfin, on intègre le terme additionnel $\sum_{t=1}^{T} \sum_{i=1}^{N} l_t^i L_t^i$ à la fonction objectif (2.11) pour pénaliser la non-satisfaction de certaines demandes. Dans le cas où $l_t^i + h_t^i > l_{t+1}^i$, une solution optimale à ce problème respecte la gestion du stock en First In, First Out (FIFO). Les modèles avec ventes perdues ont été étudiés pour la première fois par Sandbothe et Thompson [98]. Cependant, ce problème n'a pas été beaucoup traité dans la littérature en comparaison des reports de demandes (voir section 2.3.2). Sandbothe et Thompson [99] présentent une version avec une borne sur la capacité de stockage. Liu et Tu [74] ont repris ce problème pour le cas sans capacité et avec une seule référence et ont développé un algorithme polynomial basé sur une approche de *network flow*. Absi et Kedad-Sidhoum [4] ont développé des heuristiques constructives pour traiter un problème de lot-sizing avec des ventes perdues. Les heuristiques constructives se basent sur une résolution successive de sous-problèmes obtenus en découpant l'horizon de temps. L'heuristique Relax-and-Fix (R&F) présentée dans Absi et Kedad-Sidhoum [4] est fréquemment utilisée pour la résolution de problèmes de lot-sizing complexes. Cette heuristique sera présentée plus en

détail dans le Chapitre 4. Les mêmes auteurs présentent des inégalités valides [5] et des heuristiques Lagrangiennes [6] pour des problèmes similaires avec ventes perdues. Dans un contexte stochastique où la demande suit une distribution discrète, Ghamari et Sahebi [50] ont développé une heuristique pour un problème de lot-sizing avec ventes perdues appliqué au domaine de la pétrochimie.

2.3.2 Report de demandes

Dans certains problèmes, il est possible de servir les demandes non satisfaites immédiatement dans une période ultérieure. On parle alors de *mise en attente* (*backlog* en anglais) ou de *report* de demandes, par opposition aux ventes perdues introduites précédemment. Dans ce cas, on ajoute au modèle CLSP une variable B_t^i qui indique la demande de la référence i reportée en période t, ainsi qu'une pénalité unitaire de retard b_t^i. De manière équivalence à la modélisation des ventes perdues, nous pouvons reprendre la formulation (2.11)-(2.16) et intégrer la mise en attente à la contrainte (2.13) :

$$x_t^i + I_{t-1}^i - B_{t-1}^i - d_t^i = I_t^i - B_t^i \quad \forall i = 1, \ldots, N, \forall t = 1, \ldots, T \tag{2.21}$$

On ajoute également les contraintes qui définissent le domaine des variables modélisant la mise en attente :

$$B_t^i \geq 0 \quad \forall i = 1, \ldots, N, \forall t = 1, \ldots, T \tag{2.22}$$

Finalement, on intègre le terme $\sum_{t=1}^{T} \sum_{i=1}^{N} b_t^i B_t^i$ à la fonction objectif pour pénaliser le report de certaines ventes. L'intégration des mises en attente de demandes aux modèles de planification de production est une extension plus commune dans la littérature que celle des ventes perdues. À notre connaissance, ce problème a été formulé pour la première fois par Zangwill [118], qui a proposé un algorithme de résolution par programmation dynamique. Pour le problème sans contrainte de capacité, Pochet et Wolsey [91] ont proposé des modélisations plus fortes basées sur les formulations *shortest path* et *facility location* et ont décrit des familles d'inégalités valides. Millar et Yang [86] ont développé une heuristique Lagrangienne pour le CLSP avec backlog. Plusieurs auteurs se sont intéressés à des problèmes plus complexes qui intègrent les mises en attente : Toledo *et al.* [104] ont utilisé une méthode de R&F couplée au Fix-and-Optimize (F&O) pour résoudre un problème de lot-sizing avec regroupement par familles et mise en attente. L'heuristique de recherche locale F&O consiste à fixer une partie du problème par une solution obtenue afin d'optimiser uniquement une sous-partie du problème. Dans la même idée, Xiao *et al.* [117] ont traité un problème incluant des mises en attente et ont développé une méthode

hybride utilisant le F&O et une liste tabou. Dans un contexte de production de produits périssables, Abad [1] a étudié l'impact de la prise en compte des reports de demandes afin de déterminer la politique optimale à adopter. Pour plus de précision sur cette extension, Quadt et Kuhn [93] présentent une revue de la littérature des problèmes de lot-sizing en incluant les reports de demandes.

2.4 Machines parallèles

Considérer des machines parallèles (on parle aussi de ressources multiples) donne la possibilité de planifier la production de plusieurs références simultanément sur différentes machines au sein d'une même période. Plus précisément, on considère un ensemble de M machines (ou ressources, ou lignes de production) disponibles pour la production d'une ou plusieurs références. Chaque machine $m = 1, \ldots, M$ est alors caractérisée par des attributs qui lui sont propres : sa capacité disponible C_{mt} en période t, son temps de production τ_m^i et son temps de réglage λ_m^i pour la référence i. Le cas où une ressource m n'est pas en mesure de produire une référence i donnée peut alors facilement être pris en compte en définissant un temps de réglage prohibitif, par exemple $\lambda_m^i > C_{mt}$ pour tout t. Le CLSP avec machines parallèles, que nous notons CLSP-PM, peut se modéliser comme suit :

$$\min \sum_{t=1}^{T} \sum_{i=1}^{N} \left(\sum_{m=1}^{M} p_t^i x_{mt}^i + sc_t^i y_{mt}^i \right) + h_t^i I_t^i \tag{2.23}$$

$$\text{s.c.} \sum_{i=1}^{N} \lambda_m^i y_{mt}^i + \tau_m^i x_{mt}^i \leq C_{mt} \quad \forall t = 1, \ldots, T, \forall m = 1, \ldots, M \tag{2.24}$$

$$\sum_{m=1}^{M} x_{mt}^i + I_{t-1}^i - d_t^i = I_t^i \quad \forall i = 1, \ldots, N, \forall t = 1, \ldots, T \tag{2.25}$$

$$x_{mt}^i \leq M y_{mt}^i \quad \forall i = 1, \ldots, N, \forall t = 1, \ldots, T \tag{2.26}$$

$$I_t^i \geq 0 \quad \forall i = 1, \ldots, N, \forall t = 1, \ldots, T \tag{2.27}$$

$$y_{mt}^i \in \{0, 1\}, x_{mt}^i \geq 0 \quad \forall i = 1, \ldots, N, \forall t = 1, \ldots, T, \forall m = 1, \ldots, M \tag{2.28}$$

Ajouter les machines parallèles augmente le nombre de symétries (dans le cas de machines identiques) ou le nombre de solutions ce qui rend le problème plus difficile en règle générale. Dans la littérature, on distingue deux grandes classes de problèmes à ressources multiples, selon que l'on considère les machines comme identiques ([19]) ou non-identiques ([63]). Diaby *et al.* [38] étudient le problème CLSP en multi-ressource, dans lequel un unique temps de setup est nécessaire pour produire une référence sur une ou plusieurs

ressources au sein d'une période. Sahling *et al.* [97] présentent un problème de lot-sizing multi-ressource et multi-niveau et développent une heuristique F&O. Ils étudient différentes approches pour définir les sous-ensembles de variables à optimiser. Par opposition à la décomposition par périodes qu'on retrouve habituellement pour cette classe d'heuristiques, les auteurs proposent des approches alternatives basées sur des décompositions par ressource ou par article. Ils concluent qu'il peut être avantageux de combiner les décompositions pour obtenir une meilleure solution. Dans Özdamar et Barbarosoğlu [119] et Özdamar et Birbil [120], les auteurs développent des approches hybrides mêlants plusieurs métaheuristiques. Toledo et Armentano [105] ont présenté une heuristique Lagrangienne pour un problème de lot-sizing avec des machines parallèles. Les auteurs proposent une procédure permettant de reconstruire une solution réalisable en déplaçant certaines productions sur des machines ou périodes disponibles à partir d'une solution non réalisable obtenue par l'heuristique Lagrangienne. Fiorotto et de Araujo [42] ont proposé une heuristique Lagrangienne basée sur une formulation *shortest path* du problème de lot-sizing. Les auteurs arrivent à transformer le problème initial en plusieurs sous-problèmes indépendants pour chaque période et machine. Ils obtiennent de bonnes solutions sur des instances allant jusqu'à 6 machines parallèles. Fiorotto *et al.* [43] ont repris cette idée et ont proposé une approche hybride faisant intervenir une relaxation Lagrangienne et une décomposition de Dantzig-Wolfe pour le CLSP-PM.

Les machines parallèles ont aussi été introduites dans les problèmes de lot-sizing en small bucket ou des variantes de la formulation (2.23)-(2.28). Beraldi *et al.* [19] présentent des heuristiques constructives pour un problème en small bucket. Meyr [85] a développé des métaheuristiques de recherche locale pour un problème qui mixe le small bucket et big bucket.

2.5 Lot-sizing avec capacité et réglages dépendant de la séquence

Dans les formulations présentées précédemment, nous supposons que les temps et les coûts de réglage sont indépendants de la séquence de production. Cette hypothèse n'est pas toujours réaliste dans de nombreux contextes industriels. Le problème de lot-sizing avec des réglages dépendants de la séquence (*sequence dependent setups*), noté habituellement CLSD, est de plus en plus étudié dans la littérature pour répondre aux problématiques industrielles. Ce problème est parfois nommé le *Capacitated Lot-sizing and Scheduling Problem* ([66],[82],[116]). Plusieurs modélisations mathématiques existent

pour ce problème. Nous avons choisi de présenter une des modélisations décrites dans Guimarães *et al.* [55] pour la version à une machine. La version de ce problème qui considère plusieurs machines parallèles est noté CLSD-PM. À partir du problème CLSP, nous introduisons deux nouvelles variables pour modéliser les réglages dépendant de la séquence :

— w_t^{ij} une variable binaire qui remplace y_t^i égale à 1 si la référence i précède directement la référence j en période t

— z_t^i une variable binaire égale à 1 si la machine est prête pour produire la référence i en début de période t, 0 sinon.

Afin de prendre en considération les variations des *setup times* et des *setup costs* en fonction de l'ordre dans lequel on produit les références, on modifie également les paramètres correspondants comme décrit ci-dessous :

— λ_t^{ij} correspond au temps de réglage pour passer de la référence i à la référence j en période t.

— sc_t^{ij} correspond au coût de réglage pour passer de la référence i à la référence j en période t.

On obtient ainsi la formulation suivante pour le CLSD :

$$\min \sum_{t=1}^{T}\sum_{i=1}^{N} p_t^i x_t^i + h_t^i I_t^i + \sum_{j=1}^{N} sc_t^{ij} w_t^{ij} \tag{2.29}$$

$$\text{s.c.} \sum_{i=1}^{N} \tau_t^i x_t^i + \sum_{j=1}^{N} \lambda_t^{ij} w_t^{ij} \leq C_t \quad \forall t = 1, \ldots, T \tag{2.30}$$

$$x_t^i + I_{t-1}^i - d_t^i = I_t^i \quad \forall i = 1, \ldots, N, \forall t = 1, \ldots, T \tag{2.31}$$

$$x_t^i \leq M(\sum_{j=1}^{N} w_t^{ji} + z_t^i) \quad \forall i = 1, \ldots, N, \forall t = 1, \ldots, T \tag{2.32}$$

$$\sum_{i=1}^{N} z_t^i = 1 \quad \forall i = 1, \ldots, N, \forall t = 1, \ldots, T \tag{2.33}$$

$$z_t^i + \sum_{j=1}^{N} w_t^{ji} = z_{t+1}^i + \sum_{j=1}^{N} w_t^{ij} \quad \forall i = 1, \ldots, N, \forall t = 1, \ldots, T \tag{2.34}$$

$$x_t^i, I_t^i, z_t^i \geq 0 \quad \forall i = 1, \ldots, N, \forall t = 1, \ldots, T \tag{2.35}$$

$$w_t^{ij} \in \{0, 1\} \quad \forall i, j = 1, \ldots, N, \forall t = 1, \ldots, T \tag{2.36}$$

$$\{(i, j) : w_t^{ij} > 0\} \text{ n'inclut pas de sous-tours discontinus} \quad \forall t = 1, \ldots, T \tag{2.37}$$

La fonction à minimiser (2.29) prend en compte les coûts de réglage dépendant de la séquence. La contrainte (2.30) indique que les temps de production et de setup ne doivent pas dépasser la capacité en période t. La contrainte (2.31) exprime l'équilibre du stock à travers l'horizon. La contrainte (2.32) impose qu'un article i ne peut être produit en période t que si la machine est réglée pour la production de cet article au début ($z_t^i = 1$) ou au cours ($\sum_{j=1}^{N} w_t^{ji} = 1$) de la période. La contrainte (2.33) assure que la machine est réglée pour une unique référence au début de chaque période. La contrainte (2.34) correspond à la contrainte de conservation des réglages entre les périodes. Finalement, la contrainte (2.37) traduit l'idée qu'il ne doit pas y avoir de sous-tours discontinus dans la séquence. Cette contrainte est essentielle pour assurer la faisabilité des séquences de production mais rend le problème plus complexe à résoudre. Plusieurs modélisations de la contrainte (2.37) sont utilisées dans la littérature comme la contrainte Miller-Tucker-Zemlim (MTZ) ([88]) développée notamment pour la formulation du Traveling Salesman Problem (TSP). Cette contrainte est utilisée dans [60, 11]. Guimarães *et al.* [55] présentent plusieurs modélisations de cette contrainte et mettent en évidence que la modélisation *single commodity flow* (SCF) est la plus performante. Plusieurs modélisations en small bucket intègrent des coûts de réglages dépendant de la séquence ([52, 51, 45]). Fleischmann et Meyr [46] introduisent une nouvelle formulation nommée *General Lot-sizing Problem* qui mixe le small bucket et big bucket. L'idée générale est de découper les périodes en micro-périodes dans lesquelles une unique référence peut être produite. Plusieurs auteurs ont repris cette formulation par la suite ([106],[84]). Pour plus d'informations sur les différentes modélisations mathématiques du *lot-sizing and scheduling problem* en small bucket ou big bucket, Copil *et al.* [33] font une présentation et une classification complète des formulations développées dans la littérature. Dans la version classique du CLSD, nous considérons que les réglages des machines sont reportés entre deux périodes consécutives. C'est ce qu'on appelle les *setups carry-over* ([102]). Dans Clark *et al.* [30], les auteurs présentent un problème industriel dans lequel les machines sont réinitialisées entre les périodes. Pour modéliser cette réinitialisation, les auteurs introduisent une référence fictive 0 pour représenter l'état initial des machines et imposent que toute machine m soit réglée pour la produire au début de chaque période t ($z_t^0 = 1$). Dans certaines applications, on autorise également le *setup crossover* ([44]), c'est-à-dire que le réglage d'une machine peut s'effectuer à cheval sur deux périodes consécutives. Kang [64] a couplé les setup carryover et setup crossover pour le CLSD.

Les méthodes les plus utilisées pour le CLSD s'appuient sur des heuristiques. En effet, ce problème étant plus complexe que les précédents, une résolution exacte à partir

d'un solveur disponible dans le commerce peut est très coûteux en temps de calcul, ce qui pousse à l'utilisation de méthodes approchées. Lang et Shen [70] ont utilisé une heuristique F&O pour traiter ce problème. James et Almada-Lobo [60] ont développé une nouvelle heuristique basée sur une modélisation mathématique (MIP-heuristic) itérative pour le problème CLSD avec des machines parallèles. Cette heuristique est comparée aux heuristiques constructives classiques (comme le R&F) et présente de meilleurs résultats sur les instances testées. Xiao *et al.* [117] s'intéressent au problème CLSD avec des machines parallèles et des reports sur les ventes. Ils utilisent une recherche locale pour déterminer les sous-ensembles à optimiser dans le F&O. Haase [56] présente une modélisation inspirée du GLSP et développe une heuristique qui construit les séquences de production en partant de la fin de l'horizon. La méthode est testée sur un nouvel ensemble d'instances et permet d'obtenir des solutions meilleures que l'heuristique décrite dans [45] à laquelle la méthode est comparée. Kovács *et al.* [66] font l'hypothèse que les coûts de setup sont proportionnels aux temps de setup et introduisent une formulation basée sur les séquences. Un algorithme de programmation dynamique est introduit pour déterminer les séquences efficaces. Les résultats sont comparés sur des instances avec peu de références mais un nombre considérable de périodes (≥ 60) sur lesquelles leur approche donne de bons résultats. Menezes *et al.* [82] présentent une formulation dans laquelle les contraintes d'élimination des sous-tours sont représentées comme un ensemble exponentiel de contraintes. Les auteurs les ajoutent alors dynamiquement dans le Branch-and-Cut dès qu'un sous-tour discontinu est détecté. Les résultats montrent que cette méthode est efficace pour un nombre réduit de références. Xiao *et al.* [116] présentent une heuristique Lagrangienne hybride pour le problème CLSD avec des machines parallèles. L'heuristique se compare à CPLEX et obtient de meilleures solutions sur des instances de grandes tailles. Dans [15], les auteurs étudient un problème industriel de fabrication d'isolants pour les conduits dans lequel le nombre de références est supérieur à 200. Pour résoudre ce problème de grande taille qui prend en compte des réglages dépendant de la séquence et des machines parallèles, ils développent une procédure en deux phases qui résout dans un premier temps le problème avec des réglages indépendants de la séquence, puis dans un deuxième temps construit les séquences.

2.6 Instances pour le lot-sizing

La littérature compte quelques jeux d'instances classiques, générés en suivant diverses méthodes de génération au cours des dernières décennies. Cette banque de données sert en

particulier de banc d'essai aux chercheurs pour comparer les performances des différentes approches de résolution existantes. Parmi les jeux d'instances accessibles, celui proposé par Trigeiro *et al.* [108] fait partie des plus connus et des plus référencés pour le CLSP. Ce dernier génère des instances de tailles variées, en combinant différents nombres de références $N \in \{6, 12, 24\}$ et nombres de périodes $T \in \{15, 30\}$. Les instances ainsi obtenues ne contiennent qu'une seule machine, dont la capacité est définie de manière à satisfaire la demande. Plusieurs articles tels que Tempelmeier et Derstroff [103] et Jans et Degraeve [61] ont par la suite présenté des résultats basés sur une méthode de génération similaire, tandis que certains auteurs comme Absi et Kedad-Sidhoum [5] ont partiellement modifié celle-ci pour prendre en compte d'autres extensions telles que les ventes perdues. L'une des particularités de instances de Trigeiro *et al.* [108] est qu'elles balaient un large spectre de valeurs pour les différents paramètres. En particulier, certaines de ces combinaisons produisent des problèmes plus difficiles à résoudre, notamment lorsque les coûts de setup dominent les coûts linéaires de stockage. En effet, en augmentant le ratio des coûts de setup par rapport aux coûts de stockage, le *Time Between Orders (TBO)* (i.e le temps entre plusieurs productions d'une même référence) augmente. Maes et Van Wassenhove [78] et Trigeiro *et al.* [108] ont détecté que les problèmes avec un TBO élevé sont plus difficiles à résoudre. Malgré l'intérêt que représentent ces jeux d'instances, notamment pour tester la pertinence de nos solutions face à d'autres méthodes de la littérature, l'approche utilisée pour les générer est difficilement transposable au problème que nous traitons. En particulier, elle ne permet pas de prendre en compte les sequence-dependent setup times ou des machines parallèles tout en conservant une construction cohérente. De plus, les cas d'application que nous envisageons de traiter contiennent un nombre de références nettement supérieur à ceux considérés dans Trigeiro *et al.* [108], il est donc intéressant pour nous d'obtenir des problèmes de tailles plus importante pour tester la robustesse de nos méthodes de résolution.

Les constats précédents ont déjà conduit plusieurs auteurs à proposer de nouvelles méthodes de génération d'instances, dont certains pour le Capacitated Lot-Sizing with Sequence Dependent setups (CLSD). Par exemple, celle présentée par Almada-lobo *et al.* [11] pour le problème CLSD permet d'obtenir des setup times qui respectent l'inégalité triangulaire et des setup costs proportionnels aux setup times, grâce à l'introduction d'un facteur Φ. Ces instances sont généralisées aux cas des machines parallèles dans James et Almada-Lobo [60]. Dans ce dernier, les auteurs montrent que l'augmentation de Φ impacte négativement la qualité de la solution. Cette observation confirme qu'augmenter le TBO rend la résolution plus difficile. Plus récemment Xiao *et al.* [117] ont développé une

nouvelle méthode de génération d'instances pour le Capacitated Lot-sizing with Sequence Dependent setups on parallel machines (CLSD-PM). Dans ces instances, la capacité ne varie pas en fonction des périodes ni des machines ce qui simplifie la résolution. Nous présentons dans le chapitre suivant une nouvelle génération d'instances élaborées en collaboration avec notre partenaire industriel et les informations provenant des producteurs utilisant les solutions proposées par VIF. Cette génération doit permettre de mesurer l'impact de la taille des données (i.e le nombre de références, de périodes et de lignes) mais aussi l'impact de la capacité et des coûts de setup.

Chapitre 3

Modélisation mathématique du problème

Dans ce chapitre, nous nous focalisons sur la modélisation sous forme de programme linéaire en nombres entiers du problème de planification décrit au chapitre 1. Nous désignons ce problème par l'acronyme CLSSD-PM signifiant *Capacitated Lot-sizing problem with lost sales, Safety stock and Sequence Dependent setup times on Parallel Machines*. Dans un premier temps, nous présentons une description formelle du problème central de notre étude en introduisant les notations et les hypothèses nécessaires à la modélisation. Nous proposons ensuite deux modélisations mathématiques de ce problème. Afin d'améliorer la résolution de nos modèles dans un Branch-and-Cut, nous adaptons plusieurs inégalités valides de la littérature à notre problème. Finalement, afin d'éprouver nos modélisations, nous décrivons la génération de notre ensemble d'instances sur lequel nous présentons nos résultats expérimentaux.

3.1 Définition du problème

Le problème que nous étudions est un problème de lot-sizing dans lequel nous devons déterminer un plan de production pour un ensemble de N références que nous notons $\mathcal{N}$. L'horizon de planification, que nous notons $\mathcal{T}$, se compose de T périodes. De plus, on considère que l'atelier comporte M lignes de fabrication ou machines que nous notons $\mathcal{M}$. Pour ne pas porter à confusion, nous utilisons dans la suite les indices $i, j \in \mathcal{N}$ pour désigner les articles, les indices $t, s \in \mathcal{T}$ pour désigner les périodes et l'indice $m \in \mathcal{M}$ pour les lignes de production. Le tableau 3.1 présente un récapitulatif des ensembles et indices.

Au delà de la décision relative à la quantité de production à chaque période et l'affectation aux lignes de production, nous considérons les contraintes suivantes :

(C1) **Contrainte de capacité** : la somme des temps de production et des temps de réglage ne doit pas dépasser la capacité dite dure. La capacité dure correspond à la capacité allouée à laquelle est ajoutée la tolérance de dépassement. Nous utilisons

le terme anglophone *overtime* pour désigner le temps d'utilisation au-delà de la capacité allouée.

(C2) **Contrainte de production minimale** : si un lancement de production est effectué, alors cette production doit au moins être supérieure à un certain seuil appelé quantité minimale. Cette extension a été traitée dans la littérature par plusieurs auteurs ([83, 32]).

(C3) **Contrainte de stock à respecter** : le stock en fin de chaque période doit être au plus proche du stock cible. Cette contrainte "souple" est modélisée sous forme d'un coût de pénalité dans notre formulation.

(C4) **Contrainte de date limite d'utilisation** : la quantité à produire par période ne doit pas être supérieure à ce qui pourra être utilisé dans le temps compris entre la date de production et la date limite d'utilisation. En réalité cette contrainte s'apparente à de la périssabilité dans un contexte déterministe.

Pour modéliser les différentes contraintes énoncées, nous introduisons plusieurs notations. Pour tout $t \in \mathcal{T}$ et tout $i \in \mathcal{N}$, nous notons $d_t^i \geq 0$ une demande déterministe qui correspond au nombre d'unités de la référence i à fournir en période t. Nous introduisons de plus $D_{s,t}^i = \sum_{u=s}^{t} d_u^i$ qui correspond à la demande cumulée de la référence i entre les périodes s et t. Lorsqu'une référence i est produite sur une machine m dans une période t donnée, la quantité totale produite sur cette machine doit être supérieure à une valeur minimum $q_{\min}^i$. Comme indiqué précédemment, le stock cible ne correspond pas à une borne inférieure comme dans Loparic *et al.* [76] mais à un objectif à atteindre. À notre connaissance seuls Absi et Kedad-Sidhoum [6] ont présenté une modélisation similaire dans la littérature. Nous notons S_t^i le stock cible de la référence i à la fin de la période t. Nous nous concentrons sur des politiques de gestion de stock dites FIFO, c'est-à-dire que l'ordre dans lequel les unités en stock sont consommées pour servir la demande est le même que celui dans lequel elles ont été produites. En outre nous considérons qu'il est interdit de refuser de servir une demande : en d'autre terme en cas de ventes perdues pour une référence i en période t on impose que le nombre d'unités de ix en stock à la fin de cette même période soit nul. Pour modéliser la contrainte (C4), nous introduisons aussi un paramètre δ_t^i qui correspond à la dernière période à laquelle la production de la référence i en période t peut être utilisée pour satisfaire une demande. Ainsi $D_{t+1,\delta_t^i}^i$ correspond au stock maximal possible en fin de période t pour la référence i.

Nous notons respectivement C_{mt} et $\overline{C}_{mt}$ la capacité allouée et la capacité dure de la ligne m à la période t. Produire une unité de la référence i sur la ligne m (hors temps de réglage) nécessite un temps τ_m^i. De plus, le temps nécessaire aux réglages sur une ligne

$m \in \mathcal{M}$ pour passer de la production d'une référence $i \in \mathcal{N}$ à une autre $j \neq i$ est représenté comme un temps fixe λ_m^{ij} indépendant des quantités d'articles i et j produites. Ce temps est considéré non symétrique a priori. Dans le problème étudié, nous supposons que les lignes de production sont réinitialisées à la fin de chaque période. Ainsi, les réglages des machines en fin de période ne sont pas gardées pour les périodes suivantes. De plus, nous supposons que le premier réglage d'une machine dans une période est traité en dehors du temps de production et n'est donc pas comptabilisé dans la capacité de production. Cette hypothèse est peu fréquente dans les problèmes de lot-sizing avec des réglages dépendent de la séquence mais a été posée par Clark *et al.* [30]. Dans notre cas, elle est issue des pratiques clients où une équipe est généralement affectée pour préparer les lignes de production en dehors du temps de production comptabilisé dans la capacité. Nous modélisons cette spécificité grâce à l'ajout d'une référence fictive indexée par 0. Pour simplifier les notations, nous introduisons l'ensemble $\mathcal{N}_0 = \mathcal{N} \cup \{0\}$ et nous supposons que le temps de réglage pour passer de la référence 0 à n'importe quelle autre référence est nul, c'est-à-dire $\lambda_m^{0i} = 0$ pour tout $i \in \mathcal{N}$ et $m \in \mathcal{M}$.

L'un des principaux défis du problème CLSSD-PM étudié réside dans l'influence de l'ordre de la production sur les temps de réglage. L'extension des réglages dépendant de la séquence dans notre problème concilie les problèmes de planification de la production et d'ordonnancement, dans le sens où ils intègrent les deux niveaux de décision. Dans le cas du CLSD, les principaux travaux qui traitent des réglages dépendantes de la séquence sont liés au problème ATSP. En effet, une fois la production déterminée sur une période donnée, la séquence de production optimale correspond à la tournée optimale pour un ATSP sur le sous-ensemble d'articles sélectionné. La figure 3.1 présente deux séquences différentes pour un problème avec 4 éléments sur une seule machine. Les deux solutions sont équivalentes en termes de quantité produite sur la période considérée, mais leurs coûts respectifs diffèrent du fait d'un temps de production plus long dans la séquence supérieure. Comme dans l'ATSP, la partie difficile de ce problème vient en grande partie des contraintes d'élimination des sous-tours. On remarque que dans le cas du problème CLSSD-PM, cet aspect devient encore plus important avec la multiplication des lignes de production, chacune nécessitant de résoudre un problème ATSP pour tout sous-ensemble de références qui lui est affecté.

L'objectif du problème étudié est de définir un plan de production qui minimise l'ensemble des coûts engendrés par celui-ci. Cette fonction de coût se décompose en différents éléments :

— c_{mt} : coût d'utilisation de la ligne m en période t. Une ligne qui est utilisée pendant

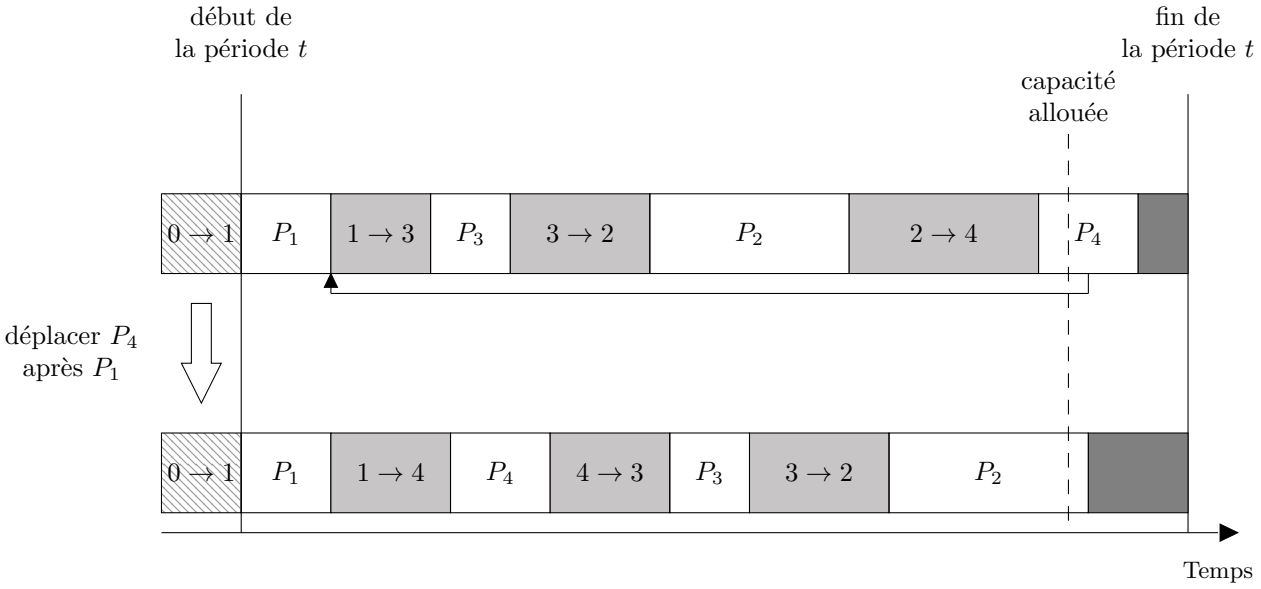


FIGURE 3.1 – Illustration d'un exemple de deux séquences de production

un temps donné coûte un montant par heure d'utilisation qui vient du coût de la main d'oeuvre et de l'énergie par exemple.

— $\overline{c}_{mt}$: coût de dépassement de la capacité allouée de la ligne m en période t. Lorsque la capacité allouée est dépassée, ce coût se cumule avec le coût d'utilisation c_{mt} et correspond aux frais additionnels qui s'appliquent (heures supplémentaires par exemple).

— p^i_{mt} : coût de production de la référence i sur la ligne m en période t. Ce coût s'applique à chaque unité produite.

— h^+_{it} : coût de sur-stock de la référence i en période t, appliqué à chaque unité de stock au dessus du stock cible en fin de période. Il prend en compte les besoins en fonds de roulement et les coûts de stockage physique supplémentaires par rapport à ceux anticipés par l'entreprise lors de la définition du stock cible.

— h^-_{it} : coût de déficit sur le stock cible de la référence i en période t encouru pour chaque unité de stock en dessous du stock cible. Il permet notamment de traduire un risque de rupture.

— l^i_t : coût de rupture sur la demande de la référence i en période t, appliqué à chaque unité de demande non satisfaite.

Tableau 3.1 – Ensembles et indices

N	Nombre de références
$\mathcal{N} = \{1, \ldots, N\}$	
$\mathcal{N}_0 = \{0, \ldots, N\}$	
i, j	indices des articles
M	Nombre de lignes
$\mathcal{M} = \{1, \ldots, M\}$	
m	indice des lignes
T	Nombre de périodes
$\mathcal{T} = \{1, \ldots, T\}$	
t, s	indices des périodes

Un récapitulatif des paramètres est présenté dans le tableau 3.2.

Tableau 3.2 – Paramètres

d_t^i	demande de la référence i sur la période t
$D_{s,t}^i$	demande cumulée de la référence i entre les périodes s et t
C_{mt}	capacité allouée en unité de temps de la ligne m pendant la période t
$\overline{C}_{mt}$	capacité dure en unité de temps de la ligne m pendant la période t
τ_m^i	temps de production unitaire de la référence i sur la ligne m
λ_m^{ij}	temps de changement entre les références i et j sur la ligne m
$q_{\min}^i$	quantité minimum de production de la référence i
S_t^i	stock cible de la référence i à la fin de la période t
δ_t^i	stock maximal autorisé de la référence i à la fin de la période t
c_{mt}	coût d'utilisation par unité de temps de la ligne m à la période t
$\overline{c}_{mt}$	sur-coût d'utilisation par unité de temps hors de la capacité allouée de la ligne m à la période t
p_{mt}^i	coût de production d'une unité de la référence i sur la ligne m à la période t
h_t^{i+}	coût unitaire de stockage de la référence i non inclus dans le stock cible à la période t
h_t^{i-}	pénalité appliquée pour chaque unité de la référence i en déficit sur stock de cible S_t^i à la période t.
l_t^i	coût unitaire de rupture sur la demande pour la référence i à la période t

3.2 Modélisation mathématique

Nous proposons deux formulations pour le problème étudié : une formulation agrégée et une formulation désagrégée basée sur le Facility Location ([67]). Pour améliorer la lisibilité de cette partie, nous introduisons en premier les variables et contraintes communes à ces

deux formulations. Dans nos deux formulations, nous utilisons des variables binaires $w_{mt}^{ij} \in \{0,1\}$ égales à 1 si la référence i est le prédécesseur direct de la référence j sur la ligne m en période t. Nous introduisons également des variables U_{mt} et $O_{mt} \in \mathbb{R}_+$ qui représentent respectivement le temps total et le temps au delà de la capacité allouée utilisé sur la ligne de production m en période t. Les domaines de ces variables communes aux différentes formulations présentées ci-dessous sont donc définis comme suit :

$$w_{mt}^{ij} \in \{0,1\} \ \forall i \in \mathcal{N}_0, \forall j \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{3.1}$$

$$U_{mt}, O_{mt} \geq 0 \qquad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{3.2}$$

Les contraintes qui se modélisent de manière équivalente dans les deux formulations sont relatives aux limites de capacité (contrainte (C1)) et à la gestion des séquences de production. Pour les contraintes de capacité, nous retrouvons :

– les contraintes qui indiquent que le temps total de mobilisation de la ligne de production m en période t est inférieur ou égal à la somme de capacité allouée et du temps supplémentaire de production :

$$U_{mt} \leq C_{mt} + O_{mt} \quad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{3.3}$$

– les contraintes qui bornent le temps supplémentaire par la différence entre la capacité dite dure et la capacité allouée :

$$O_{mt} \leq \overline{C}_{mt} - C_{mt} \quad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{3.4}$$

Les contraintes permettant de modéliser les séquences de production sont inspirées de Clark *et al.* [30]. Nous retrouvons les contraintes :

– qui imposent qu'une machine m ne peut être réglée pour une référence j que si il y a eu au moins un réglage à partir de la référence fictive :

$$\sum_{i \in \mathcal{N}} w_{mt}^{0i} \geq \sum_{i \in \mathcal{N}_0} w_{mt}^{ij} \quad \forall j \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{3.5}$$

– qui autorisent un réglage à partir d'une référence uniquement si celle-ci fait partie de la séquence de production :

$$\sum_{j \in \mathcal{N}_0} w_{mt}^{ji} \geq \sum_{j \in \mathcal{N}} w_{mt}^{ij} \quad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{3.6}$$

– qui empêchent d'avoir des sous-tours discontinus. On peut exprimer cette contrainte de manière générique comme suit :

$$\{(i,j) : w_{mt}^{ij} > 0\} \text{ n'inclut pas de sous-tours discontinus} \quad \forall t \in \mathcal{T} \tag{3.7}$$

, mais il existe différents ensembles de contraintes qui permettent d'éliminer les sous-tours discontinus de façon plus efficace. Nous discutons certaines de ces approches dans la section 3.2.1 ci-dessous.

3.2.1 Contrainte d'élimination des sous-tours

Si on restreint l'ensemble des contraintes qui s'appliquent aux séquences de production aux inégalités (3.5) et (3.6), les séquences de production considérées dans les solutions réalisables peuvent être composées de plusieurs cycles disjoints. On dit alors que celles-ci sont déconnectées. Ces dernières ne sont pas valides en pratique, puisque les productions des différentes références planifiées sur une même machine dans une période ne sont pas forcément connectées entre elles par les réglages nécessaires. Pour interdire ces solutions, on s'appuie sur des familles de contraintes appliquées dans des problèmes dérivés du voyageur de commerce (*Traveling Salesman Problem*, TSP) qu'on appelle contraintes d'élimination des sous-tours. Il existe de nombreux moyens de les modéliser. Guimarães *et al.* [55] font un état de l'art des méthodes les plus connues et utilisées, nous allons en présenter plusieurs :

Modélisation classique : L'élimination des sous-tours peut être modélisée de la façon suivante :

$$\sum_{(i,j)\in S} w_{mt}^{ij} \leq |S| - 1 \quad \forall \text{ sous-tour } S, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{3.8}$$

La principale limitation de cette modélisation provient du nombre exponentiel de contraintes qu'elle comporte, ce qui rend son implémentation impossible en pratique. En utilisant cette représentation, Clark *et al.* [30] développent une heuristique itérative qui à partir d'une solution, détecte les sous-tours présents et les ajoute pour les itérations futures. L'heuristique s'arrête lorsque qu'une solution sans sous-tours est trouvée. Dans une modélisation différente mais basée sur la même idée, Menezes *et al.* [82] ont présenté une résolution par Branch-and-Cut dans laquelle on rajoute à chaque noeud la contrainte d'élimination des sous-tours correspondante aux sous-tours détectés dans la solution courante. Ainsi, seulement les contraintes nécessaires sont ajoutées. Néanmoins, cette modélisation reste

peu efficace lorsque le nombre de référence augmente.

Modélisation Miller-Tucker-Zemlim (MTZ) : Cette modélisation est la plus utilisée dans le problème CLSD. Elle vient de la formulation de Miller, Tucker et Zemlin du problème TSP développée pour la première fois dans [88]. La contrainte (3.7) peut être remplacée par :

$$u^i_{mt} - u^j_{mt} + n_m w^{ij}_{mt} \leq n_m - 1 \quad \forall i, j \in \mathcal{N}, i \neq j, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{3.9}$$

$$u^0_{mt} = 0 \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{3.10}$$

$$1 \leq u^i_{mt} \leq N \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{3.11}$$

La variable u^i_{mt} représente la position de la référence i dans la séquence sur la ligne m à la période t. Dans (3.9), n_m représente une borne supérieure sur le nombre de références pouvant être ordonnancés sur la ligne m. Cet ensemble de contraintes permet de forcer explicitement un ordre et ainsi empêcher les sous-tours. Il est possible de renforcer la contrainte (3.9) comme dans Desrochers et Laporte [37] en appliquant une méthode de lissage :

$$u^i_{mt} - u^j_{mt} + n_m w^{ij}_{mt} + (n_m - 2) w^{ji}_{mt} \leq n_m - 1 \quad \forall i, j \in \mathcal{N}, i \neq j, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{3.12}$$

Modélisation Single Commodity Flow (SCF) : Elle est basée sur une modélisation de flot avec la variable f^{ij}_{mt} qui représente le flot qui passe de la référence i à la référence j sur la ligne m à la période t. Pour plus de détails, le lecteur peut se référer à [55] qui présente pour la première fois cette modélisation.

$$\sum_{j \in \mathcal{N}} f^{0j}_{mt} = \sum_{i \in \mathcal{N}_0} \sum_{j \in \mathcal{N}} w^{ij}_{mt} \qquad \forall m \in \mathcal{M},\ \forall t \in \mathcal{T} \tag{3.13}$$

$$\sum_{i \in \mathcal{N}_0} f^{ij}_{mt} = \sum_{i \in \mathcal{N}_0} w^{ij}_{mt} + \sum_{i \in \mathcal{N}} f^{ji}_{mt} \ \forall j \in \mathcal{N},\ \forall m \in \mathcal{M},\ \forall t \in \mathcal{T} \tag{3.14}$$

$$0 \leq f^{ij}_{mt} \leq N w^{ij}_{mt} \qquad \forall i \in \mathcal{N}_0, \forall j \in \mathcal{N},\ \forall m \in \mathcal{M},\ \forall t \in \mathcal{T} \tag{3.15}$$

Guimarães *et al.* [55] montrent que cette modélisation de la contrainte d'élimination des sous-tours est plus efficace que la modélisation MTZ pour le CLSD.

Dans la partie 3.5, nous présentons les résultats des modélisation MTZ et SCF sur un ensemble réduit d'instances.

Dans les deux sections suivantes, nous proposons deux formulations mathématiques

possibles pour le CLSSD-PM.

3.2.2 Formulation agrégée (AGG)

La formulation agrégée est la formulation la plus intuitive pour le problème de lot-sizing. Elle se base sur les variables suivantes :

- x^i_{mt} : variable indiquant la quantité produite de la référence i sur la ligne m à la période t.
- L^i_t : nombre d'unités de la référence i en rupture à la période t.
- I^{i+}_t : valeur du surstock de la référence i à la fin de la période t.
- I^{i-}_t : valeur du déficit sur le stock cible de la référence i la fin de la période t.
- I^i_t : valeur du stock de la référence i à la fin de la période t. Cette variable permet d'améliorer la lisibilité de certaines contraintes.

La formulation agrégée, que nous notons MIP-AGG est définie de la façon suivante :

$$\min \sum_{t\in\mathcal{T}}\sum_{m\in\mathcal{M}}(c_{mt}U_{mt}+\overline{c}_{mt}O_{mt})+\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}}(h^{i+}_t I^{i+}_t + h^{i-}_t I^{i-}_t + l^i_t L^i_t + \sum_{m\in\mathcal{M}} p^i_{mt}x^i_{mt}) \quad (3.16)$$

$$\text{s.c} : U_{mt} = \sum_{i\in\mathcal{N}}(\tau^i_m x^i_{mt} + \sum_{j\in\mathcal{N}}\lambda^{ij}_m w^{ij}_{mt}) \qquad \forall m\in\mathcal{M}, \forall t\in\mathcal{T} \quad (3.17)$$

$$I^i_t = S^i_t + I^{i+}_t - I^{i-}_t \qquad \forall i\in\mathcal{N}, \forall t\in\mathcal{T} \quad (3.18)$$

$$I^i_t = I^i_{t-1} + \sum_{m\in\mathcal{M}} x^i_{mt} + L^i_t - d^i_t \qquad \forall i\in\mathcal{N}, \forall t\in\mathcal{T} \quad (3.19)$$

$$x^i_{mt} \leq \min\left\{\frac{\overline{C}_{mt}}{\tau^i_m}, D^i_{t,\delta^i_t}\right\}\sum_{j\in\mathcal{N}_0} w^{ji}_{mt} \qquad \forall i\in\mathcal{N}, \forall m\in\mathcal{M}, \forall t\in\mathcal{T} \quad (3.20)$$

$$x^i_{mt} \geq q^i_{\min}\sum_{j\in\mathcal{N}_0} w^{ji}_{mt} \qquad \forall i\in\mathcal{N}, \forall m\in\mathcal{M}, \forall t\in\mathcal{T} \quad (3.21)$$

Contraintes (3.3)-(3.2)

$$L^i_t \leq d^i_t \qquad \forall i\in\mathcal{N}, \forall t\in\mathcal{T} \quad (3.22)$$

$$I^{i-}_t \leq S^i_t \qquad \forall i\in\mathcal{N}, \forall t\in\mathcal{T} \quad (3.23)$$

$$I^i_t \leq D^i_{t+1,\delta^i_t} \qquad \forall i\in\mathcal{N}, \forall t\in\mathcal{T} \quad (3.24)$$

$$L^i_t, I^i_t, I^{i+}_t, I^{i-}_t \geq 0 \qquad \forall i\in\mathcal{N}, \forall t\in\mathcal{T} \quad (3.25)$$

$$x^i_{mt} \geq 0 \qquad \forall i\in\mathcal{N}, \forall m\in\mathcal{M}, \forall t\in\mathcal{T} \quad (3.26)$$

L'objectif est de minimiser les coûts de production, les coûts d'utilisation des lignes et de temps supplémentaire, de surstock, de déficit sur le stock cible et de rupture. Les contraintes (3.17) indiquent que le temps d'utilisation de chaque ligne m en chaque période t est égal à la somme des temps de production et des temps de réglage. Les contraintes (3.18) définissent le stock à partir du stock cible et des variables de surstock et de déficit. Les contraintes (3.19) correspondent à la conservation des flux physiques sur l'horizon en prenant en compte les ventes perdues. Les contraintes (3.20) indiquent qu'il est possible de lancer une production uniquement si un réglage a été effectué. Dans ce cas, la production est bornée par le minimum entre la quantité maximale qu'il est possible de produire dans la capacité disponible et la production utilisable pour satisfaire les demandes futures. Nous modélisons l'obligation d'avoir une production supérieure à la quantité minimale $q^i_{\min}$ par les contraintes (3.21). La gestion des séquences de production et la définition des temps supplémentaires sont définis par les contraintes (3.3) à (3.2). Enfin, les contraintes (3.22) à (3.26) définissent et bornent les variables continues, binaires et entières.

3.2.3 Formulation basée sur le facility location (FL)

La formulation que nous allons présenter s'inspire du problème de facility location dans la définition des variables. Cette formulation a l'avantage en principe d'avoir une meilleure borne inférieure donnée par la relaxation linéaire en comparaison avec la formulation précédente sans dégrader la borne supérieure. Par ailleurs, la lisibilité du modèle est améliorée par la réduction du nombre de variables différentes grâce au suivi des unités sous forme de flot. Les variables de production peuvent être définies plus finement, en considérant la période de production et celle de consommation. Plusieurs références développent cette formulation pour les problèmes de lot-sizing avec diverses contraintes (voir [72, 48, 49]). Pour introduire cette nouvelle formulation, nous définissons de nouvelles variables et paramètres présentés ci-dessous :

— x^{it}_{ms} : variable linéaire comprise entre 0 et 1 indiquant la proportion de la demande en t de la référence i qui est satisfaite par la production en s sur la ligne m.

— $\theta_i(t)$: première période à laquelle la référence i doit entrer dans le stock cible pour satisfaire la demande en période t.

Cette formulation se base sur une variable de production comprise entre 0 et 1. Pour obtenir la quantité réellement produite à chaque période, nous pouvons multiplier x^{it}_{ms} par d^i_t. À noter qu'il existe des versions de la formulations Facility Location utilisant une variable de production entre 0 et d^i_t. Nous définissons le coût de production $H^{it}_{ms} =$

$d_t^i \cdot (p_{ms}^i - l_t^i + \sum_{u=s}^{\theta_i(t)-1} h_u^{i+} - \sum_{u=\max\{\theta_i(t),s\}}^{t-1} h_u^{i-})$ qui représente le coût encouru pour servir une partie de la demande d_t^i avec des unités produites sur la ligne m à la période s. Ce coût se décompose de la manière suivante : pour chaque unité produite en s sur la ligne m pour satisfaire la demande en période t de la référence i, le coût p_{ms}^i est appliqué. Si cette production est effectuée avant $\theta_i(t)$, le coût de surstock est appliqué sur l'ensemble des périodes entre s et $\theta_i(t)$. En revanche si la production est effectuée après $\theta_i(t)$, nous économisons le coût de déficit entre la période de production et t. Finalement pour chaque unité produite, nous économisons aussi un coût de rupture. Pour se comparer avec la solution de la formulation MIP-AGG, nous ajoutons le terme constant $\sum_{i\in\mathcal{N}} \sum_{t\in\mathcal{T}} \left(l_t^i + \sum_{u=\theta_i(t)}^{t-1} h_u^{i-}\right) d_t^i$ à la fonction objectif qui représente le coût de ne satisfaire aucune demande. On peut définir la nouvelle formulation, notée MIP-FL, comme suit :

$$\min \sum_{s\in\mathcal{T}} \sum_{m\in\mathcal{M}} \left(c_{mt} U_{mt} + \overline{c}_{mt} O_{mt} + \sum_{i\in\mathcal{N}} \sum_{t=s}^{\delta_s^i} H_{ms}^{it} x_{ms}^{it} \right) \tag{3.27}$$

$$\text{s.c} : U_{ms} = \sum_{i\in\mathcal{N}} (\tau_m^i \sum_{t=s}^{\delta_s^i} d_t^i x_{ms}^{it} + \sum_{j\in\mathcal{N}} \lambda_m^{ij} w_{ms}^{ij}) \qquad \forall m \in \mathcal{M}, \forall s \in \mathcal{N} \tag{3.28}$$

$$x_{ms}^{it} \leq \sum_{j\in\mathcal{N}_0} w_{ms}^{ji} \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall s,t \in \mathcal{T} \tag{3.29}$$

$$\sum_{m\in\mathcal{M}} \sum_{s=1}^{t} x_{ms}^{it} \leq 1 \qquad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{3.30}$$

$$\sum_{t=s}^{\delta_s^i} d_t^i x_{ms}^{it} \geq q_{\min}^i \sum_{j\in\mathcal{N}_0} w_{ms}^{ji} \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall s \in \mathcal{T} \tag{3.31}$$

Constraints (3.3)-(3.2)

$$x_{ms}^{it} \in [0,1] \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall s,t \in \mathcal{T} \tag{3.32}$$

L'objectif (3.27) reste le même que celui de la modélisation précédente, à savoir de minimiser les coûts d'utilisation de ligne, de production et les coûts liés au stock. La définition du coût H_{ms}^{it} permet d'agréger une partie de ces coûts. Les contraintes (3.28) indiquent que le temps d'utilisation est égal à la somme des temps de production et des temps de réglage. Les contraintes (3.29) s'assurent qu'il est possible de produire une référence i uniquement si un réglage a été effectué. Les contraintes (3.30) imposent que la production pour satisfaire la demande en période t des périodes ultérieures ne peut dépasser cette demande. Comme la variable x_{ms}^{it} représente la proportion de la demande, la somme des productions doit être inférieure à 1. Nous modélisons l'obligation de dépasser le seuil $q_{\min}^i$

lorsqu'une production est lancée par (3.31). Comme pour la formulation précédente, nous reprenons les contraintes (3.3) à (3.2). Enfin, le domaine des variables est défini par (3.32).

3.3 Inégalités valides

Ajouter des inégalités valides pour résoudre un MIP permet en théorie d'accélérer la résolution grâce à l'obtention de meilleures bornes inférieures. En effet ces inégalités permettent de couper des solutions relâchées du problème d'origine dans l'arbre de recherche. Ces inégalités valides sont généralement introduites dans un Branch-and-Cut. De nos jours, il est possible d'implémenter des inégalités valides à travers les solveurs commerciaux, comme CPLEX qui propose des méthodes d'ajout de coupe ([34]). Ces solveurs intègrent eux-même de nombreuses coupes comme des flow covers, des MIR ou encore des coupes de Gomory et ils continuent d'ajouter des coupes classiques de la littérature pour perfectionner leurs outil de résolution. Dans ce qui suit, nous présentons quelques inégalités valides classiques que nous appliquons au problème CLSSD-PM.

3.3.1 Les inégalités (l, S)

Les inégalités valides (l, S) ont été présentées pour la première fois par Barany *et al.* [17] pour le CLSP. Elles ont ensuite été reprises dans plusieurs études (Absi et Kedad-Sidhoum [5], Almada-lobo *et al.* [11] et James et Almada-Lobo [60]) pour des problèmes de lot-sizing variés dont le CSLD. Ces inégalités peuvent être adaptées au problème à machines parallèles et aux ventes perdues. En utilisant les notations de la formulation agrégée, elles se présentent de la manière suivante :

$$D^i_{1l} - \sum_{t=1}^{l} L^i_t \leq \sum_{t \in \{1,\dots,l\} \setminus S} \left(D^i_{tl} \sum_{m \in \mathcal{M}} \sum_{j \in \mathcal{N}} w^{ji}_{mt} \right) + \sum_{t \in S} \sum_{m \in \mathcal{M}} x^i_{mt} \quad \forall i \in \mathcal{N}, \forall l \in \mathcal{T}, S \subseteq \{1, \dots, l\} \tag{3.33}$$

Nous remarquons qu'il y a un nombre exponentiel d'inégalités valides ce qui en pratique empêche de les ajouter à priori. Nous les ajoutons donc en utilisant une méthode de plan sécant (*cutting plane*) qui consiste à ajouter uniquement les inégalités valides violées par la solution courante à chaque noeud dans un Branch-and-Cut. En faisant cela, on s'assure de réduire l'espace de recherche sans éliminer de solutions entières. Barany *et al.* [17] ont introduit un algorithme de séparation 1 qui détermine les inégalités (l, S) à ajouter. Cette algorithme a une complexité en $O(NT^2)$.

Algorithme 1 Algorithme de séparation (l, S)

pour $i = 1, \ldots, N$ **faire**
  pour $l = 1, \ldots, T$ **faire**
    $S_l \leftarrow \emptyset$
    pour $t' = 1, \ldots, l$ **faire**
      si $\sum_m x^{*i}_{mt'} \leq D^i_{t'l} \sum_{m\in\mathcal{M}} \sum_{j\in\mathcal{N}} w^{*ji}_{mt'}$ **alors**
        $S_l \leftarrow S_l \cup \{t'\}$
      fin si
    fin pour
    si (3.33) est violé **alors**
      Ajouter les inégalités (3.33) au noeud courant
    fin si
  fin pour
fin pour

Absi et Kedad-Sidhoum [5] ont développé une variante de ces inégalités (l, S) pour le CLSP avec ventes perdues se basant sur une structure (k, l, S, I) présentée dans Pochet et Wolsey [92]. Ces inégalités (l, S) se présentent comme suit :

$$\sum_{m\in\mathcal{M}} x^i_{mt} + \sum_{t'=t}^{\delta_{it}} L^i_{t'} + I^i_{t-1} + \sum_{t'\in U} \left(D^i_{t'\delta_{it}} \sum_{m\in\mathcal{M}} y^i_{mt'} \right) + \sum_{t'\in V} \left(\sum_{m\in\mathcal{M}} x^i_{mt'} \right) \geq D^i_{t\delta_{it}} \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{3.34}$$

où U et V représentent des partitions de l'ensemble [t+1,...,T]

3.3.2 Inégalités (t, S, R)

Les inégalités (t, S, R) ont été présentées dans Loparic *et al.* [76] pour un problème de lot-sizing sans capacité et avec la prise en compte du prix de ventes et d'un stock de sécurité. Pour définir ces inégalités sur notre problème, on introduit une nouvelle variable binaire z^i_t qui est égale à 1 si la référence i est produit à la période t. Les inégalités (t, S, R) sont définies comme suit :

$$\sum_{s\in P\setminus S} \sum_{m\in\mathcal{M}} x^i_{ms} + \sum_{s\in S} \sum_{\substack{u=s \\ u\in R}}^{t} d^i_u z^i_s \geq \sum_{s\in R} (d^i_s - L^i_s) \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T}, P = \{1, \ldots, t\}, S \subseteq P, R \subseteq P \tag{3.35}$$

Ces inégalités sont valides pour le CLSSD-PM. (voir démonstration en Annexe)

3.3.3 Inégalités (k, U)

Les inégalités (k, U) sont un cas particulier des inégalités valides (t, S, R) définies précédemment. Elles peuvent être définies de la manière suivante :

$$\begin{aligned} I_t^i &\geq \sum_{l \in U} \left(-L_l^i + d_l^i (1 - \sum_{s=t+1}^{l} z_s^i) \right) \\ &\geq \sum_{l \in U} (d_l^i - L_l^i) - \sum_{l \in U} \sum_{s=t+1}^{l} d_l^i z_s^i \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{M}, \forall U \subseteq \{t+1, \ldots, T\} \end{aligned} \tag{3.36}$$

La démonstration de la validité de ces inégalités est présentée dans [76] et ne change pas fondamentalement pour notre problème. L'algorithme de séparation 2 permet de détecter et d'ajouter ces inégalités valides en temps polynomial dans le noeud courant d'un Branch-and-Cut :

Algorithme 2 Algortihme de séparation (k, U)

pour $i = 1, \ldots, N$ **faire**
  pour $t = 1, \ldots, T$ **faire**
    pour $l = t + 1, \ldots, T$ **faire**
      si $d_l^i - L_l^i - \sum_{t+1}^{l} d_l^i z_s^i \geq 0$ **alors**
        $U \leftarrow U \cup \{l\}$
      fin si
    fin pour
    si (3.36) est violé **alors**
      Ajouter les inégalités (3.36) au noeud courant
    fin si
  fin pour
fin pour

3.3.4 Inégalités valides ajoutées a priori

Il est possible d'ajouter des inégalités valides a priori dans nos modélisations. Almada-lobo *et al.* [11] ont introduit des inégalités valides pour le problème CLSD. En introduisant la variable W_{mt} égale à 0 si il y a au moins un réglage sur la ligne m à la période t, les inégalités suivantes sont valides :

$$w_{mt}^{0i} \leq \sum_j w_{mt}^{ij} + W_{mt} \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T}, \forall m \in \mathcal{M} \tag{3.37}$$

$$1 - W_{mt} \leq \sum_{i,j} w_{mt}^{ij} \quad \forall t \in \mathcal{T}, \forall m \in \mathcal{M} \tag{3.38}$$

$$W_{mt} \leq 1 - \frac{\sum_{i,j} w_{mt}^{ij}}{N} \quad \forall t \in \mathcal{T}, \forall m \in \mathcal{M} \tag{3.39}$$

Dans le tableau A.3 en Annexe, nous comparons les gaps obtenues avec et sans l'ajout de ces coupes pour la formulation MIP-AGG.

3.4 Générateur d'instances

Dans cette partie, nous présentons la procédure de génération d'instances que nous utilisons pour tester l'ensemble des méthodes développées. Les instances que nous avons générées ont été établies avec l'aide de notre partenaire industriel. VIF n'a pas à sa disposition de données issues de ses clients pour définir rigoureusement des instances réelles, l'objectif du jeu d'instances généré est donc de reproduire les caractéristiques d'un large spectre de cas d'application proches de la réalité observée chez les producteurs. Nous avons ainsi défini la taille des problèmes (nombre de périodes, références et machines) et les valeurs des paramètres de nos modèles à partir d'ordres de grandeurs issus de cas concrets et des connaissances métiers accumulées par l'entreprise. À travers ces instances, nous cherchons à éprouver la robustesse de nos méthodes pour une future utilisation dans un contexte industriel. Il est important de noter que cette génération a pour but de tester nos méthodes allant de cas clients observés à des cas plus théorique.

Dans le but de déterminer l'influence de la taille du problème sur la qualité de la solution et le temps d'exécution, nous faisons varier le nombre de références N de 20 à 125, le nombre de machines M de 1 à 6, en considérant des horizons de temps de 15 et 30 périodes pour chaque cas. Nous obtenons ainsi des instances que nous partitionnons en trois classes en fonction de la combinaison de ces trois grandeurs. Le tableau 3.4 récapitule les différentes valeurs de ces ensembles.

Les différents paramètres de nos instances sont définis comme suit :

Capacité. Pour chaque ligne de production m et chaque période t la capacité en temps C_{mt} correspond à une durée de travail journalière. En fonction des producteurs, cette valeur peut correspondre à 8, 16 ou 24 heures. La durée de travail change rarement d'une période à une autre et suit plutôt une tendance sur plusieurs périodes. Ainsi, à partir de la capacité d'une période, la probabilité de garder une capacité en temps équivalente pour la période suivante est de 0.8. Le nombre d'heures supplémentaires

correspond à 2% de la capacité totale, cependant aucun dépassement n'est autorisé lorsque la capacité allouée est de 24 heures.

Temps de production. Pour chaque référence i, nous décidons de l'ensemble de machines $\mathcal{M}_i \subseteq \mathcal{M}$ sur lesquelles elle peut être produite. Nous commençons par tirer une variable aléatoire de Bernouilli, dont la probabilité de succès p_i est elle-même tirée uniformément dans l'intervalle $[0.25, 5]$. En cas d'échec, la référence ne peut être produite que sur une seule ligne $m_i \in \mathcal{M}$ tirée aléatoirement ($\mathcal{M}_i = \{m_i\}$). Sinon, le nombre $|\mathcal{M}_i|$ de machines sur lesquelles la référence i peut être produite est tiré uniformément dans l'ensemble $\{2, \ldots, M\}$ et nous sélectionnons alors aléatoirement dans l'ensemble $\mathcal{M}$ les machines qui constituent $\mathcal{M}_i$. Enfin pour chaque ligne $m \in \mathcal{M}_i$, le temps de production unitaire τ_m^i est tiré uniformément dans l'intervalle $[0.1, 1]$ minutes.

Temps de changement. La génération des temps de changement s'inspire des pratiques industrielles. Dans l'industrie agroalimentaire, les produits sont souvent caractérisés par plusieurs caractéristiques (par exemple la présence d'allergènes, de label "bio", ou la température de cuisson) pour lesquelles il existe des matrices de temps de changement. Chaque produit peut avoir plusieurs caractéristiques, le calcul du temps de changement entre deux produits revient alors à prendre le maximum des temps de changements entre les différentes spécificités de ces produits. Pour générer notre jeu d'instances, nous considérons que chaque référence est caractérisée par quatre propriétés. Chaque caractéristique a soit deux (par exemple bio/non bio) soit trois (par exemple des températures de cuisson) états possibles. En pratique, les temps de changement observés sont souvent non symétriques et très variables. Il peuvent ainsi atteindre une heure pour passer d'un produit à un autre alors que la transition inverse dure seulement quelques minutes. Pour chaque caractéristique, nous générons une matrice de temps de changement dont les éléments sont calculés selon les principes suivants : les temps de transition courts entre deux états d'une même caractéristiques sont tirés uniformément dans des intervalles allant de 4 à 15 minutes, tandis que les transitions longues durent entre 35 et 80 minutes, tirées selon des lois uniformes dont les paramètres dépendent de la caractéristique et des états concernés. Enfin pour toutes les spécificités prises en compte, nous considérons systématiquement que rester dans le même état induit un temps de transition quasi nul tiré entre 0 et 4 minutes.

Les matrices présentées dans le tableau 3.3 illustrent la génération des temps de changement entre les états des différentes caractéristiques considérées pour les ré-

$$\begin{pmatrix} U(0,4) & U(4,12) \\ U(40,55) & U(0,4) \end{pmatrix}$$

(a) Caractéristique 1

$$\begin{pmatrix} U(0,4) & U(4,8) \\ U(60,80) & U(0,4) \end{pmatrix}$$

(b) Caractéristique 2

$$\begin{pmatrix} U(0,4) & U(8,15) \\ U(8,15) & U(0,4) \end{pmatrix}$$

(c) Caractéristique 3

$$\begin{pmatrix} U(0,4) & U(35,55) & U(55,70) \\ U(35,55) & U(0,4) & U(35,55) \\ U(55,70) & U(35,55) & U(0,4) \end{pmatrix}$$

(d) Caractéristique 4

Tableau 3.3 – Calcul des matrices de temps de changement

férences d'articles. Nous associons à chaque référence trois propriétés à deux états et une propriétés à trois états. Les temps de changement obtenus ont ainsi une structure cohérente proche des valeurs observées dans les environnements ciblés.

Demande. Nous générons les demandes des différentes références de façon itérative jusqu'à saturer une partie de la capacité disponible, amputée d'une estimation du temps dédié aux réglages machines entre produits. Cette dernière est issue du calcul d'une séquence virtuelle « moyenne », qui inclut toutes les références et considère pour toute paire $(i,j) \in \mathcal{N}^2$ un temps de changement $\bar{\lambda}_{ij} = \frac{1}{|\mathcal{M}_i \cap \mathcal{M}_j|} \sum_{m \in \mathcal{M}_i \cap \mathcal{M}_j} \lambda_m^{ij}$ moyen sur les lignes susceptibles de produire les références i et j. Notons EST l'estimation de la durée de la séquence optimale obtenue à partir de ces données artificielles. Il est crucial d'intégrer une estimation des temps de changement dans cette formule afin de mieux appréhender leur répartition sur plusieurs machines. Les premières instances que nous avons générées pour nos expérimentations ne tenait pas compte de cet aspect, aussi l'augmentation du nombre de machines amortissait les temps de changement et produisait des instances moins contraintes. L'ajout de ce terme nous a permis de compenser cet amortissement et de produire des instances cohérentes, dont la difficulté augmente avec les dimensions du problème.

La génération des demandes se déroule alors en trois phases :

1. Nous définissons la capacité disponible pour la production sur l'horizon de planification sur l'ensemble des machines comme $\mathcal{U} \sum_{t \in \mathcal{T}} \sum_{m \in \mathcal{M}} C_{mt} - T \frac{EST}{U(2,4)}$, où $\mathcal{U}$ représente la proportion de la capacité allouée utilisée pour la production. Le paramètre $\mathcal{U}$ varie dans nos instances dans l'ensemble $\{0.9, 1.0\}$. Le dénominateur du dernier terme diminue la part dédiée aux temps de changement en estimant que la durée qui sépare deux productions se situe entre 2 et 4 périodes

pour une référence donnée.

2. Pour chaque référence $i \in \mathcal{N}$, nous tirons aléatoirement un nombre u_i à partir d'une distribution uniforme $U(0,1)$. Nous normalisons ensuite les valeurs obtenus pour obtenir la proportion $\pi_i = \frac{u_i}{\sum_{j\in\mathcal{N}} u_j}$ de la demande totale correspondante à chaque référence i possible.

3. Nous affectons la demande de chaque référence i unité par unité grâce aux probabilités π_i tant que l'inégalité $\sum_{i,t} \min_m\{\tau_m^i\} D^i < \mathcal{U} \sum_{m,t} C_{mt} - T\frac{EST}{U(2,4)}$ est respectée, où D^i représente la demande pour la référence i sur l'ensemble de l'horizon de planification. Nous répartissons ensuite les D^i unités de chaque référence aléatoirement sur les T périodes. Pour 25% des références, cette répartition se fait selon un modèle périodique, typiquement en interdisant d'affecter des unités sur des périodes multiples d'un entier donné.

Stock cible. Le stock cible d'un produit i en période t correspond à la somme de la demande de i des périodes $t+1, \ldots, t+\Delta_{it}$. Δ_{it} correspond à la couverture cible, tirée suivant une distribution uniforme discrète $U(2,4)$. À noter que la couverture cible d'une période t+1 doit englober la couverture de la période t (i.e $\Delta_{i,t+1} \geq \Delta_{it}-1$ pour chaque référence i et période t).

Stock maximal. De manière équivalente, le stock maximal d'un produit i en période t correspond à la somme de la demande de i des périodes $t+1, \ldots, t+\Omega_{it}$, avec Ω_{it} correspondant à la couverture maximale déterminée à partir d'une distribution uniforme discrète $U(6,8)$.

Quantité minimum de production. Pour chaque produit i, la quantité minimum $q^i_{\min}$ est générée en suivant une distribution uniforme $U(1,3)$.

Le tableau 3.5 décrit la génération des paramètres de coût. Nous supposons que les heures supplémentaires engendrent une augmentation de 50% du coût d'utilisation d'une ligne. Le coût h_t^{i+} est toujours inférieur à h_t^{i-} dans notre génération. En effet, les planificateurs préfèrent être au dessus du stock cible plutôt que de s'exposer à des risques de rupture. Les coûts de rupture sont construits de façon à s'assurer qu'il est toujours rentable de produire et de garder en stock (dans les pires conditions) une unité pour satisfaire une demande plutôt que d'avoir des ventes perdues. Pour chaque combinaison de N, M, T et $Util$ nous générons 10 instances pour avoir un total de 560 instances. Pour chaque instance, nous faisons aussi varier le paramètre Φ introduit dans la partie 2.6 dans l'ensemble $\{1,2,4\}$.

Tableau 3.4 – Taille des instances

Classe	N	M	T
Petite	{20,30,40}	{1, 2}	{15,30}
Moyenne	{50}		
	{75,100}	{2,4}	
Grande	{125}	{4,6}	

Tableau 3.5 – Paramètres de coûts

Paramètre	Valeurs
Φ	$\{1, 2, 4\}$
c_{mt}	$\Phi U(500, 1000)$
$\overline{c}_{mt}$	$0.5c_{mt}$
h_u^{i+}	$U(1, 3)$
h_u^{i-}	$U(15, 30)$
p_{mt}^i	$U(0.2, 0.5)$
l_t^i	$\max_{s \leq t,m,j}\{p_{ms}^i + (c_{ms} + \overline{c}_{ms})(\tau_m^i + \frac{\gamma_m^{ji}}{q_{\min}^i}) + \sum_{u=s}^{\theta_i(t)-1} h_u^{i+}\}$

3.5 Résultats expérimentaux des modélisations

Les méthodes que nous avons développées ont été exécutées sur un serveur partagé possédant un Intel Xeon Gold 6230 composé de 80 CPUS cadencé à 2.10Ghz et 200 Go de Ram. Pour résoudre nos MIP, nous avons utilisé le solveur CPLEX dans sa dernière version (20.1) avec, lorsque ce n'est pas précisé, les paramétrages par défaut pour le Branch-and-Cut. Pour chaque résolution, 4 CPUS sont utilisés. Dans la suite, nous définissons l'écart, que nous notons *gap*, entre la borne supérieure (UB) et la borne inférieure (LB) par la formulation suivante :

$$Gap = 100 \cdot \frac{UB - LB}{LB}$$

La borne supérieure correspond à la meilleure solution obtenue dans la limite de temps fixée. UB et LB sont fournis par le solveur CPLEX à la fin de la résolution. En définissant le gap comme précédemment, on peut mesurer l'écart entre la solution obtenue et la meilleure borne inférieure. Ainsi un gap de 100% nous assure que la solution obtenue est, dans le pire des cas, deux fois supérieure à la solution optimale.

Nous comparons dans un premier temps la modélisation SCF et MTZ d'élimination des sous-tours dans la formulation MIP-AGG. Le tableau 3.6 présente les résultats sur un sous-ensemble d'instances pour un temps d'exécution limité à 900 secondes. Nous présentons le gap moyen ainsi que le gap maximal et le gap minimum obtenues pour chaque ensemble d'instances (10 au total) associées à une même taille.

La première observation que nous pouvons faire en analysant les gaps obtenus est que des instances d'une même taille peuvent être plus difficiles à résoudre pour le solveur CPLEX. En effet, pour la taille d'instances (20-1-30) et $\mathcal{U} = 1$, certaines instances ont pu être résolues à l'optimal alors que pour d'autres nous avons obtenu un gap de plus de 500%. Cela montre, d'une part, que notre génération permet d'obtenir des instances variées au sein d'un même ensemble et, d'autre part, que le comportement des solveurs de programmation entière est relativement imprévisible. En effet, les stratégies de branchement ou les méthodes de détection de symétries permettent de diminuer significativement les gaps à différents instants ponctuels de l'exécution. Il est donc possible que certains des écarts très importants observés soient causés par l'arrêt de la résolution (limite de temps atteinte) alors qu'une routine sous-jacente intégrée à CPLEX était en passe de réduire l'espace des solutions réalisables.

Nous observons que pour la taille d'instance (20-1-15), modéliser les sous-tours avec les contraintes SCF permet d'obtenir de meilleurs gaps moyens en un temps d'exécution plus court. En particulier la formulation MTZ n'arrive pas à fournir de solutions optimales en 900 secondes pour l'ensemble (20-1-15) et $\mathcal{U} = 1.0$. Pour les instances plus grandes, nous observons que la modélisation SCF trouve presque toujours le meilleur Gap_{min} mais obtient dans certains cas un Gap_{max} largement supérieur à la modélisation MTZ. Cela implique que nous obtenons souvent un meilleur Gap_{moy} avec la modélisation MTZ. Cela est particulièrement visible pour les instances (30-2-15), $\mathcal{U} = 0.9$ dans lequel le Gap_{max} est égale à 1610% ce qui augmente considérablement le gap moyen. Ce phénomène est tout de même à nuancer par le fait que pour $N \geq 30$ et $T = 30$, les solutions obtenues sont de mauvaises qualités dans les deux cas. Nous voyons facilement que ces deux paramètres influencent grandement la complexité du problème. Le nombre de machine M a un impact moins flagrant sur le gap, bien que présent. Finalement, nous observons que les instances dont les gaps sont les plus élevés présentent un paramètre d'utilisation de ligne élevé ($\mathcal{U} = 1$).

De façon générale, nous avons observé que la modélisation SCF permet de trouver de meilleurs gaps pour un plus grand nombre d'instances. Dans nos expérimentations, nous n'avons pas vu de différence entre la forme renforcée (3.12) et la forme classique (3.9) sur les solutions. Dans la suite dus résultats, nous supposons que la modélisation utilisée pour éliminer les sous-tours correspond donc à cette dernière.

Le tableau 3.7 présente les résultats obtenus en fonction de la formulation. Nous observons que la formulation désagrégée obtient de meilleurs gaps en moyenne par rapport à la formulation agrégée. Sur les petites instances, cette tendance est moins prononcée et

Tableau 3.6 – Résultats expériementaux de SCF et MTZ dans la formulation agrégée

Taille du problème N-M-T	$\mathcal{U}$		Gap_{moy} (%)	Gap_{min} (%)	Gap_{max} (%)	Temps (s)
20-1-15	0.9	SCF	0.35	0.0	2.74	393.1
20-1-15	0.9	MTZ	2.22	0.0	13.36	888.9
20-1-15	1.0	SCF	0.58	0.0	2.2	591.0
20-1-15	1.0	MTZ	2.71	0.42	6.07	900.0
20-2-15	0.9	SCF	6.6	0.0	26.42	833.6
20-2-15	0.9	MTZ	7.91	0.11	17.43	900.1
20-2-15	1.0	SCF	9.72	0.0	46.48	858.3
20-2-15	1.0	MTZ	5.97	0.27	12.32	900.0
20-1-30	0.9	SCF	11.78	0.0	57.01	865.0
20-1-30	0.9	MTZ	24.07	0.59	69.29	900.0
20-1-30	1.0	SCF	66.98	0.0	514.53	887.2
20-1-30	1.0	MTZ	32.15	1.92	162.32	900.0
20-2-30	0.9	SCF	5.81	0.51	14.1	900.0
20-2-30	0.9	MTZ	33.64	1.26	120.65	900.0
20-2-30	1.0	SCF	87.33	2.01	258.37	900.0
20-2-30	1.0	MTZ	55.83	18.5	102.82	900.0
30-1-15	0.9	SCF	13.09	0.06	46.63	900.0
30-1-15	0.9	MTZ	37.0	3.21	104.05	900.1
30-1-15	1.0	SCF	13.6	0.0	41.66	893.6
30-1-15	1.0	MTZ	31.58	3.31	97.99	900.0
30-2-15	0.9	SCF	183.02	2.85	1610.55	900.0
30-2-15	0.9	MTZ	75.15	8.39	211.87	900.0
30-2-15	1.0	SCF	30.31	5.0	109.52	900.0
30-2-15	1.0	MTZ	140.93	17.32	554.35	900.0
30-1-30	0.9	SCF	2333.04	0.98	19348.01	900.0
30-1-30	0.9	MTZ	1295.19	5.37	4162.49	900.5
30-1-30	1.0	SCF	8678.81	27.71	24287.13	900.0
30-1-30	1.0	MTZ	1285.0	54.64	3874.09	900.9
30-2-30	0.9	SCF	3968.54	5.6	30884.03	900.0
30-2-30	0.9	MTZ	751.58	64.91	4870.01	900.0
30-2-30	1.0	SCF	2037.21	9.78	8531.96	900.0
30-2-30	1.0	MTZ	561.4	85.04	3356.44	900.0
40-1-15	0.9	SCF	70.29	0.19	307.76	900.0
40-1-15	0.9	MTZ	702.31	11.68	2193.5	901.0
40-1-15	1.0	SCF	2923.54	25.02	22406.81	900.0
40-1-15	1.0	MTZ	739.33	22.49	3320.73	900.0
40-2-15	0.9	SCF	537.94	2.61	3563.97	900.0
40-2-15	0.9	MTZ	537.15	15.64	2316.76	900.0
40-2-15	1.0	SCF	1395.85	5.0	7396.69	900.0
40-2-15	1.0	MTZ	267.93	55.29	593.02	900.0
40-1-30	0.9	SCF	6190.99	713.69	17149.16	900.0
40-1-30	0.9	MTZ	6320.55	630.24	12885.09	901.9
40-1-30	1.0	SCF	7587.47	366.54	20331.88	900.0
40-1-30	1.0	MTZ	7898.93	629.91	15602.95	900.0
40-2-30	0.9	SCF	9253.24	630.78	24584.58	900.1
40-2-30	0.9	MTZ	4900.77	175.7	17433.77	900.0
40-2-30	1.0	SCF	11449.44	2606.27	28378.05	900.0
40-2-30	1.0	MTZ	2888.1	392.88	21653.91	900.0

on observe d'ailleurs de meilleurs temps de résolution pour la formulation agrégée pour les instances (20-1-15). Au delà, nous observons que la formulation basée sur le Facility location fournit régulièrement un meilleur gap plus rapidement, même lorsque le gap obtenu par MIP-AGG en 900 secondes est élevé. On peut également noter que les bornes inférieures obtenues par MIP-FL sont de meilleures qualités. Les gaps obtenus avec cette formulation restent acceptables pour les instances avec $N \leq 30$.

Des tableaux similaires au tableau 3.7 sont présentés en Annexe. En particulier, nous présentons dans le tableau A.1 les gaps obtenus par les résolutions en fonction de la valeur du paramètre Φ. Nous voyons que ce paramètre influence grandement la qualité des solutions obtenues pour les instances avec $N = 20$. Pour l'ensemble d'instances (20-1-30), $\mathcal{U} = 1.0$, nous voyons que passer de $\Phi = 1$ à $\Phi = 4$ multiplie par 10 le gap moyen obtenu. Sur les instances de plus grandes tailles avec $N = 30$, il est difficile de conclure sur l'influence du paramètre Φ sur la qualité des solutions. Néanmoins, dans le cas de gaps aussi grands, la solution obtenue par le solveur correspond généralement à la première solution trouvée qui n'est pas améliorée jusqu'au temps limite d'exécution. Ainsi, en fonction de la stratégie de branchement du solveur, il est possible d'obtenir une meilleure première solution pour $\Phi = 4$.

3.5.1 Analyse des inégalités valides

Dans les solveurs récents, des méthodes spécifiques permettent d'ajouter des inégalités valides directement dans le Branch-and-Cut. À chaque noeud, en plus des coupes directement ajoutées par CPLEX telles que les coupes MIR ou les coupes de Gomory, nos propres coupes sont ajoutées. Dans CPLEX, ces coupes sont appelées des *User Cuts*. Comme la figure 3.2 le montre, ces coupes renforcent la relaxation linéaire de notre problème en réduisant la taille du polyèdre tout en s'assurant qu'aucune solution entière ne soit retirée de l'ensemble réalisable. C'est aussi le cas pour les coupes que CPLEX ajoute dans le Branch-and-Cut. De façon automatique, CPLEX peut retirer des coupes si elles sont redondantes avec celles déjà implémentées nativement. De même lorsque CPLEX détecte l'ajout d'inégalités valides par l'utilisateur dans le modèle, certaines heuristiques pour améliorer le branchement sur des variables dans l'arbre de recherche sont désactivées. Ainsi l'impact de l'ajout d'inégalités valides via le solveur peut être compliqué à mesurer. Le tableau A.2 présente les gaps obtenues en ajoutant les inégalités (l, S) et les inégalités (k, U). Ce tableau montre aussi le nombre de coupes générées par CPLEX de façon native (CPLEX Cut) et le nombre de nos coupes qui sont retenues par CPLEX (User Cut) dans le Branch-and-Cut. La première observation est que le nombre de nos coupes ajoutées par

Tableau 3.7 – Résultats expériementaux des formulations MIP-AGG et MIP-FL

Taille du problème N-M-T	$\mathcal{U}$		Gap_{moy} (%)	Gap_{min} (%)	Gap_{max} (%)	Temps (s)
20-1-15	0.9	MIP-AGG	0.35	0.0	2.74	393.1
20-1-15	0.9	MIP-FL	0.56	0.0	5.08	419.4
20-1-15	1.0	MIP-AGG	0.58	0.0	2.2	591.0
20-1-15	1.0	MIP-FL	0.51	0.0	2.47	592.2
20-2-15	0.9	MIP-AGG	6.6	0.0	26.42	833.6
20-2-15	0.9	MIP-FL	4.23	0.0	11.05	830.9
20-2-15	1.0	MIP-AGG	9.72	0.0	46.48	858.3
20-2-15	1.0	MIP-FL	4.79	0.0	10.36	851.1
20-1-30	0.9	MIP-AGG	11.78	0.0	57.01	865.0
20-1-30	0.9	MIP-FL	4.19	0.0	11.51	892.6
20-1-30	1.0	MIP-AGG	66.98	0.0	514.53	887.2
20-1-30	1.0	MIP-FL	9.11	0.0	32.23	880.4
20-2-30	0.9	MIP-AGG	5.81	0.51	14.1	900.0
20-2-30	0.9	MIP-FL	3.3	0.41	5.97	900.0
20-2-30	1.0	MIP-AGG	87.33	2.01	258.37	900.0
20-2-30	1.0	MIP-FL	19.34	2.1	38.74	900.0
30-1-15	0.9	MIP-AGG	13.09	0.06	46.63	900.0
30-1-15	0.9	MIP-FL	17.43	0.3	76.17	900.0
30-1-15	1.0	MIP-AGG	13.6	0.0	41.66	893.6
30-1-15	1.0	MIP-FL	19.12	0.0	81.03	883.3
30-2-15	0.9	MIP-AGG	183.02	2.85	1610.55	900.0
30-2-15	0.9	MIP-FL	26.11	4.33	101.06	900.0
30-2-15	1.0	MIP-AGG	30.31	5.0	109.52	900.0
30-2-15	1.0	MIP-FL	23.12	3.66	59.48	900.0
30-1-30	0.9	MIP-AGG	2333.04	0.98	19348.01	900.0
30-1-30	0.9	MIP-FL	395.73	0.6	2199.72	900.0
30-1-30	1.0	MIP-AGG	8678.81	27.71	24287.13	900.0
30-1-30	1.0	MIP-FL	106.61	12.66	670.92	900.0
30-2-30	0.9	MIP-AGG	3968.54	5.6	30884.03	900.0
30-2-30	0.9	MIP-FL	584.69	3.34	5530.61	900.0
30-2-30	1.0	MIP-AGG	2037.21	9.78	8531.96	900.0
30-2-30	1.0	MIP-FL	805.85	14.08	7802.98	900.0
40-1-15	0.9	MIP-AGG	70.29	0.19	307.76	900.0
40-1-15	0.9	MIP-FL	45.25	0.33	194.14	900.0
40-1-15	1.0	MIP-AGG	2923.54	25.02	22406.81	900.0
40-1-15	1.0	MIP-FL	82.24	10.86	175.25	900.0
40-2-15	0.9	MIP-AGG	537.94	2.61	3563.97	900.0
40-2-15	0.9	MIP-FL	75.92	5.1	258.56	900.0
40-2-15	1.0	MIP-AGG	1395.85	5.0	7396.69	900.0
40-2-15	1.0	MIP-FL	64.55	8.08	117.85	900.0
40-1-30	0.9	MIP-AGG	6190.99	713.69	17149.16	900.0
40-1-30	0.9	MIP-FL	6844.53	28.58	20550.26	900.0
40-1-30	1.0	MIP-AGG	7587.47	366.54	20331.88	900.0
40-1-30	1.0	MIP-FL	4871.43	13.02	19110.24	900.0
40-2-30	0.9	MIP-AGG	9253.24	630.78	24584.58	900.1
40-2-30	0.9	MIP-FL	16177.08	1882.69	22562.97	900.0
40-2-30	1.0	MIP-AGG	11449.44	2606.27	28378.05	900.0
40-2-30	1.0	MIP-FL	13840.07	2058.11	28027.61	900.0

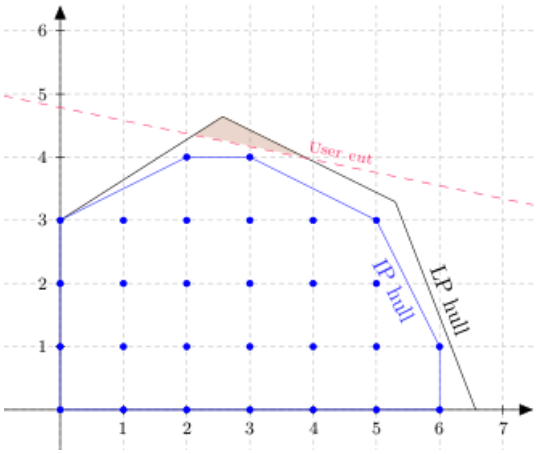


FIGURE 3.2 – User Cuts ajoutés dans CPLEX [96]

le solveur est très faible et représente moins de 1% des coupes totales dans de nombreux cas. Cette observation se traduit par des solutions qui ne sont pas meilleures que celles obtenues avec les paramétrages de base de CPLEX. Dans le cas des coupes (l, S), Absi et Kedad-Sidhoum [5] ont observé des résultats concluants pour un problème similaire sur une version de CPLEX antérieure (9.0). Ils ont noté qu'ajouter leurs coupes dans le Branch-and-Cut permettait de réduire le gap et d'obtenir de meilleures bornes inférieures. Dans notre cas, l'ajout de ces coupes a été inefficace. Notre conclusion est donc que le solveur CPLEX intègre de nos jours directement des coupes plus efficaces dans son Branch-and-Cut, et que nos coupes ne permettent pas d'accélérer significativement la résolution.

3.6 Conclusion

Dans ce chapitre, nous avons présenté le problème central que nous traitons dans cette thèse. Après avoir introduit les notations, nous avons proposé deux formulations mathématiques : une formulation agrégée et une formulation désagrégée basée sur le Facility location. Les résultats permettent de mettre en évidence que la deuxième formulation donne de meilleures bornes supérieures et bornes inférieures en règle générale. Une des principales difficultés de notre problème vient de la prise en compte des temps de réglage qui dépendent de la séquence. En particulier, leur modélisation amène à l'introduction d'un ensemble de contraintes s'assurant qu'aucun sous-tour n'est présent dans les séquences obtenues. Plusieurs modélisations existent pour éliminer les sous-tours, nous avons choisi de baser nos modèles sur la formulation SCF. Pour éprouver nos différentes approches,

nous avons introduit une nouvelle génération d'instances, basée sur la combinaison de plusieurs paramètres définis en collaboration avec notre partenaire industriel. Nous obtenons ainsi de nombreuses instances dont la difficulté varie au sein d'un sous-ensemble de même taille. Nous avons exploré l'ajout d'inégalités valides pour accélérer la résolution de nos méthodes. Les dernières versions de CPLEX offrant déjà un ensemble important de coupes intégrées automatiquement dans le Branch-and-Cut, nous avons testé d'y ajouter nos coupes à travers des fonctions proposées par CPLEX. Nos expérimentations ont montré que nos coupes sont souvent redondantes avec celles ajoutées par CPLEX et sont donc supprimées par ce dernier. Ainsi l'ajout de coupes n'a pas été concluant pour améliorer la qualité de nos solutions. De manière générale, les résultats que nous avons obtenus montrent les limitations de la simple utilisation d'un solveur pour résoudre le problème que nous considérons sur les instances générées. Le facteur aléatoire de certains paramètres dans la génération nous a permis d'obtenir des instances variées pour lesquelles, pour une même taille, nous observons des résultats drastiquement différents. D'un point de vue industriel, il est nécessaire de pouvoir anticiper la qualité des solutions que nous obtenons pour une même taille d'instances. Dans la suite, nous allons donc nous concentrer sur la définition de nouvelles méthodes de résolution, permettant d'obtenir un bon compromis entre le temps d'exécution et la qualité de la solution pour l'ensemble de nos instances.

Chapitre 4

MIP-heuristiques appliquées à une approche de clustering

Dans ce chapitre, nous présentons plusieurs heuristiques pour résoudre le problème étudié. Le but de ces méthodes est d'obtenir une solution acceptable dans un temps réduit. Nous introduisons dans un premier temps des heuristiques constructive et de recherche locale classiques pour résoudre des problèmes de lot-sizing de grande taille. Ensuite, nous proposons une approche originale de *clustering* pour réduire le nombre de variables dans notre modélisation du problème, qui se base sur les spécificités des temps de réglage dans l'agroalimentaire.

4.1 MIP-heuristiques

Comme expliqué au chapitre 1, l'enjeu pour VIF est de mettre au point une méthode algorithmique rapide pour obtenir une solution au problème générique décrit. Malheureusement, les deux formulations présentées au chapitre 3 sont trop complexes et de trop grande taille pour pouvoir obtenir rapidement de bonnes solutions (ou même réalisables) sur la plupart des ensembles de données réalistes avec l'un des solveurs commerciaux existants. Dans cette section, nous proposons des approches alternatives pour résoudre ce problème. L'utilisation d'heuristiques est souvent privilégiée pour obtenir de bonnes solutions, bien que sous-optimales, en un temps de calcul court. Un exemple d'heuristique fréquemment rencontré dans la littérature pour résoudre le lot-sizing est le Relax-and-Fix (R&F).

4.1.1 Relax-and-Fix basée sur une décomposition de l'horizon de temps

Le principe général de la méthode Relax-and-Fix est de résoudre itérativement plusieurs MIP simplifiés dans lesquels certaines variables sont fixées et d'autres relâchées.

Elle appartient donc à une classe de procédures approchées appelées *MIP-heuristiques* ou encore *matheuristiques* car elle se base sur la résolution de programmes linéaires en nombres entiers. La solution des sous-problèmes ainsi définis est obtenue plus rapidement grâce à une diminution de la taille de l'arbre de recherche dans la méthode de Brand-and-Cut utilisée. La réduction du problème se fait habituellement par une approche de décomposition de l'horizon de temps. À chaque itération, nous partitionnons l'horizon de planification en différentes fenêtres qui se déplacent :

Fenêtre gelée : la fenêtre gelée correspond à une partie de l'horizon où les variables entières et binaires sont fixées par des résolutions antérieures. Plusieurs variantes sont utilisées pour fixer ces variables.

Fenêtre de décision : la fenêtre de décision correspond à une partie de l'horizon dans laquelle les variables du problème original ne changent pas de nature. En particulier dans notre cas, les variables entières et binaires sont identiques à celles des formulations présentées au chapitre précédent.

Fenêtre d'approximation : La fenêtre d'approximation est une partie de l'horizon où les toutes les variables entières et binaires sont relâchées. On obtient ainsi un plan de production relâché sur cette portion de l'horizon, qui produit ainsi une solution dont la valeur est une borne inférieure sur les coûts encourus.

La figure 4.1, inspirée de [4], montre une représentation des différentes fenêtres dans l'heuristique R&F.

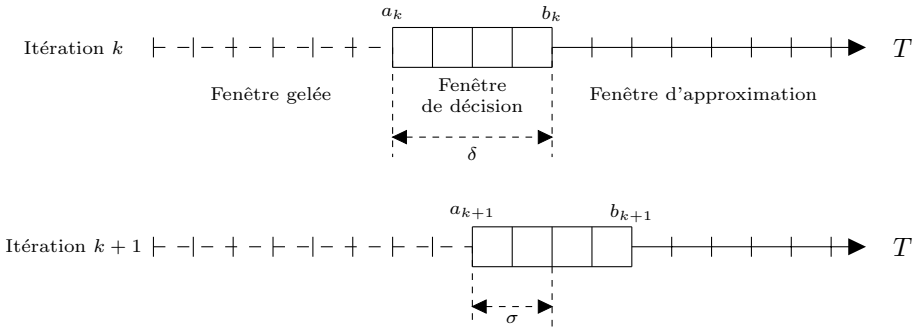


FIGURE 4.1 – Décomposition de l'horizon dans le R&F

Le principe de l'heuristique Relax-and-Fix est dans un premier temps de résoudre le problème avec une fenêtre de décision de taille δ en début d'horizon et une fenêtre d'approximation sur le reste de l'horizon. À l'itération suivante, la fenêtre de décision est

déplacée de $\delta - \sigma$ périodes vers la droite, le plus souvent en gardant un chevauchement avec la fenêtre de décision précédente ($\sigma < \delta$). Les périodes qui précèdent la nouvelle fenêtre de décision entrent dans la fenêtre gelée. Cette procédure est répétée jusqu'à la disparition de la fenêtre d'approximation, c'est-à-dire quand la fenêtre de décision atteint la fin de l'horizon.

L'heuristique R&F construit donc une solution réalisable de manière itérative en résolvant plusieurs sous-problèmes avec moins de variables binaires. Plusieurs types de décomposition de ce type ont été proposées, cependant nous nous concentrons sur la décomposition basée sur les périodes, largement appliquée dans la littérature et généralement considérée comme plus efficace que la décomposition alternative basée sur les produits (voir [57] pour une comparaison des deux approches)

4.1.2 Heuristique Relax-and-Fix appliquée au CLSSD-PM

La procédure conserve à tout instant une valeur pour l'ensemble des variables du programme, stockées dans un vecteur noté sol^k à l'issue de l'itération k. On définit ainsi $sol^k(x)$ la valeur de la variable de décision x dans la solution obtenue à l'itération k de l'algorithme. De plus, on note $a_k = 1 + (k-1)(\delta - \sigma)$ et $b_k = k\delta - (k-1)\sigma$ respectivement la première et la dernière période de la fenêtre de décision associée. Soit $P^k_{R\&F}$ le sous-problème relaxé à l'itération k, qui est résolu en utilisant soit MIP-AGG, soit MIP-FL introduite dans la Section 3.2. Le problème $P^k_{R\&F}$ est similaire à celui d'origine, à l'exception des contraintes de définition des variables (3.1) qui sont remplacées par les suivantes :

$$w^{ij}_{mt} \in \{0,1\} \quad \forall i, \forall j, \forall m, \forall t \in [a_k, b_k] \tag{4.1}$$

$$w^{ij}_{mt} \in [0,1] \quad \forall i, \forall j, \forall m, \forall t \in [b_k + 1, T] \tag{4.2}$$

$$w^{ij}_{mt} = sol^{k-1}(w^{ij}_{mt}) \quad \forall i, \forall j, \forall m, \forall t \in [1, a_k - 1], \forall k \geq 2 \tag{4.3}$$

Afin de contrôler le temps total nécessaire pour obtenir une solution avec la procédure R&F, nous ajoutons deux paramètres qui limitent le temps de calcul alloué à chaque sous-problème $P^k_{R\&F}$. Nous définissons d'abord le gap limite $\gamma > 0$ pour la solution de $P^k_{R\&F}$. Le gap que nous utilisons est défini directement dans le solveur de la manière suivante :

$$Gap_{\text{CPLEX}} = 100 \cdot \frac{UB - LB}{UB} \tag{4.4}$$

, avec UB et LB représentant respectivement la borne supérieure et inférieure de la solution du problème. Ainsi, à chaque itération k, le solveur renvoie la première solution trouvée dont le gap est inférieur ou égal à γ. Pour éviter les situations dans lesquelles atteindre γ induit un temps de calcul long, nous fixons également une limite de temps ρ pour chaque itération, c'est-à-dire que l'algorithme renvoie la meilleure solution réalisable obtenue en ρ unités de temps, indépendamment du gap final atteint.

Lors de la première itération, l'algorithme résout le problème MIP-AGG (ou MIP-FL) en considérant les variables binaires w_{mt}^{ij} dans la fenêtre de décision ($t = 1, \ldots, \delta$) et leur version relâchée de la fenêtre d'approximation, soit $0 \leq w_{mt}^{ij} \leq 1$ pour tout $t = \delta + 1, \ldots, T$. Dans la deuxième itération, la fenêtre de décision est décalée vers la droite de $\delta - \sigma$ périodes, où $\sigma < \delta$ est le nombre de périodes de chevauchement dans la fenêtre de décision entre deux itérations consécutives. Les variables d'affectation de la fenêtre d'approximation précédente qui entrent dans la fenêtre de décision sont alors ramenées à des variables binaires. Les variables binaires qui quittent la fenêtre de décision, c'est-à-dire dont l'index appartient à $\{1, \ldots, \delta - \sigma\}$, se figent et conservent leur valeur calculée à l'étape de résolution précédente. L'algorithme répète la procédure de décalage et de fixation des variables jusqu'à ce que la fenêtre de décision atteigne la fin de l'horizon.

Algorithme 3 Relax-and-Fix

procédure R&F($\delta, \sigma, \gamma, \rho$)
  $k \leftarrow 1, a_k \leftarrow 1, b_k \leftarrow \delta, sol^k \leftarrow \emptyset$
  tant que $b_k < T$ **faire**
    $sol^k \leftarrow$ Résoudre $P_{R\&F}^k$ avec le gap limite γ en ρ unités de temps
    $a_{k+1} \leftarrow b_k - \sigma + 1$, $b_{k+1} \leftarrow \min\{b_k + \delta - \sigma, T\}$, $k \leftarrow k + 1$
  fin tant que
  Résoudre $P_{R\&F}^k$
  Retourner sol^k
fin procédure

L'heuristique R&F que nous décrivons dans l'algorithme 3 utilise une décomposition en période partant du début jusqu'à la fin de l'horizon. Une version dans le sens inverse, c'est-à-dire qui commence à la fin de l'horizon et se termine à la première période, a également été testée dans la littérature ([106]). Les deux versions ont affiché des performances équivalentes lors de nos premiers tests, nous nous concentrons donc sur la version

classique dans la suite.

4.1.3 Heuristique Fix-and-Optimize

L'heuristique R&F est fréquemment combinée avec une procédure de recherche locale appelée Fix-and-Optimize (F&O) qui améliore la solution réalisable trouvée. Comme pour l'heuristique R&F, la procédure repose sur une décomposition du problème d'origine en sous-problèmes plus simples, où la décomposition la plus classique est à nouveau basée sur les périodes. Partant d'une solution de référence initiale, l'idée est de « libérer » un sous-ensemble de variables binaires dont l'indice temporel appartient à une fenêtre de décision donnée et de les « réoptimiser » afin d'améliorer la solution. Si une meilleure solution est trouvée, elle devient la nouvelle référence et est utilisée comme telle dans les itérations suivantes. Lors de chaque itération, les variables binaires n'appartenant pas au sous-ensemble mentionné ci-dessus sont fixées soit à leur valeur initiale, soit à une nouvelle valeur définie dans une solution de référence de l'itération précédente. À chaque itération, seules les variables binaires (c'est-à-dire les variables de réglage) en dehors de la fenêtre de décision sont fixées alors que les autres peuvent être modifiées pour améliorer la solution courante. La Figure 4.2 montre la décomposition en période dans le F&O. Les paramètres utilisés pour définir la fenêtre à optimiser et son avancement dans l'horizon peuvent différer de ceux du R&F, même si il est fréquent de conserver les mêmes valeurs σ et δ.

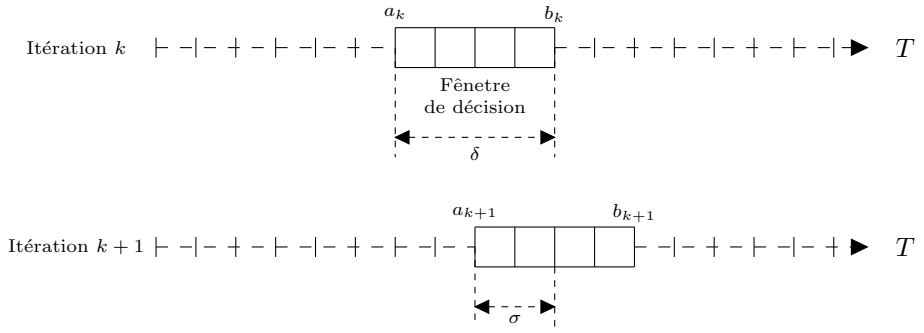


FIGURE 4.2 – Décomposition de l'horizon dans le F&O

Le problème $P^k_{F\&O}$ à résoudre à l'itération k est défini en remplaçant les contraintes de définition des variables (3.1) par :

$$w^{ij}_{mt} \in \{0,1\} \quad \forall i, \forall j, \forall m, \forall t \in [a_k, b_k]$$

$$w_{mt}^{ij} = sol^{k-1}(w_{mt}^{ik}) \quad \forall i, \forall j, \forall m, \forall t \in [1, a_k - 1] \cup [b_k + 1, T]$$

où $sol^{k-1}(x)$ correspond à la valeur de x dans la solution obtenue à l'itération précédente. Le pseudo-code de l'algorithme de Fix-and-Optimize est le suivant :

Algorithme 4 Fix-and-Optimize

1: **procédure** F&O$(\delta, \sigma, \gamma, \rho)$
2: $k \leftarrow 1, a_k \leftarrow 1, b_k \leftarrow \delta, sol^0 \leftarrow sol(R\&F)$
3: **tant que** $b_k < T$ **faire**
4: $sol^k \leftarrow$ Résoudre $P^k_{F\&O}(sol^{k-1}, [a_k, b_k], \gamma, \rho)$
5: $a_{k+1} \leftarrow b_k - \sigma + 1$, $b_{k+1} \leftarrow \min\{b_k + \delta - \sigma, T\}$
6: $k \leftarrow k + 1$
7: **fin tant que**
8: Résoudre $P^k_{F\&O}(sol^{k-1}, [a_k, T], \gamma, \rho)$
9: **fin procédure**

4.1.4 RFFO : Relax-and-Fix et Fix-and-Optimize

La combinaison des heuristiques Relax-and-Fix et Fix-and-Optimize (RFFO), est particulièrement efficace pour les problèmes complexes de lot-sizing et offre généralement un bon compromis entre le temps d'exécution et la qualité de la solution. Par exemple, [60] présente ces deux approches pour le problème de lot-sizing avec des réglages dépendants de la séquence sur des machines parallèles et obtient de bons résultats en peu de temps de calcul. Dans les deux cas, la taille de la fenêtre de décision influence grandement le temps de calcul de la procédure. Les différentes étapes de la procédure RFFO que nous proposons sont décrites ci-dessous :

Initialisation Définir les paramètres $\delta, \sigma, \gamma, \rho$ pour le R&F et F&O.

Étape 1 Appliquer l'heuristique R&F$(\delta, \sigma, \gamma, \rho)$ pour obtenir une solution réalisable.

Étape 2 Appliquer l'heuristique F&O sur la solution courante avec les mêmes paramètres $\delta, \sigma, \gamma, \rho$.

Dans la suite, nous notons RFFO$(\delta, \sigma, \gamma, \rho)$ la procédure RFFO en utilisant les paramètres $\delta, \sigma, \gamma, \rho$ dans l'étape 1 et l'étape 2. Une analyse de l'impact des différents paramètres (c'est-à-dire la taille de la fenêtre de décision, le chevauchement entre deux fenêtres de

décision consécutives, le gap limite et le temps limite dans chaque itération) est présentée dans la partie 4.3. Nous verrons notamment que cette procédure ne suffit pas pour obtenir des solutions acceptables sur les instances les plus compliquées à résoudre. Dans la suite, nous proposons une approche basée sur du clustering pour réduire la taille des instances.

4.2 Approche de clustering

Généralement, l'utilisation de MIP-heuristiques comme le R&F permet d'obtenir de bons résultats sur la plupart des problèmes de lot-sizing. Dans notre cas, ces heuristiques ne suffissent pas pour obtenir des solutions satisfaisantes dans un délai raisonnable sur des instances de grande taille (voir Partie 4.3). L'une des causes principales à cette difficulté est la prise en compte des réglages dépendants de la séquence. Retirer ou réduire cette composante pour obtenir des temps qui dépendent uniquement de la référence à produire est susceptible de nettement améliorer la convergence de la procédure. Comme expliqué dans la partie 2.6 qui présente la génération d'instances, les temps de réglage des lignes de production observés dans l'agroalimentaire sont le plus souvent influencés par les caractéristiques des différents produits. On constate donc assez naturellement que des regroupements peuvent s'effectuer entre produits qui présentent des similarités. Ainsi notre idée est de déterminer une approche permettant de partitionner automatiquement les références en familles, de telle sorte que le temps d'une séquence soit principalement influencé par l'ordonnancement des différentes familles et qu'on puisse négliger celui des articles au sein d'une même famille. L'approche que nous développons dans la suite s'appuie une méthode de clustering classique en apprentissage machine. Dans la suite, pour simplifier la lisibilité, nous utilisons l'abréviation SD (pour *sequence dependent*) qui désigne un réglage qui dépend de la séquence. Par opposition, SI (pour *sequence independent*) désigne un réglage qui ne dépend pas de la séquence, c'est-à-dire que pour tout $i \in \mathcal{N}$ et $m \in \mathcal{M}$, $\lambda_m^{ij} = \lambda_m^j$ pour tout $j \in \mathcal{N} \setminus \{i\}$.

Notre approche consiste à trouver un moyen de transformer une instance I du problème CLSSD-PM en une instance $\tilde{I}$ dans laquelle certains réglages sont considérés comme indépendants de la séquence et dont la formulation nécessite moins de variables binaires w_{mt}^{ij}. Comme cette opération peut supprimer des informations cruciales sur les véritables temps de réglage associés à une séquence de production, nous veillons à ce que (i) il existe une séquence de production réalisable sur chaque machine dans chaque période qui produit la même quantité de chaque référence en respectant la capacité disponible, et (ii) les temps de réglages SI associés à un plan de production pour $\tilde{I}$ restent comparables à leur

homologue SD en I.

Nous construisons notre procédure de conversion à partir de l'observation des caractéristiques des temps de réglage. En effet, la majorité des temps de changement entre les références sont soit « longs » (entre 30 et 80 minutes) soit plutôt « courts » (moins de 15 minutes) et satisfont l'inégalité triangulaire, c'est-à-dire qu'il n'est jamais préférable d'effectuer plusieurs réglages intermédiaires (avec une ou plusieurs références de transition) pour arriver au réglage voulu. Prenons par exemple le cas d'un producteur qui produit trois types de cookies. Les cookies de type 1 contiennent des ingrédients allergènes tandis que les types 2 et 3 sont tous deux exempts d'allergènes. Le temps de nettoyage entre le type 1 et l'un des deux autres est généralement beaucoup plus long que celui entre les types 2 et 3, et prolonge les temps de réglage en conséquence. Des asymétries similaires peuvent également se produire en raison des différences de températures du four, des produits bio/non bio etc, qui correspondent à l'ensemble des « caractéristiques » envisageables pour les références considérées semblables à celles décrites dans la partie 2.6. En pratique, les planificateurs s'appuient sur ces caractéristiques pour réduire les temps de réglage globaux et maximiser l'utilisation des machines. Ils adoptent ainsi une politique intuitive qui consiste à regrouper les références avec des ingrédients ou des caractéristiques de recette similaires. Celle-ci leur permet de limiter le nombre de réglages longs dans chaque séquence de production et donc d'éliminer les longues périodes d'inactivité avec un fort impact sur le temps net de production disponible. Le facteur déterminant du temps de réglages total devient alors l'agencement de ces groupes de référence, plutôt que la séquence des références au sein de chaque groupe particulier.

Nous cherchons à reproduire cette pratique, mais en l'automatisant et en utilisant de l'apprentissage machine pour utiliser les données à notre disposition sur les temps de changement pour créer des regroupements pertinents. Cela nous permet de définir des instances plus simples pour lesquelles nous ne considérons que les temps de réglage dépendants de la séquence entre les familles, tandis que nous utilisons une approximation des temps de réglage au sein d'une même famille par des temps virtuels indépendants de la séquence. L'approche que nous développons a pour but de réduire la taille du problème par la diminution du nombre de variables binaires dans la modélisation. Une classe de problèmes dans laquelle des *major setups* par famille et des *minor setups* par référence a déjà été étudié dans la littérature et est présenté sous le nom de *coordinated lot sizing problem* ([94, 29, 95]). Ce problème ne prend pas en compte les séquences pour l'intégration des *setups*.

4.2.1 Partionning Around Medoid (PAM)

Notre procédure consiste à partitionner les références en clusters ou familles de telle sorte que le temps de réglage entre deux références quelconques d'un même cluster soit négligeable par rapport au temps de réglage entre deux références appartenant à des clusters différents. Pour cela, nous nous appuyons sur un algorithme de partitionnement appelé PAM (Partitioning Around Medoids) parfois aussi appelé k-medoids, dont les seules entrées sont un entier k (indiquant le nombre de clusters) et les temps de réglage entre toutes les paires de référence. Par conséquent, PAM ne nécessite qu'une matrice de distances pour chaque paire de points, une caractéristique souhaitable par rapport aux procédures alternatives telles que l'algorithme k-means qui nécessite de définir les coordonnées de chaque point dans un espace normé. Il est important de noter que l'algorithme PAM original considère les distances symétriques pour déterminer quel médoïde est le plus proche de chaque point. Pour répondre à cette exigence tout en garantissant la robustesse de notre approche, nous définissons un temps de réglage virtuel $\tilde{\lambda}_m^{ij} = \max\{\lambda_m^{ij}, \lambda_m^{ji}\}$ pour tous les $i, j \in \mathcal{N}$ et $m \in \mathcal{M}$. Ainsi, deux éléments avec des temps de réglage asymétriques (causés par la présence de composants issus de l'agriculture biologiques par exemple), sont susceptibles de se retrouver dans deux groupes différents si la valeur maximale de leurs temps de changement est grande. L'idée générale de la procédure PAM est de construire des clusters de manière itérative autour de points (références) spécifiques, considérés comme le medoïde (centre) de leur cluster respectif. À n'importe quelle étape de l'algorithme, la qualité de la partition actuelle sur une machine donnée $m \in \mathcal{M}$ est évaluée à l'aide d'une fonction de coût E_m qui est définie à partir des temps de réglage virtuels comme suit :

$$E_m = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{C}_i} \tilde{\lambda}_m^{ij} \tag{4.5}$$

, où $\mathcal{I} \subseteq \mathcal{N}$ est l'ensemble des médoïdes de la solution courante et pour tout $i \in \mathcal{I}$, $\mathcal{C}_i$ représente l'ensemble des références qui appartiennent au cluster associé au médoïde i. La procédure commence par une partition initiale basée sur k médoïdes sélectionnés au hasard et se poursuit en effectuant des permutations successives entre l'une des références médoïdes actuelles et d'autres (non-médoïdes). Elle assigne ensuite chaque point à son médoïde le plus proche et prend cette nouvelle configuration comme référence si elle améliore la qualité des clusters obtenus (sur la base de la valeur E_m). La procédure répète les étapes précédentes jusqu'à ce qu'aucune nouvelle amélioration ne soit observée. L'Algorithme 5 présente la procédure PAM.

Comme mentionné précédemment, l'une des entrées requises par PAM est le nombre

Algorithme 5 PAM

```
1: procédure PAM(k, 𝒩, m)
2:     actualCost = +∞, ℐ ← ∅
3:     Choisir aléatoirement k références ∈ 𝒩 pour être les premiers médoïdes
4:     Mettre à jour ℐ
5:     Associer chaque référence i ∈ 𝒩 \ ℐ au plus proche médoïde
6:     actualCost = Calculer E_m
7:     tant que actualCost diminue faire
8:         pour i in ℐ faire
9:             pour o in 𝒩 \ ℐ faire
10:                Calculer E_m pour l'ensemble des médoïdes ℐ' = ℐ ∪ {o} \ {i}
11:                si E_m < actualCost alors
12:                    actualCost ← E_m
13:                    ℐ ← ℐ'
14:                    Retourner en 7
15:                fin si
16:            fin pour
17:        fin pour
18:    fin tant que
19:    Retourner ℐ
20: fin procédure
```

k de clusters à définir. Il reste donc à estimer une valeur de k qui assure une partition pertinente des références. À cette fin, nous utilisons un indicateur appelé score *silhouette* pour chaque référence. Ce score mesure la qualité d'appariement d'une référence avec les autres appartenant au même groupe. Plus formellement, soit $m \in \mathcal{M}$ et $i \in \mathcal{I}$: le score obtenu par la référence $j \in \mathcal{C}_i$ repose sur une combinaison entre la distance moyenne de j avec les autres références de $\mathcal{C}_i$, par rapport à la distance moyenne entre j et les références de n'importe quel autre cluster $\mathcal{C}_{i'}$ pour $i' \neq i$. Définissons $a_m(j)$ comme le temps de réglage moyen de la référence j aux autres références du cluster $\mathcal{C}_i$ sur la machine m :

$$a_m(j) = \frac{1}{|\mathcal{C}_i| - 1} \sum_{j' \in \mathcal{C}_i, j' \neq j} \tilde{\lambda}_m^{jj'} \tag{4.6}$$

et $b_m(i)$ comme le temps de réglage moyen avec le plus proche cluster de j sur la machine m :

$$b_m(j) = \min_{i' \neq i} \frac{1}{|\mathcal{C}_{i'}|} \sum_{j' \in \mathcal{C}_{i'}} \tilde{\lambda}_m^{jj'} \tag{4.7}$$

Le score silhouette de la référence j, que l'on note $s_m(j)$ est ainsi obtenu à partir de (4.6)

et (4.7) :

$$s_m(j) = \frac{b_m(j) - a_m(j)}{\max\{a_m(j), b_m(j)\}} \tag{4.8}$$

Plus le score silhouette est proche de 1, meilleur est le positionnement de la référence dans le cluster.

Pour obtenir plus de détails sur la signification du score silhouette dans une approche de clustering, voir [68] et [25]. Plusieurs autres méthodes peuvent être utilisés pour déterminer le nombre de clusters dont la méthode *elbow* ([100]).

Afin de sélectionner le meilleur nombre de clusters k_m^* pour chaque machine m, on boucle sur les valeurs possibles de k et on applique l'algorithme PAM à chaque itération. Pour un k donné et une machine donnée $m \in \mathcal{M}$, nous considérons le résultat de la procédure et nous calculons la moyenne du score de silhouette sur toutes les références que nous notons $S_m(k)$:

$$S_m(k) = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} s_m(i) \tag{4.9}$$

Plus ce score moyen est élevé, meilleure est la qualité du clustering pour la machine que nous considérons. On considère généralement qu'une silhouette moyenne supérieure à 0.75 signifie que le clustering global est de bonne qualité. Nous sélectionnons le clustering avec la plus grande silhouette moyenne, qui est considéré comme le meilleur possible sur la machine pour l'instance considérée. Dans la partie 4.3, nous montrons que $S_m(k)$ est un bon indicateur *a priori* pour obtenir le meilleur compromis entre le temps de calcul et la qualité de la solution. Pour une description rapide de la procédure, nous présentons le pseudo-code de l'approche de clustering dans l'Algorithme 6. Elle est appliquée comme un pré-traitement avant la définition du modèle MIP, pour transformer une instance du problème d'origine en une nouvelle dont les temps de setups sont modifiés comme indiqué précédemment. Il est important de noter que pour une instance CLSSD-PM, nous exécutons cet algorithme $|\mathcal{M}|$ fois puisque nous considérons que les temps réglages dépendent de la machine. Son temps d'exécution est négligeable (moins de 15 secondes pour les plus grosses instances) par rapport à celui nécessaire au solveur pour obtenir une solution au CLSSD-PM.

4.2.2 Formulation mathématique basée sur le clustering

Pour formuler notre problème en intégrant les modifications issues du clustering, nous définissons des temps de réglage virtuels $v_m^j = \max_{k \in \mathcal{C}_i}\{\lambda_m^{kj}\}$ pour chaque référence $j \in \mathcal{C}_i$, qui dépendent uniquement du point d'arrivée de l'arc correspondant dans le chemin orienté

Algorithme 6 Clustering

```
1: procédure CLUSTERING(maxClusters, 𝒩, m)
2:     actualClustering ← ∅, bestClustering ← ∅, actualScore ← 0, bestScore ← 0,
3:     pour k = 2, ..., maxClusters faire
4:         actualClustering ← Exécuter PAM(k, 𝒩, m)
5:         actualScore ← Calculer AvgSilhouette(actualClustering)
6:         si actualScore > bestScore alors
7:             bestScore ← actualScore
8:             bestClustering ← actualClustering
9:         fin si
10:    fin pour
11:    Retourner bestClustering
12: fin procédure
```

représentant la séquence de production. Pour chaque paire de médoïdes (i, i') et pour chaque machine nous définissons également un nouveau temps de réglage dépendant de la séquence $f_m^{ii'} = \max_{j \in \mathcal{C}_i, j' \in \mathcal{C}_{i'}} \{\lambda_m^{jj'} - v_m^{j'}\}$ encouru lorsqu'une référence de $\mathcal{C}_{i'}$ suit une référence de $\mathcal{C}_i$ dans la séquence de production. À noter que ce temps de réglage exclut le temps de réglage individuel des références définies ci-dessus, puisque ce dernier s'ajoute au temps total s'ils sont inclus dans la séquence de production.

Avec ces nouveaux temps de réglage virtuels, nous considérons désormais que seuls les temps de transition entre deux clusters différents restent dépendants de la séquence En particulier, nous ne prenons plus en compte les sous-séquences de production au sein d'un cluster et remplaçons donc les variables de transition w_{mt}^{ij} par des variables binaires d'affectation plus communes y_{mt}^{j}. Dans notre modèle modifié, la décision de basculement entre deux clusters est représentée par de nouvelles variables binaires $z_{mt}^{ii'}$, égales à 1 si la machine m est configurée pour des articles du cluster i' juste après ceux du cluster i dans la période t et 0 sinon. D'autre part, chaque fois qu'une référence est produite sur la machine m dans une période de temps donnée, nous ne comptons que le temps de réglage spécifique à la référence v_m^i dans le temps actif total de la machine. Il est ainsi possible de modéliser le problème à partir de ces nouvelles données. En partant de la formulation MIP-AGG (3.16)-(3.26) décrite en 3.2, nous pouvons obtenir la formulation pour le problème prenant en compte le clustering. Cette formulation, que nous notons MIP-AGG-C peut être obtenue en remplaçant les contraintes suivantes :

— (3.17) est remplacé par :

$$\tilde{U}_{mt} = \sum_{j \in \mathcal{N}} (\tau_m^j x_{mt}^j + v_m^j y_{mt}^j) + \sum_{(i,i') \in \mathcal{I}^2} f_m^{ii'} z_{mt}^{ii'} \leq C_{mt} + O_{mt} \quad \forall m = 1, \ldots, M, \forall t = 1, \ldots, T \tag{4.10}$$

— les contraintes pour modéliser les réglages dépendants de la séquence peuvent être directement adaptées pour prendre en compte les clusters au lieu des références. En l'occurrence, en introduisant $\mathcal{I}_0$ l'ensemble des clusters auquel on ajoute un cluster fictif d'indice 0, nous obtenons :

$$\sum_{i \in \mathcal{I}} z_{mt}^{0i} \geq \sum_{i \in \mathcal{I}_0} z_{mt}^{ii'} \qquad \forall i' \in \mathcal{I}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{4.11}$$

$$\sum_{i' \in \mathcal{I}_0} z_{mt}^{i'i} \geq \sum_{i' \in \mathcal{I}} z_{mt}^{ii'} \qquad \forall i \in \mathcal{I}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{4.12}$$

$$z_{mt}^{ii'} \in \{0, 1\} \qquad \forall i \in \mathcal{I}_0, \forall i' \in \mathcal{I}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{4.13}$$

$$\sum_{i' \in \mathcal{I}} f_{mt}^{0i'} = \sum_{i \in \mathcal{I}_0} \sum_{i' \in \mathcal{I}} z_{mt}^{ii'} \qquad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{4.14}$$

$$\sum_{i \in \mathcal{I}} f_{mt}^{ii'} = \sum_{i \in \mathcal{I}_0} z_{mt}^{ii'} + \sum_{i \in \mathcal{I}} f_{mt}^{i'i} \quad \forall i' \in \mathcal{I}, \forall m \in \mathcal{M}, \ \forall t \in \mathcal{T} \tag{4.15}$$

$$0 \leq f_{mt}^{ii'} \leq |\mathcal{I}| z_{mt}^{ii'} \qquad \forall i \in \mathcal{I}_0, \forall i' \in \mathcal{I}, \forall m \in \mathcal{M}, \ \forall t \in \mathcal{T} \tag{4.16}$$

Finalement, nous ajoutons une nouvelle contrainte (4.17) pour forcer le réglage d'un cluster pour lancer la production d'une référence appartenant à celui-ci :

$$\sum_{i \in \mathcal{I}_0} z_{mt}^{ii'} \geq y_{mt}^j \quad \forall i' \in \mathcal{I}, \forall j \in \mathcal{C}_{i'}, \forall m = 1, \ldots, M, \forall t = 1, \ldots, T \tag{4.17}$$

De manière équivalente, nous pouvons passer de la formulation MIP-FL à une nouvelle formulation MIP-FL-C en changeant les mêmes contraintes. Il reste à prouver que l'on peut facilement construire une solution réalisable du problème original MIP-AGG (ou. MIP-FL) à partir d'une solution réalisable de MIP-AGG-C (ou MIP-FL-C) sans surcoût. Nous prouvons cette propriété pour MIP-AGG, mais une analyse similaire peut être menée pour obtenir les mêmes résultats dans le cas de MIP-FL. Soit $(\mathbf{x}, \mathbf{z}, \mathbf{y}, \mathbf{O})$ une solution de MIP-AGG-C. Pour tous les $j \in \mathcal{N}$, nous définissons $\mu(j)$ comme le medoïde du cluster de la référence j obtenu après l'application de l'algorithme PAM, c'est-à-dire $j \in \mathcal{C}_{\mu(j)}$. Nous proposons la procédure simple suivante pour créer une solution pour MIP-AGG : Pour

toutes les $m \in \mathcal{M}$, $t \in \mathcal{T}$, nous définissons les variables de transition suivantes :

$$w_{mt}^{0j} = \begin{cases} 1 & \text{si } j = \min\{j' \in \mathcal{N} : z_{mt}^{0\mu(j')} = 1 \text{ et } y_{mt}^{j'} = 1\} \\ 0 & \text{sinon} \end{cases} \tag{4.18}$$

$$w_{mt}^{jj'} = \begin{cases} 1 & \text{si } j' = \min\{j'' \in \mathcal{C}_{\mu(j)}, j'' > j : y_{mt}^{j''} = 1\} \\ & \text{ou } j = \max \mathcal{C}_{\mu(j)} \text{ et } j' = \min\{j'' : z_{mt}^{\mu(j)\mu(j'')} = 1 \text{ et } y_{mt}^{j''} = 1\} \\ 0 & \text{sinon} \end{cases} \tag{4.19}$$

La proposition suivante indique que la transformation ci-dessus donne une solution réalisable à MIP-AGG :

Proposition 4.2.1. Soit $\mathbf{s}_C = (\mathbf{x}, \mathbf{z}, \mathbf{y}, \mathbf{O})$ une solution pour MIP-AGG-C. Alors $\mathbf{s} = (\mathbf{x}, \mathbf{w}, \mathbf{O})$ avec $\mathbf{w} = \left(w_{mt}^{ij}\right)_{i,j \in \mathcal{N}, m \in \mathcal{M}, t \in \mathcal{T}}$ défini à partir des équations (4.18) et (4.19) est une solution réalisable pour MIP-AGG qui est au plus égale au coût de $\mathbf{s}_C$.

Démonstration. Soit $m \in \mathcal{M}$ et $t \in \mathcal{T}$. Le temps total d'activité de la machine m en période t dans une solution $\mathbf{s}$ est :

$$\begin{aligned}
U_{mt} &= \sum_{j \in \mathcal{N}} \left(\tau_m^j x_{mt}^j + \sum_{i \in \mathcal{N}} \lambda_m^{ij} w_{mt}^{ij} \right) \\
&= \sum_{j \in \mathcal{N}} \left(\tau_m^j x_{mt}^j + \sum_{i \in \mathcal{N}} (\lambda_m^{ij} - v_{mt}^j + v_{mt}^j) w_{mt}^{ij} \right) \\
&\leq \sum_{j \in \mathcal{N}} \left(\tau_m^j x_{mt}^j + v_{mt}^j \sum_{i \in \mathcal{N}} w_{mt}^{ij} + \sum_{i \in \mathcal{N}} \max_{i' \in \mathcal{C}_{\mu(i)}} \{\lambda_m^{i'j} - v_{mt}^j\} w_{mt}^{ij} \right) && (4.20) \\
&= \sum_{j \in \mathcal{N}} \left(\tau_m^j x_{mt}^j + v_{mt}^j y_{mt}^j + \sum_{i \notin \mathcal{C}_{\mu(j)}} \max_{i' \in \mathcal{C}_{\mu(i)}} \{\lambda_m^{i'j} - v_{mt}^j\} w_{mt}^{ij} \right) && (4.21) \\
&= \sum_{j \in \mathcal{N}} \left(\tau_m^j x_{mt}^j + v_{mt}^j y_{mt}^j \right) + \sum_{j \in \mathcal{N}} \sum_{i \notin \mathcal{C}_{\mu(j)}} \max_{i' \in \mathcal{C}_{\mu(i)}} \{\lambda_m^{i'j} - v_{mt}^j\} z_{mt}^{\mu(i)\mu(j)} \\
&= \sum_{j \in \mathcal{N}} (\tau_m^j x_{mt}^j + v_m^j y_{mt}^j) + \sum_{(i,i') \in \mathcal{I}^2} f_m^{ii'} z_{mt}^{ii'} \\
&= \tilde{U}_{mt} \leq C_{mt} + O_{mt}
\end{aligned}$$

L'inégalité (4.20) vient du fait que $\max_{i' \in \mathcal{C}_{\mu(j)}} \{\lambda_m^{i'j} - v_{mt}^j\} = 0$, quand (4.21) est déduit de la définition (4.19) des w_{mt}^{ij}. Pour conclure la preuve, nous pouvons remarquer que toutes les variables avec des coefficients non nuls dans la fonction objectif sont les mêmes dans $\mathbf{s}$ et $\mathbf{s}_C$, à l'exception de $U_{mt} \leq \tilde{U}_{mt}$ qui est un coefficient positif. On en déduit que le coût

total induit par $\mathbf{s}$ est au mieux égal au coût total induit par $\mathbf{s}_C$. □

En général, la partition des références au sein des clusters réduit de façon significative le nombre de variables de décision relatives aux séquences de production. Dans la version du CLSSD-PM avec des clusters, le nombre de variables décrivant les séquences de production est quadratique en le nombre de clusters, alors que le nombre de variables est quadratique en le nombre de références dans la version originale.

Notre définition des temps de réglage virtuels garantit que toute séquence de production des références intra-cluster et inter-cluster est inférieure ou égale à $\tilde{U}_{mt}$, comme défini dans (4.10), il est donc toujours possible de construire une solution réalisable avec au plus un temps de production $\tilde{U}_{mt}$ pour tous les $m \in \mathcal{M}$ et $t \in \mathcal{T}$. En particulier, tout algorithme qui reconstruit des séquences de production intra-cluster (aléatoire, plus proche voisin, etc.) à partir d'une solution réalisable de MIP-AGG-C puis les concatène selon la séquence de clusters correspondante aux variables $z^{ii'}_{mt}$ produit une solution réalisable pour MIP-AGG. Dans nos résultats expérimentaux, nous utilisons un algorithme glouton pour reconstruire les séquences de production optimales. Notons enfin qu'en plus de la flexibilité qu'offre cette méthode, cela permet également de trouver une solution de MIP-AGG qui réduit les temps de production par rapport à la solution de MIP-AGG-C. Cette réduction permet de diminuer les coûts de production ou encore de servir des demandes qui sont considérées comme perdues dans MIP-AGG-C.

4.3 Résultats expérimentaux

Nos algorithmes sont implémentés en Java et nous utilisons la version de CPLEX 20.1 pour résoudre les différents MIP. Les expériences sont réalisées à l'aide de 4 CPUs sur un serveur partagé Intel Xeon Gold 6230 composé de 80 CPUS cadencé à 2.10Ghz et 200 Go de Ram. Nous utilisons les abréviations suivantes pour identifier les algorithmes utilisés dans notre test :

- CPLEX : Branch-and-Cut de CPLEX sur MIP-AGG
- CPLEX-C : Branch-and-Cut de CPLEX sur MIP-AGG-C
- RFFO : Relax-and-Fix & Fix-and-Optimize appliqué avec MIP-AGG
- RFFO-C : Relax-and-Fix & Fix-and-Optimize appliqué avec MIP-AGG-C

Pour CPLEX et CPLEX-C, nous définissons une limite de temps de calcul de 900 secondes. Pour chaque méthode testée, nous évaluons la qualité de la solution en fonction de l'écart de entre sa valeur de la fonction objectif par rapport à la borne inférieure LB

obtenue après 4 heures avec CPLEX en utilisant la formulation MIP-FL, connue pour donner une meilleure borne inférieure. Si UB correspond à la meilleure solution réalisable trouvée par la méthode testée, nous définissons l'écart entre la meilleure solution et la borne inférieure de la manière suivante :

$$Gap = 100 \cdot \frac{UB - LB}{LB}$$

4.3.1 Analyse des paramètres de RFFO

Les performances de la procédure RFFO dépendent fortement des valeurs δ, σ, γ et ρ. Il est donc crucial de déterminer la combinaison de paramètres qui offre le meilleur compromis attendu entre le coût des solutions obtenues et le temps de calcul moyen pour chaque classe d'instances. Après une analyse préliminaire empirique, nous avons décidé de nous concentrer sur un nombre limité de paramètres. Nous fixons le nombre de périodes de chevauchement σ à 1 car l'augmentation de sa valeur n'améliore pas significativement les solutions mais détériore sévèrement le temps de calcul pour les instances moyennes et grandes. Même pour les petites instances, nous n'avons pas observé de gain substantiel de performances pour $\sigma > 1$. Nous avons testé différentes combinaisons de longueur de fenêtre de décision $\delta \in \{2, 3\}$, de gap limite (donnée par CPLEX) $\gamma \in \{1, 3, 5, 8, 10\}\%$ et de limite de temps par itération $\rho \in \{30, 60, 90, 120\}$ secondes.

Afin de décider quels critères d'arrêt γ et ρ sont les plus performants, nous avons analysé leur impact sur les solutions. Il faut noter que cette analyse a été faite sur RFFO-C car RFFO n'a parfois pas réussi à trouver une solution réalisable lors de l'étape R&F sur certains instances.

Les résultats pour un sous-ensemble d'instances sont présentés en Annexe. Comme prévu, le gap de la solution finale augmente avec γ, mais cette influence est légère lorsque $\gamma \leq 5\%$ (Figure A.1). D'autre part, l'influence de γ sur le temps de calcul est plus prononcée, par exemple fixer $\gamma = 3\%$ réduit significativement le temps de calcul par rapport à $\gamma = 1\%$. La première valeur offre ainsi le meilleur compromis entre le gap final et le temps de calcul. Une analyse similaire est menée pour différentes valeurs de ρ. On observe sur la figure A.3 que le temps limite n'est jamais dépassé pour les instances avec $N = 50$. Pour d'autres cas, les résultats suggèrent que ρ a un faible impact à la fois sur le temps de calcul total et sur le gap final. Par conséquent, d'après les résultats, $\rho = 60$ semble être le meilleur choix car la limite de temps est rarement le paramètre limitant lorsque γ est ajusté de manière appropriée.

Le tableau A.4 présente les résultats utilisant RFFO-C sur chaque classe d'instances.

Pour chaque paramétrage RFFO présenté, nous laissons $\sigma = 1$ et $\rho = 60$. Nous observons que $\delta = 3$ conduit dans presque la quasi-totalité des cas à un meilleur gap sur les petites et moyennes instances pour toutes les valeurs de γ que nous avons testées.

- Pour $N \leq 40$, $\gamma = 1\%$ semble produire les meilleures solutions en moins de 180 secondes sauf pour le cas $N = 40$, $M = 2$, $T = 30$ pour lequel $\gamma = 3\%$ est plus pertinent si l'on souhaite rester en dessous des 180 secondes. Nous avons donc décidé d'appliquer RFFO-C$(3, 1, 1, 60)$ pour $N \leq 40$ sauf pour le dernier cas où nous appliquons RFFO-C$(3, 1, 3, 60)$.
- Pour les instances plus importantes, le paramètre RFFO-C$(3, 1, 3, 60)$ s'avère être un bon choix pour $N = 50$. En revanche, lorsque $N = \{75, 100\}$ et $T = 30$ le temps d'exécution explose. Pour le cas $N = 75$, utiliser le paramétrage RFFO-C$(2, 1, 5, 60)$ permet d'obtenir une solution en moins de 300 secondes ce qui reste acceptable dans notre cadre applicatif. Dans le cas $N = 100$, RFFO-C$(2, 1, 8, 60)$ permet d'obtenir une solution en moins de 500 secondes mais avec un gap dégradé. Finalement, fixer $\delta = 2$ semble pertinent pour rester sous la barre des 180 secondes pour les instances avec $N = 100$ et $T = 15$.
- Finalement, lorsque $N = 125$, le paramètre $\delta = 2$ semble être un choix plus robuste en général. Pour $T = 15$, RFFO-C$(2, 1, 5, 60)$ permet d'obtenir des solutions acceptables pour cette taille de problème. Dans le cas où $T = 30$, aucun paramétrage ne permet d'obtenir à la fois une solution et un temps qui sont acceptables.

Il est important de noter que le nombre de périodes T impacte fortement la qualité de la solution et le temps d'exécution. En particulier, pour un même nombre de références et de machines, faire varier T demande de modifier le paramètre γ pour rester dans la limite de temps souhaitée. Un récapitulatif des configurations RFFO pour les différents types d'instances est présenté dans le tableau 4.1.

Tableau 4.1 – Configuration pour RFFO et RFFO-C

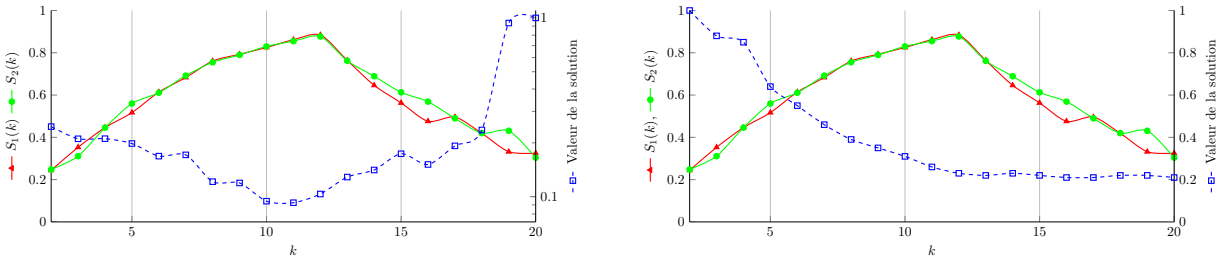
N	20		30		40		50		75		100		125	
T	15	30	15	30	15	30	15	30	15	30	15	30	15	30
γ (%)	1						3		5	3	8		5	8
δ	3								2					

4.3.2 Analyse du clustering

Pour étudier les avantages de notre approche de clustering, nous avons effectué des expérimentations en utilisant directement le solveur CPLEX puis en appliquant la procédure RFFO à la fois sur MIP-AGG et MIP-AGG-C. Les résultats sont présentés dans les tableaux 4.2 et 4.3. Comme nous l'avons vu au chapitre 3, l'utilisation directe d'un solveur commercial sur la formulation MIP s'avère inadéquate, même pour certains cas où le nombre de références est limité à 30. Nous accordons à CPLEX un temps de calcul maximum de 900 secondes, ce qui dépasse déjà les exigences industrielles de 3 à 5 minutes. Alors que le solveur est capable d'obtenir des solutions optimales sur certaines des instances, la qualité générale des solutions obtenus n'est pas satisfaisante en moyenne. D'après nos observations, un grand nombre de solutions acceptables trouvées par CPLEX ont été atteinte à la fin des 900 secondes allouées à la résolution, ce qui augmente la variabilité des solutions et la rend incompatible avec une approche industrielle.

En revanche, le pré-traitement de clustering réduit la taille du problème et permet de trouver des solutions nettement meilleures dans la plupart des scénarios, avec un taux de convergence plus rapide. La perte de qualité de la solution due à l'approximation est compensée par une résolution plus rapide. Les résultats obtenus suggèrent que c'est le cas pour les instances avec $N \geq 30$, pour lesquelles CPLEX-C atteint de meilleures solutions que son homologue dépendant de la séquence. Nous observons que cette tendance devient de plus en plus prononcée à mesure que N augmente. Enfin, il apparaît que la longueur de l'horizon de planification T affecte significativement la qualité de la solution pour les deux approches.

Nous testons également si le score de silhouette introduit dans la partie 4.2 est un critère pertinent pour déterminer le nombre de clusters considérés. C'est-à-dire que nous évaluons les performances de MIP-AGG-C pour différents nombres de clusters k en comparant la qualité des solutions obtenues et le score silhouette moyenne correspondant $S_m(k)$ tel que défini dans (4.9). La conclusion générale est que la valeur de la fonction objectif diminue à mesure que ce nombre atteint sa « meilleure » valeur $k^* = \arg\max\{S_m(k)\}$. Lorsque le nombre de clusters dépasse k^*, les solutions obtenues se détériorent significativement avec MIP-AGG-C. En effet, à mesure que le nombre de familles augmente, nous introduisons plus de variables de transition entre les références dans la formulation MIP-AGG-C, ce qui rend le problème plus proche de celui d'origine mais plus difficile à résoudre. En utilisant RFFO-C, nous n'observons pas ce phénomène mais plutôt un seuil atteint en k^* pour lequel les solutions ne s'améliorent plus (ou de façon peu significative). Dans notre cas, le score silhouette apparaît donc comme un indicateur adapté et facile à



(a) CPLEX-C

(b) RFFO-C

FIGURE 4.3 – Impact du nombre de clusters sur la qualité de la solution et du score silhouette

calculer.

La figure 4.3 illustre cette tendance dans le pire des cas que nous avons observé sur les instances générées pour $M = 2$ et $N = 100$ pour un nombre différent de clusters. Les courbes rouges et vertes montrent le score de silhouette moyen $S_m(k)$ obtenu sur la machine $m \in \{1, 2\}$ pour différents nombres de clusters k. La courbe bleue fournit un indicateur de la qualité de la solution, exprimé comme une fraction de la pire solution obtenue sur l'ensemble des valeurs de k considérées. La qualité de la solution s'améliore nettement à mesure que le nombre de clusters augmente vers k^*.

Dans le cas de la résolution RFFO, on constate que des solutions légèrement meilleures sont trouvées pour un plus grand nombre de clusters, mais l'augmentation du temps de calcul est significative. Au contraire pour une résolution CPLEX simple en 900 secondes, on observe qu'au-delà d'un certain point, la qualité de la solution se dégrade à mesure que l'on considère plus de clusters. Nous concluons que ce critère est un candidat viable pour choisir un bon nombre de clusters a priori afin d'obtenir une bonne solution.

Les expérimentations pour les procédures RFFO sont réalisées avec les valeurs δ, σ, γ et ρ ajustées pour chaque classe d'instances selon l'analyse de la section précédente.

Dans le tableau 4.2, nous observons clairement une réduction de l'écart moyen par rapport à une résolution MIP simple. Sur les petites instances, l'écart obtenu par RFFO combiné à l'approche clustering reste faible par rapport aux autres méthodes. De manière générale, nous observons que la combinaison de ces deux approches réduit considérablement le temps de calcul tout en fournissant des solutions acceptables, voire de bonne qualité, sur toutes les instances.

Bien qu'elle soit plus performante que CPLEX, la procédure RFFO directement appliquée au problème CLSSD-PM conduit généralement à des résultats insuffisants ou

à un dépassement de temps même sur les instances avec $N = 40$. Pour les instances plus importantes, cela provient souvent d'itérations incomplètes pendant la partie RF de l'heuristique, qui ne peuvent pas trouver de solution dans le temps imparti. Une solution pourrait être d'augmenter le temps limite ρ autorisé pour chaque itération mais cela se fait au détriment d'un temps de calcul total plus long, qui dépasse alors fréquemment le seuil acceptable fixé par les planificateurs. En revanche, le temps de calcul et l'écart moyen obtenus restent acceptables lorsque nous combinons la procédure RFFO avec l'approche de clustering. Dans le tableau 4.3, nous comparons les résultats obtenus avec CPLEX et RFFO sur les problèmes modifiés obtenus après le pré-traitement de clustering. Le pré-traitement de clustering définit un nombre de clusters variant entre 10 et 14 sur nos instances. Les résultats sur les instances de taille moyenne, montrés dans le tableau 4.3, indiquent que RFFO-C atteint de meilleurs gaps moyens tout en gardant un temps de calcul inférieur à 3 minutes. Cependant l'écart minimum (Gap_{min}) et l'écart maximum (Gap_{max}) obtenus montrent qu'il existe une grande variation dans les qualités de la solution même au sein d'une même classe d'instances. Quoi qu'il en soit, RFFO-C a les meilleures performances dans le pire des cas parmi les deux méthodes. De plus, nous avons observé que le temps de calcul moyen pour RFFO-C est sensible aux différents paramètres et à la taille du problème. Enfin, pour les instances de taille de problème 125-6-30, nous dépassons l'objectif de temps de calcul, ce qui nécessite des ajustements supplémentaires des méthodes pour obtenir une convergence plus rapide. Néanmoins, au vu de la taille des instances qui sont construites pour tester les limites de nos méthodes et du cadre applicatif, les solutions obtenues permettent d'envisager une application industrielle.

4.4 Conclusion

Nous avons adapté deux heuristiques bien connues au problème CLSSD-PM qui sont le R&F et le F&O. Étant donné que le nombre de variables pour modéliser les séquences de production influence fortement la complexité à résoudre le problème, nous avons profité des caractéristiques des temps de réglage pour développer et appliquer un algorithme de clustering. Lorsque nous combinons cette approche avec le RFFO, nous sommes en mesure d'obtenir des solutions acceptables en un temps de calcul très court par rapport à la complexité du problème. Nos expérimentations nous ont permis de tester plusieurs combinaisons de paramètres et de déterminer quelle configuration est susceptible de fournir les meilleurs résultats pour différentes classes d'instances. La perte due à l'approximation du clustering est récupérée grâce à l'efficacité de la méthode sur les grands cas industriels. De

Tableau 4.2 – Résultats obtenus avec l'ensemble des méthodes sur les petites instances

Taille		Gap Moy. (%)				Temps d'exé. (s)			
N-M-T	$\mathcal{U}$	CPLEX	CPLEX-C	RFFO	RFFO-C	CPLEX	CPLEX-C	RFFO	RFFO-C
20-1-15	0.9	0.14	10.17	3.0	9.96	393.1	235.8	18.32	5.04
20-1-15	1.0	0.08	11.14	2.43	10.44	591.0	312.6	27.04	5.68
20-2-15	0.9	5.17	14.29	3.03	12.98	833.6	648.7	65.36	27.28
20-2-15	1.0	8.56	14.26	3.64	12.68	858.3	848.9	39.44	27.2
20-1-30	0.9	10.68	18.09	6.33	17.35	865.0	815.8	83.6	17.6
20-1-30	1.0	63.5	26.82	7.36	20.93	887.2	841.9	89.92	24.64
20-2-30	0.9	4.96	10.73	7.04	10.71	900.0	900.0	76.56	36.8
20-2-30	1.0	84.65	43.65	10.39	20.16	900.0	900.0	125.12	107.92
30-1-15	0.9	10.09	29.59	5.41	26.56	900.0	506.7	107.12	16.0
30-1-15	1.0	10.89	26.25	5.42	23.6	893.6	500.0	112.88	13.92
30-2-15	0.9	175.38	33.83	8.56	24.51	900.0	900.0	232.72	80.48
30-2-15	1.0	26.8	36.08	8.55	26.19	900.0	900.0	236.56	115.04
30-1-30	0.9	2248.96	30.85	11.1	22.85	900.0	900.0	209.04	40.32
30-1-30	1.0	8479.14	54.97	11.7	25.27	900.0	900.0	254.32	49.52
30-2-30	0.9	3898.95	57.68	13.9	23.14	900.0	900.0	354.48	135.2
30-2-30	1.0	1952.23	79.22	16.28	27.34	900.0	900.0	418.48	203.12
40-1-15	0.9	60.22	27.54	10.92	24.89	900.0	648.4	202.4	14.8
40-1-15	1.0	2795.31	55.48	14.62	43.98	900.0	855.6	310.8	34.24
40-2-15	0.9	508.41	44.81	16.81	30.09	900.0	900.0	347.84	89.92
40-2-15	1.0	1341.18	66.73	17.24	35.03	900.0	900.0	327.6	157.52
40-1-30	0.9	5833.42	69.8	25.52	30.21	900.0	900.0	428.16	55.92
40-1-30	1.0	7256.03	168.89	24.89	44.21	900.0	900.0	403.28	92.48
40-2-30	0.9	8006.58	289.59	41.18	42.0	900.1	900.0	569.04	80.48
40-2-30	1.0	10105.01	175.95	114.04	56.63	900.0	900.0	616.48	125.36

Tableau 4.3 – Résultats obtenus avec CPLEX-C et RFFO-C sur les instances moyennes et grandes instances

Taille		CPLEX-C				RFFO-C			
N-M-T	$\mathcal{U}$	$Gap_{min}(\%)$	$Gap_{max}(\%)$	$Gap_{moy}(\%)$	Temps (s)	$Gap_{min}(\%)$	$Gap_{max}(\%)$	$Gap_{moy}(\%)$	Temps (s)
50-1-15	0.9	26.17	946.28	206.37	857.7	30.82	230.87	81.67	30.56
50-1-15	1.0	26.04	240.05	82.95	877.5	33.27	170.02	61.32	21.12
50-2-15	0.9	20.05	211.97	77.44	900.0	25.56	122.65	49.4	41.92
50-2-15	1.0	33.27	223.38	105.11	900.0	36.28	115.56	57.55	62.24
50-1-30	0.9	34.38	479.95	155.09	900.0	41.11	132.69	51.37	40.8
50-1-30	1.0	44.6	689.2	225.4	900.0	41.31	147.0	62.35	54.8
50-2-30	0.9	28.5	182.71	114.83	900.0	30.35	104.28	56.98	144.56
50-2-30	1.0	83.73	293.32	175.17	900.0	64.92	137.64	73.54	228.32
75-2-15	0.9	76.34	319.13	185.06	900.0	58.53	169.85	86.4	77.76
75-2-15	1.0	84.72	4296.11	618.0	900.0	54.92	183.66	105.81	136.96
75-4-15	0.9	15.9	271.12	77.84	900.1	16.57	93.03	37.14	95.04
75-4-15	1.0	23.66	165.88	90.7	900.0	27.49	82.0	46.99	139.52
75-2-30	0.9	171.58	1198.31	482.43	900.0	102.89	438.68	169.63	282.4
75-2-30	1.0	270.98	5569.56	1224.91	900.1	149.42	468.35	229.67	307.04
75-4-30	0.9	104.4	24481.27	7503.81	900.0	47.08	162.83	79.38	237.12
75-4-30	1.0	128.09	25596.78	5305.99	900.0	46.14	247.27	88.52	329.76
100-2-15	0.9	121.64	689.39	424.53	900.0	145.64	369.01	181.76	132.56
100-2-15	1.0	316.87	1462.93	620.58	900.0	173.87	507.35	221.09	159.68
100-4-15	0.9	24.48	290.67	165.18	900.0	27.25	166.53	82.34	131.92
100-4-15	1.0	80.76	360.84	162.38	900.0	65.1	150.9	82.84	167.36
100-2-30	0.9	2326.57	25151.24	15048.63	900.0	247.37	1310.56	454.04	444.96
100-2-30	1.0	599.61	21443.94	6574.43	900.1	213.21	735.7	320.61	438.24
100-4-30	0.9	5050.56	27579.62	15251.2	900.1	99.36	353.01	166.35	465.12
100-4-30	1.0	287.58	34502.35	12827.76	900.0	98.8	501.7	197.44	464.08
125-4-15	0.9	198.27	577.88	365.83	900.0	120.33	290.47	156.34	139.52
125-4-15	1.0	118.0	635.8	294.24	900.0	89.84	335.78	132.73	173.28
125-6-15	0.9	49.39	9542.56	1194.14	900.0	50.26	162.67	79.96	127.68
125-6-15	1.0	117.61	1060.09	415.51	900.0	81.88	258.82	137.77	206.88
125-4-30	0.9	2865.74	34799.47	21274.28	900.0	108.59	499.73	228.45	550.16
125-4-30	1.0	2696.36	28578.93	15457.53	900.0	147.3	2088.9	397.75	731.36
125-6-30	0.9	10349.25	34920.57	23978.21	900.0	99.84	470.47	155.44	563.84
125-6-30	1.0	4290.39	43026.49	26177.84	900.0	101.2	536.91	215.65	713.68

plus, nous fournissons des preuves que le score de silhouette est un bon indicateur pour définir un nombre approprié de clusters par rapport à la solution finale. Nos résultats confirment que notre procédure fonctionne bien sur des instances industrielles moyennes à grandes, pour lesquelles nous avons pu obtenir des plans de production réalisables en moins de 3 et 5 minutes, respectivement. Les bonnes performances de nos méthodes sont encourageantes, même si elles n'ont été testées que sur des instances dont la structure est propre au cadre d'application des clients de VIF. Nos tests révèlent qu'un mauvais score silhouette se traduit *a priori* par une diminution de la qualité des solutions obtenues, aussi étendre cette approche à des problèmes dont les setups sont plus généraux constitue une perspective de recherche séduisante pour des travaux futurs.

La version simplifiée du problème obtenue après application de l'algorithme PAM en pré-traitement reste assez riche et inclut de nombreuses contraintes industrielles. D'un point de vue académique, évaluer l'impact de chaque type de contrainte comme l'inclusion de stocks de sécurité ou d'heures supplémentaires peut aider à identifier les sous-problèmes qui bénéficient le plus de cette transformation. On peut également envisager de simplifier encore plus le problème en définissant des temps de configuration indépendants de la séquence entre les groupes d'éléments. Cependant, une application directe de la procédure de clustering proposée aux familles de références risque de se révéler trop imprécise car les temps inter-cluster contribuent beaucoup plus que leurs homologues intra-cluster au temps total d'utilisation de la machine. Les temps de réglage virtuels qui sont actuellement définis comme le maximum sur tous les temps de réglage possibles vers un élément donné sont susceptibles d'être trop éloignés de la valeur réelle atteinte. Par conséquent, il convient de rechercher une borne supérieure différente qui conserve plus d'informations sur la séquence de production pour affiner l'approximation.

Chapitre 5

Approches alternatives pour le CLSSD-PM

Dans ce chapitre, nous détaillons des approches alternatives que nous avons explorées durant cette thèse pour résoudre le problème CLSSD-PM. La première correspond à un algorithme génétique qui a été étudié durant un stage de Master que j'ai co-encadré. En effet, à la suite des premières modélisations introduites au chapitre 3 et aux difficultés rencontrées pour les traiter efficacement, nous souhaitions explorer une première résolution dans un temps d'exécution maîtrisé, ce à quoi se prêtent bien les métaheuristiques. La deuxième est une approche itérative en trois phases qui se base sur un décomposition des décisions du problème. Au moment de la soumission de ce document, la qualité des solutions obtenues grâce à ces méthodes alternatives n'est pas encore satisfaisante par rapport à l'approche heuristique introduite au chapitre 4. Néanmoins, nous les présentons dans ce chapitre et proposons des pistes de recherche envisagées pour en améliorer les performances.

5.1 Algorithme génétique

L'approche introduite dans cette section est issue d'un stage co-encadré réalisé par Benoît Le Badezet dont le sujet visait à explorer les métaheuristiques existantes appliquées à des problèmes de lot-sizing dans le but de les adapter au CLSSD-PM. Le travail initialement réalisé comprenait un état de l'art de ces méthodes ainsi qu'un algorithme génétique implémenté en Java dont les résultats ont été analysés sur un sous-ensemble des instances générées. Cette partie ne présente que les principales idées de l'algorithme, tandis que nous exposons les résultats obtenus dans la partie 5.3 afin de comparer les performances de cette procédure avec les autres heuristiques développées spécifiquement pour le problème.

L'algorithme génétique que nous proposons se base sur la génération successive d'une population d'individus, en utilisant des opérateurs spécifiques de croisement et de mu-

tation inspirés de la nature. Chaque individu correspond à une solution du problème. Entre chaque génération, nous gardons une partie des meilleures solutions (i.e individus) dans la population pour conserver l'information qu'ils contiennent. Pour éviter que la méthode ne se retrouve bloquée dans un optimum local, l'algorithme réinitialise un sous-ensemble aléatoire des individus dans la population courante, lorsqu'aucune amélioration de la meilleure solution connue n'a été constatée. Les phases de croisement et de mutation sont des étapes déterminantes dans le bon fonctionnement d'un algorithme génétique. Pour les exécuter au mieux, les individus de la population sont représentés sous forme de chromosomes, modélisés comme suit dans notre cas :

— X_{mt} un ensemble de couples ⟨référence ; quantité⟩ pour chaque machine m et période t. Ce chromosome indique pour chaque référence, la quantité produite sur les différentes machines et périodes.

— W_{mt} correspond à l'ordre des références dans la séquence de production de la machine m en période t.

Les tableaux 5.1 et 5.2 montrent un exemple de chromosome basé sur ces représentations pour une machine et une période.

Tableau 5.1 – Exemple d'un chromosome basé sur les quantités produites

Référence	1	4	3
Quantité	15	13	9

Tableau 5.2 – Exemple d'un chromosome basé sur la séquence de production

séquence	1	4	3	7	2	6	5

Dans la littérature, plusieurs méthodes de croisement qui se basent sur les périodes, références ou séquences existent ([59, 101]). En pratique, coupler les différentes méthodes peut permettre d'augmenter l'espace de recherche et d'améliorer la convergence de la méthode. La mutation que nous considérons se base sur un échange de la position de deux références choisis aléatoirement dans une séquence de production.

L'algorithme 7 présente le pseudo-code de l'algorithme global dans lequel nous voyons les différents étapes (génération, croisements, mutations) de l'algorithme. La phase d'initialisation consiste à générer une population de façon aléatoire et garder le meilleur individu dans une variable notée s^*. Par la suite, la procédure met à jour la population courante à chaque génération en se basant sur les meilleurs individus de la population précédente. Cette étape est inutile lors de la première génération mais permet dans la suite de

garder des individus d'une génération à la suivante. À partir de cet ensemble d'individus, l'algorithme va dans l'ordre proposer des croisements entre plusieurs individus, que nous appelons parents, ainsi que des mutations sur des individus tirés aléatoirement. À la fin de chaque itération, la meilleure solution trouvée s^* est mise à jour si le meilleur individu de la nouvelle population représente une solution de meilleure qualité que celle stockée dans s^*. Lorsque l'un des critères d'arrêt est atteint (temps limite d'exécution ou nombre limite de générations), l'algorithme s'arrête et renvoie le meilleur individu sur l'ensemble des populations. Pour plus de détails sur les différentes étapes (croisement, mutation, ...) et le paramétrage de l'algorithme, le lecteur peut se référer à [71]. Dans la partie 5.3, nous présentons des résultats en utilisant cette approche.

Algorithme 7 Algorithme Génétique

```
1: genActu ← GenerationPopulation()
2: s* ← MeilleurIndividu
3: tant que stoppingCriteria = false faire
4:     nextGen ← mettreAJour(GenActu)
5:     tant que |nextGen| < maxPopSize faire
6:         parent1, parent2 ← selectionParents(genActu)
7:         nextGen.ajoute(crossover(parent1, parent2))
8:         muté ← selectionMutés(genActu)
9:         nextGen.ajoute(mutation(muté))
10:    fin tant que
11:    si cost(nextGen.MeilleurIndividu) < cost(s*) alors
12:        s* ← nextGen.MeilleurIndividu
13:    fin si
14:    genActu ← nextGen
15: fin tant que
16: retourne s*
```

5.2 Approche itérative en trois phase

Dans cette partie, nous nous intéressons à une classe de méthodes de résolution en plusieurs phases qui permettent de ne pas traiter un problème complexe de façon globale, mais plutôt de le diviser en sous-problèmes qui ne contiennent qu'un sous-ensemble de décisions et dans lesquels certains paramètres sont approximés. Pour des problèmes combinant la planification de production et l'ordonnancement, plusieurs auteurs ont proposé des

approches de ce type. En général, on retrouve principalement des décompositions en deux phases dans lesquelles les deux niveaux de décision sont séparés. Par exemple, Dauzère-Péres et Lasserre [35] ont proposé une heuristique intégrant une phase de lot-sizing et une phase d'ordonnancement pour le *Capacitated lot-sizing and Scheduling Problem*. Plusieurs autres auteurs ont adopté des approches similaires ([77, 80] où la phase d'ordonnancement (ou de séquencement) et la phase de lot-sizing sont séparées. Récemment, Alves *et al.* [13] ont proposé plusieurs approches de résolution itératives pour résoudre un problème similaire associant le lot-sizing et l'ordonnancement. Ils les ont comparées à des méthodes plus classiques où aucune information ne transite entre les phases et ont montré que le partage d'information améliore significativement leur efficacité.

Dans le même ordre d'idées, nous proposons une méthode itérative en trois phases pour résoudre le problème CLSSD-PM. L'idée centrale de cette approche est de décomposer le problème d'origine en trois sous-problèmes de sorte que l'information contenue dans le résultat obtenu à l'issue d'une phase est utilisé comme paramètre d'entrée des suivantes. La procédure met ensuite à jour une solution courante en exécutant successivement ces trois phases jusqu'à converger ou atteindre un critère d'arrêt. Plus précisément, notre approche vise à résoudre le CLSSD-PM de façon itérative en décomposant le problème en trois phases :

— la première est une phase de lot-sizing dans laquelle les temps de réglage sont approchés de façon à ne pas dépendre de la séquence de production. Elle permet de pré-affecter la production des différentes références à des paires machine-période.

— la deuxième correspond à une phase d'ordonnancement dans laquelle nous construisons des séquences de production de façon à minimiser le temps requis par les réglages.

— la troisième phase calcule l'unique les quantités de chaque référence à produire pour chaque paire machine-période.

Ainsi, la deuxième phase utilise l'information d'affectation transmise par la première, tandis que la troisième calcule les quantités à produire en tenant compte des temps de réglages nécessaires pur les séquences obtenues durant la deuxième phase. La procédure exécute séquentiellement ces différentes phases jusqu'à atteindre un critère d'arrêt (soit la limite de temps, soit un nombre donné d'itérations sans amélioration). Dans la suite nous détaillons formellement chacune des trois phases de notre approche.

5.2.1 Première Phase : Affectation des références aux machines et périodes

Nous décrivons la première phase avec une formulation mathématique basée sur le Facility Location. Contrairement à la version originale du problème, nous introduisons une nouvelle variable binaire y^i_{mt} égale à 1 si la référence i est affecté à la machine m en période t et nous retirons l'ensemble des variables w^{ij}_{mt} qui permettent de modéliser l'affectation des références au sein des séquences de production.

L'objectif est d'optimiser les coûts induits par la gestion de la production et des stocks, en considérant des temps de réglage fictifs indépendants de la séquence pour chaque référence. Ainsi, nous introduisons ST^i_{mt} le temps de réglage de la référence i pour la machine m à la période t. À la première itération, nous initialisons ces temps à 0. Au cours de son exécution, l'algorithme conserve l'information des décisions prises lors de l'itération précédente. Ainsi, les paramètres $\tilde{y}^i_{mt}$ et $\tilde{x}^{it}_{ms}$ sont calculés à partir des valeurs prises par les variables correspondantes durant la phase précédente. Elles sont éventuellement modifiées par l'algorithme au fur et à mesure de l'exécution des phases, mais ne peuvent être affectées par le solveur lorsqu'elles sont intégrées à un programme mathématique. Avant la première itération, tous ces paramètres sont initialisés à 0.

Les restrictions de capacité sont également assouplies sur la base des informations fournies par les autres phases. Nous introduisons pour cela le paramètre Cap^{res}_{ms}, initialisé à $\overline{C}_{ms}$ pour chaque machine m et période s, qui correspond à la capacité restante après la production et les temps de réglages décidés lors de l'itération précédente.

La formulation du problème traité dans cette phase correspond au MIP suivant :

$$\min \sum_{s\in\mathcal{T}}\sum_{m\in\mathcal{M}}\left(c_{ms}U_{ms} + \overline{c}_{ms}O_{ms} + \sum_{i\in\mathcal{N}}\sum_{t=s}^{\delta^i_s} H^{it}_{ms}x^{it}_{ms}\right) \tag{5.1}$$

$$\text{s.c} : U_{ms} = \sum_{i\in\mathcal{N}}(\tau^i_m \sum_{t=s}^{\delta^i_s} d^i_t x^{it}_{ms} + ST^i_{ms}y^i_{ms}) \qquad \forall m\in\mathcal{M}, \forall s\in\mathcal{N} \tag{5.2}$$

$$U_{ms} \leq C_{ms} + O_{ms} \qquad \forall m\in\mathcal{M}, \forall s\in\mathcal{T} \tag{5.3}$$

$$O_{ms} \leq \overline{C}_{ms} - C_{ms} \qquad \forall m\in\mathcal{M}, \forall s\in\mathcal{T} \tag{5.4}$$

$$\sum_{i\in\mathcal{N}}\left(\tau^i_m(\sum_{t=s}^{T} d^i_t(x^{it}_{ms} - \widetilde{x}^{it}_{ms})) + (y^i_{ms} - \widetilde{y}^i_{ms})ST^i_{ms}\right) \leq Cap^{res}_{ms} \; \forall m\in\mathcal{M}, \forall s\in\mathcal{T} \tag{5.5}$$

$$x^{it}_{ms} \leq y^i_{ms} \qquad \forall i\in\mathcal{N}, \forall m\in\mathcal{M}, \forall s,t\in\mathcal{T} \tag{5.6}$$

$$\sum_{m\in\mathcal{M}}\sum_{s=1}^{t} x^{it}_{ms} \leq 1 \qquad \forall i\in\mathcal{N}, \forall t\in\mathcal{T} \tag{5.7}$$

$$\sum_{t=s}^{\delta_s^i} d_t^i x_{ms}^{it} \geq q_{\min}^i y_{ms}^i \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall s \in \mathcal{T} \quad (5.8)$$

$$U_{ms}, O_{ms} \geq 0 \qquad \forall m \in \mathcal{M}, \forall s \in \mathcal{T} \quad (5.9)$$

$$y_{ms}^i \in \{0,1\} \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall s \in \mathcal{T} \quad (5.10)$$

$$x_{ms}^{it} \in [0,1] \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall s,t \in \mathcal{T} \quad (5.11)$$

Cette formulation désagrégée est très similaire à celle du problème central. Nous ne présentons donc pas chaque contrainte. Le principal changement vient de la gestion de l'affectation des références dans la modélisation qui ne dépend plus de la séquence de production, ce qui permet de diminuer le nombre de variables et contraintes et ainsi réduire le temps de traitement. Nous faisons une approximation de l'impact des séquences sur le temps de production par les nouveaux temps fictifs ST_{ms}^i pour chaque référence i. À l'issue de cette phase, nous obtenons une affectation des références aux machines grâce aux variables y_{ms}^i et nous redéfinissons $\tilde{y}_{ms}^i = y_{ms}^i$ pour tout $i \in \mathcal{N}, m \in \mathcal{M}$ et $s \in \mathcal{T}$.

5.2.2 Deuxième phase : Ordonnancer les références

Cette deuxième phase a pour but d'ordonnancer les références affectées (lors de la première phase) en prenant en compte les temps de réglage dépendants de la séquence. Cette étape nécessite donc la résolution de $M \times T$ problèmes distincts. Il est possible de modéliser chacun des problèmes par une formulation mathématique simple adaptée du Asymmetric Traveling Salesman Problem (ATSP). Pour la machine m en période t, nous notons $\mathcal{N}_{mt} = \{i \in \mathcal{N} : \tilde{y}_{mt}^i = 1\}$. Le problème d'ordonnancement de la machine m en période t peut être modélisé comme suit :

$$\min \sum_{i,j \in \mathcal{N}_{mt}} \lambda_m^{ij} w^{ij} \quad (5.12)$$

$$\text{s.c} \sum_{j \in \mathcal{N}_{mt}} w^{ij} = 1 \quad \forall i \in \mathcal{N}_{mt} \quad (5.13)$$

$$\sum_{j \in \mathcal{N}_{mt}} w^{ji} = 1 \quad \forall i \in \mathcal{N}_{mt} \quad (5.14)$$

$$u^i - u^j + (N_{mt}+1) w^{ij} \leq N_{mt} \quad \forall i,j \in \mathcal{N}_{mt}, i \neq j \quad (5.15)$$

$$u^0 = 0 \quad (5.16)$$

$$w^{ij} \in \{0,1\} \quad \forall i,j \in \mathcal{N}_{mt} \quad (5.17)$$

$$1 \leq u^i \leq N_{mt} \quad \forall i \in \mathcal{N}_{mt} \quad (5.18)$$

, avec la variable w^{ij} indiquant si la référence i précède directement la référence j et u^i une variable donnant l'ordre de la référence i dans la séquence. Cette phase peut être résolue avec un solveur pour un nombre de références réduit ou grâce à des heuristiques venant directement du ATSP. À la fin de cette phase, nous obtenons le temps total nécessaire pour affecter les références aux machines, que l'on note Λ_{mt} pour la machine m en période t. Dans le cas où $\Lambda_{mt} > \overline{C}_{mt}$, l'algorithme relance cette phase en enlevant la référence i dont le temps de réglage impacte le plus la séquence trouvée. Cette évaluation du temps total nécessaire pour effectuer les réglages est ici obtenue de façon exacte en fonction des décisions précédentes. On a ainsi une évaluation directe du temps disponible pour la production, qui sera réparti lors de la phase suivante entre les références sélectionnées.

5.2.3 Troisième phase : Déterminer la taille des lots de production

La dernière phase a pour but de déterminer la quantité à produire de chaque référence affectée au sein de chaque période en considérant le temps restant après les temps de réglage. Il est important de noter que les décisions d'affectation sont fixées en amont de cette phase et les variables associées sont donc remplacées par $\tilde{y}^i_{mt}$. Ainsi dans cette dernière phase, le problème à résoudre correspond à un programme linéaire.

$$\min \sum_{s\in\mathcal{T}}\sum_{m\in\mathcal{M}}\left(c_{mt}U_{mt} + \overline{c}_{mt}O_{mt} + \sum_{i\in\mathcal{N}}\sum_{t=s}^{\delta^i_s} H^{it}_{ms}x^{it}_{ms}\right) \tag{5.19}$$

$$\text{s.c} : U_{ms} = \sum_{i\in\mathcal{N}}(\tau^i_m \sum_{t=s}^{\delta^i_s} d^i_t x^{it}_{ms}) + \Lambda_{ms} \qquad \forall m \in \mathcal{M}, \forall s \in \mathcal{N} \tag{5.20}$$

$$U_{mt} \leq C_{mt} + O_{mt} \qquad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{5.21}$$

$$O_{mt} \leq \overline{C}_{mt} - C_{mt} \qquad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{5.22}$$

$$x^{it}_{ms} \leq \tilde{y}^i_{ms} \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall s,t \in \mathcal{T} \tag{5.23}$$

$$\sum_{m\in\mathcal{M}}\sum_{s=1}^{t} x^{it}_{ms} \leq 1 \qquad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{5.24}$$

$$\sum_{t=s}^{\delta^i_s} d^i_t x^{it}_{ms} \geq q^i_{\min}\tilde{y}^i_{ms} \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall s \in \mathcal{T} \tag{5.25}$$

$$U_{mt}, O_{mt} \geq 0 \qquad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{5.26}$$

$$x^{it}_{ms} \in [0,1] \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall s,t \in \mathcal{T} \tag{5.27}$$

Soit $\mathbf{x}^*$ la solution obtenue. L'algorithme met à jour les variables $\tilde{x}^{it}_{ms} = x^{it*}_{ms}$ pour tout $i \in \mathcal{N}, m \in \mathcal{M}$ et $s, t \in \mathcal{T}$ qui sont ensuite passées à l'itération suivante, avec les variables Λ_{ms} et $\tilde{y}^i_{ms}$ obtenues lors des phases précédentes.

5.2.4 Mise à jour des paramètres

Avant le passage à l'itération suivante, la procédure procède à la mise à jour des paramètres utilisés par la première phase. Pour tout $m \in \mathcal{M}$ et $s \in \mathcal{T}$, la nouvelle capacité résiduelle est calculée grâce à l'équation suivante :

$$Cap^{res}_{ms} = \overline{C}_{ms} - \sum_{i \in \mathcal{N}} (\tau^i_m \sum_{t=s}^{\delta^i_s} d^i_t \tilde{x}^{it}_{ms}) - \Lambda_{ms} \tag{5.28}$$

Dans un deuxième temps, nous mettons à jour les temps de réglage fictifs ST^i_{mt} en prenant en compte les séquences construites lors de la deuxième phase. Notre approche se base sur celle présenté dans Absi *et al.* [2] pour le Production Routing Problem. Deux cas sont à considérer :

— pour $m \in \mathcal{M}$, $s \in \mathcal{T}$ et $i \in \mathcal{N}$ tel que $\tilde{y}^i_{ms} = 1$, ST^i_{ms} correspond au temps marginal sur la séquence de production : soit j le prédécesseur de i et k son successeur sur la séquence de production de la machine m en période t : $ST^i_{ms} = \lambda^{ji}_m + \lambda^{ik}_m - \lambda^{jk}_m$

— pour $m \in \mathcal{M}$, $s \in \mathcal{T}$ et $i \in \mathcal{N}$ tel que $\tilde{y}^i_{ms} = 0$, ST^i_{ms} correspond au temps d'insertion minimum dans la séquence de production actuelle : $ST^i_{ms} = \min_{j,k \in E}\{\lambda^{ji}_m + \lambda^{ik}_m - \lambda^{jk}_m\}$

Ainsi, à partir de cette procédure, nous obtenons des temps de réglage fictifs qui représentent l'impact de l'ajout une références à la séquence actuelle (figure 5.1). À partir

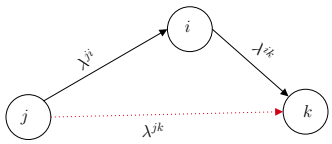


FIGURE 5.1 – Mise à jour des temps de réglage virtuels

de la mise à jour des contraintes et temps de réglage fictifs, nous pouvons réitérer les trois phases de notre approche. La figure 5.2 représente notre approche en trois phases sous forme de diagramme. Pour plus de détails, l'algorithme 8 présenté en l'annexe A.4 présente les différentes étapes de la méthode.

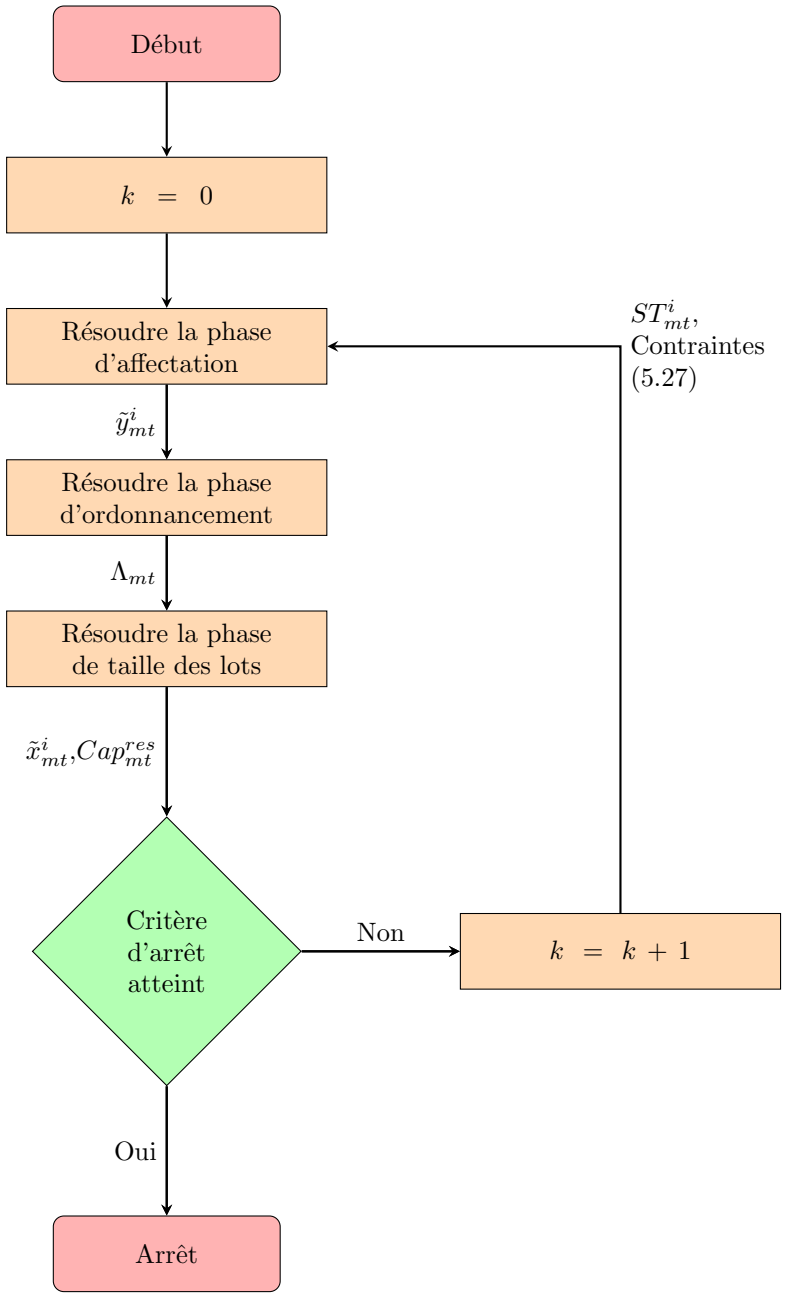


FIGURE 5.2 – Diagramme de l'approche en trois phases

5.3 Résultats préliminaires des approches

Dans cette partie, nous présentons des résultats préliminaires de nos méthodes. Nos algorithmes sont implémentés en Java et nous utilisons la version de CPLEX 20.1 pour résoudre les différents MIP dans l'approche itérative en trois phases. Les résultats décrits dans les tableaux 5.3 et 5.4 sont obtenus en lançant nos algorithmes sur un serveur partagé Intel Xeon Gold 6230 composé de 80 CPUS cadencé à 2.10Ghz et 200 Go de Ram. Pour l'algorithme génétique, un seul CPU est utilisé. Pour l'approche en trois phase, 4 CPUS sont utilisés pour le Branch-and-Cut de CPLEX. Pour comparer les deux approches, nous utilisons les instances proposées au chapitre 3 pour lesquelles nous avons déjà calculé des bornes inférieures (LB) en 4 heures. Ainsi nous reprenons le Gap définie au chapitre précédent :

$$\text{Gap} = 100 \cdot \frac{UB - LB}{LB} \tag{5.29}$$

, avec UB le coût de la solution obtenue avec nos deux méthodes.

Pour l'algorithme génétique, nous bornons le nombre d'itérations à 30000. Nous voyons à partir du tableau 5.4, que cette méthode ne permet pas d'obtenir des gaps acceptables. Même si pour les instances avec $N = 50$, nous obtenons un gap moyen meilleur que celui en utilisant CPLEX, le gap reste toujours supérieur à 400%. En revanche, l'approche en trois phases permet d'obtenir des solutions de meilleure qualité. Dans le tableau 5.3, nous constatons que les gaps obtenus sont bien plus proches de ceux obtenus avec l'approche RFFO. Les temps d'exécution sont aussi réduits et ne dépassent pas les 3 minutes en moyenne. L'impact du nombre de références ou de périodes sur la qualité des solutions n'est pas visible sur l'ensemble des instances présentées dans le tableau 5.3. Le temps d'exécution est par contre impacté par le nombre de périodes et de machines. En effet, la phase 2 de notre approche demande de résoudre $M \times T$ MIP ce qui impacte fortement le temps total de résolution.

5.4 Perspectives

Dans ce chapitre, nous avons proposé deux approches de résolution. La première correspond à un algorithme génétique. Nous avons vu que cette approche ne permet pas d'obtenir de bonnes solutions bien que sur les instances de grandes tailles, les résultats obtenus soient meilleurs que ceux obtenus par CPLEX. Une première direction de recherche serait d'améliorer la population initiale, en particulier en calculant des séquences de production prédéfinies. Une autre approche pourrait être de proposer une hybridation

Tableau 5.3 – Gaps moyens obtenus par l'approche en trois phase

Taille N-M-T	$\mathcal{U}$	Gap (%)	#Iteration	Temps d'exé. (s)
20-1-15	0.9	22.19	30.0	17.2
20-1-15	1.0	23.58	30.0	14.5
20-2-15	0.9	68.33	30.0	22.3
20-2-15	1.0	64.38	30.0	20.9
20-1-30	0.9	43.6	30.0	66.9
20-1-30	1.0	205.3	30.0	37.1
20-2-30	0.9	46.57	30.0	79.8
20-2-30	1.0	171.69	30.0	47.9
30-1-15	0.9	66.1	30.0	23.4
30-1-15	1.0	26.72	30.0	20.9
30-2-15	0.9	108.6	30.0	28.8
30-2-15	1.0	95.05	30.0	29.2
30-1-30	0.9	53.19	30.0	74.4
30-1-30	1.0	96.01	30.0	63.9
30-2-30	0.9	93.71	30.0	93.1
30-2-30	1.0	111.88	30.0	80.0
40-1-15	0.9	15.22	30.0	27.5
40-1-15	1.0	52.5	30.0	25.7
40-2-15	0.9	70.43	30.0	40.5
40-2-15	1.0	81.57	30.0	39.1
40-1-30	0.9	40.42	30.0	98.2
40-1-30	1.0	63.81	30.0	76.3
40-2-30	0.9	61.32	30.0	119.4
40-2-30	1.0	168.41	30.0	103.6
50-1-15	0.9	123.47	30.0	35.2
50-1-15	1.0	33.58	30.0	30.5
50-2-15	0.9	60.23	30.0	54.1
50-2-15	1.0	85.13	30.0	44.6
50-1-30	0.9	29.81	30.0	120.0
50-1-30	1.0	56.62	30.0	164.3
50-2-30	0.9	122.43	30.0	139.9
50-2-30	1.0	90.99	30.0	125.6

Tableau 5.4 – Gaps moyens obtenus par l'algorithme génétique

Taille N-M-T	$\mathcal{U}$	Gap (%)	#Iteration	Temps d'exé. (s)
20-1-15	0.9	438.32	30000.0	99.7
20-1-15	1.0	515.71	30000.0	87.4
20-2-15	0.9	452.12	30000.0	143.0
20-2-15	1.0	487.06	30000.0	132.0
20-1-30	0.9	686.84	30000.0	177.1
20-1-30	1.0	712.16	30000.0	162.5
20-2-30	0.9	862.67	30000.0	257.4
20-2-30	1.0	636.12	30000.0	252.6
30-1-15	0.9	788.57	30000.0	128.0
30-1-15	1.0	764.28	30000.0	130.4
30-2-15	0.9	970.82	30000.0	182.8
30-2-15	1.0	799.93	30000.0	160.1
30-1-30	0.9	991.97	30000.0	221.2
30-1-30	1.0	1022.68	30000.0	222.7
30-2-30	0.9	1054.79	30000.0	315.5
30-2-30	1.0	880.8	30000.0	287.0
40-1-15	0.9	1124.73	30000.0	165.1
40-1-15	1.0	914.54	30000.0	170.6
40-2-15	0.9	1193.8	30000.0	211.2
40-2-15	1.0	1248.79	30000.0	199.9
40-1-30	0.9	1481.57	30000.0	255.0
40-1-30	1.0	1558.81	30000.0	280.3
40-2-30	0.9	1468.33	30000.0	362.3
40-2-30	1.0	1498.64	30000.0	350.3
50-1-15	0.9	1245.6	30000.0	201.7
50-1-15	1.0	1530.51	30000.0	199.4
50-2-15	0.9	1821.22	30000.0	245.7
50-2-15	1.0	1582.91	30000.0	227.5
50-1-30	0.9	1807.8	30000.0	307.3
50-1-30	1.0	1662.28	30000.0	332.4
50-2-30	0.9	2060.57	30000.0	433.3
50-2-30	1.0	1628.32	30000.0	388.4

avec une heuristique de recherche locale pour générer de nouvelles solutions. Enfin, même si les solutions obtenues sont de mauvaise qualité, les utiliser comme *warm start* dans une approche de résolution plus classique peut être envisagé.

Notre deuxième approche, plus pertinente, correspond à une méthode en trois phases. Les solutions obtenues pour le moment laissent envisager une utilisation dans un contexte industriel. Néanmoins, plusieurs pistes d'amélioration sont à considérer. La première est d'utiliser des heuristiques venant du ATSP pour la deuxième phase. Pour le moment, nous avons résolu chaque sous-problème avec un solveur de programmation entière. Or, il peut être plus intéressant d'appliquer des heuristiques dans cette phase pour réduire le temps d'exécution et pour permettre d'appliquer notre approche pour des instances de plus grandes tailles. La deuxième piste est de modifier la procédure de mise à jour des temps de réglage fictifs pour mieux prendre en compte leurs impacts sur la capacité dans la première phase. L'approche en regardant le temps marginal de chaque référence ne permet pas de considérer plusieurs changements dans les séquences de façon optimale. Utiliser notre approche de clustering dans l'évaluation des temps de réglage peut être plus intéressant pour regarder l'impact de retirer un sous ensemble de références d'une même famille de la séquence. La troisième piste envisagée est d'adapter des techniques de *LP-rounding* à notre problème. Ces techniques se basent sur une relaxation de certains variables binaires pour rendre le problème plus simple à résoudre. Les variables sont arrondies à l'entier le plus proche sous certaines conditions. Dans notre cas, l'idée peut être de définir un paramètre compris entre 0 et 1 pour lequel les variables binaires relâchées sont arrondi à 1 si elles sont supérieures à ce dernier. Plusieurs auteurs ont déjà introduit des approches basées sur du *LP-rounding* dans différents problèmes ([81, 72]). Au vu des pistes d'amélioration que nous avons identifié, nous avons décidé de proposer un stage de fin d'étude pour poursuivre ces travaux effectuées en fin de thèse qui n'ont pas pu être finalisés.

Chapitre 6

Application industrielle

Dans ce chapitre, nous présentons certaines extensions du CLSSD-PM, moins communes, que l'on retrouve dans les problématiques rencontrées par certains clients de VIF. Nous proposons dans un premier temps une modélisation du problème de planification pour les industriels produisant des produits secs que nous avons introduit dans la partie 1.3. À partir d'une instance réelle caractéristique, nous présentons les résultats obtenus que nous analysons grâce à des outils de visualisation développés. Finalement, nous présentons des modélisations mathématiques pour les différents points marqués au chapitre 1 par le symbole $\star$, adaptées à la formulation MIP-AGG et à la formulation MIP-FL.

6.1 Planification de la production de produits secs

Le problème de planification rencontré par les industriels qui fabriquent des produits secs diffère du CLSSD-PM par plusieurs points. D'une part, dans ce problème, nous ne considérons pas de temps de réglage, ce qui simplifie la problématique. Comme expliqué au chapitre 1, ces temps sont intégrés de façon implicite dans la cadence, dont la valeur est réduite par rapport à la cadence de production réelle. Le temps de production par unité de chaque référence est donc allongé. Nous notons ce nouveau temps de production modifié $\bar{\tau}_m^i$ pour la référence i sur la machine m. Ce temps moyen est relativement stable car les personnes en charge de la planification font en même temps le choix de regrouper sur chaque période de production des articles ayant des caractéristiques communes, entre lesquels on a donc des temps de changements faibles. Sans cette restriction, la variabilité des temps de réglage serait trop importante pour que le temps moyen soit estimé de façon correcte. Ainsi pour lancer une production, il est nécessaire d'allouer un créneau durant lequel seules les références appartenant à une même famille peuvent être produites. Les créneaux ont des durées fixes prédéfinies généralement associées aux temps de travail des équipes, par exemple des durées de 4 heures ou 8 heures. Dans le cas où le temps nécessaire pour la production des lots retenus est inférieure à la durée du créneau, le temps restant est perdu. Cette pratique des planificateurs ressemble à une approche de clustering simplifiée

dans le sens où elle force le regroupement de références similaires en famille sur une durée donnée. Pour chaque famille $k \in \mathcal{F}$, nous définissons Γ_m^k la durée fixe de la famille k sur la machine m. Nous introduisons la notation $\mathcal{N}_k$ indiquant l'ensemble des références qui appartiennent à la famille k. Chaque référence ne peut appartenir qu'à une unique famille. L'objectif reste le même que pour le CLSSD-PM, à savoir de minimiser la somme des coûts de production de la solution retenue. Pour un récapitulatif des différents coûts, des notations et des variables introduites pour modéliser le CLSSD-PM, le lecteur peut se référer à la Partie 3.1. Pour modéliser ce problème avec une formulation agrégée, nous introduisons les variables suivantes :

— f_{mt}^k une variable entière indiquant le nombre de créneaux en période t sur la machine m affectés pour produire des références de la famille k.

— y_{mt}^i une variable binaire indiquant si la référence i est produite sur la machine m en période t

$$\min \sum_{t\in\mathcal{T}} \sum_{m\in\mathcal{M}} c_{mt}U_{mt} + \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{N}} (h_t^{i+}I_t^{i+} + h_t^{i-}I_t^{i-} + l_t^i L_t^i + \sum_{m\in\mathcal{M}} p_{mt}^i x_{mt}^i) \tag{6.1}$$

$$\text{s.c} : U_{mt} = \sum_{k\in\mathcal{F}} \Gamma_m^k f_{mt}^k \qquad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{6.2}$$

$$U_{mt} \leq C_{mt} \qquad \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{6.3}$$

$$\sum_{i\in\mathcal{N}_k} \bar{\tau}_m^i x_{mt}^i \leq \Gamma_m^k f_{mt}^k \qquad \forall k \in \mathcal{F}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{6.4}$$

$$I_t^i = S_t^i + I_t^{i+} - I_t^{i-} \qquad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{6.5}$$

$$I_t^i = I_{t-1}^i + \sum_{m\in\mathcal{M}} x_{mt}^i + L_t^i - d_t^i \qquad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{6.6}$$

$$x_{mt}^i \leq \min\left\{\frac{C_{mt}}{\bar{\tau}_m^i}, D_{t,\delta_t^i}^i\right\} y_{mt}^i \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{6.7}$$

$$x_{mt}^i \geq q_{\min}^i y_{mt}^i \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{6.8}$$

$$L_t^i \leq d_t^i \qquad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{6.9}$$

$$I_t^{i-} \leq S_t^i \qquad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{6.10}$$

$$I_t^i \leq D_{t+1,\delta_t^i}^i \qquad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{6.11}$$

$$L_t^i, I_t^i, I_t^{i+}, I_t^{i-} \geq 0 \qquad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{6.12}$$

$$y_{mt}^i \in \{0,1\} \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{6.13}$$

$$f_{mt}^j \in \mathbb{N} \qquad \forall j \in \mathcal{F}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{6.14}$$

$$x_{mt}^i \in \mathbb{N} \qquad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall t \in \mathcal{T} \tag{6.15}$$

Les contraintes (6.2) indiquent que le temps d'utilisation des lignes est égal à la somme des créneaux horaires utilisés pour produire les références. Par ailleurs, le temps d'utilisation de chaque ligne est borné par la capacité comme exprimé par les contraintes (6.3). Les contraintes (6.4) assurent que le temps de production des références appartenant à une même famille ne dépasse pas le temps total permis par les créneaux horaires associés à cette famille. Les contraintes (6.5) et (6.6) permettent de modéliser la conservation des flux physiques à travers l'horizon. Les contraintes (6.7) indiquent qu'il est possible de lancer une production uniquement si la décision est prise. Nous modélisons l'obligation d'avoir une production supérieure à la quantité minimale $q^i_{\min}$ par les contraintes (6.8). Finalement les contraintes (6.9) à (6.15) bornent et définissent les différentes variables.

En pratique ce problème est plus simple que son homologue pour les producteurs fabriquant des produits frais de part sa structure qui ne permet pas de prendre en compte les réglages. De plus, dans les cas d'application que nous avons rencontrés, les durées Γ^k_m sont généralement indépendantes de la machine et de la famille.

6.2 Résultats sur une instance client

Contrairement au problème central traité dans cette thèse pour lequel nous n'avons pas pu obtenir d'instances issues des clients, nous avons pu obtenir des données partielles d'un client fabriquant des produits secs. Ces instances se basent sur les demandes d'un ensemble de références de produit sur un horizon de temps de 15 semaines. Au total 213 références ont une demande pour au moins une semaine sur cet horizon de temps et 10 lignes de production sont disponibles. La capacité des lignes de production correspond à des semaines de travail, où le temps travaillé sur une journée jour est égal à 8h ou 16h. La production se fait par famille de produits sur des durées fixes de 8 heures. Aucune donnée sur les coûts des différentes opérations n'a cependant pu être obtenue dans le cadre de notre étude. Nous avons donc généré les coûts en suivant la même procédure que celle introduite au chapitre 3.

Nous avons utilisé le solveur CPLEX pour résoudre ce problème. Au bout de 60 secondes nous obtenons un gap de 0.27%. Bien que le nombre de références soit important, la capacité de production permet largement de satisfaire la demande. De plus, le fait de ne pas considérer de temps ou de coûts de réglage simplifie aussi le problème.

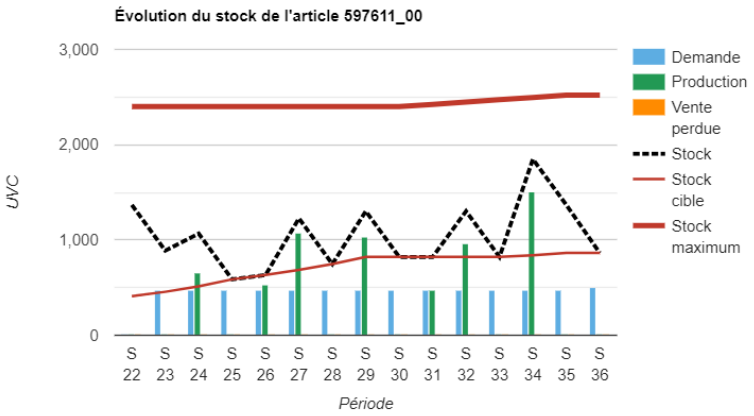
Pour mieux visualiser les solutions obtenues, nous avons développé des interfaces graphiques permettant de voir à la fois le plan de production des références et la charge des lignes de production. Ces interfaces se basent sur des librairies Google Chart. La figure

6.1 montre, pour deux références, l'évolution du stock en fonction de la demande et de la production sur l'horizon de temps. Les deux courbes rouges représentent le stock cible et le stock maximal. Ce dernier, vu comme une limite de stock à ne pas dépasser, n'a aucune influence sur la solution obtenue. Nous voyons que le stock des deux références reste très proche du stock cible. La première référence a une production irrégulière et cyclique tandis que la production de la deuxième assure un stock égal au stock cible au sein de chaque période. Dans la solution obtenue, la demande a toujours été satisfaite.

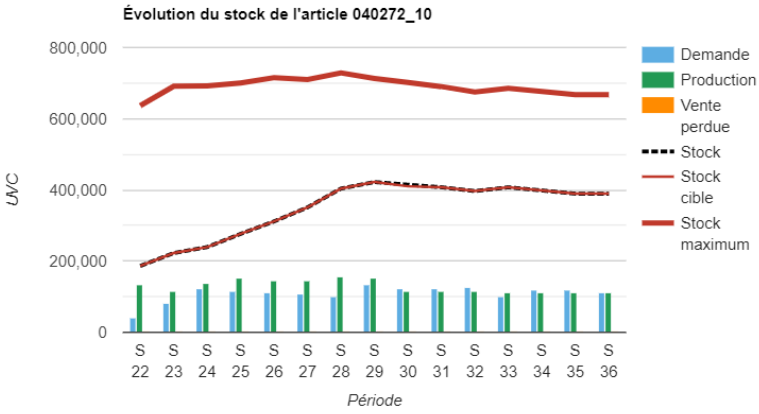
La figure 6.2 représente le plan de charge d'une ligne de production sous forme de graphique. Nous voyons à la fois le temps total d'affectation des familles sur la ligne et la répartition de ces familles de produits au sein de chaque période. En particulier, la figure 6.2 (a) permet de visualiser en rouge la partie affectée à la production qui n'est pas utilisée. Cette gestion de la production par familles définies sur des durées fixes semble donc être sous-optimale car une partie non négligeable du temps est perdue.Plus généralement, nous constatons que la capacité nécessaire à la production dans la solution obtenue est surévaluée de 10% à 20% par les planificateurs lors de la définition de la capacité allouée. Un axe de travail futur est de remettre en question cette pratique actuelle, qui vise à assurer la production, lorsque nous aurons davantage de données disponibles pour estimer précisément les temps de réglage réels et les cadences réelles. Celui-ci permettra de comparer ces solutions obtenues sur la base de regroupements par famille établis *a priori* avec une approche alternative dans laquelle ceux-ci sont définis de façon automatisée par le clustering. Notre contribution pourrait ainsi être mise à profit pour favoriser les regroupements les plus pertinents, sans engendrer de temps perdu. En l'état, pour traiter ce type de problème, l'utilisation d'un solveur suffit pour obtenir de bonnes solutions en un temps rapide.

6.3 Contraintes additionnelles

Dans cette partie, nous présentons une modélisation mathématique des contraintes qui n'ont pas été intégrées dans le problème central. Le Capacitated lot-sizing problem with lost sales, Safety stock and Sequence Dependent setup times on Parallel Machines (CLSSD-PM) étant déjà très riche, certaines d'entre elles ont été écartées du problème générique pour éviter d'augmenter encore davantage sa complexité. D'autres sont relativement atypiques, aussi nous préférons les présenter dans cette partie dédiée. Pour chacune des contraintes, nous présentons les nouvelles notations et variables permettant de les intégrer aux différentes formulations du CLSSD-PM.



(a) Référence 597611_00



(b) Référence 040272_10

FIGURE 6.1 – Plan de production

6.3.1 Lots de matière première

L'ajout des lots de matières premières dans le problème implique qu'il faut adapter la production en fonction de leur disponibilité. En effet, les matières premières qui sont acheminées par lots sont nécessaires pour la fabrication des produits et doivent être utilisées dans une limite de temps. Cette limite de temps peut aller d'une seule à plusieurs périodes. Ce problème est donc différent d'un problème de lot-sizing multi-niveau. Nous introduisons dans un premier temps l'ensemble des matières premières $\mathcal{R}$ et l'ensemble des références $\mathcal{N}_r$ utilisant la matière première r dans leur fabrication. Nous introduisons les paramètres suivants :

— M_r la quantité de la matière première r par lot.

— n_r^i le nombre d'unité de la matière première r nécessaire dans la recette de la référence i.

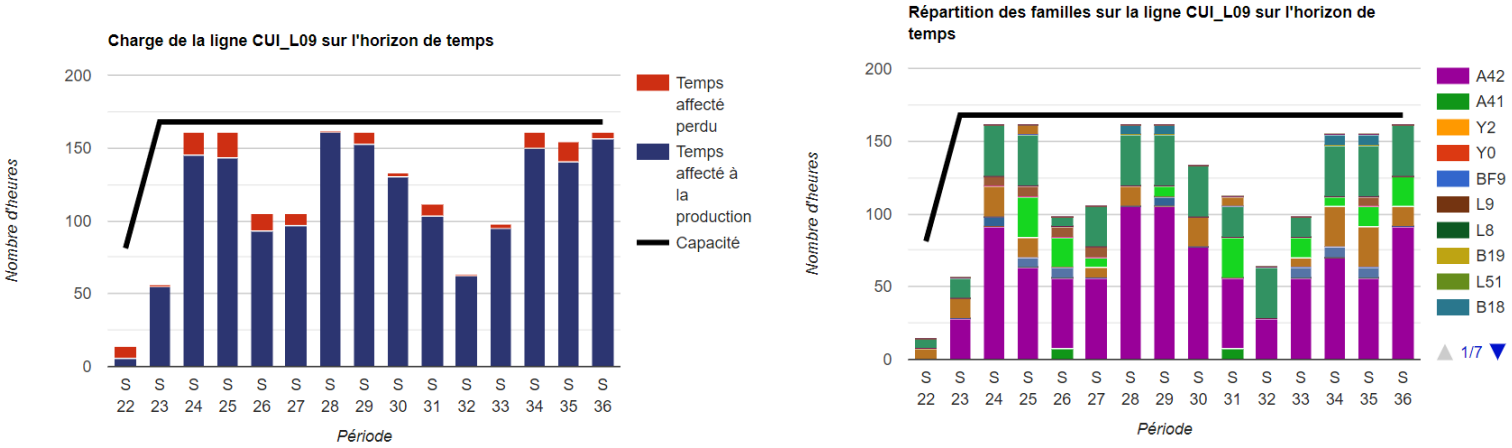
— G_r le coût unitaire de dégagement de la matière première r.

Nous basons notre formulation sur les variables suivantes :

— g_t^r : nombre d'unité de r arrivées en t qui ne sont pas utilisées dans la limite du temps d'utilisation.

— z_t^r : nombre de lots de matière première r qui arrivent en début de période t.

Dans beaucoup de cas pratiques, les matières premières ne sont pas stockées d'une période à une autre. Dans ce cas, pour les prendre en compte dans notre modélisation, nous pouvons ajouter les contraintes suivantes :



(a) Charge

(b) Répartition des familles

FIGURE 6.2 – Plan de charge

$$g_t^r = M_r z_t^r - \sum_{i \in \mathcal{N}_r} \sum_{m \in \mathcal{M}} n_r^i x_{mt}^i \quad \forall r \in \mathcal{R}, \forall t \in \mathcal{T} \tag{6.16}$$

$$g_t^r \geq 0,\ z_t^r \in \mathbb{N} \quad \forall r \in \mathcal{R}, \forall t \in \mathcal{T} \tag{6.17}$$

Les contraintes (6.16) indiquent que le nombre d'unités perdues correspond à la différence entre le nombre d'unités qui arrivent par lot et celles qui sont utilisées dans la période. En s'assurant que g_t^r est une variable continue positive, (6.17) force la production à ne pas dépasser ce qu'il est possible de produire avec les matières premières disponibles.

Dans certains cas, il est possible de stocker les matières premières sur plusieurs périodes. Pour modéliser cela, nous introduisons une variable permettant de suivre le stock des matières premières à travers l'horizon de temps. Soit I_t^r la variable qui représente le stock de la matière première r en fin de période t, le modèle inclut alors les contraintes suivantes :

$$I_t^r = I_{t-1}^r + M_r z_t^r - \sum_{i \in \mathcal{N}_r} \sum_{m \in \mathcal{M}} n_r^i x_{mt}^i - g_t^r \quad \forall r \in \mathcal{R}, \forall t \in \mathcal{T} \tag{6.18}$$

$$I_t^r \leq \sum_{t'=t}^{t+\delta_{rt}} \sum_{i \in \mathcal{N}_r} \sum_{m \in \mathcal{M}} n_r^i x_{mt'}^i \quad \forall r \in \mathcal{R}, \forall t \in \mathcal{T} \tag{6.19}$$

$$g_t^r, I_t^r \geq 0,\ z_t^r \in \mathbb{N} \quad \forall r \in \mathcal{R}, \forall t \in \mathcal{T} \tag{6.20}$$

Finalement, on peut également introduire une pénalité associée au dégagement des ma-

tières premières inutilisées dans notre modélisation en ajoutant le terme (6.21) à la fonction objectif :

$$\sum_{t=1}^{T} \sum_{r \in \mathcal{R}} G_r g_t^r \tag{6.21}$$

Comme nous l'avons vu, deux cas sont présents, à savoir le flux tiré et le flux poussé. En réalité il est possible de modéliser ces deux cas de la même façon en transformant certaines variables en paramètres : Concrètement, dans le cas où l'arrivée des matières premières se fait en flux poussé, la variable z_t^r ne traduit plus une décision à prendre et devient donc un paramètre d'entrée du problème pour tout r et t.

6.3.2 Délai de rétention

La modélisation des délais de rétention (ou des *lead time* pour le terme anglophone) correspond au fait de bloquer une partie de la production pour une durée finie. Ce délai a des raisons bien précises d'être considéré en pratique, par exemple pour pratiquer des contrôles sanitaires sur certains produits. Cette extension a déjà été traitée dans la littérature mais généralement sous forme stochastique ([58, 12]). Dans notre cas, les délais de rétention sont simplifiés par le fait qu'ils sont systématiques et d'une durée fixe pour les références concernées. Pour chaque référence i, nous définissons le nombre de périodes de rétention, $LT_i \geq 0$. Par défaut, il est égal à 0 pour les références sans délai de rétention. Dans le formulation MIP-AGG, nous modifions les contraintes (3.19) pour qu'elles prennent en compte que la production qui entre dans le stock en période t a été produite en période $t - LT_i$ pour chaque référence i :

$$I_{t-1}^i + L_t^i + \sum_{m \in \mathcal{M}} x_{m,t-LT_i}^i = d_t^i + I_t^i \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T} \tag{6.22}$$

Dans le formulation MIP-FL, il suffit d'ajouter les contraintes suivantes qui interdisent de produire pour satisfaire une période pendant la période de rétention :

$$x_{ms}^{it} = 0 \quad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall s \in \mathcal{T}, \forall t \leq s + LT_i - 1 \tag{6.23}$$

Par ailleurs il faut aussi modifier dans l'expression de coût H_{ms}^{it} la date d'entrée en stock, pour prendre en compte uniquement les coûts de stockage liés au stock disponible.

6.3.3 Report de demande

Le report de demande n'a pas été introduit dans nos modélisations pour ne pas surcharger le problème déjà riche que nous traitons. Néanmoins, plusieurs producteurs rencontrent ce type de problématique. C'est principalement le cas des producteurs travaillant en B2B pour lesquelles il est possible de décaler la date où la demande est satisfaite au prix d'une pénalité sur le prix de vente. Dans certains cas, les productions peuvent reporter la demande uniquement pour un sous-ensemble de références qui sont considérées comme moins critiques. Pour intégrer cette extension, en plus des ventes perdues, nous introduisons l'ensemble $\mathcal{N}_b$ correspondant à l'ensemble des références pouvant être reportées. Pour intégrer le report de demande dans la formulation MIP-AGG, nous introduisons la variable B_t^i correspondant au nombre d'unités de la référence $i \in \mathcal{N}_b$ qui sont repoussées après t. Les contraintes (3.19) ne deviennent valables que pour les références dans l'ensemble $\mathcal{N} \setminus \mathcal{N}_b$. Pour l'ensemble $\mathcal{N}_b$, nous ajoutons les contraintes suivantes :

$$I_{t-1}^i + L_t^i + \sum_{m \in \mathcal{M}} x_{m,t}^i - B_{t-1}^i = d_t^i + I_t^i - B_t^i \quad \forall i \in \mathcal{N}_b, \forall t \in \mathcal{T} \tag{6.24}$$

On note également b_t^i que le coût de report d'une unité de demande de la référence i par période t. Pour pénaliser les reports de demande, nous ajoutons à la fonction objectif (6.21) le terme suivant :

$$\sum_{t=1}^{T} \sum_{i \in \mathcal{N}_b} b_t^i B_t^i$$

Pour la formulation MIP-FL, nous autorisons les références de l'ensemble $\mathcal{N}_b$ à être produite de façon à satisfaire une demande d'une période antérieure, la valeur de coût H_{ms}^{it} devient alors :

$$H_{ms}^{it} = d_t^i.(p_{ms}^i - l_t^i + \sum_{u=s}^{\theta_i(t)-1} h_u^{i+} - \sum_{u=\max\{\theta_i(t),s\}}^{t-1} h_u^{i-} - \sum_{u=s}^{t} b_u^i)$$

6.3.4 Taille de lot fixe

Prendre en compte les tailles de lot dans la formulation implique que la production de chaque référence doit être un multiple d'une taille de lot prédéfinie. Plusieurs travaux présentent cette contrainte sur la production ([92, 73, 111, 9]). Néanmoins celle-ci reste peu étudiée dans la littérature : même si son intégration ne demande pas beaucoup de changement dans la formulation classique, elle rend sa résolution plus difficile. Dans la

formulation MIP-AGG, il est possible de prendre en compte cette contrainte en transformant les variables de production x^i_{mt} en variables entières. En notant Q_i la taille de lot de la référence i, le terme $Q_i x^i_{mt}$ permet d'obtenir le nombre d'unités de la référence i produite sur la machine m en période t.

Dans la formulation MIP-FL, il est possible de modéliser cette contrainte sur la production en ajoutant les variables q^i_{mt} indiquant le nombre de lot produit de la référence i sur la machine m à la période t et d'ajouter la contrainte suivante :

$$\sum_{t=s}^{\delta^i_s} \frac{d^i_t}{Q_i} x^i_{ms} = q^i_{ms} \quad \forall i \in \mathcal{N}, \forall m \in \mathcal{M}, \forall s \in \mathcal{T} \tag{6.25}$$

Pour contourner la complexité induite par cette contrainte justifie une pratique courante, qui consiste à résoudre d'abord le problème sans considérer les tailles de lot fixes, avant d'appliquer un lissage ou simplement prendre la partie entière des variables de production pour obtenir une solution réalisable.

6.4 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle modélisation mathématique d'un problème distinct concernant le plan de production des fabriquants de produits secs. Nous avons pu tester partiellement nos méthodes sur une instance client pour laquelle certaines informations ont du être générées par nos soins. Plusieurs sorties graphiques ont été développées pour mieux visualiser les solutions obtenues. Nous avons dans un deuxième temps proposé des modélisations d'extensions au problème CLSSD-PM qui n'ont pas été prises en compte dans le problème central. La prochaine étape cruciale dans le transfert des algorithmes développés visera à les intégrer dans la solution *Planification* de VIF afin de pouvoir se comparer avec les plans de production construits par les producteurs. Ce travail sera aussi l'occasion d'échanger avec un ou plusieurs producteurs pour s'assurer que les hypothèses sur lesquelles nous nous sommes appuyés durant notre étude sont bien cohérentes avec leur pratique. Cette phase va bien entendu nécessiter de très nombreux tests, et permettra certainement de mettre en évidence des difficultés ou des sous-cas que nous n'avons pas encore considérés. Des améliorations de nos méthodes et différents paramétrages logiciel seront ainsi apportés avant la commercialisation pour couvrir l'ensemble des besoins clients.

Plus généralement, nous avons vu que de nombreux sous-cas sont à traiter pour proposer un outil flexible capable de s'adapter aux utilisateurs. Il est alors primordial de

pouvoir les accompagner dans le paramétrage du module suivant leur besoin.

Conclusion générale et perspectives

Dans cette thèse, nous avons développé des solutions algorithmiques pour répondre à des besoins industriels identifiés par VIF. Les développements produits s'intègrent dans la volonté de compléter l'outil PDP actuel par un module d'optimisation. Au-delà des développements réalisés, ce projet a été pour VIF une opportunité significative pour comprendre et clarifier les problématiques rencontrées par les producteurs, mais aussi acquérir une plus grande maîtrise des techniques d'optimisation, aujourd'hui largement attendue par la clientèle.

Dans ce manuscrit, nous nous sommes focalisés sur la modélisation et la résolution d'un problème complexe de planification de production. Celui-ci, noté CLSSD-PM, est défini à partir de différentes extensions du lot-sizing qui apparaissent dans la littérature et dont la combinaison rend particulièrement difficile une résolution directe en un temps raisonnable. Notre première contribution a été de proposer deux modélisations mathématiques pour ce problème, l'une basée sur une formulation agrégée et l'autre basée sur une formulation désagrégée. À notre connaissance, aucune référence de la littérature existante ne formalise l'intégration de ventes perdues, d'heures supplémentaires et d'un stock cible dans un problème de lot-sizing avec des *sequence dependent setup times* sur plusieurs machines parallèles. Nous avons également proposé une nouvelle génération d'instances basée sur des données métier qui permet de prendre en compte l'ensemble des caractéristiques considérées dans ce problème générique. De plus, celle-ci enrichit les jeux d'instances déjà existants de la littérature pour des problèmes connexes, qui peuvent être considérés comme des cas particuliers du CLSSD-PM. Cette génération d'instances peut être appliquée à de nombreuses autres problématiques de lot-sizing.

Une contribution significative de cette thèse a été de proposer un pré-traitement basé sur une procédure d'apprentissage machine et qui permet d'agréger les références en *clusters* de façon à réduire la taille des données de nos instances. Nous avons mesuré l'impact des paramètres retenus pour notre algorithme de pré-traitement pour déterminer sa pertinence sur les instances générées. Les résultats montrent qu'appliquer ce pré-traitement, en plus du Relax-and-Fix et de l'heuristique de recherche locale Fix-and-Optimize, per-

met d'obtenir des solutions satisfaisantes pour une application industrielle sur un grand nombre d'instances. Dans cette thèse, nous avons aussi exploré plusieurs autres pistes de recherche qui demandent encore des améliorations pour obtenir des résultats pleinement satisfaisants. La première piste a été de proposer une résolution par l'utilisation d'un algorithme génétique. Nous avons obtenu de meilleures solutions que le solveur CPLEX pour les instances les plus difficiles à résoudre, mais les solutions obtenues avec celle-ci étaient de moins bonne qualité que celles obtenues avec les heuristiques dédiées proposées précédemment. La deuxième piste a été de développer une approche itérative originale, dans laquelle chaque itération se divise en trois phases qui fixent chacune un sous-ensemble des variables de décision. Cette approche, plus prometteuse, a été proposée en fin de thèse et plusieurs pistes d'amélioration sont encore à étudier pour atteindre de meilleurs résultats. En particulier, une estimation plus fine des temps de réglage dans la première phase à l'aide d'un pré-traitement similaire à celui développé au chapitre 4 est susceptible d'accélérer l'exécution et la convergence de la procédure.

Plusieurs autres perspectives dans l'ensemble des travaux réalisés sont à envisager. La première concerne l'utilisation des méthodes pour détecter le meilleur nombre de *clusters* dans le pré-traitement. Nous avons décidé d'utiliser le score *silhouette* mais d'autres méthodes peuvent être envisagés. Il est par exemple possible d'utiliser la méthode *elbow* ([20]) et une piste de recherche consiste ainsi à comparer ces différentes mesures de performance du clustering et ainsi d'estimer leur pertinence vis-à-vis de l'optimisation. Une deuxième perspective pourrait être d'améliorer le Fix-and-Optimize par une hybridation de la même manière que Turhan et Bilgen [109] qui ont utilisé un algorithme de recuit simulé pour choisir les ensembles de variables à ré-optimiser. Plus récemment, Carvalho et Nascimento [27] ont également étudié d'autres heuristiques pour améliorer cette phase de recherche locale, en particulier des méthodes dites de *path-relinking* ([89, 26]) ou de *kernel search* ([14, 54]. Une étude similaire serait susceptible de dégager des pistes d'amélioration pour les matheuristiques introduites au chapitre 4. Finalement, en l'état notre approche itérative en trois phases comporte de nombreuses pistes d'amélioration détaillées en 5.4 qui peuvent constituer une piste de recherche pour des problèmes similaires.

Dans les travaux futurs sur les problèmes de lot-sizing, nos contributions peuvent être adaptées et nourrir de nouvelles approches. Notre idée de réduire la taille des données par un pré-traitement pour des instances de grande taille pourrait être notamment étendue à des problèmes plus vastes, comme le Production Routing Problem, où il est possible de regrouper des clients lorsque la structure des données le permet (regroupements par régions géographiques par exemple).

Pour conclure d'un point de vue recherche, une piste que nous n'avons pas eu le temps d'explorer est de résoudre le problème à l'aide d'une génération de colonnes. Plusieurs auteurs ont en effet proposé une formulation du CLSD dite parfois *sequence oriented* ([55]) dans laquelle un nombre exponentiel de variables représentent les séquences possibles. Une génération de colonnes peut permettre de prendre en compte dans le modèle seulement les variables pertinentes. En particulier, dans le cas où le problème de base serait modifié pour prendre en compte uniquement des séquences entre clusters, la génération de colonnes serait simplifiée par un nombre réduit de cas possibles.

Au niveau du travail à effectuer pour une industrialisation des travaux de recherche, la prochaine étape est de mener des tests sur des données clients pour avoir un cadre de comparaison de nos résultats avec les pratiques actuelles. Ceci n'a malheureusement pas été possible durant cette thèse en raison de la difficulté à obtenir des informations précises sur lesquelles construire le cadre expérimental de mes travaux. Cette étape nécessite de transférer et potentiellement d'adapter le code des algorithmes dans le projet Java sur lequel se base le PDP. Il sera aussi nécessaire de trouver le bon paramétrage des algorithmes en fonction des contextes d'application. Ces tâches relèvent donc d'un travail d'ingénierie que je réaliserai prochainement dans le cadre de mes nouvelles fonctions au sein de VIF. La deuxième étape concerne le véritable déploiement dans le module PDP pour une utilisation par les clients. Elle nécessitera la mise en place d'interfaces graphiques permettant à l'utilisateur d'avoir accès à ces méthodes et de les paramétrer de façon simple et intuitive. De plus, des questions autour de l'architecture du serveur de calcul dédié à faire tourner les algorithmes d'optimisation vont se poser. Pour terminer cette mise en production, il sera sans doute nécessaire d'accompagner les clients durant cette première phase de déploiement, puis d'assurer le suivi de leurs retours pour améliorer en continu le logiciel et les méthodes de résolution employées.

L'intégration d'outils d'aide à la décision dans le logiciel doit permettre à terme de faire gagner du temps aux planificateurs, d'avoir un plan de production plus efficace et robuste amenant à moins de gaspillage. L'évolution logique pour VIF et pour leurs clients est de prolonger cette approche à des niveaux de décision plus fins au sein de l'ordonnancement, en tenant en compte des informations issues du PDP. Une nouvelle thèse CIFRE, que j'encadrerai, va débuter sur cette problématique. L'enjeu est de développer des méthodes de résolution permettant d'obtenir des ordonnancements adaptés aux contraintes des ateliers de production. Une contrainte importante qui n'est pas intégrée à la maille temporelle du PDP est l'intégration de temps minimum/maximum entre deux opérations sur des machines distinctes. De même, certains ordres de fabrication nécessitent des res-

sources auxiliaires spécifiques pour effectuer des réglages et les décisions d'affectation de ces dernières ne sont généralement traitées qu'au niveau de l'ordonnancement. Celui-ci a donc pour objectif d'utiliser les solutions obtenues par le PDP en intégrant ces nouvelles informations. Ainsi, imbriquer ces deux modules et proposer des outils dédiés pour optimiser les décisions permettra à terme de réduire les coûts et l'impact global de la production pour les industriels.

Cette thèse a été, pour la société VIF, une occasion de dégager de très prometteuses pistes pour mettre à disposition des producteurs de l'agroalimentaire des outils d'optimisation avancés. L'intégration d'algorithmes d'optimisation dans les différentes étapes de décision au sein de la chaîne logistique a pour objectif d'apporter une avancée significative dans le monde de l'agroalimentaire. Il pourrait être intéressant dans ce but d'étendre les apports d'algorithmes à d'autres problématiques comme la prévision des ventes ou les étapes de distribution dans la Supply Chain.

ANNEXE

A.1 Inégalité valide

Les inégalités (t, S, R) définies comme suit :

$$\sum_{s\in\mathcal{P}\setminus\mathcal{S}}\sum_{m\in\mathcal{M}} x^i_{ms}+\sum_{s\in\mathcal{S}}\sum_{\substack{u=s\\u\in\mathcal{R}}}^{t} d^i_u z^i_s \geq \sum_{s\in\mathcal{R}}(d^i_s-L^i_s) \quad \forall i\in\mathcal{N}, \forall t\in\mathcal{T}, \mathcal{P}=\{1,\ldots,t\}, \mathcal{S}\subseteq\mathcal{P}, \mathcal{R}\subseteq\mathcal{P} \tag{26}$$

sont valides pour le CLSSD-PM.

Démonstration. La démonstration se base sur celle présentée dans Loparic *et al.* [76]. Nous allons simplement l'adapter à notre problème et nos notations. Soit $i \in \mathcal{N}$, $t \in \mathcal{T}$, et trois sous-ensembles de $\mathcal{T}$: $\mathcal{P} = \{1,\ldots,t\}, \mathcal{S} \subseteq \mathcal{P}, \mathcal{R} \subseteq \mathcal{P}$. Pour démonter que les inégalités (3.35) sont valides, nous vérifions deux cas :

1. Supposons $z^{*i}_s = 0$ pour tout $i \in \mathcal{N}$ et $s \in \mathcal{S}$. Cela implique que $x^{*i}_{ms} = 0$ pour tout $i \in \mathcal{N}$, $m \in \mathcal{M}$ et $s \in \mathcal{S}$. On a alors pour tout $i \in \mathcal{N}$:

$$\begin{aligned}\sum_{s\in\mathcal{P}\setminus\mathcal{S}}\sum_{m\in\mathcal{M}} x^{*i}_{ms} + \sum_{s\in\mathcal{S}}\sum_{\substack{u=s\\u\in\mathcal{R}}}^{t} d^i_u z^{*i}_s &= \sum_{s\in\mathcal{P}}\sum_{m\in\mathcal{M}} x^{*i}_{ms}\\ &\geq \sum_{s\in\mathcal{P}}(d^i_s - L^{*i}_s)\\ &\geq \sum_{s\in\mathcal{R}}(d^i_s - L^{*i}_s)\end{aligned}$$

2. Sinon, posons $k^i = \min\{s \in \mathcal{S} : z^i_s = 1\}$ pour tout $i \in \mathcal{N}$. Cela implique que $x^{*i}_{ms} = 0$ pour tout $i \in \mathcal{N}$, $m \in \mathcal{M}$ et $s \in \mathcal{S},\ s \leq k^i - 1$. On a alors :

$$\begin{aligned}\sum_{s\in\mathcal{P}\setminus\mathcal{S}}\sum_{m\in\mathcal{M}} x^{*i}_{ms} \geq \sum_{\substack{s\in\mathcal{P}\setminus\mathcal{S}\\s\leq k^i-1}}\sum_{m\in\mathcal{M}} x^{*i}_{ms} &= \sum_{s\leq k^i-1}\sum_{m\in\mathcal{M}} x^{*i}_{ms}\\ &\geq \sum_{s\leq k^i-1}(d^i_s - L^{*i}_s)\\ &\geq \sum_{\substack{s\in\mathcal{R}\\s\leq k^i-1}}(d^i_s - L^{*i}_s)\end{aligned} \tag{27}$$

Nous avons de plus :

$$\sum_{s\in\mathcal{S}}\sum_{\substack{u=s\\u\in\mathcal{R}}}^{t} d_u^i z_s^{*i} \geq \sum_{\substack{u=k^i\\u\in\mathcal{R}}}^{t} d_u^i \geq \sum_{\substack{s=k^i\\s\in\mathcal{R}}}^{t} (d_s^i - L_s^{*i}) \tag{28}$$

Finalement, à partir de (27) et (28), nous obtenons :

$$\sum_{s\in\mathcal{P}\setminus\mathcal{S}}\sum_{m\in\mathcal{M}} x_{ms}^{*i} + \sum_{s\in\mathcal{S}}\sum_{\substack{u=s\\u\in\mathcal{R}}}^{t} d_u^i z_s^{*i} \geq \sum_{\substack{s\leq k^i-1\\s\in\mathcal{R}}} (d_s^i - L_s^{*i}) + \sum_{\substack{s=k^i\\s\in\mathcal{R}}}^{t} (d_s^i - L_s^{*i}) \geq \sum_{s\in\mathcal{R}} (d_s^i - L_s^i)$$

Ce qui prouve que (3.35) est valide pour le CLSSD-PM. □

A.2 Tableaux de résultats des modèles MIP

Tableau A.1 – Impact de Φ sur le gap

Taille du problème N-M-T	$\mathcal{U}$		$Gap_{moy}(\%)$	$Gap_{min}(\%)$	$Gap_{max}(\%)$	Temps (s)
20-1-15	0.9	$\Phi = 1$	0.35	0.0	2.74	393.1
20-1-15	0.9	$\Phi = 2$	0.26	0.0	1.45	523.6
20-1-15	0.9	$\Phi = 4$	0.3	0.0	1.19	655.1
20-1-15	1.0	$\Phi = 1$	0.58	0.0	2.2	591.0
20-1-15	1.0	$\Phi = 2$	0.82	0.0	3.93	634.1
20-1-15	1.0	$\Phi = 4$	0.84	0.0	3.39	801.7
20-2-15	0.9	$\Phi = 1$	6.6	0.0	26.42	833.6
20-2-15	0.9	$\Phi = 2$	8.43	0.0	61.6	876.5
20-2-15	0.9	$\Phi = 4$	51.89	0.02	498.16	900.0
20-2-15	1.0	$\Phi = 1$	9.72	0.0	46.48	858.3
20-2-15	1.0	$\Phi = 2$	15.62	0.12	74.59	900.0
20-2-15	1.0	$\Phi = 4$	177.5	0.21	1631.34	900.0
20-1-30	0.9	$\Phi = 1$	11.78	0.0	57.01	865.0
20-1-30	0.9	$\Phi = 2$	31.17	0.11	277.56	900.0
20-1-30	0.9	$\Phi = 4$	61.62	0.3	580.15	900.0
20-1-30	1.0	$\Phi = 1$	66.98	0.0	514.53	887.2
20-1-30	1.0	$\Phi = 2$	103.39	0.57	631.49	900.0
20-1-30	1.0	$\Phi = 4$	687.65	1.23	6364.65	900.0
20-2-30	0.9	$\Phi = 1$	5.81	0.51	14.1	900.0
20-2-30	0.9	$\Phi = 2$	7.86	0.35	45.86	900.0
20-2-30	0.9	$\Phi = 4$	13.97	0.72	115.55	900.0
20-2-30	1.0	$\Phi = 1$	87.33	2.01	258.37	900.0
20-2-30	1.0	$\Phi = 2$	181.49	1.57	938.11	900.0
20-2-30	1.0	$\Phi = 4$	540.99	1.41	2838.52	900.0
30-1-15	0.9	$\Phi = 1$	13.09	0.06	46.63	900.0
30-1-15	0.9	$\Phi = 2$	154.24	0.18	1424.29	900.0
30-1-15	0.9	$\Phi = 4$	18.1	0.31	103.16	900.0
30-1-15	1.0	$\Phi = 1$	13.6	0.0	41.66	893.6
30-1-15	1.0	$\Phi = 2$	29.57	0.4	234.33	900.0
30-1-15	1.0	$\Phi = 4$	8.94	0.67	21.2	900.0
30-2-15	0.9	$\Phi = 1$	183.02	2.85	1610.55	900.0
30-2-15	0.9	$\Phi = 2$	135.95	5.81	1162.42	900.0
30-2-15	0.9	$\Phi = 4$	242.22	1.45	1736.82	900.0
30-2-15	1.0	$\Phi = 1$	30.31	5.0	109.52	900.0
30-2-15	1.0	$\Phi = 2$	52.15	1.91	283.13	900.0
30-2-15	1.0	$\Phi = 4$	66.2	3.78	512.02	900.0
30-1-30	0.9	$\Phi = 1$	2333.04	0.98	19348.01	900.0
30-1-30	0.9	$\Phi = 2$	949.71	0.49	3433.25	900.1
30-1-30	0.9	$\Phi = 4$	2899.99	2.23	16478.53	900.0
30-1-30	1.0	$\Phi = 1$	8678.81	27.71	24287.13	900.0
30-1-30	1.0	$\Phi = 2$	5141.49	45.49	27133.84	900.0
30-1-30	1.0	$\Phi = 4$	4590.99	27.77	24234.37	900.0
30-2-30	0.9	$\Phi = 1$	3968.54	5.6	30884.03	900.0
30-2-30	0.9	$\Phi = 2$	2729.32	3.45	18043.49	900.0
30-2-30	0.9	$\Phi = 4$	2335.45	52.86	6178.03	900.0
30-2-30	1.0	$\Phi = 1$	2037.21	9.78	8531.96	900.0
30-2-30	1.0	$\Phi = 2$	4597.57	349.71	11309.69	900.0
30-2-30	1.0	$\Phi = 4$	4241.26	842.36	15838.8	900.0

Tableau A.2 – Analyse des inégalités valides

Taille du problème N-M-T	$\mathcal{U}$		$Gap_{moy}(\%)$	$Gap_{min}(\%)$	$Gap_{max}(\%)$	User Cut	CPLEX Cut	Temps(s)
20-1-15	0.9	(l,S)	0.47	0.0	3.73	0.6	1696.4	427.7
20-1-15	0.9	(k,U)	0.38	0.0	2.76	3.1	1528.8	398.5
20-1-15	1.0	(l,S)	0.91	0.0	4.26	43.1	1951.5	591.5
20-1-15	1.0	(k,U)	0.89	0.0	5.13	22.2	1828.2	569.6
20-2-15	0.9	(l,S)	5.18	0.0	26.46	175.1	2246.2	843.4
20-2-15	0.9	(k,U)	5.21	0.0	25.43	31.3	2279.5	820.7
20-2-15	1.0	(l,S)	11.88	0.0	58.66	120.9	1920.9	843.7
20-2-15	1.0	(k,U)	6.56	0.0	17.74	44.4	1945.9	842.6
20-1-30	0.9	(l,S)	13.55	0.0	76.75	17.7	3972.7	855.4
20-1-30	0.9	(k,U)	8.07	0.0	42.8	7.6	4039.4	857.2
20-1-30	1.0	(l,S)	50.99	0.06	367.31	145.2	4030.6	900.0
20-1-30	1.0	(k,U)	60.21	0.0	515.55	80.5	3890.0	895.9
20-2-30	0.9	(l,S)	6.08	0.52	21.71	98.1	4281.2	900.0
20-2-30	0.9	(k,U)	5.72	0.52	18.18	48.1	4255.2	900.0
20-2-30	1.0	(l,S)	83.97	2.09	254.35	455.0	3424.8	900.0
20-2-30	1.0	(k,U)	50.59	1.88	189.9	270.6	3395.6	900.0
30-1-15	0.9	(l,S)	140.9	0.01	1271.86	80.1	2792.6	900.0
30-1-15	0.9	(k,U)	25.34	0.0	171.04	52.5	2934.9	893.2
30-1-15	1.0	(l,S)	12.32	0.0	54.87	53.9	2845.5	889.8
30-1-15	1.0	(k,U)	29.52	0.0	183.02	28.8	2689.0	881.6
30-2-15	0.9	(l,S)	197.06	7.88	1621.61	87.5	2939.6	900.1
30-2-15	0.9	(k,U)	200.81	2.84	1615.23	52.1	2922.8	900.0
30-2-15	1.0	(l,S)	40.32	4.16	131.56	91.7	2924.6	900.0
30-2-15	1.0	(k,U)	38.26	5.63	132.31	52.3	2965.0	900.0
30-1-30	0.9	(l,S)	2605.86	2.28	19380.92	22.7	5469.2	900.0
30-1-30	0.9	(k,U)	2702.33	0.74	19377.12	26.9	5610.3	900.0
30-1-30	1.0	(l,S)	11391.36	14.96	24272.0	37.3	5433.5	900.0
30-1-30	1.0	(k,U)	10150.29	14.08	24211.03	39.2	5394.6	900.0
30-2-30	0.9	(l,S)	4927.65	5.57	23634.8	46.9	6427.2	900.0
30-2-30	0.9	(k,U)	5875.89	3.05	30649.25	37.0	6376.9	900.0
30-2-30	1.0	(l,S)	2167.46	13.12	8525.93	151.1	5927.2	900.0
30-2-30	1.0	(k,U)	1712.13	59.9	8529.05	149.1	6055.0	900.0
40-1-15	0.9	(l,S)	2288.14	0.24	20399.9	6.5	4119.1	900.0
40-1-15	0.9	(k,U)	3280.16	0.15	20245.71	7.1	3782.2	900.0
40-1-15	1.0	(l,S)	1449.29	5.65	13230.57	96.8	3371.7	900.1
40-1-15	1.0	(k,U)	2569.32	13.71	22614.95	122.8	3343.0	900.0
40-2-15	0.9	(l,S)	475.05	21.18	2305.24	30.1	4094.8	900.0
40-2-15	0.9	(k,U)	1708.04	1.49	9402.62	28.4	4183.7	900.0
40-2-15	1.0	(l,S)	2323.22	4.4	10110.81	27.5	4036.7	900.0
40-2-15	1.0	(k,U)	1485.91	30.73	7413.64	21.9	3974.6	900.0
40-1-30	0.9	(l,S)	8713.09	365.55	19820.5	20.3	7327.3	900.0
40-1-30	0.9	(k,U)	6021.96	22.93	17149.25	27.5	7139.6	900.0
40-1-30	1.0	(l,S)	8799.13	115.31	20570.41	36.6	7011.8	900.0
40-1-30	1.0	(k,U)	5761.68	1655.05	18621.53	45.7	7010.4	900.0
40-2-30	0.9	(l,S)	10360.64	1709.48	23931.11	30.2	7922.6	900.0
40-2-30	0.9	(k,U)	9541.0	1757.72	24520.82	31.1	7886.7	900.0
40-2-30	1.0	(l,S)	12562.88	1857.98	28372.61	102.4	8055.4	900.0
40-2-30	1.0	(k,U)	9848.52	1966.79	28350.8	64.1	6934.2	900.0

Tableau A.3 – Résultats expérimentaux des coupes W

Taille du problème N-M-T	$\mathcal{U}$		$Gap_{moy}(\%)$	$Gap_{min}(\%)$	$Gap_{max}(\%)$	Temps (s)
20-1-15	0.9		0.35	0.0	2.74	393.1
20-1-15	0.9	W Cut	0.5	0.0	4.36	378.2
20-1-15	1.0		0.58	0.0	2.2	591.0
20-1-15	1.0	W Cut	0.58	0.0	2.34	564.7
20-2-15	0.9		6.6	0.0	26.42	833.6
20-2-15	0.9	W Cut	5.75	0.0	23.27	819.5
20-2-15	1.0		9.72	0.0	46.48	858.3
20-2-15	1.0	W Cut	21.12	0.0	148.6	850.1
20-1-30	0.9		11.78	0.0	57.01	865.0
20-1-30	0.9	W Cut	12.44	0.0	86.39	857.8
20-1-30	1.0		66.98	0.0	514.53	887.2
20-1-30	1.0	W Cut	60.78	0.0	393.34	879.2
20-2-30	0.9		5.81	0.51	14.1	900.0
20-2-30	0.9	W Cut	3.37	0.48	7.14	900.0
20-2-30	1.0		87.33	2.01	258.37	900.0
20-2-30	1.0	W Cut	92.51	2.01	248.37	900.0
30-1-15	0.9		13.09	0.06	46.63	900.0
30-1-15	0.9	W Cut	16.82	0.0	69.94	890.5
30-1-15	1.0		13.6	0.0	41.66	893.6
30-1-15	1.0	W Cut	27.75	0.0	157.9	869.8
30-2-15	0.9		183.02	2.85	1610.55	900.0
30-2-15	0.9	W Cut	494.44	2.78	4088.65	900.0
30-2-15	1.0		30.31	5.0	109.52	900.0
30-2-15	1.0	W Cut	66.78	3.52	169.82	900.0
30-1-30	0.9		2333.04	0.98	19348.01	900.0
30-1-30	0.9	W Cut	3093.95	0.66	19378.96	900.0
30-1-30	1.0		8678.81	27.71	24287.13	900.0
30-1-30	1.0	W Cut	8858.57	15.19	21100.04	900.0
30-2-30	0.9		3968.54	5.6	30884.03	900.0
30-2-30	0.9	W Cut	859.67	23.08	3046.46	900.0
30-2-30	1.0		2037.21	9.78	8531.96	900.0
30-2-30	1.0	W Cut	1368.24	10.85	5884.53	900.0
40-1-15	0.9		70.29	0.19	307.76	900.0
40-1-15	0.9	W Cut	541.32	0.25	2752.62	900.0
40-1-15	1.0		2923.54	25.02	22406.81	900.0
40-1-15	1.0	W Cut	1349.84	38.63	11094.23	900.0
40-2-15	0.9		537.94	2.61	3563.97	900.0
40-2-15	0.9	W Cut	832.03	8.59	3849.28	900.0
40-2-15	1.0		1395.85	5.0	7396.69	900.0
40-2-15	1.0	W Cut	1462.19	5.22	7873.93	900.0
40-1-30	0.9		6190.99	713.69	17149.16	900.0
40-1-30	0.9	W Cut	11757.52	91.91	21563.62	900.0
40-1-30	1.0		7587.47	366.54	20331.88	900.0
40-1-30	1.0	W Cut	11924.37	57.77	22452.52	900.0
40-2-30	0.9		9253.24	630.78	24584.58	900.1
40-2-30	0.9	W Cut	13472.17	3737.52	24348.27	900.1
40-2-30	1.0		11449.44	2606.27	28378.05	900.0
40-2-30	1.0	W Cut	11621.63	79.28	28340.47	900.0

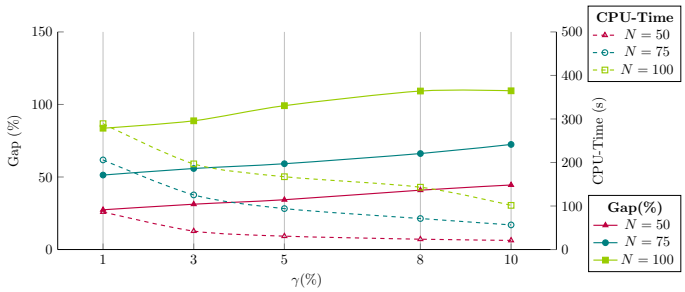
A.3 Analyse RFFO

Tableau A.4 – Résultats du RFFO ($\sigma = 1$, $\rho = 60$s)

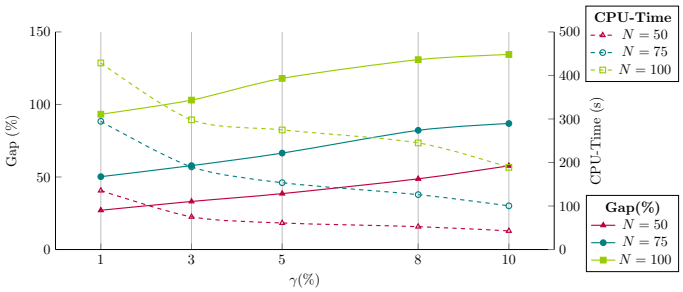
Taille		20-1-15	20-1-30	20-2-15	20-2-30	30-1-15	30-1-30	30-2-15	30-2-30	40-1-15	40-1-30	40-2-15	40-2-30
RFFO-C(2, 1, 1, 60)	Gap Moy. (%)	10.62	20.92	13.5	16.16	26.86	25.72	27.52	27.04	35.52	39.5	33.82	45.04
	Temps (s)	6.12	24.72	17.48	53.24	12.72	37.92	44.68	107.08	17.92	65.16	55.68	172.96
RFFO-C(2, 1, 3, 60)	Gap Moy. (%)	12.38	26.16	16.43	23.13	28.09	31.09	31.22	34.85	38.6	45.19	38.37	53.76
	Temps (s)	4.88	18.6	13.16	39.48	10.08	34.76	27.72	71.44	13.56	45.28	32.72	111.08
RFFO-C(3, 1, 1, 60)	Gap Moy. (%)	10.2	19.14	12.83	15.43	25.08	24.06	25.35	25.24	34.44	37.21	32.56	42.6
	Temps (s)	5.36	21.12	27.24	72.36	14.96	44.92	97.76	169.16	24.52	74.2	123.72	243.0
RFFO-C(3, 1, 3, 60)	Gap Moy. (%)	11.79	23.97	15.14	21.16	26.99	28.22	28.97	31.69	35.61	41.99	36.0	49.31
	Temps (s)	3.6	14.84	11.04	30.96	9.8	26.96	31.32	72.12	13.32	40.04	38.2	102.92

Taille		50-1-15	50-1-30	50-2-15	50-2-30	75-2-15	75-2-30	75-4-15	75-4-30	100-2-15	100-2-30	100-4-15	100-4-30
RFFO-C(2, 1, 3, 60)	Gap Moy. (%)	74.86	62.05	56.45	69.69	99.48	187.22	44.75	77.07	201.43	346.86	82.59	158.7
	Temps (s)	21.48	55.68	41.2	150.52	70.04	351.56	85.0	347.04	146.12	543.44	149.64	624.0
RFFO-C(2, 1, 5, 60)	Gap Moy. (%)	80.66	68.29	61.44	78.09	105.41	199.65	48.05	83.95	206.14	364.3	87.93	170.27
	Temps (s)	18.44	53.4	34.68	132.96	59.12	294.72	69.8	283.44	115.0	484.92	111.04	529.84
RFFO-C(2, 1, 8, 60)	Gap Moy. (%)	87.58	87.58	69.96	97.71	111.12	220.78	56.66	98.68	216.64	387.32	92.39	181.9
	Temps (s)	15.08	46.56	28.2	109.2	51.24	272.68	52.56	252.08	95.92	441.6	91.84	464.6
RFFO-C(3, 1, 3, 60)	Gap Moy. (%)	71.5	56.86	53.48	65.26	96.11	189.0	42.07	74.71	196.61	431.28	78.22	197.7
	Temps (s)	25.84	47.8	52.08	186.44	107.36	466.2	117.28	411.6	205.56	653.44	219.8	709.48
RFFO-C(3, 1, 5, 60)	Gap Moy. (%)	74.68	62.43	56.22	71.72	100.18	200.32	45.55	82.02	200.91	436.96	85.02	188.78
	Temps (s)	18.72	43.4	34.76	124.96	64.88	352.48	78.04	311.44	145.92	509.56	157.36	560.88
RFFO-C(3, 1, 8, 60)	Gap Moy. (%)	80.9	71.88	61.48	85.59	103.92	221.63	50.1	91.88	209.47	407.89	89.08	226.99
	Temps (s)	15.2	38.12	27.56	106.24	60.72	289.56	61.52	239.2	134.16	410.88	104.0	508.52

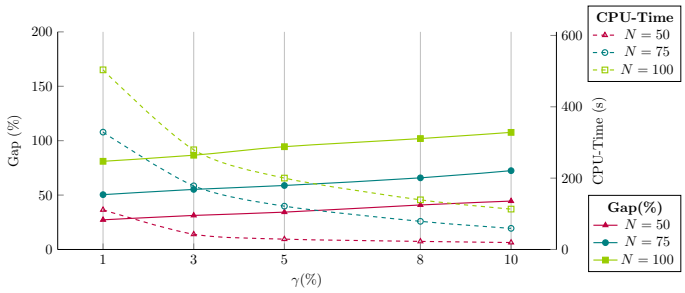
Taille		125-4-15	125-4-30	125-6-15	125-6-30
RFFO-C(2, 1, 5, 60)	Gap Moy. (%)	144.53	230.41	108.87	168.89
	Temps (s)	156.4	759.68	167.28	710.92
RFFO-C(2, 1, 8, 60)	Gap Moy. (%)	151.71	313.1	115.98	185.55
	Temps (s)	140.28	640.76	136.6	638.76
RFFO-C(2, 1, 10, 60)	Gap Moy. (%)	158.28	280.64	120.45	193.33
	Temps (s)	134.04	656.76	126.04	577.52
RFFO-C(3, 1, 5, 60)	Gap Moy. (%)	145.58	340.52	105.09	339.16
	Temps (s)	228.32	830.12	218.04	818.64
RFFO-C(3, 1, 8, 60)	Gap Moy. (%)	159.16	342.51	114.29	367.16
	Temps (s)	200.0	677.28	161.68	646.08
RFFO-C(3, 1, 10, 60)	Gap Moy. (%)	154.64	364.66	114.27	359.77
	Temps (s)	165.2	651.44	142.28	611.8



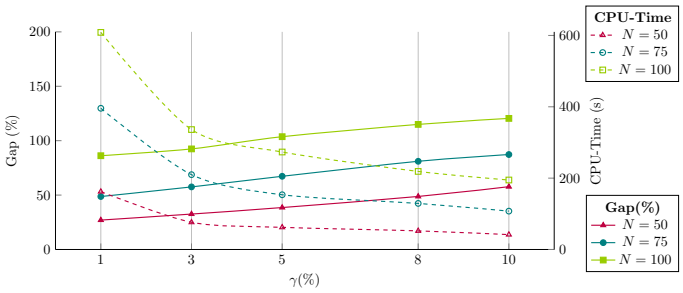
(a) $\rho = 30,\ T = 15$



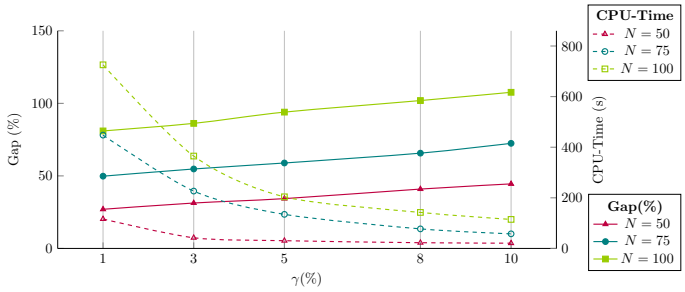
(b) $\rho = 30,\ T = 30$



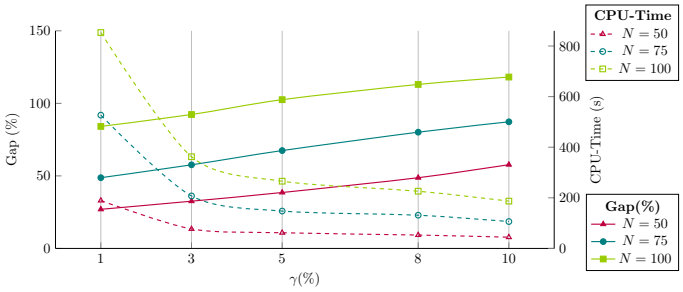
(c) $\rho = 60,\ T = 15$



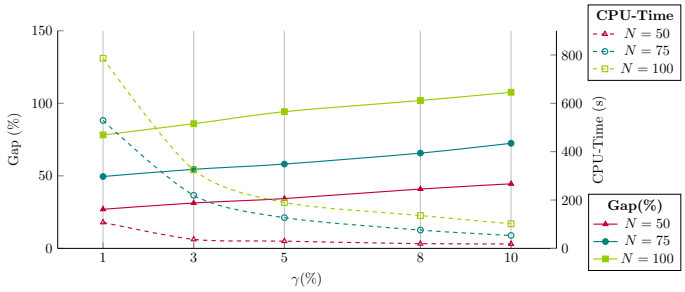
(d) $\rho = 60,\ T = 30$



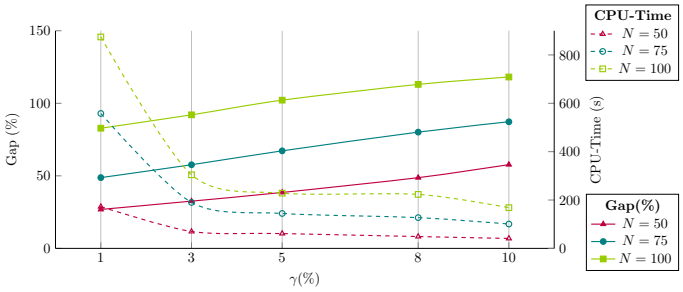
(e) $\rho = 90,\ T = 15$



(f) $\rho = 90,\ T = 30$

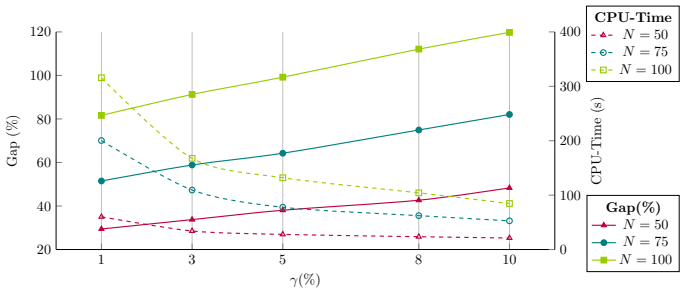


(g) $\rho = 120,\ T = 15$

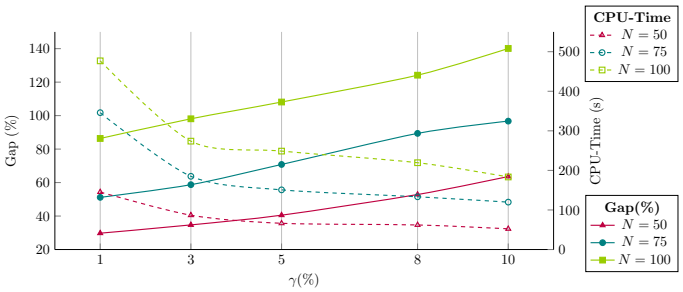


(h) $\rho = 120,\ T = 30$

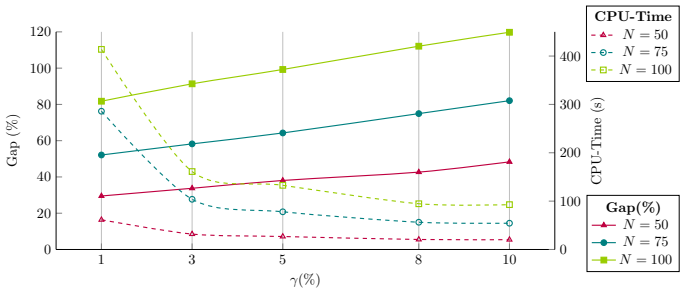
FIGURE A.1 – Résultats pour RFFO-C avec $\delta = 3$, $\sigma = 1$



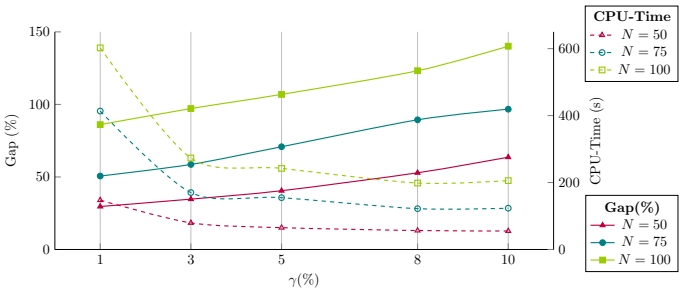
(a) $\rho = 30,\ T = 15$



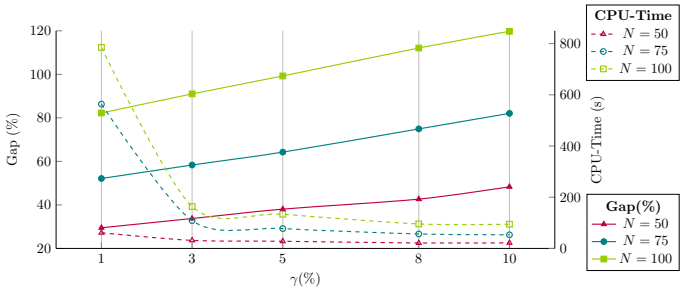
(b) $\rho = 30,\ T = 30$



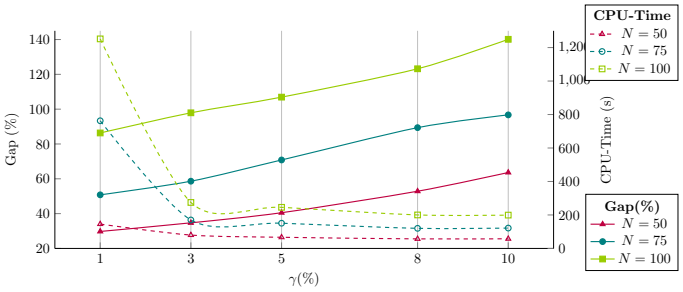
(c) $\rho = 60,\ T = 15$



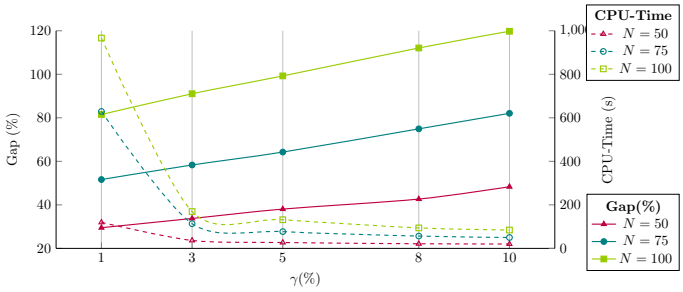
(d) $\rho = 60,\ T = 30$



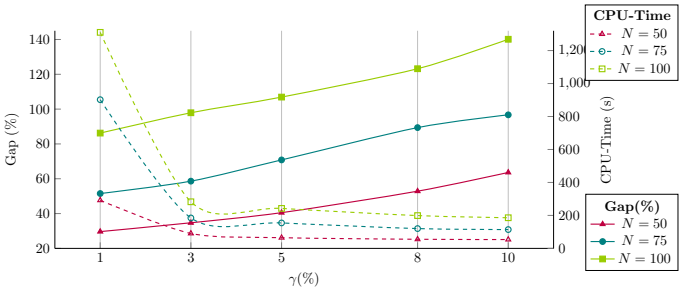
(e) $\rho = 90,\ T = 15$



(f) $\rho = 90,\ T = 30$

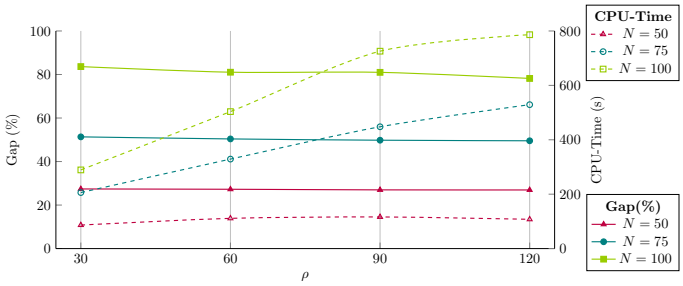


(g) $\rho = 120,\ T = 15$

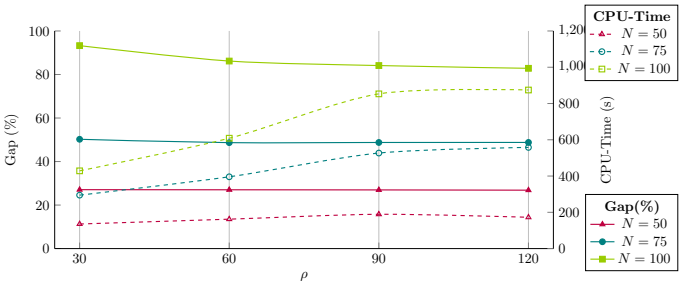


(h) $\rho = 120,\ T = 30$

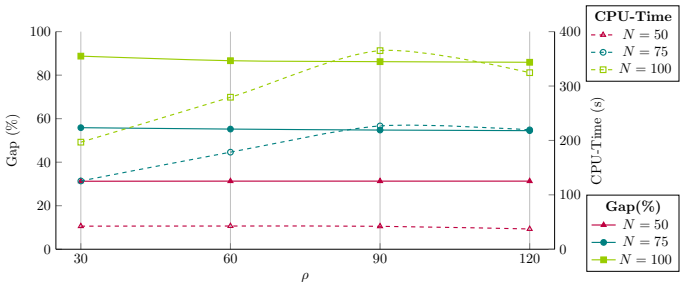
FIGURE A.2 – Résultats pour RFFO-C avec $\delta = 2$, $\sigma = 1$



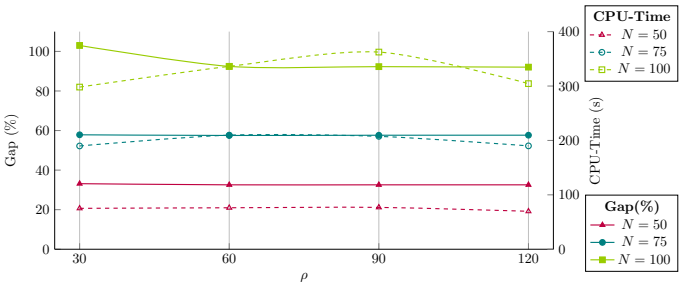
(a) $\gamma = 1,\ T = 15$



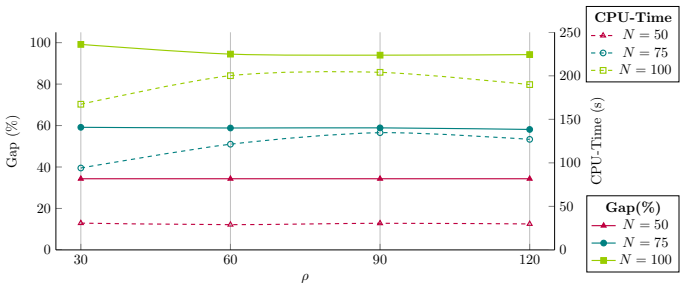
(b) $\gamma = 1,\ T = 30$



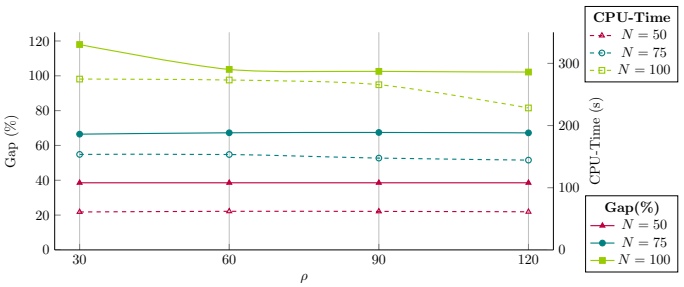
(c) $\gamma = 3,\ T = 15$



(d) $\gamma = 3,\ T = 30$

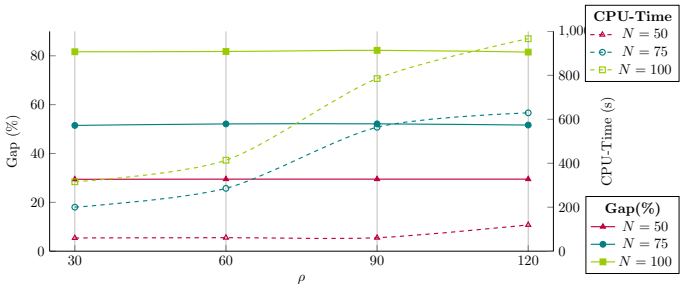


(e) $\gamma = 5,\ T = 15$

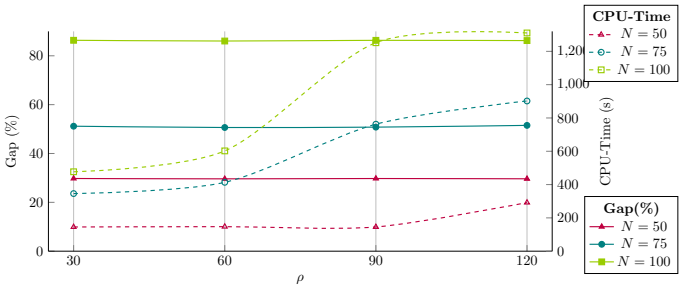


(f) $\gamma = 5,\ T = 30$

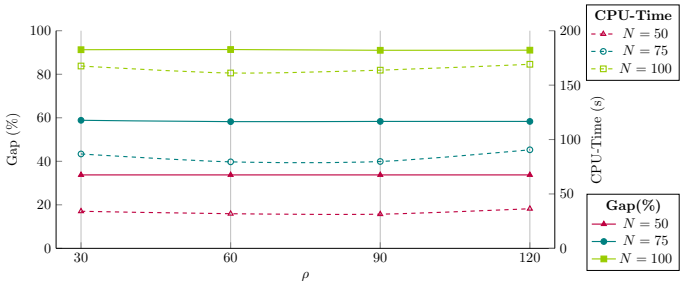
FIGURE A.3 – Résultats pour RFFO-C avec $\delta = 3$, $\sigma = 1$



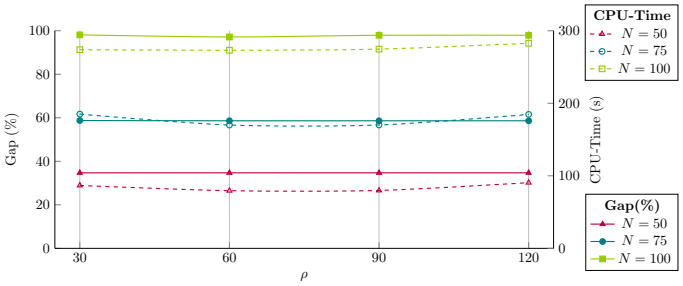
(a) $\gamma = 1,\ T = 15$



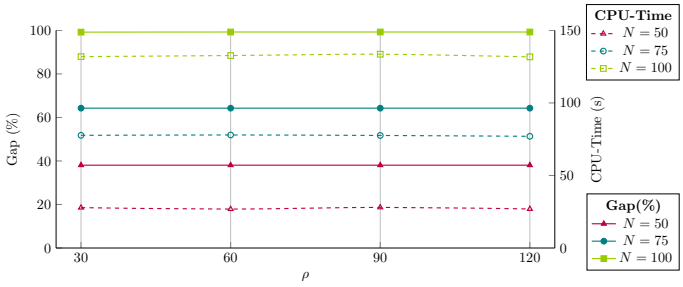
(b) $\gamma = 1,\ T = 30$



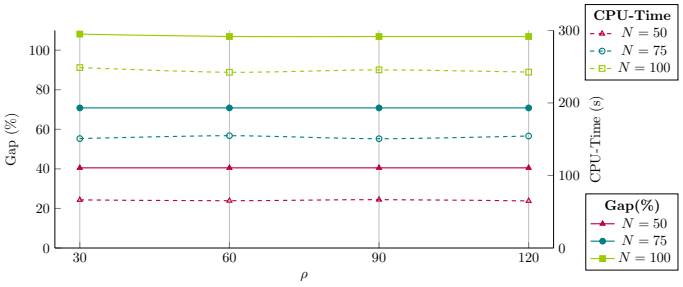
(c) $\gamma = 3,\ T = 15$



(d) $\gamma = 3,\ T = 30$



(e) $\gamma = 5,\ T = 15$



(f) $\gamma = 5,\ T = 30$

FIGURE A.4 – Résultats pour RFFO-C avec $\delta = 2$, $\sigma = 1$

A.4 Algorithme de l'heuristique en trois phase

Algorithme 8 3-phase

procédure 3-PHASE(stoppingCriteria)
    $\widetilde{x}^i_{mt} \leftarrow 0$,$\widetilde{y}^i_{mt} \leftarrow 0$ $ST^i_{mt} \leftarrow 0$, $Cap^{res}_{mt} \leftarrow \overline{C}_{mt}$
    $k \leftarrow 0$
    tant que stoppingCriteria = false **faire**
        $k \leftarrow k + 1$
        Résoudre Phase1($\widetilde{x}^i_{mt}$, $\widetilde{y}^i_{mt}$, Cap^{res}_{mt}, ST^i_{mt})
        pour m in $\mathcal{M}$ **faire**
            pour t in $\mathcal{T}$ **faire**
                $(\Lambda_{mt}, \widetilde{y}^i_{mt})$ = Résoudre Phase2($\bar{y}^i_{mt}$,m,t,$\alpha^{(k)}_{mt}$)
            fin pour
        fin pour
        Résoudre Phase3(Λ_{mt}, $\widetilde{y}^i_{mt}$)
        MiseAJour($\widetilde{x}^i_{mt}$,$\widetilde{y}^i_{mt}$,Cap^{res}_{mt},ST^i_{mt})
    fin tant que
fin procédure

BIBLIOGRAPHIE

[1] Prakash L ABAD : Optimal pricing and lot-sizing under conditions of perishability, finite production and partial backordering and lost sale. *European Journal of Operational Research*, 144(3):677–685, 2003.

[2] Nabil ABSI, Claudia ARCHETTI, Stéphane DAUZÈRE-PÉRÈS et Dominique FEILLET : A two-phase iterative heuristic approach for the production routing problem. *Transportation Science*, 49(4):784–795, 2015.

[3] Nabil ABSI, Boris DETIENNE et Stéphane DAUZÈRE-PÉRÈS : Heuristics for the multi-item capacitated lot-sizing problem with lost sales. *Computers & Operations Research*, 40(1):264–272, 2013.

[4] Nabil ABSI et Safia KEDAD-SIDHOUM : MIP-based heuristics for multi-item capacitated lot-sizing problem with setup times and shortage costs. *RAIRO - Operations Research*, 41(2):171–192, 2007.

[5] Nabil ABSI et Safia KEDAD-SIDHOUM : The multi-item capacitated lot-sizing problem with setup times and shortage costs. *European Journal of Operational Research*, 185(3):1351–1374, 2008.

[6] Nabil ABSI et Safia KEDAD-SIDHOUM : The multi-item capacitated lot-sizing problem with safety stocks and demand shortage costs. *Computers & Operations Research*, 36(11):2926–2936, 2009.

[7] Alok AGGARWAL et James K. PARK : Improved algorithms for economic lot size problems. *Operations Research*, 41(3):549–571, 1993.

[8] Kerem AKARTUNALI et Andrew J. MILLER : A heuristic approach for big bucket multi-level production planning problems. *European Journal of Operational Research*, 193(2):396–411, 2009.

[9] Ayse AKBALIK et Christophe RAPINE : Polynomial time algorithms for the constant capacitated single-item lot sizing problem with stepwise production cost. *Operations Research Letters*, 40(5):390–397, 2012.

[10] A. Alfieri, P. Brandimarte et S. D'Orazio : LP-based heuristics for the capacitated lot-sizing problem : The interaction of model formulation and solution algorithm. *International Journal of Production Research*, 40(2):441–458, 2002.

[11] Bernardo Almada-lobo, Diego Klabjan, Maria Antónia carravilla et José F. Oliveira : Single machine multi-product capacitated lot sizing with sequence-dependent setups. *International Journal of Production Research*, 45(20):4873–4894, 2007.

[12] Christian Almeder, Diego Klabjan, Renate Traxler et Bernardo Almada-Lobo : Lead time considerations for the multi-level capacitated lot-sizing problem. *European Journal of Operational Research*, 241(3):727–738, 2015.

[13] Fernanda F. Alves, Thiago H. Nogueira, Mauricio C. de Souza et Martín G. Ravetti : Approaches for the joint resolution of lot-sizing and scheduling with infeasibilities occurrences. *Computers & Industrial Engineering*, 155:107176, 2021.

[14] Enrico Angelelli, Renata Mansini et M. Grazia Speranza : Kernel search : A general heuristic for the multi-dimensional knapsack problem. *Computers & Operations Research*, 37(11):2017–2026, 2010.

[15] Jesica de Armas et Manuel Laguna : Parallel machine, capacitated lot-sizing and scheduling for the pipe-insulation industry. *International Journal of Production Research*, 58(3):800–817, 2020.

[16] Imre Barany, Tony Van Roy et Laurence A. Wolsey : Uncapacitated lot-sizing : The convex hull of solutions. *In Mathematical Programming at Oberwolfach II*, pages 32–43. Springer, 1984.

[17] Imre Barany, Tony J. Van Roy et Laurence A. Wolsey : Strong formulations for multi-item capacitated lot sizing. *Management Science*, 30(10):1255–1261, 1984.

[18] Gaetan Belvaux et Laurence A. Wolsey : Modelling practical lot-sizing problems as mixed-integer programs. *Management Science*, 47(7):993–1007, 2001.

[19] Patrizia Beraldi, Gianpaolo Ghiani, Antonio Grieco et Emanuela Guerriero : Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. *Computers & Operations Research*, 35(11):3644–3656, 2008.

[20] Purnima Bholowalia : Ebk-means : A clustering technique based on elbow method and k-means in wsn. *International Journal of Computer Applications*, 105(9), 2014.

[21] Gabriel R. Bitran et Horacio H. Yanasse : Computational complexity of the capacitated lot size problem. *Management Science*, 28(10):1174–1186, 1982.

[22] Nadjib Brahimi, Nabil Absi, Stéphane Dauzère-Pérès et Atle Nordli : Single-item dynamic lot-sizing problems : An updated survey. *European Journal of Operational Research*, 263(3):838–863, 2017.

[23] Nadjib Brahimi, Stéphane Dauzere-Peres, Najib M. Najid et Atle Nordli : Single item lot sizing problems. *European Journal of Operational Research*, 168 (1):1–16, 2006.

[24] Lisbeth Buschkühl, Florian Sahling, Stefan Helber et Horst Tempelmeier : Dynamic capacitated lot-sizing problems : a classification and review of solution approaches. *OR Spectrum*, 32(2):231–261, 2010.

[25] R. J. G. B. Campello et E. R. Hruschka : A fuzzy extension of the silhouette width criterion for cluster analysis. *Fuzzy Sets and Systems*, 157(21):2858–2875, 2006.

[26] Desiree M. Carvalho et Mariá C.V. Nascimento : Lagrangian heuristics for the capacitated multi-plant lot sizing problem with multiple periods and items. *Computers & Operations Research*, 71:137–148, 2016.

[27] Desiree M. Carvalho et Mariá C.V. Nascimento : Hybrid matheuristics to solve the integrated lot sizing and scheduling problem on parallel machines with sequence-dependent and non-triangular setup. *European Journal of Operational Research*, 296 (1):158–173, 2022.

[28] W. H. Chen et J. M. Thizy : Analysis of relaxations for the multi-item capacitated lot-sizing problem. *Annals of Operations Research*, 26(1):29–72, 1990.

[29] Chia-Shin Chung, Sin-Hoon Hum et Omer Kirca : The coordinated replenishment dynamic lot-sizing problem with quantity discounts. *European Journal of Operational Research*, 94(1):122–133, 1996.

[30] Alistair R. Clark, Reinaldo Morabito et Eli A. V. Toso : Production setup-sequencing and lot-sizing at an animal nutrition plant through atsp subtour elimination and patching. *Journal of Scheduling*, 13(2):111–121, 2010.

[31] Alistair Richard Clark et Vinicius Amaral Armentano : The application of valid inequalities to the multi-stage lot-sizing problem. *Computers & Operations Research*, 22(7):669–680, 1995.

[32] Miguel Constantino : Lower bounds in lot-sizing models : A polyhedral study. *Mathematics of Operations Research*, 23(1):101–118, 1998.

[33] Karina Copil, Martin Wörbelauer, Herbert Meyr et Horst Tempelmeier : Simultaneous lotsizing and scheduling problems : a classification and review of models. *OR Spectrum*, 39(1):1–64, 2017.

[34] IBM ILOG CPLEX : Cplex user's manual. *International Business Machines Corporation*, page 596, 2018.

[35] Stephane Dauzère-Péres et Jean-Bernard Lasserre : Integration of lotsizing and scheduling decisions in a job-shop. *European Journal of Operational Research*, 75(2):413–426, 1994.

[36] Zeger Degraeve et Raf Jans : A new dantzig-wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Operations Research*, 55(5):909–920, 2007.

[37] Martin Desrochers et Gilbert Laporte : Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27–36, 1991.

[38] Moustapha Diaby, Harish C. Bahl, Mark H. Karwan et Stanley Zionts : A lagrangean relaxation approach for very-large-scale capacitated lot-sizing. *Management Science*, 38(9):1329–1340, 1992.

[39] A. Drexl et A. Kimms : Lot sizing and scheduling — Survey and extensions. *European Journal of Operational Research*, 99(2):221–235, 1997.

[40] Gary D. Eppen et R. Kipp Martin : Solving Multi-Item Capacitated Lot-Sizing Problems Using Variable Redefinition. *Operations Research*, 1987.

[41] Awi Federgruen et Michal Tzur : A simple forward algorithm to solve general dynamic lot sizing models with n periods in 0 (n log n) or 0 (n) time. *Management Science*, 37(8):909–925, 1991.

[42] Diego Jacinto Fiorotto et Silvio Alexandre de Araujo : Reformulation and a Lagrangian heuristic for lot sizing problem on parallel machines. *Annals of Operations Research*, 217(1):213–231, 2014.

[43] Diego Jacinto Fiorotto, Silvio Alexandre de Araujo et Raf Jans : Hybrid methods for lot sizing on parallel machines. *Computers & Operations Research*, 63:136–148, 2015.

[44] Diego Jacinto Fiorotto, Jackeline del Carmen Huaccha Neyra et Silvio Alexandre de Araujo : Impact analysis of setup carryover and crossover on lot sizing problems. *International Journal of Production Research*, 58(20):6350–6369, 2020.

[45] Bernhard Fleischmann : The discrete lot-sizing and scheduling problem with sequence-dependent setup costs. *European Journal of Operational Research*, 75(2): 395–404, 1994.

[46] Bernhard Fleischmann et Herbert Meyr : The general lotsizing and scheduling problem. *Operations-Research-Spektrum*, 19(1):11–21, 1997.

[47] Michael Florian et Morton Klein : Deterministic production planning with concave costs and capacity constraints. *Management Science*, 18(1):12–20, 1971.

[48] J.-P. Gayon, G. Massonnet, C. Rapine et G. Stauffer : Constant approximation algorithms for the one warehouse multiple retailers problem with backlog or lost-sales. *European Journal of Operational Research*, 250(1):155–163, 2016.

[49] Jean-Philippe Gayon, Guillaume Massonnet, Christophe Rapine et Gautier Stauffer : Fast approximation algorithms for the one-warehouse multi-retailer problem under general cost structures and capacity constraints. *Mathematics of Operations Research*, 42(3):854–875, 2017.

[50] Ali Ghamari et Hadi Sahebi : The stochastic lot-sizing problem with lost sales : A chemical-Petrochemical case study. *Journal of Manufacturing Systems*, 44:53–64, 2017.

[51] C. Gicquel, M. Minoux et Y. Dallery : On the discrete lot-sizing and scheduling problem with sequence-dependent changeover times. *Operations Research Letters*, 37(1):32–36, 2009.

[52] Celine Gicquel, Michel Minoux, Yves Dallery et Jean-Marie Blondeau : A tight mip formulation for the discrete lot-sizing and scheduling problem with parallel resources. *In 2009 International Conference on Computers & Industrial Engineering*, pages 1–6. IEEE, 2009.

[53] Michel Gourgand, David Lemoine et Sylvie Norre : Metaheuristic for the capacitated lot-sizing problem : a software tool for MPS elaboration. *International Journal of Mathematics in Operational Research*, 2(6):724–747, 2010.

[54] G. Guastaroba, M. Savelsbergh et M.G. Speranza : Adaptive kernel search : A heuristic for solving mixed integer linear programs. *European Journal of Operational Research*, 263(3):789–804, 2017.

[55] Luis Guimarães, Diego Klabjan et Bernardo Almada-Lobo : Modeling lotsizing and scheduling problems with sequence dependent setups. *European Journal of Operational Research*, 239(3):644–662, 2014.

[56] Knut Haase : Capacitated lot-sizing with sequence dependent setup costs. *Operations-Research-Spektrum*, 18(1):51–59, 1996.

[57] Stefan Helber et Florian Sahling : A fix-and-optimize approach for the multilevel capacitated lot sizing problem. *International Journal of Production Economics*, 123(2):247–256, 2010.

[58] Faicel Hnaien et Hasan Murat Afsar : Robust single-item lot-sizing problems with discrete-scenario lead time. *International Journal of Production Economics*, 185:223–229, 2017.

[59] Y.-F. Hung, C.-C. Shih et C.-P. Chen : Evolutionary algorithms for production planning problems with setup decisions. *The Journal of the Operational Research Society*, 50(8):857, 1999.

[60] Ross J. W. James et Bernardo Almada-Lobo : Single and parallel machine capacitated lotsizing and scheduling : New iterative MIP-based neighborhood search heuristics. *Computers & Operations Research*, 38(12):1816–1825, 2011.

[61] Raf Jans et Zeger Degraeve : Meta-heuristics for dynamic lot sizing : A review and comparison of solution approaches. *European Journal of Operational Research*, 177(3):1855–1875, 2007.

[62] Raf Jans et Zeger Degraeve : Modeling industrial lot sizing problems : a review. *International Journal of Production Research*, 46(6):1619–1643, 2008.

[63] Waldemar Kaczmarczyk : Modelling set-up times overlapping two periods in the proportional lot-sizing problem with identical parallel machines. *Decision Making in Manufacturing and Services*, 7, 2013.

[64] Jangha Kang : Capacitated lot-sizing problem with sequence-dependent setup, setup carryover and setup crossover. *Processes*, 8(7):785, 2020.

[65] B. Karimi, S.M.T. Fatemi Ghomi et J.M. Wilson : The capacitated lot sizing problem : a review of models and algorithms. *Omega*, 31(5):365–378, 2003.

[66] András Kovács, Kenneth N. Brown et S. Armagan Tarim : An efficient MIP model for the capacitated lot-sizing and scheduling problem with sequence-dependent setups. *International Journal of Production Economics*, 118:282–291, 2009.

[67] Jakob Krarup et Ole Bilde : Plant location, set covering and economic lot size : An 0 (mn)-algorithm for structured problems. *In Numerische Methoden bei Optimierungsaufgaben Band 3*, pages 155–180. Birkhäuser, Basel, 1977.

[68] Mark Van der Laan, Katherine Pollard et Jennifer Bryan : A new partitioning around medoids algorithm. *Journal of Statistical Computation and Simulation*, 73 (8):575–584, 2003.

[69] Manuel Laguna : A heuristic for production scheduling and inventory control in the presence of sequence-dependent setup times. *IIE Transactions*, 31(2):125–134, 1999.

[70] Jan Christian Lang et Zuo-Jun Max Shen : Fix-and-optimize heuristics for capacitated lot-sizing with sequence-dependent setups and substitutions. *European Journal of Operational Research*, 214(3):595–605, 2011.

[71] François Larroche, Odile Bellenguez, Guillaume Massonnet et Benoît Le Badezet : A Genetic Algorithm for a capacitated lot-sizing problem with lost sales, overtimes, and safety stock constraints. 2021.

[72] Retsef Levi, Andrea Lodi et Maxim Sviridenko : Approximation algorithms for the capacitated multi-item lot-sizing problem via flow-cover inequalities. *Mathematics of Operations Research*, 33(2):461–474, 2008.

[73] Chung-Lun Li, Vernon Ning Hsu et Wen-Qiang Xiao : Dynamic lot sizing with batch ordering and truckload discounts. *Operations Research*, 52(4):639–654, 2004.

[74] X. Liu et Yl. Tu : Production planning with limited inventory capacity and allowed stockout. *International Journal of Production Economics*, 111(1):180–191, 2008.

[75] Stratégies Logistique : Pénurie de composants électroniques : les nouveaux risques des supply chains, 2021. URL `https://strategieslogistique.com/Penurie-de-composants,10949`.

[76] Marko Loparic, Yves Pochet et Laurence A. Wolsey : The uncapacitated lot-sizing problem with sales and safety stocks. *Mathematical Programming*, 89 (3):487–504, 2001.

[77] Johan Maes, John O. McClain et Luk N. Van Wassenhove : Multilevel capacitated lotsizing complexity and LP-based heuristics. *European Journal of Operational Research*, 53(2):131–148, 1991.

[78] Johan Maes et Luk Van Wassenhove : Multi-item single-level capacitated dynamic lot-sizing heuristics : A general review. *The Journal of the Operational Research Society*, 39(11):991–1004, 1988.

[79] Alan S. Manne : Programming of economic lot sizes. *Management Science*, 4 (2):115–135, 1958.

[80] Geraldo R. Mateus, Martín G. Ravetti, Maurício C. de Souza et Taís M. Valeriano : Capacitated lot sizing and sequence dependent setup scheduling : an iterative approach for integration. *Journal of Scheduling*, 13(3):245–259, 2010.

[81] T. Melo, S. Nickel et F. Saldanha-da Gama : An LP-rounding heuristic to solve a multi-period facility relocation problem. 2009.

[82] António Aroso Menezes, Alistair Clark et Bernardo Almada-Lobo : Capacitated lot-sizing and scheduling with sequence-dependent, period-overlapping and non-triangular setups. *Journal of Scheduling*, 14(2):209–219, 2011.

[83] C. Mercé et G. Fontan : MIP-based heuristics for capacitated lotsizing problems. *International Journal of Production Economics*, 85(1):97–111, 2003.

[84] H. Meyr : Simultaneous lotsizing and scheduling by combining local search with dual reoptimization. *European Journal of Operational Research*, 120(2):311–326, 2000.

[85] Herbert Meyr : Simultaneous lotsizing and scheduling on parallel machines. *European Journal of Operational Research*, 139(2):277–292, 2002.

[86] Harvey H. Millar et Minzhu Yang : Lagrangian heuristics for the capacitated multi-item lot-sizing problem with backordering. *International Journal of Production Economics*, 34(1):1–15, 1994.

[87] Andrew J Miller, George L Nemhauser et Martin W. P Savelsbergh : On the capacitated lot-sizing and continuous 0–1 knapsack polyhedra. *European Journal of Operational Research*, 125(2):298–315, 2000.

[88] C. E. Miller, A. W. Tucker et R. A. Zemlin : Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329, 1960.

[89] Mariá C.V. Nascimento, Mauricio G.C. Resende et Franklina M.B. Toledo : Grasp heuristic with path-relinking for the multi-plant capacitated lot sizing problem. *European Journal of Operational Research*, 200(3):747–754, 2010.

[90] National School of Business : Electronic Supply Chain Management, 2015. URL `https://fr.slideshare.net/hari3hhh/e-scm-45891860`.

[91] Yves Pochet et Laurence A. Wolsey : Lot-size models with backlogging : Strong reformulations and cutting planes. *Mathematical Programming*, 40(1):317–335, 1988.

[92] Yves Pochet et Laurence A. Wolsey : Lot-sizing with constant batches : Formulation and valid inequalities. *Mathematics of Operations Research*, 18(4):767–785, 1993.

[93] Daniel Quadt et Heinrich Kuhn : Capacitated lot-sizing with extensions : a review. *4OR*, 6(1):61–83, 2008.

[94] E. Powell Robinson et F. Barry Lawrence : Coordinated capacitated lot-sizing problem with dynamic demand : A lagrangian heuristic. *Decision Sciences*, 35 (1):25–53, 2004.

[95] Powell Robinson, Arunachalam Narayanan et Funda Sahin : Coordinated deterministic dynamic demand lot-sizing problem : A review of models and algorithms. *Omega*, 37(1):3–15, 2009.

[96] Paul A. Rubin : User Cuts versus Lazy Constraints, 2012. URL `https://orinanobworld.blogspot.com/2012/08/user-cuts-versus-lazy-constraints.html`.

[97] Florian Sahling, Lisbeth Buschkühl, Horst Tempelmeier et Stefan Helber : Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic. *Computers & Operations Research*, 36(9):2546–2553, 2009.

[98] Richard A. Sandbothe et Gerald L. Thompson : A forward algorithm for the capacitated lot size model with stockouts. *Operations Research*, 38(3):474–486, 1990.

[99] Richard A. Sandbothe et Gerald L. Thompson : Decision horizons for the capacitated lot size model with inventory bounds and stockouts. *Computers & Operations Research*, 20(5):455–465, 1993.

[100] Congming Shi, Bingtao Wei, Shoulin Wei, Wen Wang, Hai Liu et Jialei Liu : A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm. *EURASIP Journal on Wireless Communications and Networking*, 2021(1):31, 2021.

[101] Riyaz Sikora : A genetic algorithm for integrating lot-sizing and sequencing in scheduling a capacitated flow line. *Computers & Industrial Engineering*, 30(4):969–981, 1996.

[102] Charles R. Sox et Yubo Gao : The capacitated lot sizing problem with setup carry-over. *IIE Transactions*, 31(2):173–181, 1999.

[103] Horst Tempelmeier et Matthias Derstroff : A lagrangean-based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. *Management Science*, 42(5):738–757, 1996.

[104] Claudio Fabiano Motta Toledo, Márcio da Silva Arantes, Marcelo Yukio Bressan Hossomi, Paulo Morelato França et Kerem Akartunali : A relax-and-fix with fix-and-optimize heuristic applied to multi-level lot-sizing problems. *Journal of Heuristics*, 21(5):687–717, 2015.

[105] Franklina Maria Bragion Toledo et Vinícius Amaral Armentano : A Lagrangian-based heuristic for the capacitated lot-sizing problem in parallel machines. *European Journal of Operational Research*, 175(2):1070–1083, 2006.

[106] Eli A. V. Toso, Reinaldo Morabito et Alistair R. Clark : Lot sizing and sequencing optimisation at an animal-feed plant. *Computers & Industrial Engineering*, 57 (3):813–821, 2009.

[107] William W. Trigeiro : A Dual-Cost Heuristic For The Capacitated Lot Sizing Problem. *IIE Transactions*, 19(1):67–72, 1987.

[108] William W. Trigeiro, L. Joseph Thomas et John O. McClain : Capacitated lot sizing with setup times. *Management Science*, 35(3):353–366, 1989.

[109] Aykut Melih Turhan et Bilge Bilgen : A hybrid fix-and-optimize and simulated annealing approaches for nurse rostering problem. *Computers & Industrial Engineering*, 145:106531, 2020.

[110] C. P. M. van Hoesel et A. P. M. Wagelmans : An o (t 3) algorithm for the economic lot-sizing problem with constant capacities. *Management Science*, 42(1): 142–150, 1996.

[111] Mathieu Van Vyve : Algorithms for single-item lot-sizing problems with constant batch size. *Mathematics of Operations Research*, 32(3):594–613, 2007.

[112] François Vanderbeck et Martin W. P. Savelsbergh : A generic view of Dantzig–Wolfe decomposition in mixed integer programming. *Operations Research Letters*, 34(3):296–306, 2006.

[113] Albert Wagelmans, Stan Van Hoesel et Antoon Kolen : Economic lot sizing : An o (n log n) algorithm that runs in linear time in the wagner-whitin case. *Operations Research*, 40(1-supplement-1):S145–S156, 1992.

[114] Harvey M. Wagner et Thomson M. Whitin : Dynamic version of the economic lot size model. *Management Science*, 5(1):89–96, 1958.

[115] Laurence A. Wolsey : *Integer Programming*. John Wiley & Sons, 2020.

[116] Jing Xiao, Huasheng Yang, Canrong Zhang, Li Zheng et Jatinder N.D. Gupta : A hybrid Lagrangian-simulated annealing-based heuristic for the parallel-machine

capacitated lot-sizing and scheduling problem with sequence-dependent setup times. *Computers I& Operations Research*, 63:72–82, 2015.

[117] Jing Xiao, Canrong Zhang, Li Zheng et Jatinder N. D. Gupta : MIP-based fix-and-optimise algorithms for the parallel machine capacitated lot-sizing and scheduling problem. *International Journal of Production Research*, 51(16):5011–5028, 2013.

[118] Willard I. Zangwill : A deterministic multi-period production scheduling model with backlogging. *Management Science*, 13(1):105–119, 1966.

[119] L Özdamar et G Barbarosoğlu : Hybrid heuristics for the multi-stage capacitated lot sizing and loading problem. *Journal of the Operational Research Society*, 50(8):810–825, 1999.

[120] Linet Özdamar et Şevket İlker 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.

[121] Linet Özdamar et Mehmet Ali Bozyel : The capacitated lot sizing problem with overtime decisions and setup times. *IIE Transactions*, 32(11):1043–1057, 2000.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Optimisation d'un plan de production de produits périssables dans un contexte multi-ressources à capacité finie

Mot clés : Planification de la production, agroalimentaire, lot-sizing, programmation linéaire en nombres entiers, heuristique

Résumé : La planification de la production est une étape importante dans de nombreux secteurs industriels. Définir la production pour répondre à la demande, s'assurer d'avoir un niveau de stock adéquat et maîtriser sa capacité de production sont des tâches complexes. L'utilisation d'outils mathématiques pour optimiser les prises de décision dans la planification de la production est donc totalement pertinent pour les industriels. Dans cette thèse, nous nous intéressons à des problèmes de planification de la production dans le domaine de l'agroalimentaire rencontrés par VIF, une entreprise spécialisée dans la création de logiciel pour ce secteur. Nous nous concentrons sur la modélisation et la résolution du problème de lot-sizing sous différentes contraintes : capacité finie, rupture sur la demande, séquence de production, machines parallèles, etc. Nous proposons plusieurs modélisations mathématiques et des heuristiques pour résoudre ce problème. La première heuristique se base sur une décomposition du problème en sous-problèmes résolus de façon itérative. Nous proposons ensuite un algorithme de pré-traitement basé sur une approche de regroupement pour diminuer la taille du problème. Finalement, nous proposons une approche de résolution en trois phases dans laquelle les décisions relatives à notre problème sont séparées et résolues indépendamment.

Title: Optimization of a production plan for perishable products in a multi-resources context with finite capacity

Keywords: Production planning, food industry, lot-sizing, mixed integer program, heuristic

Abstract: Production planning is an important step in many industrial sectors. Defining production to meet demand, ensuring that the level of stock is adequate and controlling your production capacity are complex tasks. The use of mathematical tools to optimize decision-making in production planning is therefore totally relevant for planners. In this thesis, we focus on production planning problems in the food industry encountered by VIF, a company specializing in the creation of software for this sector. We focus on modeling and solving the lot-sizing problem with different constraints: finite capacity, lost sales, production sequence, parallel machines, etc. We propose different mathematical models and heuristics to solve this problem. The first heuristic is based on a decomposition of the problem into subproblems which are solved iteratively. We then propose a preprocessing algorithm based on a clustering approach to reduce the size of the problem. Finally, we propose a three-phase solution approach in which the decisions of our problem are separated and solved independently.