Thèse de Doctorat de l'Université Pierre et Marie Curie

Présentée par : Anna ROBERT

Pour obtenir le grade de
Docteur de l'Université Pierre et Marie Curie

Spécialité : Informatique

Optimisation des batches de production

Soutenue le 14 Septembre 2007

Devant le jury composé de :

Directeur de thèse :	Philippe Chrétienne	Professeur, Université Pierre et Marie Curie, Paris 6
Co-Directeurs :	Claude Le Pape	Directeur R&D Manufacturing, ILOG S.A.
	Francis Sourd	Chargé de Recherche, CNRS
Rapporteurs :	Marie-Claude Portmann	Professeur, École des Mines de Nancy
	Stéphane Dauzère-Pérès	Professeur, École des Mines de Saint Etienne
Examinateurs :	Gerd Finke	Professeur, Université Joseph Fourier, Grenoble
	Michel Minoux	Professeur, Université Pierre et Marie Curie, Paris 6
	Eric Pinson	Professeur, Université Catholique de l'Ouest, Angers

À Sédou,

Remerciements

C'est Philippe Chrétienne qui, en 2002, m'ouvrit les portes de la Recherche Opérationnelle, traçant au tableau des cercles qu'il reliait par des traits, et nommant *graphes*, ces figures mathématiques et ludiques. Mon goût pour cette discipline ne devait alors plus me quitter. Toute ma gratitude va donc vers Philippe, qui a supervisé mes travaux de doctorat, en confiant à Francis Sourd la tâche d'un encadrement plus étroit. Rencontré aussi il y a 5 ans, Francis m'a montré et transmis rigueur, méthode, connaissances, en de multiples occasions. Ce doctorat est un aboutissement, du moins un moment majeur, significatif, chargé de symboles, au sein de la relation que nous avons eue jusqu'ici. Travailler à ses côtés fut pour moi un immense plaisir. En 2004, Francis m'a alors mise entre les mains de Claude Le Pape, qui devint mon troisième directeur de thèse. C'est pour moi un honneur d'avoir effectué mon doctorat sous sa responsabilité. L'honnêté intellectuelle et la créativité scientifique dont Claude fait preuve quotidiennement sont autant de facteurs qui rendent le travail agréable et efficace. Je remercie ces trois personnes pour leur profonde humanité et la confiance qu'ils m'ont témoignée dès les premiers temps de nos collaborations.

Mon profond respect et ma gratitude se dirigent aussi vers Marie-Claude Portmann et Stéphane Dauzère-Pérès, qui ont rapporté ma thèse, enrichissant, chacun à leur manière, mes travaux, de remarques et commentaires. Je remercie en particulier Stéphane pour les conseils qu'il m'a prodigués lors d'échanges privilégiés que nous avons pu avoir au cours de la dernière année de mon doctorat.

Je remercie Gerd Finke, Eric Pinson et Michel Minoux d'avoir accepter de prendre part au jury de ma thèse. Leur renommée dans le milieu de la Recherche Opérationnelle confère à ces positions d'examinateurs une grande qualité, et m'honore beaucoup. Je tiens notamment à témoigner ma gratitude à Michel Minoux, qui a suivi mon évolution depuis 2002 jusqu'à aujourd'hui, et avec qui j'ai eu le plaisir de collaborer dans le milieu enseignant.

Pour sa bonne humeur, son esprit de corps et la disponibilité de chacun de ses membres, je veux remercier toute l'équipe d'ILOG Plant PowerOps, qui, malgré un contexte parfois délicat, sait garder la rage et la foi en ses qualités et son potentiel. Merci à Wim, Frédéric, Filippo, Daniel, Julien, Ali, Emmanuel, Xavier et Dominique. Ce fut pour moi une excellente expérience que celle

d'évoluer à ILOG aux côtés de nombreuses autres personnes (et personnages !) : Virginie, Viu, Laurent, Jérôme, Georges-Henri, Michel, Philippe, Bruno, Vincent, Sylvain.

Je suis revenue, pour mes derniers mois de thèse, dans mon environnement d'origine, là où tout avait commencé, c'est-à-dire au Laboratoire d'Informatique de Paris 6 (mais dans de nouveaux locaux !). Avec beaucoup de plaisir j'y ai retrouvé les membres permanents et les thésards (ainsi que ceux qui ne sont ni l'un ni l'autre) d'une équipe qui porte un nom inoubliable : DESIR([r]emplaçables ?) ! Pour la gaieté, l'écoute et la finesse d'esprit qui vous caractérisent, je tiens à tous vous remercier : Safia, Claire, Alix, Caroline, Lucie, Nina, Fanny, Pierre, Emmanuel, Patrice, Jean-Yves, Olivier, Christophe, Paul, Nico, Loux, Gildas, Clément ("Play-mo-bil, en avant les histoires !"), Sergio, Jox. Et les anciens : Yasmin, Nathalie, Cathy, Bénédicte, Yann, Nabil.

Ils contribuent à l'adoucissement de mes mœurs, à raison d'une écoute quotidienne d'au moins 4h, dans le désordre, je remercie Therion, Lacuna Coil, Jeff Buckley, The Old Dead Tree, Wolfgang Amadeus Mozart, Moonspell, Keith Jarret, Kamelot, Piotr Ilitch Tchaikovski, Lou Reed, Epica, Within Temptation, Sergueï Prokofiev, Georges Brassens, Apocalyptica, Pearl Jam, Nightwish, Alan Stivell, After Forever, Lunatica, Antonio Vivaldi, Wim Mertens, Rammstein.

Pour nos moments de détente et de discussion partagés depuis de nombreuses années, je veux remercier mes amis les plus chers : Christelle, AnCé, LN, Del, Chun, Daï, Noon, C, Ivan, Guet, Charles et Tiflopin. Je souhaite laisser un message d'affection à Sébastien M. qui m'a accompagnée pendant les deux premières années de ma thèse. Pour être auprès de moi aujourd'hui, j'exprime ma plus grande tendresse à Jean-Sébastien G., je te remercie d'être là.

Je fais un gros bisou à mes petites sœurs chéries : Diane et Léa, vous êtes les oursons les plus oursons du monde !

Pour ce que vous m'avez apporté, et m'apportez chaque jour, ce que vous avez fait pour moi et *de* moi, pour l'amour que vous m'exprimez en chaque occasion, je souhaite vous témoigner, Maman, Papa, ma plus grande gratitude, et mon plus profond respect. Je vous aime très fort...

Enfin, pour me soutenir depuis toujours, pour son écoute et son intelligence, pour ses forces et sa sensibilité, je remercie ma grande sœur, Jeanie, qui en plus de tout ce que je lui dois, m'a fait l'honneur de rédiger le prologue de cette thèse, j'en suis très fière et très émue. Sédou, tu sais comme je tiens à toi, tu es comme mon double, je t'aime.

Prologue, *par ma sœur Jeanie.*

Avant de pénétrer dans le champ, très spécifique, qui est celui de notre travail, nous souhaitons formuler ici quelques brèves remarques relatives à la manière dont nous concevons la recherche scientifique, ainsi que son rôle et sa place en ce monde tissé non seulement de complexités, mais d'énigmes sans cesse renaissantes.

Il fut un temps où était nommé savant l'homme versé dans tous les champs du savoir sans exception, qu'il s'agît des sciences, des lettres ou de la philosophie. Aujourd'hui, l'accroissement des connaissances a contraint le chercheur à se spécialiser, c'est-à-dire à restreindre son champ d'investigation à une discipline ou à un domaine déterminé du savoir, voire à un problème unique. Le chercheur, sans doute, gagne alors en précision et en possibilité d'approfondissement de sa recherche. Mais il court également le risque, les liens aux autres champs du savoir devenant plus lâches, de se couper d'une pluridisciplinarité souvent fort féconde. Toutefois, il existe un lien fondamental, présent de tout temps, et qui ne saurait jamais être rompu sans que la recherche elle-même ne s'arrête : c'est la relation intrinsèque entre le travail du chercheur et le monde vivant et habité par l'homme. En ce sens, la connaissance n'est jamais entièrement "désintéressée", n'ayant pour fin qu'elle-même : elle enveloppe toujours des causes et des conséquences, des moyens et des fins, qui touchent à l'existence concrète, matérielle et morale, des hommes. Si nous tenons à rappeler cette évidence, c'est qu'elle ne peut, au sein d'un travail de recherche à ce point spécialisé, qu'apparaître soit en filigrane, soit très localement. Elle confère pourtant, nous semble-t-il, son sens et ses enjeux les plus profonds à tout travail authentique de recherche.

Avec Socrate, qui fut l'un des premiers en Europe à penser la science et son exigence de vérité, nous pouvons définir la recherche comme tout ensemble aspiration au savoir et quête de la sagesse : la connaissance n'a de sens que si elle aboutit à une amélioration de la condition, intérieure et extérieure, des hommes. Un corollaire de cette définition de la connaissance est que la recherche théorique doit toujours trouver, même lointainement, parfois à très long terme, des applications dans le monde, matériel ou moral, des hommes : après le chemin qui mène vers les cimes de la connaissance pure, le chercheur se doit, selon la célèbre Allégorie platonicienne — *l'Allégorie de la Caverne*, au livre VII de La République —, de *redescendre* vers le monde des hommes, afin que la pensée trouve sa réalisation dans le concret de la matière, dit *obscur* en tant qu'il n'est pas encore organisé par la pensée. Cette pensée lui confère alors son ordre —

réfléchi et rationnel — et sa valeur — éthique. Le vrai et le bien, la connaissance et la moralité, le savoir et la justice sont en effet, par Socrate comme par son disciple, éprouvés et conçus comme indissociables. Enfin, cette mise en acte, ou application de l'esprit, est éminemment le fait des activités humaines qui sont, selon la classification grecque, non seulement la pensée — *noesis* —, mais l'action — *praxis* — et la production — *poiesis*.

Au sens ancien, la production signifie le fait, pour l'homme, de fabriquer ou de créer des objets artificiels, soit esthétiquement beaux, soit pratiquement utiles. Or cette dimension de la production peut paraître bien modeste, voire inférieure, relativement aux activités théoriques et morales de l'homme. Pourtant, comme a pu le souligner Hannah Arendt, le fait de produire, qui définit pour ainsi dire le travail humain, est, en tant que tel, essentiel à l'homme, et constitue la condition d'existence d'un monde véritablement *habité* par l'homme, c'est-à-dire constitué d'objets dont il est l'origine, et qui prolonge et complète les créations de la nature. L'activité de production touche en fait aux conditions mêmes de la vie, humaine mais également terrestre en général, au point d'être fondatrice du rapport de l'homme à la nature, ainsi que des échanges et de la coopération des humains entre eux. Toutefois, selon Arendt, travail et production ne concernent que la dimension biologique de la vie humaine, la simple survie pour ainsi dire.

Or les dimensions éthique et politique, loin d'en être absentes, pénètrent ce champ du travail et de la production ; et ce peut-être avec d'autant plus d'acuité aujourd'hui, que la production désigne non plus la tâche d'un seul homme, mais un processus multiple et complexe, véritable chaîne d'activités humaines. En outre, ce processus connaît un essor et des mutations sans précédent, dont l'homme est certes l'origine créatrice, mais auxquels il doit également, par un effet de retour, s'adapter. Cette évolution, et les transformations souvent très rapides, parfois même imprévues, qu'elle engendre dans le champ du travail et des échanges humains, imposent donc à l'homme de répondre à de nouvelles exigences, à de nouveaux problèmes, à de nouveaux défis. Et ceux-ci sont cruciaux non seulement pour l'efficacité des processus de production, mais pour le respect et la préservation des normes écologiques et éthiques vitales à notre monde. La rentabilité ne peut en effet être réelle que si elle s'accompagne soit du maintien, soit du progrès des conditions de la vie humaine, concernant notamment l'environnement, mais aussi la durée du temps de travail ou la qualité des conditions où il s'effectue. C'est d'ailleurs ceci qui fait de la production un ensemble de *processus* — décidés, mis en œuvre, et maîtrisés, autant que possible, par l'homme — et non un pur *mécanisme* — aveugle parce que dénué de pensée et d'intention. Ainsi la rationalisation, qui est l'empreinte de la pensée dans la matière, ne peut se défaire de sa dimension réflexive et éthique sans devenir une succession aveugle d'actes et de procédures, alors coupée de son but final : l'amélioration des conditions physiques et morales de l'existence humaine.

Face à cette situation nouvelle, la science, ici la Recherche Opérationnelle, et ses outils informatique et mathématique, a un rôle essentiel à jouer. Elle donne en effet à l'homme, pour

ainsi dire, la possibilité d'étendre les bornes de son entendement, en permettant la figuration (représentation ?), l'analyse et l'anticipation de situations d'une très haute complexité. En outre, elle lui permet de prendre en compte une diversité de critères, et de les comparer, afin de l'aider à opter pour la décision la meilleure, face à telle situation donnée. Le "meilleur", qui a toujours été la fin des recherches morales et économiques de l'homme, apparaît donc ici comme un but qui est un sommet, une excellence, non pas fixé une fois pour toutes, mais variable selon les circonstances bigarrées et contingentes du *champ des affaires humaines*. Nous reprenons cette dernière expression à Aristote, qui définissait le meilleur choix, ou décision juste, comme variable selon une multitude de circonstances et de critères, tels que, par exemple, le temps ou le moment opportun — *kairos* —, ou encore les individus sujets et objets de l'action ; de sorte que le juste choix et l'action qui le suit se doivent de s'adapter à un champ qui contient, en dépit de ses régularités, une part non négligeable de contingence, de hasard et d'imprévisibilité : le domaine des choix et des actions des hommes. Par conséquent, la meilleure décision, loin d'être contingente au sens où elle pourrait être autre qu'elle n'est, demeure, pour chaque situation donnée, unique ; et la tâche des hommes, par un commun effort, est de la découvrir. Dans cette perspective, notre travail se donne pour fin d'apporter une modeste contribution aux recherches visant la découverte de cette décision juste.

Jeanie Robert.

Introduction générale

Environnement de travail

C'est dans le contexte du développement du logiciel APS (*Advanced Planning and Scheduling*) *Plant PowerOps* (PPO), de la société ILOG S.A., que s'est effectuée la présente thèse. Durant ces trois années, c'est en effet essentiellement au sein de l'équipe de Recherche et Développement du département "Optimisation" d'ILOG que nous avons conduit notre étude et développé nos travaux. La thèse s'est déroulée dans le cadre d'un contrat CIFRE entre la société d'édition de logiciels ILOG S.A. et l'Université Pierre et Marie Curie. Les problèmes rencontrés au cours du développement de l'application PPO, relatifs à la problématique générale d'optimisation de la production, ont conduit ILOG à entreprendre, par le biais de ce contrat CIFRE, une étude en profondeur afin d'identifier, analyser et résoudre de nouveaux problèmes qui pourraient être la cause de lacunes constatées dans les logiciels visant à répondre à ce type de questions.

Problématique générale

De nos jours, les problèmes liés à la gestion et l'optimisation de la production sont cruciaux pour les industriels. En effet, d'une part, ils recherchent l'accroissement de leur productivité tout en minimisant les coûts afférents (mise en œuvre, fabrication, utilisation des ressources, stockage...), d'autre part, de nombreuses exigences métier, sanitaires ou commerciales se doivent d'être respectées. Pour cela, il est nécessaire de concevoir des outils d'optimisation et d'aide à la décision perfectionnés, adaptés aux difficultés que rencontrent les industriels vis-à-vis de l'organisation de leur production. Dans ce contexte, on distingue principalement deux catégories d'industrie : l'industrie de fabrication et l'industrie de *process*. Dans la première, les usines comportent en général des chaînes d'assemblage de pièces, et visent à fabriquer des articles dont il est possible d'isoler une unité sans avoir à peser ou évaluer un volume (construction d'automobiles, de composants électroniques, d'équipements mécaniques). La seconde, quant à elle, fait appel à des procédés de transformation pouvant être très complexes et délicats. L'élaboration des produits finis nécessite la manipulation de matières premières et de matériaux intermédiaires, respectant des protocoles précis. Les industries chimiques, pharmaceutiques ou agroalimentaires appartiennent à cette catégorie. En outre, en gestion de production, plusieurs niveaux de décisions peuvent être examinés : points de vue stratégique (long terme), tactique (moyen terme) et

opérationnel (court terme).

L'industrie de *process* est le domaine d'application pour lequel est spécialisé le logiciel PPO. Son rôle est d'optimiser la planification globale (décisions tactiques) et l'ordonnancement détaillé (décisions opérationnelles) de la production dans des situations complexes et fortement contraintes. Dans le contexte de l'industrie de *process*, la production est en général réalisée par batch. Un batch de taille donnée définit un ensemble d'activités de production dont les durées et quantités de produits transformés dépendent de la taille du batch. Ces activités s'exécutent alors sur des ressources réalisant physiquement la production.

La question posée dans cette thèse est celle de l'optimisation de la taille des batches de production, elle intervient entre les niveaux tactique de planification et opérationnel d'ordonnancement. En effet, ce problème, résolu de manière heuristique par les APS du marché, concerne des décisions soumises à des contraintes métier difficiles, et en général cruciales pour l'optimisation globale de la chaîne de production. Le dimensionnement des batches est critique puisqu'il s'agit de trouver un compromis entre la création de trop gros batches, entraînant des délais, et celle de trop petits batches, induisant des surcoûts de production. L'objectif de cette thèse est multiple. Dans un premier temps, une définition formelle du problème sous-tendu par la question permettra de dégager un cadre d'étude précis et de situer le problème dans son contexte, puis d'en effectuer une analyse théorique. Dans un second temps, l'élaboration et l'implantation de méthodes de résolution s'intégrant dans le logiciel PPO permettront de traiter le problème et ses variantes spécifiques aux cas pratiques, dans le but de valider, d'une part, la problématique, et d'autre part, les approches de résolution proposées. Nous montrerons que l'optimisation des batches permet effectivement d'obtenir, en moyenne, de meilleurs ordonnancements de la production.

Organisation du manuscrit

Dans un premier chapitre, nous définissons précisément le contexte général dans lequel se positionne notre étude. Nous introduisons la problématique de gestion de la chaîne d'approvisionnement (*Supply Chain Management*) dans laquelle s'inscrivent nos travaux, et présentons les principaux modèles et logiciels d'aide à la décision relatifs à ce domaine. Nous dressons un panorama général des problèmes de planification, d'ordonnancement et de leur coordination. Nous mettons alors en évidence que la connexion entre ces deux niveaux de prises de décisions est souvent ardue à mettre en œuvre, en particulier dans une approche que l'on souhaite la plus générique possible, ce qui est en général le cas des APS. Enfin, nous définissons les notions majeures relatives aux chaînes de production et les spécificités particulières à l'industrie de *process* que nous considérons dans cette thèse. Nous concluons cette partie en dégageant les motivations et objectifs de la thèse.

Le second chapitre est consacré à la définition du problème d'optimisation qui est au cœur de nos travaux. Se situant entre la planification et l'ordonnancement, il vise à définir l'ensemble des batches qui réaliseront physiquement la production globale, nous le dénommons dans la suite par le terme de *batching*. Nous donnons tout d'abord une définition informelle du problème et montrons les intérêts que présente une solution *optimisée*, ainsi que la difficulté intrinsèque à la prise de ces décisions. Dans un second temps, nous formalisons précisément le problème global résolu par PPO, d'une manière d'abord générale, puis dans le contexte de l'approche novatrice tri-modulaire de résolution employée par le logiciel (planification, *batching*, ordonnancement), cela permet de donner le cadre précis du problème original de *batching* traité dans cette thèse. Nous montrons ensuite les liens qui existent entre notre problème ainsi posé et les problèmes déjà étudiés d'ordonnancement par batches.

Dans le troisième chapitre de cette thèse, nous proposons une étude de la complexité de différents sous-cas du problème de *batching*. La question de la faisabilité est d'abord examinée. Puis, nous divisons l'étude en deux parties, en fonction du critère d'optimisation considéré. La dernière partie est dédiée à la preuve de complexité d'un sous-cas particulièrement intéressant du point de vue de cette étude.

Le quatrième chapitre se concentre sur la résolution du problème *"cœur"* de *batching*, qui en est la variante élémentaire. Dans un premier temps, une méthode exacte de résolution est proposée : nous présentons un algorithme de programmation dynamique permettant de traiter un cas particulier du problème *"cœur"*. Dans un second temps, nous formulons un modèle linéaire, noté $\mathcal{P}$, pour le cas général. La fonction objectif, quadratique dans le problème général, est ici linéarisée par une approximation. Nous proposons enfin une courte étude expérimentale permettant de comparer les deux approches.

Un certain nombre d'extensions du problème *"cœur"* de *batching* sont proposées dans le Chapitre 5. Chacune d'entre elles correspond à la modélisation d'une spécificité majeure issue du contexte de la production. Pour chaque cas, nous donnons une description générale, sa prise en compte dans PPO et enfin la formulation mathématique correspondant à la contrainte additionnelle à ajouter à $\mathcal{P}$ afin de considérer l'extension au niveau du *batching*. Ces nouvelles spécificités, indépendantes les unes par rapport aux autres, permettent d'établir une classification des problèmes par combinaison de chacune des extensions.

Une série d'expérimentations est réalisée au Chapitre 6, à partir d'un benchmark élaboré selon la classification établie au chapitre précédent. Le comportement de différentes variantes du programme linéaire que nous avons proposé dans cette thèse, alors résolu par ILOG CPLEX, est évalué, afin de déterminer une variante plus efficace en moyenne. Cette dernière est ensuite comparée à trois autres méthodes élaborées en parallèle de nos travaux, dans le cadre du

développement du logiciel PPO : un autre programme linéaire résolu par ILOG CPLEX, et deux heuristiques.

L'intégration logicielle, d'une part, et l'étude de problèmes provenant d'industriels, d'autre part, sont développées au Chapitre 7. Nous présentons tout d'abord plus en détail l'APS ILOG PPO, au sein duquel nous avons implanté nos travaux : l'algorithme de programmation dynamique, mais essentiellement le programme linéaire. Nous effectuons ensuite une étude d'un cas réel rencontré dans le contexte de l'industrie laitière. Cette application a permis de mettre en évidence des spécificités métier critiques que nous avons alors modélisées comme contraintes additionnelles de notre programme linéaire. Nous avons alors pu éprouver la robustesse et la flexibilité de notre approche en résolvant les instances fournies par l'industriel.

Enfin, nous terminons par une synthèse des études, travaux réalisés et résultats obtenus au cours de cette thèse. Nous indiquons aussi de possibles perspectives de recherche permettant de poursuivre le travail engagé.

Table des matières

1 Gestion et optimisation de la production dans l'industrie **19**
1.1 Problématique de la gestion de la production 19
1.1.1 Organisation de la chaîne d'approvisionnement (*Supply Chain*) 19
1.1.2 Planification et ordonnancement de la production industrielle 22
1.1.3 Émergence des APS (*Advanced Planning and Scheduling*) 24
1.2 Optimisation de la production par les APS 25
1.2.1 Définition et vocation d'un APS 25
1.2.2 Problèmes de planification à moyen terme 27
1.2.3 Problèmes d'ordonnancement détaillé 35
1.2.4 Coordination des niveaux tactiques et opérationnels 37
1.3 Positionnement du travail de thèse 47
1.3.1 Environnement de production 47
1.3.2 Motivations et objectifs de la thèse 50

2 Le problème de *"batching"* **53**
2.1 Description générale et intérêts opérationnels 53
2.1.1 Une quantité de production à partager 53
2.1.2 Définition informelle d'une bonne solution 56
2.1.3 Un problème de décision difficile 58
2.2 Formulation du problème central ou problème *"cœur"* 60
2.2.1 Description des données, notations générales et simplifiées 60
2.2.2 Solution générale du problème *"cœur"* 63
2.2.3 Architecture de résolution 64
2.2.4 Position du problème de *batching* étudié dans cette thèse 67
2.3 Liens avec l'ordonnancement par batch 68
2.3.1 Points communs entre *batching* et ordonnancement par batch 68
2.3.2 D'autres contraintes et de nouveaux objectifs 69
2.3.3 Conclusion et extensions 70

3 Étude de complexité **73**
3.1 Présentation des paramètres et critères considérés . . . 73
3.1.1 Restrictions du problème pour l'étude de complexité . . . 73
3.1.2 Paramètres et contraintes critiques . . . 74
3.1.3 Critères pour la fonction objectif . . . 75
3.2 Étude de problèmes de faisabilité et d'optimisation . . . 76
3.2.1 Décider de l'existence d'une solution . . . 76
3.2.2 Minimiser les coûts de production . . . 79
3.2.3 Minimiser les coûts de retard . . . 80
3.3 Preuve de complexité pour le problème $T4$ . . . 88
3.3.1 Définition du problème d'optimisation . . . 88
3.3.2 Complexité . . . 89
3.4 Conclusion . . . 95

4 Résolution du problème *"cœur"* de *batching* **97**
4.1 Programmation dynamique . . . 97
4.1.1 Problème traité, propriété et expression d'une solution . . . 98
4.1.2 Schéma de l'algorithme de programmation dynamique . . . 101
4.1.3 Complexité de l'algorithme . . . 104
4.2 Modélisation linéaire . . . 105
4.2.1 Problème abordé et complément de notations . . . 105
4.2.2 Approximation linéaire de la fonction objectif . . . 107
4.2.3 Contraintes du programme linéaire . . . 108
4.2.4 Synthèse sur les différents modèles présentés . . . 113
4.3 Étude expérimentale . . . 114

5 Extensions du problème *"cœur"*, classification des problèmes de *batching* **117**
5.1 Arrêt complet et productivité variable des ressources . . . 117
5.1.1 Description de l'extension . . . 117
5.1.2 Modélisation dans ILOG *Plant PowerOps* . . . 118
5.1.3 Intégration au programme linéaire en variables mixtes . . . 118
5.2 Modes alternatifs et capacité quelconque pour les ressources . . . 119
5.2.1 Description de l'extension . . . 119
5.2.2 Modélisation dans ILOG PPO . . . 119
5.2.3 Intégration au programme linéaire en nombres entiers . . . 120
5.3 Recette multi-activités et précédences internes . . . 121
5.3.1 Description de l'extension . . . 121
5.3.2 Modélisation dans ILOG PPO . . . 121
5.3.3 Intégration au programme linéaire en nombres entiers . . . 121
5.4 Précédences externes . . . 122
5.4.1 Description de l'extension . . . 122

5.4.2 Modélisation dans ILOG PPO . . . 123
5.4.3 Intégration au programme linéaire en nombres entiers . . . 123
5.5 Ruptures sur les demandes et hypothèse de surcapacité . . . 124
5.5.1 Description de l'extension . . . 124
5.5.2 Modélisation dans ILOG PPO . . . 125
5.5.3 Intégration au programme linéaire en nombres entiers . . . 125
5.6 Pénalisation du niveau de stock . . . 126
5.6.1 Description de l'extension . . . 126
5.6.2 Modélisation dans ILOG PPO . . . 126
5.6.3 Intégration au programme linéaire en nombres entiers . . . 126
5.7 Temps et coûts de setup . . . 129
5.7.1 Description de l'extension . . . 129
5.7.2 Modélisation dans ILOG PPO . . . 129
5.8 Synthèse sur les variantes de problèmes . . . 129

6 Expérimentations et analyse de performances 131
6.1 Jeu de données et indicateurs de qualités . . . 131
6.1.1 Benchmark pour l'évaluation d'algorithmes . . . 131
6.1.2 Critères de performance . . . 133
6.2 Étude des variations du programme linéaire . . . 134
6.2.1 Résultats numériques . . . 134
6.2.2 Analyse . . . 134
6.3 Comparaisons de différentes méthodes de résolution . . . 136
6.3.1 Description des trois autres approches de résolution . . . 137
6.3.2 Écart relatif moyen aux meilleures solutions connues . . . 138
6.3.3 Qualité de l'ordonnancement . . . 140
6.3.4 Étude des différents critères d'optimisation . . . 141
6.4 Conclusion . . . 142

7 Intégration logicielle et étude de problèmes industriels 145
7.1 L'APS ILOG *Plant PowerOps* . . . 145
7.1.1 Présentation de l'application : vocation et objectifs . . . 145
7.1.2 Architecture générale et intégration des travaux de thèse . . . 146
7.2 Cas d'étude en industrie laitière . . . 149
7.2.1 Analyse des procédés de transformation . . . 149
7.2.2 Modélisation des contraintes spécifiques . . . 151
7.2.3 Synthèse . . . 158

8 Conclusions et perspectives 161

A Notations mathématiques 165

Chapitre 1

Gestion et optimisation de la production dans l'industrie

Dans ce chapitre, nous décrivons le contexte global dans lequel s'inscrit la thèse, c'est-à-dire la gestion de la chaîne d'approvisionnement (*Supply Chain Management*). Plus particulièrement, nous nous intéressons aux décisions relatives à la production au sein d'une usine, en considérant deux niveaux de granularité temporelle : le moyen terme qui concerne des prises de décisions tactiques, et le court terme qui a trait à des questions d'ordre opérationnel. Cette thèse s'inscrivant dans le cadre du développement du produit ILOG *Plant PowerOps* (PPO), logiciel d'optimisation de la production, nous orientons nos travaux de recherche dans la perspective d'élaboration d'une solution générique.

Dans un premier temps nous présentons la problématique globale de la gestion de la chaîne d'approvisionnement ainsi que les premières réponses qui ont été apportées pour résoudre les problèmes qui y ont trait. Nous nous intéressons par la suite à des modèles mathématiques plus élaborés et aux outils de résolution issus de la Recherche Opérationnelle employés pour ces problèmes ; nous rapprochons ces méthodes des solutions logicielles d'aide à la décision effectivement commercialisées : les APS (*Advanced Planning and Scheduling*), qui font eux-mêmes appel aux techniques d'optimisation. Enfin, nous dégageons les motivations qui ont conduit à effectuer cette thèse dont nous explicitons les objectifs.

1.1 Problématique de la gestion de la production

1.1.1 Organisation de la chaîne d'approvisionnement (*Supply Chain*)

Réseau coordonné d'entités coopératives. La problématique de la gestion de production, dans laquelle s'inscrit la thèse, appartient elle-même à une problématique plus globale qu'est la gestion de la chaîne d'approvisionnement. Ce concept de "chaîne d'approvisionnement" peut être défini comme un réseau d'entités impliquées dans différents processus et activités permettant de produire de la valeur sous forme de marchandises ou de services, à destination d'un client final,

la coopération entre ces entités s'effectuant selon des flots bidirectionnels (du fournisseur vers le client final et inversement). Cette définition est inspirée de celle proposée par Christopher [33]. Dans le contexte de l'industrie de production, cela se traduit par la coopération entre les producteurs de matières premières, les usines réalisant la production proprement dite, les centres de distribution, les entrepôts, qui interviennent à différents niveaux de la chaîne (stockage de matières premières, de produits intermédiaires ou encore, de produits finis). Les centres administratifs et financiers peuvent aussi être inclus dans un tel réseau. Ainsi, les différentes entités qui le composent échangent non-seulement de la marchandise, mais aussi de l'information et de l'argent. Ces échanges sont conditionnés par une négociation entre chaque entité concernée par la transaction et la mise en place d'une coopération (c'est-à-dire l'intégration d'une nouvelle entité dans le réseau ou l'instauration d'un nouveau flot de marchandise, d'information et/ou d'argent) ne se réalise que s'il en résulte une stratégie "gagnant-gagnant" sur le long terme pour toutes les entités impliquées. Ces décisions relèvent de la *conception* de la chaîne et de la *coordination* entre toutes ses entités. Récemment, Li et Wang [81] ont dressé un panorama des mécanismes relatifs à ces prises de décisions en considérant différentes structures pour le système étudié (centralisé ou distribué).

Du modèle global au modèle à une unique entité. Dans le paragraphe qui précède, nous présentons la problématique de la gestion de la chaîne d'approvisionnement comme adoptant une vision très globale d'un système. En réalité, elle se décompose elle-même en sous-problématiques, chacune posant des questions s'appliquant au niveau d'une entité donnée, au sein de laquelle des décisions d'ordre stratégique, tactique ou opérationnel, pour ce qui concerne la dimension **temporelle**, peuvent être considérées. Le défi que pose la gestion de la chaîne d'approvisionnement consiste alors à trouver un compromis entre la vision globale et la vision distribuée du réseau. En effet, il s'agit de poser le problème de niveau global de façon suffisament simple pour qu'il soit résoluble mais suffisamment détaillé pour que les solutions proposées soient cohérentes et compatibles avec les solutions raffinées proposées au niveau d'une entité particulière. Ainsi, la question de la gestion de la chaîne d'approvisionnement se pose au sein de chaque entité. Ces décisions peuvent concerner différents champs d'activité. Pour ce qui concerne les flux de marchandises — qui est le sujet qui nous intéresse —, on trouvera par exemple les domaines suivants : prévision des besoins en matières premières, de la demande finale, de l'inventaire disponible, contrôle de la production, du stockage, du transport, des livraisons. Des distinctions similaires peuvent être identifiées lorsque l'on traite des flux d'informations ou de flux financiers. Ainsi, il existe non-seulement une hiérarchie relative au système considéré, mais aussi un découpage d'ordre **conceptuel** lors du traitement d'un problème donné, au sein de la problématique initiale. La structure à deux dimensions (temporelle et conceptuelle), propre à chaque sous-problématique appartenant à un niveau hiérarchique donné du réseau, sera reprise à la Section 1.2.1. Le paradigme de chaîne d'approvisionnement est donc très tentaculaire et en effectuer l'étude est un travail ardu. Tan [119] propose un cadre pour en faire l'analyse en remarquant l'existence de deux approches ("approvisionnement-satisfaction de la demande" et "transport-logistique") évoluant

de manière indépendante, qu'il tente alors d'harmoniser. Sahin et Robinson [107] étudient la relation entre la coordination des flots physiques et les flots d'informations au sein d'une chaîne d'approvisionnement.

Hiérarchie temporelle. Une dernière dimension doit être considérée lorsque l'on traite un problème appartenant à la problématique de la gestion de la chaîne d'approvisionnement, il s'agit de la granularité temporelle. Nous illustrons ici nos propos en nous plaçant dans le cas où, au sein d'une unique entité qu'est une usine de transformation de matériaux, le problème concerne la gestion de la production elle-même. D'une manière générale, du point de vue temporel, le fonctionnement d'un système peut être étudié à plusieurs niveaux. Nous avons évoqué ci-dessus trois degrés de prises de décisions : stratégique, tactique et opérationnel. Le premier a trait au long terme, les problèmes sont posés sur un horizon d'une dizaine d'années selon une granularité de prises de décisions annuelle ou semestrielle. Ces décisions concernent par exemple la diversification de l'activité de production, l'intensification de l'activité déjà en place ou la mise en application de nouvelles règles sanitaires. Cela suppose de modifier l'infrastructure globale, d'envisager l'achat de ressources coûteuses (machines servant à la production ou au traitement des produits) ou l'embauche de nouveau personnel. Le niveau tactique pose des questions à moyen terme, sur un horizon pouvant aller de six mois à deux ans, selon une maille temporelle hebdomadaire, mensuelle ou trimestrielle. Dans notre contexte, cela concerne des décisions sur la production globale de l'usine, la satisfaction de demandes, les niveaux de stocks... Enfin, le niveau opérationnel étudie la production sur le court terme. En fonction de l'industrie considérée, l'horizon ainsi que la granularité temporelle peuvent varier entre une semaine et un an, pour le premier, entre une minute et une journée, pour le second. Se posent ici les problèmes d'allocation de ressources (techniques ou humaines), d'affectation de dates précises pour l'exécution des différentes tâches... Un bilan des différentes approches sur la coordination d'un système mettant en jeu deux partenaires, à des niveaux stratégiques, tactiques et opérationnels a été proposé par Thomas et Griffin [123].

Objectif global du modèle et mise en pratique. Le terme *Supply Chain Management* que nous avons traduit dans cette thèse par "gestion de la chaîne d'approvisionnement" a été introduit en 1982 [96]. Le modèle sous-jacent a pour objectif d'accroître la compétitivité globale d'un système comprenant de multiples entités indépendantes mais intervenant au sein d'un même réseau de flux, en proposant à chacune d'elles une stratégie "gagnant-gagnant". Comme nous l'avons dit, la résolution pratique de l'ensemble des problèmes attenant à cette problématique générale, se fait en considérant plusieurs dimensions (conceptuelle et temporelle) et différents degrés de précision. À chaque situation correspond un problème clairement identifié qui peut alors être modélisé fonctionnellement par la mise en correspondance entre le problème et sa formulation en termes formels, ce qui permet alors d'envisager sa résolution. En général, les outils mathématiques permettent de mener à bien ce travail de modélisation, ainsi que celui de résolution. En particulier, les modèles et les méthodes issus de la Recherche

Opérationnelle apparaissent comme des réponses efficaces pour résoudre ces problèmes. Cependant, des méthodes basées sur l'approche par agents peuvent aussi être employées pour traiter de tels problèmes ; nous n'aborderons pas cette direction de recherche et renvoyons à [27, 51, 63, 12] à titre d'exemples.

Conclusion et références. La problématique générale de la gestion de la chaîne d'approvisionnement englobe, en plus des problèmes abordés dans le cadre de cette thèse, des problèmes d'approvisionnement des matières premières, de prévision et planification des demandes, de transport, de distribution... Dans [117], Stadtler propose une vue d'ensemble détaillée des notions fondamentales relatives à cette problématique. Dans [116], il présente un abrégé de ces différentes questions, ainsi qu'une description très générale des modèles et approches logiciels. Une étude en profondeur du paradigme défini ci-dessus dépasse le cadre de cette thèse ; cependant, il convient d'y faire référence dans la mesure où le problème abordé dans la suite s'y rapporte. La littérature traitant de cette problématique est très fournie car de nombreux domaines peuvent s'y rattacher : économie, gestion, logistique, transport, distribution, production... De cela résulte qu'il est difficile que chacun s'accorde sur une définition précise pour la *gestion de la chaîne d'approvisionnement*, comme l'illustre l'article de Croom et al. [34] qui discute des diverses approches d'analyse en étudiant différentes hiérarchisations et classifications. Néanmoins, nous ajoutons aux références déjà mentionnées quelques ouvrages majeurs et articles auxquels le lecteur pourra se référer comme point de départ au développement d'une connaissance plus poussée du domaine : [44, 48, 108, 109, 128].

1.1.2 Planification et ordonnancement de la production industrielle

Dans cette thèse, nous nous concentrons sur le problème de la gestion de la production au sein d'une seule usine, du point de vue des flux de matériaux. Nous nous positionnons dans le cadre de prises de décisions tactiques et opérationnelles, c'est-à-dire que nous couvrons les visions à moyen et à court termes. Par la suite, nous verrons que, plus précisément, nous nous plaçons à l'intermédiaire entre ces deux niveaux. Cependant, il est nécessaire d'étudier les deux problèmes décisionnels qui encadrent le nôtre afin de bien appréhender nos motivations et notre démarche. Dans la terminologie de la gestion de la production, ces deux problèmes se retrouvent sous les termes de *planification de production* pour les décisions tactiques et *ordonnancement* pour les décisions opérationnelles.

Motivations conduisant à l'identification des problèmes. À l'heure actuelle, les industries de production ont de plus en plus d'exigences en matière d'optimisation de leur fonctionnement. D'une part, la production doit satisfaire la demande, en évitant le stockage de matériaux et l'anticipation des livraisons (entraînant le plus souvent un stockage chez le client et des pénalités d'avance), ainsi que la pénurie ou le retard de la distribution. D'autre part, les industriels cherchent à accroître le rendement de leurs usines, en réduisant les coûts de production tout en augmentant le débit de celles-ci. Mais à ces exigences s'ajoutent des impératifs

liés à des contraintes physiques ou des exigences sanitaires : limitation des capacités, conditions techniques spécifiques, périssabilité des produits, nettoyage régulier des machines, traçabilité. Dans le but d'exercer un meilleur contrôle sur leur activité et de l'anticiper de façon réaliste, les industriels souhaitent donc obtenir des informations à moyen terme, tout au long de la chaîne de production, les aidant à prendre de bonnes décisions, souvent cruciales. En outre, afin de réaliser pratiquement la production, des décisions fines, prises à plus court terme et considérant les aspects détaillés de l'usine sont aussi nécessaires. Ces dernières permettent en outre aux industriels de faire preuve d'une forte réactivité de sorte que, si une modification de l'état du système de production survient (défaillance technique — temporaire ou non —, modification soudaine de la demande), une solution réajustée ou alternative puisse être proposée. Les industriels se trouvent alors face à deux problèmes connus de la Recherche Opérationnelle : la *planification de production* et l'*ordonnancement*.

Les premières solutions de gestion de production. Le développement de logiciels d'aide à la prise de décisions, dans le domaine du contrôle de la production et, plus globalement, dans celui de la gestion de la chaîne d'approvisionnement apparaît en réponse à l'identification de cette problématique dans les années 60-70. Le modèle MRP I (*Material Requirement Planning*), décrit pour la première fois par Orlicky en 1968 dans [97], est pionnier en matière de solution proposée aux industriels. Le principe sous-tendu par ce concept consiste à calculer les quantités nécessaires des matériaux, le long de la chaîne de production, afin de satisfaire les demandes en produits finaux. Il s'agit d'un modèle à capacité infinie, c'est-à-dire, qui ne tient pas compte de contraintes liées à la charge des ressources. Connaissant la nomenclature de l'usine considérée (lien entre matières premières, produits intermédiaires et produits finis) et le niveau de stock initial de chaque matériau, il s'agit de calculer la quantité à produire, pour chaque matériau, en remontant des demandes en produits finis vers les matières premières. Le calcul du MRP I se fait sur un certain horizon de temps et selon une maille temporelle donnée. Les demandes étant réparties tout au long de l'horizon, les quantités de produits à délivrer sont ainsi planifiées dans le temps [97]. Nous l'avons dit, dans ce modèle, les limites de capacité de ressources ne sont pas prises en compte, et cette insuffisance critique a conduit, dans les années 70, à l'élaboration d'un modèle plus précis : le MRP II (*Manufacturing Resources Planning*). Ce nouveau concept absorbe les contraintes et propriétés du MRP I qu'il étend dans un système plus complexe contrôlant l'utilisation des capacités. La procédure gloutonne du MRP I est désormais employée de manière itérative, chaque étape effectuant un lissage de la solution courante, de manière à répartir sur l'horizon les quantités de production planifiées ne respectant pas les capacités des ressources mises en jeu. Cette procédure est itérée jusqu'à l'obtention d'une solution satisfaisant toutes les contraintes de capacité.

Accroissement de la complexité des systèmes. Dans le même temps, on assiste à l'apparition d'architectures multi-modulaires, hiérarchisées temporellement (long, moyen et court termes) et conceptuellement (approvisionnements, productions, demandes, livraisons, aspects

financiers...). Dans les années 80, le concept d'ERP (*Enterprise Resource Planning*) émerge, élargissant encore celui de MRP II. L'ERP permet d'adapter les précédents modèles, jusqu'ici réservés aux industries de manufacture, à n'importe quel type de domaine (aérospatial, défense, industrie automobile, banque, assurance, industrie chimique et pharmaceutique, santé, hautes technologies, électronique, télécommunications...). L'ERP ajoute encore un degré de hiérarchisation en découpant l'entreprise en multiples entités, toutes dotées de leur propre système de gestion (MRP I ou II, ou procédure dédiée) mais communiquant entre elles. De nombreux ouvrages décrivent en détail les modèles MRP I et MRP II, qui sont à la base de tous les systèmes de gestion et contrôle de la chaîne d'approvisionnement, ainsi que les ERP qui conduisent à une grande complexité de coordination dans la résolution des problèmes. Pour de plus amples détails, le lecteur pourra se référer à [82, 91, 129, 47, 65, 99, 83].

1.1.3 Émergence des APS (*Advanced Planning and Scheduling*)

Il est important de faire remarquer au lecteur qu'à ce point de notre exposé concernant les solutions logicielles d'aide à la décision, nous n'avons pas employé le terme *optimisation* mais seulement ceux de *gestion* ou *contrôle*. En effet, jusqu'au milieu des années 90, les logiciels déployés dans les ERP ont surtout vocation à automatiser le travail des planificateurs, qui, jusqu'à l'émergence de tels systèmes, établissaient manuellement et quotidiennement les plannings, considérant en début de journée l'état courant de l'usine (demandes, ressources disponibles, niveaux de stocks...) et déduisant l'ensemble des tâches à accomplir et l'affectation des moyens permettant leur réalisation. Or, de nouveaux facteurs économiques, physiques, ainsi que des exigences humaines et sanitaires, sont à l'origine d'une augmentation de la complexité des problèmes. Ceux-ci ont mis en évidence les limites et les insuffisances des systèmes existants, précédemment décrits, et la nécessité d'y intégrer des techniques d'optimisation afin d'obtenir des solutions satisfaisant les industriels tant du point de vue de leur qualité que du temps passé par le logiciel à les déterminer. Le déploiement de ces méthodes de résolution de problèmes, en remplacement des MRP I et II, au sein des ERP, a conduit à un nouveau concept : les APS (*Advanced Planning and Scheduling*, on trouve parfois *Advanced Planning System*).

Suivant le contexte dans lequel le terme d'APS est employé, ce dernier peut désigner une procédure de résolution d'un problème spécifique d'optimisation de la production (on peut alors dire qu'il y a équivalence entre les notions d'APS et d'algorithme) ou bien, par extension, il peut désigner un logiciel complet dédié à la résolution de ce type de problèmes. On se trouve en général dans le premier cas lorsque l'on traite un problème précis et qu'un algorithme a été élaboré exclusivement pour résoudre le dit problème. Au contraire, lorsque le terme d'APS fait référence à une solution logicielle, une propriété de généricité apparaît alors. L'APS n'est alors pas dédié à un seul problème très précisément décrit mais, tout en demeurant dans un certain domaine d'application, il est capable de traiter différentes variantes d'un problème. Dans la suite, nous employons le terme d'APS pour faire référence à la solution logicielle, celui d'algorithme (ou procédure) dédié est utilisé pour se référer au premier cas.

1.2 Optimisation de la production dans les APS

Dans cette section, nous proposons une étude technique des problèmes d'optimisation de la production au niveau tactique (planification) et au niveau opérationnel (ordonnancement), ainsi que de la coordination entre ces deux niveaux. En particulier, nous nous intéressons aux modèles et méthodes issus de la Recherche Opérationnelle développés pour traiter ces questions et étudions dans quelle mesure ils sont intégrés dans les APS du marché.

1.2.1 Définition et vocation d'un APS

Architecture générale des APS. L'accroissement des marchés, la multiplication de la concurrence, la volonté des industriels d'augmenter leur productivité et de limiter leurs pertes sont autant de facteurs liés à l'optimisation du fonctionnement d'une usine. Cependant, cela nécessite une modélisation très fine du système que l'on souhaite optimiser, afin de ne pas fournir des solutions non-réalisables (car elles ne satisfont pas certaines contraintes critiques) ou, au contraire, sous-optimales. Or, aujourd'hui, les variations liées au domaine d'application, les spécificités des ressources, l'augmentation de la technicité, la complexification des procédés de production, le renforcement de la rigueur sanitaire sont bien souvent des aspects qui contribuent à une très grande complexité des problèmes. De cela résulte que les APS sont en général dédiés à un certain domaine d'application. On trouvera par exemple le domaine de l'industrie de transformation (dite industrie de *process*), avec la chimie, la pharmacie, l'agro-alimentaire, ou l'industrie manufacturière (d'assemblage) avec l'automobile, l'aérospatial... Chacune de ces industries présente des contraintes très diversifiées et il est clair qu'un même outil ne peut se révéler efficace dans tous les contextes. De surcroît, au sein d'un même domaine, voire d'une même société, il existe encore des différences d'une usine à l'autre, le logiciel employé pour traiter ces problèmes doit donc être capable de prendre en charge ces disparités tout en restant générique et efficace.

Malgré ces contraintes qui poussent les APS à se spécialiser, une architecture commune peut tout de même être mise en évidence. Nous l'avons dit en début de chapitre, les deux dimensions conceptuelle et temporelle permettent de définir des problèmes particuliers, communs à tout domaine d'application. Cela permet d'inférer un certain nombre de modules, chacun étant affecté à la résolution d'un problème donné, que Meyr [90] a représentés par le diagramme de la Figure 1.1. Nous y retrouvons les deux dimensions d'étude d'un système : horizontalement, la dimension conceptuelle et verticalement, la dimension temporelle.

Dans cette thèse, nous nous concentrons sur les modules de planification et d'ordonnancement de la production, grisés sur la figure. Désormais nous ferons l'hypothèse que nous nous situons dans ce cadre d'étude où un APS est réduit à ces deux modules. Nous le verrons par la suite, cette restriction est tout à fait réaliste puisqu'il est peu probable de trouver des logiciels à haut niveau d'expertise prenant en charge l'ensemble des modules représentés sur la figure. Un APS, en plus de s'inscrire dans un domaine d'application, est en général focalisé sur un concept et un horizon temporel donné.

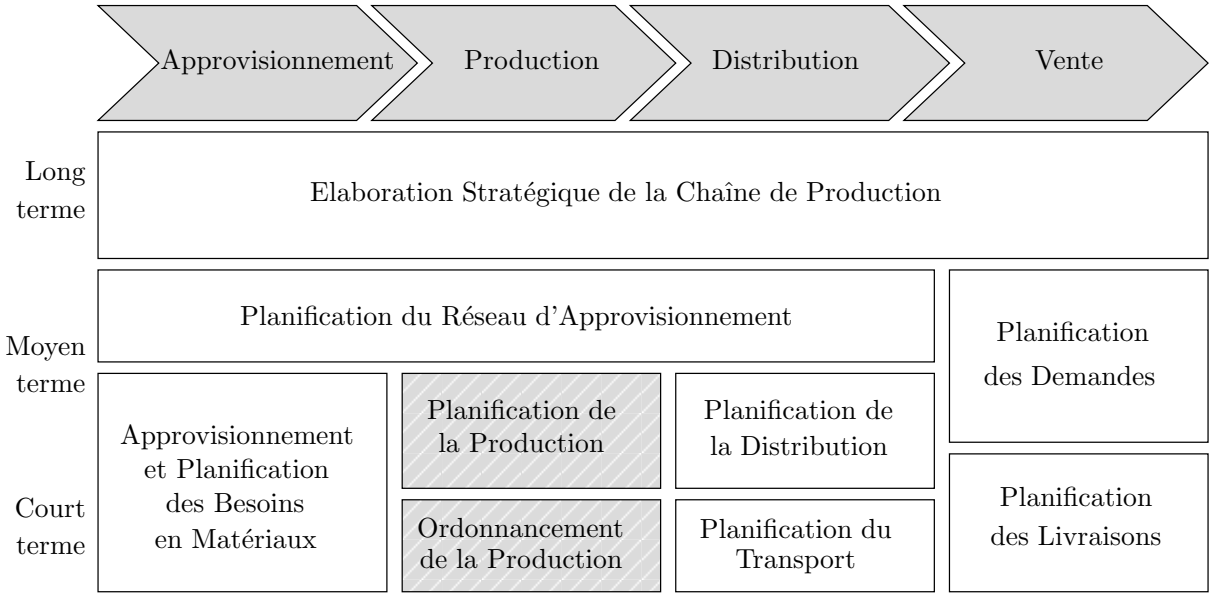

FIG. 1.1 – Modules supportés dans les APS (traduit de [90])

Objectifs et défis des APS. Les APS considérés dans cette thèse ont pour but d'apporter aux industriels des solutions pour la gestion de la production, au sein d'une usine donnée, à moyen et court termes. Ces solutions doivent d'une part être réalisables, c'est-à-dire respecter toutes les contraintes imposées par le système de production, d'autre part, optimiser les coûts définis par l'industriel. Ces coûts intégrent en général des critères qui sont de deux types : le revenu réel généré par une solution donnée (recettes induites par la satisfaction de demandes, ou au contraire, manque à gagner en cas de rupture de stocks, coûts de production intrinsèques à l'élaboration des produits finis...), et des pénalités plus difficilement mesurables financièrement (coûts de stockage des matériaux, pénalités liées à la livraison en avance ou en retard de produits finis dus à une date préalablement négociée avec le client dans une perspective de livraison "juste-à-temps"...).

Dans notre étude, deux modules sont concernés par la modélisation et l'optimisation des coûts du système considéré. Les APS adoptent en général le même type de décomposition pour coordonner ces modules : un plan de production, calculé à moyen terme, est fourni en entrée du calcul à court terme de l'ordonnancement. Les décisions de planning deviennent alors des contraintes pour l'ordonnancement [62]. Ces deux problèmes sont donc fortement liés. Lors de l'élaboration de l'APS, deux difficultés se posent. D'une part, chaque module considéré indépendamment doit résoudre un problème d'optimisation complexe, il faut donc trouver un compromis, entre un modèle réaliste et un modèle relativement adaptable à diverses situations, ainsi qu'entre l'obtention d'une solution de bonne qualité et le temps nécessaire à son calcul. Sur ce point, les méthodes de Recherche Opérationnelle se révèlent comme des outils indispensables qu'il convient

d'utiliser avec pertinence et discernement afin de profiter au maximum de leur potentiel d'efficacité. D'autre part, le problème de la coordination entre les deux modules est crucial. En effet, la façon dont est établie la connexion entre eux, au sein de l'APS, est un indicateur de cohérence du système dans sa globalité. L'obtention de solutions de bonne qualité, tant pour le plan de production que pour l'ordonnancement, est conditionnée par une bonne cohérence du système. Comment assurer qu'à partir d'un plan de production, même optimal, on puisse calculer un ordonnancement, lui-même optimal, voire seulement réalisable ? Certes, les deux modules considèrent le même système de production et optimisent les mêmes coûts, cependant, la modélisation à moyen terme diffère de la modélisation à court terme et, sur ce point, la difficulté réside donc dans l'établissement d'une grande cohérence entre planification et ordonnancement.

Démarche suivie pour l'étude. Dans la suite, nous nous proposons d'étudier l'intégration des méthodes d'optimisation, dans les modules de planification et d'ordonnancement des APS. Nous tentons de mettre en parallèle les modèles et méthodes classiques proposés par la littérature, les algorithmes dédiés à certains cas de production et les solutions mises en place dans les logiciels. Dans la mesure du possible, nous illustrons nos propos par l'étude d'APS de marché. Cependant, compte tenu des problèmes liés à la confidentialité, les informations sur les algorithmes implantés dans ces solutions sont sensibles et rarement accessibles de manière détaillée. Seul le produit développé par ILOG, PPO, qui a offert un cadre d'application pour nos recherches et travaux, est analysé avec plus de précision lorsque cela s'avère nécessaire. En outre, comme nous nous intéressons particulièrement à la coordination entre les modules de planification et d'ordonnancement au sein d'un APS, nous mettons en évidence les problèmes sous-jacents et montrons comment ils sont résolus dans les divers APS étudiés.

1.2.2 Problèmes de planification à moyen terme

Dans une perspective à moyen terme de la gestion de la production, les problèmes de planification étudient des questions de niveau tactique, essentiellement relatives à la quantité de matériaux à produire globalement dans un intervalle de temps donné, de manière à satisfaire les demandes distribuées le long de l'horizon. Il s'agit donc de décider *exactement combien* et *approximativement quand* produire. Cette thématique générale a donné lieu à la formulation de nombreux problèmes qui sont abondammment étudiés dans la littérature depuis une cinquantaine d'années. Ils appartiennent à la problématique de dimensionnement de lots, ou *lot-sizing*. Dans cette section, nous commençons par présenter les principaux modèles mathématiques élaborés pour résoudre ces problèmes en les classifiant suivant divers critères. Nous verrons dans un second temps dans quelle mesure ils sont intégrés à des APS et quelles autres méthodes peuvent aussi être employées dans la pratique.

1.2.2.1 Classification des principaux modèles

Modèle initial et premiers constats. Le problème qui est à l'origine des nombreux travaux réalisés par la suite, a été formulé dans les années 50. En particulier, les travaux de Wagner et Whitin [130] ont mené à cette première variante, simple mais fondamentale, du *lot-sizing*. Ce problème considère un seul produit fini et peut être posé de la façon suivante : étant donnés un horizon divisé en un certain nombre de périodes de temps, une certaine quantité de produit fini requise pour chaque période et le niveau de stock initial du produit, il s'agit de décider quelle quantité produire par période pour satisfaire la demande. Dans ce modèle, les contraintes de capacité sont ignorées, et la fonction objectif consiste en trois termes : un coût proportionnel à la quantité produite dans une période, un coût fixe de mise en production dans une période et un coût de stockage pénalisant l'inventaire en fin de période. Nous notons ce problème ULSP (*Uncapacitated Lot-Sizing Problem*). Il est résolu en temps polynomial [130, 49, 3] et les contraintes de flot sous-jacentes en autorisent une formulation linéaire en variables entières. Ainsi, les problèmes de *lot-sizing* qui étendent ULSP se modélisent en général par des programmes linéaires, souvent en variables mixtes, ce qui permet leur résolution par un solveur, même dans le cas de problèmes difficiles.

Le modèle ULSP, de par les réalités qu'il exprime ainsi que celles qu'il ignore, permet de mettre en évidence certaines notions centrales dans les problèmes généraux de *lot-sizing*. Parmi elles, et pour guider notre analyse, nous relevons les quatre suivantes :

- la capacité des ressources réalisant la production, totalement ignorée dans ULSP ;
- la définition des périodes, qui pose la question de la modélisation du temps ;
- les coûts de mise en production, parfois considérés comme des coûts de reconfiguration des ressources (setup), et dont l'interprétation dépend du contexte pratique ;
- les quantités de production, à mettre en rapport avec la réalité physique des lots de production.

Il convient cependant de souligner l'importance d'autres notions comme le nombre de produits pris en compte, le nombre de niveaux de la chaîne de production (modélisation des matières premières et produits intermédiaires, selon des procédés de transformation spécifiques). Ces paramètres font souvent l'objet d'études dans les classifications des problèmes de *lot-sizing*. Néanmoins, il est moins pertinent dans notre contexte d'en faire des articulations centrales car, la généralisation à plusieurs produits et plusieurs niveaux n'y est pas problématique. Dans la suite, nous présentons des résultats sur le cas multi-produit à un seul niveau de production. Nous donnons ensuite quelques généralisations de ces résultats aux variantes multi-niveaux. Cependant, notre propos n'est pas de faire un état de l'art des travaux sur les problèmes de *lot-sizing* et nous portons notre attention sur les considérations pratiques qui ont guidé ces travaux.

Capacité des ressources. Ignorer les contraintes de capacité des ressources permet de faire l'hypothèse que l'on peut exécuter ce que l'on veut quand on le veut. Il est clair que cette hypothèse n'est pas réaliste dans la majorité des cas et très rapidement, les modèles de *lot-sizing* à capacité finie ont été introduits. Le problème général dans lequel la capacité de production

peut varier d'une période à l'autre est noté CLSP pour *Capacitated Lot-Sizing Problem*, il a été prouvé NP-dificile par Bitran et Yanasse [14], néanmoins, un algorithme pseudo-polynomial a été proposé par Chen *et al.* dans [30]. En revanche, le cas particulier où la capacité est constante au cours du temps (CCLSP : *Constant Capacity Lot-Sizing Problem*) est polynomial [126]. Des états de l'art sur les problèmes avec ou sans contraintes de capacité peuvent être trouvés dans [5, 132, 19, 69].

Modélisation du temps. De manière générale, on peut distinguer deux façons de considérer le temps : la modélisation continue ou discrétisée. Bien que notre intérêt se concentre ici sur les approches discrètes, citons le modèle ELSP (*Economic Lot Scheduling Problem*) qui considère un niveau de production et une demande constante au cours du temps et adopte une modélisation continue du temps [46]. En général, ces formulations sont réservées à des problèmes plus détaillés tels que l'ordonnancement, nous reviendrons sur cette approche et concentrons ici notre intérêt sur les approches discrètes. Dans les modèles de *lot-sizing*, la discrétisation du temps se fait en subdivisant l'horizon en intervalles réguliers, les décisions sont alors prises par période (e.g. les quantités de production) et aux bords de chaque période (e.g. les niveaux de stock). Les modèles à formulation discrète peuvent à leur tour être classifiés suivant deux approches, en fonction de la durée des périodes. Il existe les modèles à courtes périodes (micropériodiques ou encore *small buckets*) et les modèles à longues périodes (macropériodiques ou encore *big buckets*). L'appartenance d'un modèle à l'une ou l'autre classe est établie en répondant aux questions : est-il possible de produire des lots de différents matériaux sur une même ressource pendant une période ? et, est-il possible de produire plusieurs lots d'un même matériau sur une même ressource pendant une période ? La première question est reliée à celle de la modélisation des setups comme reconfiguration des ressources en cas de changement de produit, la seconde cherche à mettre en rapport les quantités planifiées et les lots physiquement mis en production dans l'usine. On peut trouver une classification dans [10] et [11].

Le modèle CLSP est macropériodique, de cela résulte que la modélisation des setups y est approximée puisque la séquence d'exécution des lots ne fait pas partie de la solution. Au contraire, les modèles micropériodiques subdivisent les longues périodes définies dans le CLSP de façon à ce qu'un seul matériau ne soit produit par période. C'est le cas des modèles DLSP (*Discrete Lot-sizing and Scheduling Problem*) [67, 24, 50] et CSLP (*Continuous Setup Lot-sizing and scheduling Problem*) [70]. La réalisation d'un seul produit par période est autorisée, soit en utilisant toute la capacité disponible ("tout ou rien") dans le cas DLSP, soit en permettant à la ressource d'être inactive, dans le cas CSLP. Le modèle PLSP (*Proportional Lot Sizing and scheduling Problem*) élargit le CSLP en autorisant l'utilisation du restant de la capacité pour la production d'un autre matériau, l'exécution d'un seul setup étant permise dans une période (au début ou à la fin) [41]. Pour les formulations mathématiques et une revue de ces modèles, nous renvoyons le lecteur à [42, 118]. Notons que les modèles à courtes périodes déterminent exactement la séquence des lots, c'est pourquoi ils résolvent en général le problème d'ordonnancement associé. Cependant, remarquons que, dans la pratique, des problèmes de très grande taille en résultent,

le nombre de variables et de contraintes est souvent prohibitif et des modèles macropériodiques sont privilégiés.

Nous l'avons dit, la modélisation des setups dans ces formulations est approximée, un setup étant exécuté dès lors qu'un matériau est produit, sans pouvoir considérer la séquence d'exécution. Or, dans les cas où les coûts et temps de setup sont importants, cette approximation se révèle très pessimiste. Ainsi, des modèles macropériodiques issus du CLSP ont été élaborés dans le but d'obtenir une meilleure modélisation de la réalité, en maintenant une taille des problèmes raisonnable. En particulier, la possibilité de conserver un setup d'une période à l'autre a mené aux problèmes CLSP*L* (*Capacitated Lot Sizing Problem with Linked lot sizes*) [58, 113, 111] et ou CLSD (*Capacitated Lot Sizing problem with sequence Dependent setup costs*) [59].

Coût de production et setup. Bien que toutes deux associées à une pénalité fixe intervenant lors de l'exécution d'un lot (pénalité souvent accompagnée d'un temps fixe), ces notions ne renvoient pas toujours à la même réalité. Dans la suite du manuscrit, nous utiliserons les définitions suivantes : un *coût fixe de production* est associé à l'exécution d'un lot et est automatiquement dû, quoi qu'il arrive ; au contraire, un *setup* correspond à une reconfiguration de la machine sur laquelle s'exécute un lot, cette reconfiguration est nécessaire seulement si la machine n'est pas dans l'état requis avant l'exécution du lot, les coûts (et temps) de setup dépendent donc de la séquence. Ainsi, les modèles discrets de planification micropériodiques, permettant de connaître la séquence d'exécution, modélisent exactement les setups tandis que les modèles macropériodiques en font une approximation en associant un coût fixe à chaque fois qu'il est décidé de produire un certain matériau, ce coût doit être précalculé à partir des différents coûts de setup pouvant survenir.

Par ailleurs, nous l'avons dit, les coûts fixes de production doivent être payés à chaque lot exécuté, cependant les modèles macropériodiques déterminent des quantités de production importantes qui nécessitent souvent d'être recoupées en lots réellement exécutables, afin de satisfaire des contraintes physiques ou d'optimiser l'avance et le retard de la production à l'intérieur d'une période, lors de l'ordonnancement (c'est l'objet du paragraphe suivant). Dans de telles situations, le nombre exact de lots n'est pas toujours déterminé par la procédure de planification et les coûts fixes de production sont à nouveau une approximation de la réalité.

Quantité par période et lots réels. Nous venons d'évoquer cette question dans le paragraphe précédent et nous souhaitons y consacrer une attention particulière. En effet, la problématique du découpage des quantités planifiées en lots afin d'obtenir un ordonnancement réalisable au niveau de l'usine (qui respecte les contraintes physiques) et de bonne qualité (production juste-à-temps) est au cœur de nos travaux. C'est elle qui a émergé des constats faits lors des études de cas pratiques. En effet, l'APS ILOG PPO au sein duquel s'inscrivent nos travaux adopte un modèle macropériodique pour son module de planification de production, et l'étape de redécoupage de la production s'est avérée nécessaire dès lors que des contraintes physiques

interviennent (e.g. taille des lots bornée inférieurement et supérieurement), ce qui a souvent été mis en évidence. En outre, l'aspect lié à l'optimisation est évidemment présent dans cette question. Dans des environnements réels où les problèmes de production sont complexes il est difficile de prendre cet aspect en compte dans un modèle générique, dès le plan de production, de manière à optimiser correctement les coûts. Nous reviendrons sur ce point de manière plus détaillée au prochain chapitre.

Remarque sur l'industrie de *process*. On trouve une littérature fournie concernant les problèmes de planification en industrie de process. La récente thèse de doctorat de Lütke Entrup [82] est une importante contribution. Après y avoir effectué une synthèse très complète des problèmes, des modèles et solutions logicielles, pour les décisions tactiques, l'auteur propose des approches pour intégrer les contraintes de durée de vie des produits dans la chaîne de production dès les modèles de planification. Dans l'industrie chimique et pharmaceutique, on trouve les travaux de Blömer, Günther, Kallrath, Westenberger, Neumann, Trautmann, Schwindt, Timpe, Maravelias, Grossmann... Les approches qu'ils proposent sont souvent dédiées à des problèmes particuliers et sont en général couplées avec un problème d'ordonnancement. Ces méthodes se révèlent en général fructueuses. Nous reviendrons sur leurs travaux en Section 1.2.4. Une nouvelle terminologie apparaît dans ce domaine pour désigner les quantités planifiées : le batch (on parle d'ailleurs de problèmes de *batch-sizing*). Du point de vue du problème de planification, la distinction entre les concepts de lot et de batch n'est pas toujours explicite. Kallrath [68] emploie les deux termes pour désigner deux notions différentes : d'un point de vue hiérarchique dans la décomposition du problème, le batch se situe à un niveau de décision supérieur, puisqu'il peut à son tour être fractionné en lots. Cela fait écho à la terminologie utilisée dans les problèmes de *batching* (qui seront étudiés dans le chapitre suivant), où un batch est un regroupement d'activités. Par la suite, nous emploierons de préférence le terme de batch (éventuellement de lot) pour faire référence à ce qui s'exécute en une fois au niveau de l'usine (un batch ne peut être interrompu) ; quant aux quantités résultant du plan de production, nous les désignons par le terme de "quantités planifiées". Le concept de campagne peut aussi être introduit pour désigner la production d'un même matériau sur une longue durée, ce qui peut correspondre à l'exécution de plusieurs lots ou batches à la suite (sans nécessiter un nouveau setup) [112, 102, 56].

Méthodes de résolution. La modélisation par programmation linéaire est la plus répandue pour les problèmes de *lot-sizing*, en effet, le problème de flot sous-jacent conduit naturellement à cette approche [112, 113, 76, 114]. Les variantes les plus simples (en général, les cas à un seul produit, avec capacité infinie, constante ou finie), mènent à des problèmes qui restent polynomiaux ou NP-difficiles au sens faible. Des méthodes exactes polynomiales ou pseudo-polynomiales ont pu être développées, souvent par la programmation dynamique [3, 130]. Pour les problèmes difficiles, des méthodes exactes basées sur des algorithmes de séparation (*branch and bound* ou *branch and cut*) ont été étudiées [1, 100]. On trouve aussi des résolutions approchées avec le développement d'heuristiques, souvent basées sur des résultats de programma-

tion linéaire [1, 61, 84].

Conclusion et références. Un problème de planification de la production au sein d'une usine est en général exprimé par une formulation découlant des modèles de *lot-sizing*. Ceux-ci ont été décrits de manière à nous permettre d'en dégager, dans la section suivante, les forces et les limites en vue d'une implantation au sein d'un APS.

Outre les références proposées ci-dessus, de plus amples informations peuvent être trouvées sur ces sujets qui sont l'objet de nombreuses recherches. Signalons la thèse doctorale de Absi [1] qui présente un état de l'art détaillé des modèles à un ou plusieurs matériaux. Ces travaux considèrent les aspects additionnels de non-satisfaction de demandes (ruptures) et de maintien d'un stock de sécurité. Les problèmes considérant plusieurs niveaux de production, modélisant ainsi matières premières, produits intermédiaires et finis, plus difficiles que les problèmes à un niveau, sont souvent traités par des méthodes moins directes que la programmation linéaire. Le problème est NP-difficile dans le cas général [39]. Pochet et Wolsey dans [100] traitent ce type de problèmes avec la méthode des plans sécants. Des méthodes basées sur la relaxation lagrangienne [2, 13, 121] ou la génération de colonnes [26] sont aussi employées. Enfin, des heuristiques spécifiques ont été élaborées pour traiter des problèmes particuliers [122, 115]. Par ailleurs, la plupart des états de l'art réalisés sur le problème de planification de production incluent souvent des problèmes d'ordonnancement [42, 118, 19, 69].

Les contraintes prises en compte dans ces modèles sont souvent réalistes, ou tout au moins, sont absolument nécessaires pour refléter la réalité de la production. Par ailleurs, différents objectifs sont définis. Dans certains modèles, on minimise les coûts de setup (fixes), de production (fixes et variables), d'utilisation des ressources, les niveaux d'inventaire (ou d'écart par rapport aux niveaux minimal et maximal), les pénalités d'avance, de retard ou de pénurie. Dans d'autres, on maximise la production totale, l'utilisation des ressources ou encore la satisfaction des demandes. Sans même parler des contraintes spécifiques à des cas industriels réels, contraintes et fonctions objectifs réalistes laissent présager un grand nombre de problèmes en planification de production. Bien souvent, ces aspects se retrouvent en effet dans les applications pratiques, mais de nouvelles contraintes sont souvent découvertes par l'étude de cas concrets, ce qui conduit à la définition de nouveaux problèmes.

1.2.2.2 Planification dans les APS

Comme nous l'avons dit, les applications réelles sont en même temps dédiées à un domaine particulier d'application (pharmaceutique, agroalimentaire, automobile...), et en même temps génériques du point de vue des problèmes qui leur sont soumis. En effet, au sein d'un domaine, des contraintes communes se retrouvent d'une société à une autre, d'une usine à une autre, et souvent avec le même degré d'importance ; mais le problème à résoudre n'est jamais exactement le même, les équipements varient, les processus de transformation ou de fabrication sont différents, les exigences sanitaires ou relatives à la qualité des solutions ne sont pas les mêmes ou ne se situent pas sur les mêmes critères... De cela résulte qu'un logiciel de planification doit être à

même de prendre en compte de nombreuses contraintes pouvant survenir seulement dans certains cas particuliers, et d'optimiser des coûts très divers.

La programmation linéaire comme méthode fondamentale. La programmation linéaire, bien souvent en variables mixtes, s'avère être une méthode privilégiée pour résoudre les problèmes de planification. Cela s'explique par plusieurs points. D'une part, les études théoriques des modèles de *lot-sizing* permettent de mettre en évidence les forces et les faiblesses inhérentes aux différents modèles, et ainsi, d'appréhender *a priori* les difficultés que présentent un problème pratique ; modéliser un problème de planification par une formulation linéaire est en général naturel. Par ailleurs, la résolution de programmes linéaires par des solveurs du marché est de plus en plus efficace, l'intégration de méthodes transversales à celle classique de séparation et évaluation (coupes, heuristiques, recherche locale) permet de rendre ces logiciels de plus en plus robustes. De cela résulte que, pour le concepteur d'APS, la difficulté réside essentiellement dans la bonne modélisation du problème qui lui est posé. La généricité qu'offre intrinsèquement la méthode de la programmation linéaire répond parfaitement à l'exigence qu'ont les concepteurs d'APS de proposer une solution logicielle adaptable à de nombreux cas particuliers.

La modélisation linéaire. Les impératifs des industriels et la spécificité des systèmes de production rendent les problèmes de planification complexes, c'est pourquoi des efforts de modélisation et de résolution doivent être réalisés afin de mettre en œuvre des méthodes efficaces. Les modèles présentés dans la section précédente guident en général l'élaboration de ces solutions logicielles lors de la création du module de planification d'un APS. Ainsi, au sein de la plupart des APS du marché, on trouve au cœur du module de planification un programme linéaire inspiré du modèle macropériodique CLSP. C'est le cas de l'APS ILOG PPO, mais aussi de FuturMaster, SAP *Advanced Planner and Optimizer* (APO) dans son module SNP (*Supply Network Planning*), OMPartners, AZAP, INFOR, n.Skep (DynaSys). Ces APS du marché résolvent les programmes linéaires ainsi formulés en utilisant un solveur (e.g. ILOG CPLEX).

Tous sont capables de traiter les cas à plusieurs produits et plusieurs niveaux. Cependant, ignorant les contraintes de précédences entre niveaux (relation production-consommation), ainsi que les séquences d'exécution qui seront physiquement mises en œuvre sur chaque ressource (cette décision est du ressort de l'ordonnancement), l'un des défis consiste à en faire des approximations les plus fines possibles, tout en restant linéaire. Compte tenu de la longueur des périodes, le délai qui sépare les exécutions d'un batch de produit intermédiaire et d'un batch de produit fini utilisant la production du premier batch est en général ignoré. De manière très optimiste, les modèles considèrent que l'exécution peut se faire en parallèle. On peut trouver des appoximations plus réalistes qui limitent la quantité de production possible de matériaux appartenant au bas de la chaîne en estimant un délai minimal (pour les modèles optimistes), maximal (pour les modèles pessimistes), ou moyen, entre les différents niveaux de production. Cependant, le délai est souvent négligeable devant la longueur des périodes, ce qui diminue l'impact de cette amélioration sur les résultats. L'approximation des setups peut quant à elle se révéler

primordiale pour la qualité d'un modèle. En effet, dans de nombreux cas, le temps nécessaire à la reconfiguration d'une ressource peut être très élevé. Plus courts que la durée d'une période elle-même, ils ne sont cependant plus négligeables et selon que l'on adopte un modèle gros grain ou précis, pessimiste ou optimiste, les résultats peuvent se révéler très différents. Les modèles les plus simples comptent un setup par produit si celui-ci est fabriqué au cours de la période (CLSP classique) ; d'autres décident qu'un setup exécuté dans une période est conservé sur la période suivante, ce qui permet alors d'en faire l'économie (CLSPL). Parallèlement, des modèles utilisent le regroupement par produits requérant la même configuration de la ressource, ils sont une bonne approximation de la réalité dans de nombreux cas pratiques. Enfin, les modèles les plus sophistiqués combinent les deux dernières approches. Le choix du modèle doit être fait avec le plus grand soin car opter pour la formulation la plus fine n'est pas nécessairement bénéfique. En effet, plus le modèle est raffiné et plus il génère un programme linéaire de grande taille (en nombre de variables — notamment binaires — et de contraintes). De surcroît, il n'est pas évident que le modèle sophistiqué s'approche au plus de la réalité. Sur ce sujet, une étude du cas pratique est primordiale pour décider de sa modélisation.

De nombreuses contraintes peuvent être exprimées dans un modèle linéaire : utilisation des ressources (choix de la ressource et pour quelle capacité), respect de niveaux minimal et maximal d'inventaire, ruptures de stock, report des livraisons (*backlog*), satisfation totale ou partielle des demandes (ruptures). Il est cependant nécessaire de prendre garde à l'expression mathématique des contraintes, car deux formulations a priori équivalentes ne sont pas toujours aussi bien résolues par un solveur. La fonction objectif peut aussi prendre en compte différents critères : coûts d'utilisation des ressources, de setup, de production en général, coûts de surstock ou de rupture de stock, pénalité d'avance ou de retard des livraisons planifiées, revenus induits par les livraisons prévues... À nouveau, la pondération de chaque critère dans la fonction objectif doit être faite avec soin, afin de ne pas déséquilibrer le programme de manière inconsidérée.

Les méthodes dérivées. Il est clair que la programmation linéaire est une approche fondamentale dans la résolution de problèmes de planification. Cependant, la linéarité requise pour la formulation peut être rédhibitoire. Il peut s'agir d'exprimer des contraintes non-linéarisables (ou linéarisables au prix d'un modèle de taille peu raisonnable), pourtant indispensables, ou d'optimiser une fonction objectif quelconque non-linéaire. Il va de soi que des approximations linéaires peuvent d'abord être envisagées mais elles ne sont pas toujours possibles ou satisfaisantes. Dans ces cas-là, des méthodes de relaxations lagrangiennes ou des heuristiques de décomposition à base de programmation linéaire peuvent être élaborées (DynaSys n.Skep, [1] ; FuturMaster, SNP de SAP [82, 40]). Des métaheuristiques ou plus simplement, des heuristiques à base de règles (en général des méthodes gloutonnes issues d'un calcul MRP modifié) peuvent être envisagées comme des alternatives (FuturMaster, SNP de SAP).

1.2.2.3 Conclusion

Les problèmes de planification de la production à moyen terme conduisent à des prises de décisions relatives à l'activité globale de la production d'une usine. Aussi bien dans les modèles classiques que dans ceux utilisés en pratique, le temps y est décomposé en un ensemble de périodes au sein desquelles il s'agit de décider globalement ce qui doit être produit, en quelle quantité, et afin de satisfaire quelles demandes. La capacité des ressources et les coûts liés à la production (coûts fixes indépendants de la séquence ou setup) d'une part et les niveaux de stocks et les livraisons des demandes d'autre part, y sont en général aggrégés ou approximés par période et selon des formulations plus ou moins sophistiquées. La méthode classique pour la résolution est la programmation linéaire, tant pour son utilisabilité que pour la qualité des solutions des solveurs du marché.

1.2.3 Problèmes d'ordonnancement détaillé

Au sein de la problématique d'optimisation de la production, les problèmes d'ordonnancement se situent au niveau de détail le plus fin. Il s'agit de décider *exactement quand* et *par quelle ressource* est exécutée chaque activité de production. Dans l'absolu, on peut dire qu'une solution d'ordonnancement doit pouvoir s'exécuter telle quelle dans l'usine considérée, c'est-à-dire que le degré de précision du modèle doit permettre de prendre en compte les complexités et spécificités du problème de manière à produire une solution physiquement réalisable. Dans ce qui suit, nous n'étudions pas en détail la problématique d'ordonnancement comme nous l'avons fait pour la planification, nous décrivons succinctement le problème sous-jacent ainsi que les difficultés qui lui sont inhérentes. Nous présentons certaines contraintes provenant de l'industrie de production. Enfin, nous abordons les méthodes couramment employées dans les APS pour résoudre ces problèmes.

Le problème général de l'ordonnancement. Dans tout problème d'ordonnancement, on trouve des décisions et contraintes classiques, qui font l'essence de cette problématique. Un tel problème se pose comme suit : étant donnés un horizon de temps, un ensemble de ressources ainsi que leur disponibilité sur l'horizon, un ensemble d'activités auxquelles sont associées les ressources permettant leur exécution, la capacité qu'elles requièrent, ainsi que leur durée d'exécution, le problème consiste à décider, pour chaque activité, ses dates de début et de fin ainsi que les ressources réalisant son exécution.

Contraintes et fonctions d'optimisation classiques. De nombreuses extensions ou restrictions peuvent venir enrichir le problème formulé ci-dessus. En général, des informations relatives aux ressources (nombre, capacité, vitesse d'exécution, disponibilité au cours du temps...) ou aux activités (dates d'exécution au plus tôt ou au plus tard, dates d'échéance préférées, contraintes de précédences, durée dépendant de la ressource...) viennent ajouter des contraintes au problème. Les fonctions objectifs mettent en général en jeu les dates de fin d'exécution des activités,

pénalisant l'avance ou le retard, les coûts d'exécution, qui peuvent varier d'une ressource à l'autre...

Méthodes classiques de résolution. La résolution d'un problème d'ordonnancement est une tâche réputée ardue. Situé à un niveau très détaillé, les contraintes qu'il se doit de considérer sont souvent très spécifiques et il n'existe pas de modèle particulièrement adapté dans le cas général. Les problèmes faciles sont généralement résolus par des algorithmes de listes. La programmation dynamique permet souvent de traiter des cas NP-difficiles au sens faible. Bien qu'utilisée parfois pour traiter certains problèmes d'ordonnancement assez purs, la programmation linéaire (en variables continues et/ou binaires) devient très vite inapplicable face aux difficultés rencontrées. On peut trouver des heuristiques basées sur l'approche linéaire, mais dans ce cas, on ne peut parler de résolution complète du problème par programmation linéaire. Pour des problèmes quelconques, la programmation par contraintes (PPC) qui permet d'exprimer des contraintes de tous types (non-nécessairement linéaires, logiques...) est une méthode très répandue. Un solveur de contraintes dédié ou générique (e.g. ILOG Solver) est alors nécessaire. Enfin, les métaheuristiques (e.g. algorithmes génétiques) se sont révélées comme des approches très efficaces en pratique.

Il n'est ni pertinent ni raisonnable d'effectuer ici un état de l'art des problèmes d'ordonnancement tant cette discipline est intensivement et extensivement étudiée. En reprenant la notation de Graham [55], une classification des variantes du problème peut être réalisée en fonction de l'environnement machine, des caractéristiques des activités et du critère d'optimalité considéré. Les ouvrages de Leung [80], Pinedo [98] et Brucker [20] dressent un panorama très complet des problèmes d'ordonnancement, de leur complexité et des méthodes de résolution. Nous redirigeons le lecteur vers ces ouvrages de référence pour de plus amples détails.

Contexte de l'industrie de production. L'étude de problèmes d'ordonnancement dans ce domaine particulier met en évidence de nouvelles contraintes. En général, une activité à ordonnancer a besoin d'une certaine quantité d'ingrédient(s) disponible au moment d'être exécutée, qu'elle transforme pour obtenir une certaine quantité d'un nouveau produit. Les niveaux de stock doivent donc être contrôlés à tout instant du temps pour s'assurer qu'on ne consomme pas plus que ce qui est disponible. En outre, la consommation et la production par une activité peut se faire continuement, au cours de son exécution, ou au contraire, en une fois, au début ou à la fin de l'exécution, cet aspect intervient naturellement dans le contrôle des stocks. Par ailleurs des contraintes sanitaires imposent que les resources soient nettoyées régulièrement. Il en résulte l'introduction d'activités spécifiques, occupant la ressource qu'elles nettoient, au cours de l'ordonnancement, rendant alors impossible l'exécution d'activités de production. Les règles régissant l'exécution de ces activités de nettoyage sont de divers types et varient d'une usine à l'autre, et même d'une ressource à l'autre : elle peut dépendre de la séquence des activités, de la durée écoulée ou du nombre d'activités exécutées depuis le dernier nettoyage. On peut aussi

trouver des cas où la durée d'exécution des activités dépend du temps écoulé depuis le dernier nettoyage (*"deteriorating job"*). La prise en compte de ces contraintes peu classiques sont autant de nouveaux défis pour les concepteurs d'algorithmes traitant les problèmes d'ordonnancement dans le domaine de l'industrie de *process*.

Par ailleurs, on trouve beaucoup de procédures dédiées à un problème particulier issu de la problématique de l'optimisation de la production. La question y est bien souvent posée à un niveau global et les concepteurs envisagent en général de résoudre le problème de planification et celui d'ordonnancement en coopération très étroite. Nous reviendrons donc sur ce sujet dans la Section 1.2.4 traitant de la coordination entre planification et ordonnancement.

Ordonnancement dans les APS. Nous l'avons dit, les problèmes d'ordonnancement, particulièrement lorsqu'ils s'inscrivent dans une réalité pratique mettant en jeu de multiples contraintes classiques et exotiques, sont des problèmes difficiles (au sens général comme au sens de la complexité). Il est courant que l'élaboration de solutions logicielles soit motivée, d'abord et avant tout, par l'obtention d'une solution réalisable prenant en considération les contraintes principales posées par l'usine. Dans ces cas, heuristiques, algorithmes de listes, algorithmes à base de règles sont bien souvent mis en œuvre. Dans ces approches, l'aspect optimisation est alors souvent ignoré (FuturMaster, OMPartners, INFOR, SAP APO — module PP/DS *Production Planning and Detailed Scheduling*). Lorsqu'au contraire un effort de généricité et d'optimisation est fait au cours de l'élaboration du module d'ordonnancement, les méthodes classiques de Recherche Opérationnelle sont alors employées. L'utilisation de la PPC est très répandue (ILOG PPO, INFOR, OMPartners). ILOG Scheduler permet de modéliser des problèmes d'ordonnancement pouvant être très complexes. Un modèle de contraintes est alors formulé et sa résolution est effectuée par le solveur de contraintes ILOG Solver (ILOG PPO, INFOR). ILOG PPO intègre en outre des heuristiques de recherche locale, plus spécifiques, à la recherche classique du solveur, afin de mieux prendre en compte les critères d'optimisation. Enfin, des méta-heuritiques peuvent aussi être intégrées, PP/DS de SAP implante des algorithmes génétiques [40], OMPartners fait aussi appel à ces méthodes.

1.2.4 Coordination des niveaux tactiques et opérationnels

Nous l'avons signalé en Section 1.1.3, un APS est un système dont le rôle est de résoudre un ou des problème(s) issu(s) de la problématique de l'optimisation de la production. Nous nous intéressons dans cette section aux différentes façons de mettre en relation les niveaux tactique (planification) et opérationnel (ordonnancement) dans l'objectif d'obtenir de meilleures solutions d'un point de vue global. Nos recherches nous ont conduit à distinguer deux types d'approches : celles traitant d'un problème précisément décrit, et les approches qui se veulent plus génériques. Bien que notre travail de thèse s'inscrive dans le cadre de l'élaboration d'un APS générique, s'appliquant à différents cas pratiques, nous nous sommes intéressés aux algorithmes dédiés afin de mettre en évidence leurs atouts et leurs limites.

L'étude de problèmes généraux de planification et d'ordonnancement est une tâche ardue. De nombreux travaux s'y rapportent, et dans des domaines divers. De cela résulte que différentes terminologies, notamment dans le cas de la planification, ont été introduites pour désigner les mêmes notions, et il n'est pas toujours aisé d'identifier avec exactitude le problème concerné par une étude particulière. Néanmoins, nous tentons dans ce qui suit d'étudier des problèmes de planification et ordonnancement issus de différents contextes ; en faire une étude exhaustive n'étant pas raisonnable dans le cadre de nos travaux, nous proposons quelques approches permettant d'appréhender les difficultés sous-jacentes.

Dans un premier temps nous présentons des modèles monolithiques dans lesquels planification et ordonnancement sont totalement intégrés dans une unique formulation. Des approches bi-modulaires où planification et ordonnancement se coordonnent étroitement, tout en conservant une certaine indépendance au sein du système, sont ensuite abordées. À cette occasion, nous étudions des modèles traitant de problèmes classiques ou spécifiques. Après avoir mis en évidence les points critiques motivant notre recherche, nous présentons alors l'approche hiérarchique totalement bi-modulaire mise en œuvre par les solutions logicielles du marché. Ce modèle flexible pouvant traiter des problèmes complexes et, cependant, selon une approche générique, fera l'objet de notre dernière étude.

1.2.4.1 Approche monolithique intégrée

Les modèles s'inscrivant dans cette approche ont trait à la planification et à l'ordonnancement dans le sens où ils déterminent simultanément la taille des batches et leur séquence sur les ressources dans un même problème résolu en une étape. En effet, si l'on définit la planification comme le processus de prise de décisions relative aux quantités de production, et l'ordonnancement comme celui de placement d'activités dans le temps et sur des ressources, les approches monolithiques peuvent alors se rapporter aux deux domaines. Les deux conceptions sont, de ce point de vue, défendables : en fonction du contexte, la planification peut ne considérer que des décision de production globale au cours du temps ou bien décider des batches physiques exécutés dans l'usine. Le premier cas est en général traité par des modèles macropériodiques et dont la formulation intègre rarement le problème d'ordonnancement ; cela sera étudié dans les sections suivantes. Au contraire, le second cas, en général formulé avec des modèles temporels micropériodiques voire continus, traite souvent le problème d'ordonnancement détaillé, associé au cas étudié. Ces modèles sont l'objet de cette section. En outre, la question du dimensionnement des batches est centrale dans notre thèse ; en ce sens, il se positionne sans ambiguïté à l'intermédiaire entre planification et ordonnancement, il est donc tout à fait naturel de s'attarder sur ces problèmes.

Planification micropériodique et ordonnancement. Nous avons distingué en Section 1.2.2 les modèles de *lot-sizing* micro et macropériodiques, en signalant que les modèles à courtes périodes considèrent en général des contraintes du problème d'ordonnancement. Cela provient

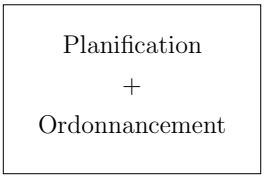

FIG. 1.2 – Résolution intégrée (monolithique) du problème général

du fait que la modélisation micropériodique permet de déterminer la séquence des batches (équivalents aux activités dans la terminologie de l'ordonnancement), de tenir compte d'éventuelles contraintes sur leur taille, d'évaluer avec exactitude les setups car elle n'autorise en général la production que d'un ou deux produits par ressource par période, c'est-à-dire de considérer un degré de détail tel que le problème d'ordonnancement y est résolu.

Dans [112], Suerie propose un programme linéaire en nombres entiers (MILP) pour le problème PLSP à une resource avec diverses contraintes sur les tailles de batches : multiple d'une taille prédéfinie, taille minimale ou maximale. Un solveur résout le programme ainsi formulé. Pour le PLSP à plusieurs niveaux et plusieurs produits, un algorithme génétique est développé par Kimms [71]. Un problème de dimensionnement et ordonnancement de batches est posé par Blömer et Günther dans [15] et [16]. Ils considèrent deux jeux de données issus de l'industrie chimique (Kondili *et al.* [72] et Westenberger et Kallrath [68]), intensivement étudiés dans le domaine, où sont définis plusieurs produits, plusieurs niveaux et où de nombreuses contraintes spécifiques doivent être satisfaites. Un MILP est formulé mais sa taille rend le problème intraitable par un solveur et ils développent alors des heuristiques à base de décomposition utilisant la relaxation linéaire du programme grâce auxquelles ils obtiennent de bons résultats. Toujours dans le domaine de la chimie, Maravelias et Grossmann se sont eux aussi intéressés à des problèmes de dimensionnement et ordonnancement de batches. Dans [86], ils proposent aussi une formulation en temps discret pour des problèmes présentant des contraintes sur les tailles de batches et sur les politiques de stockage des produits. Un MILP résulte de leurs travaux, cependant, la taille du modèle résultant ne permet d'aborder que des problèmes de petites taille. Nous revenons dans la suite sur d'autres approches qu'ils ont développées pour résoudre ce type de problèmes.

Autres modélisations du temps. Les approches où le temps est modélisé par micropériodes induit un grand nombre de variables et contraintes. Dans les approches permettant une haute précision de la formulation, le temps continu apparaît alors comme une modélisation possible et moins coûteuse puisque le nombre et la durée des intervalles de temps n'y est pas défini à l'avance. À ce sujet, Maravelias et Grossmann comparent formulations discrètes et formulations continues dans [89], pour des programmes linéaires en nombres entiers ayant pour but de formuler un même problème. Ils présentent l'approche discrète comme un cas particulier de l'approche continue, menant en théorie vers de moins bonnes solutions. Cependant, l'approche continue présente elle-aussi des difficultés de résolution en raison la faiblesse de sa relaxation linéaire.

Ils présentent un modèle continu dans [87] qui permet de résoudre, à nouveau, des instances de petite taille. Dans un domaine plus général, Jodlbauer [66] propose un programme non-linéaire pour le problème DLSP, en temps continu.

La subdivision du temps en longues périodes peut tout de même parfois être envisagée dans le cadre d'une résolution monolithique des problèmes de planification et d'ordonnancement. Gupta et Magnusson [57] proposent un MILP pour le problème CLSP multi-produits, avec des temps et coûts de setup dépendants de la séquence des produits. Les contraintes relatives à l'établissement de cette séquence, au sein d'une période donnée, sont donc toutes exprimées dans le programme. Cela mène à une formulation d'une taille telle qu'un solveur n'est pas en mesure de la traiter. Ainsi, les auteurs ont développé des heuristiques pour résoudre le problème.

Conclusion. Les approches monolithiques présentent l'avantage incontestable de formuler le problème intégré avec exactitude. Toutes les contraintes peuvent être exprimées simultanément. Cependant, dès lors que l'on sort du domaine des problèmes purs, relativement simplifiés de par les contraintes qu'ils considèrent et donc souvent artificiels, une résolution exacte du problème complet est inenvisageable car sa taille devient vite prohibitive. Des heuristiques sont alors élaborées, ce qui peut conduire à une sous-optimalité de la solution, on perd alors l'avantage de la formulation intégrée. C'est pourquoi, dans de nombreuses approches, le problème complet est décomposé de manière à faciliter sa résolution. En outre, de multiples méthodes peuvent être couplées, ce qui permet de profiter de différents atouts, souvent complémentaires.

1.2.4.2 Modèles décomposés fortement coopératifs

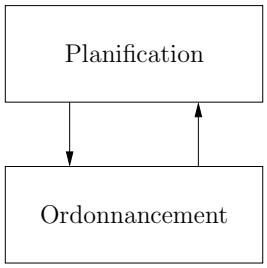

FIG. 1.3 – Résolution coopérative (par alternance) du problème général

Planification à séquence fixée. Dans [35] Dauzère-Pérès et Lasserre proposent un modèle totalement intégré pour le problème de planification et d'ordonnancement. Cependant l'approche de résolution consiste en la définition de deux sous-problèmes résolus de manière coopérative. Le problème global s'inscrit dans une chaîne de production multi-niveaux de type *job shop* (ce cas est particulièrement abordé dans [36]) où l'exécution d'un batch dans une période induit un coût fixe indépendant de la séquence ou de la quantité produite. Il s'agit alors de déterminer les

quantités de production de chaque produit par période, c'est-à-dire de décider de la production et, le cas échéant, de la taille de chaque batch par période pour un produit donné. Le schéma itératif décrit dans [38] est le suivant : le problème de dimensionnement des batches se fait à séquence fixée, leur taille est alors déterminée, puis un problème de séquencement est à nouveau résolu, à tailles de batches fixées. L'objectif de cette approche est de déterminer un plan de production (c'est-à-dire un ensemble de batches non-ordonnancés) réalisable du point de vue de la capacité des ressources. Lorsque la procédure itérative a convergé (il existe une séquence possible pour les activités ainsi dimensionnées), le problème complet d'ordonnancement est alors résolu. Wolosewicz, dans son travail de thèse, reprend ce modèle en résolvant le problème de planification par une méthode de relaxation lagrangienne et celui d'ordonnancement par une métaheuristique (recuit simulé) [131].

Programmation linéaire et programmation par contraintes. D'autres approches coopératives ont été étudiées et proposées pour intégrer les problèmes de planification et d'ordonnancement. Timpe, dans [124], présente une architecture de ce type permettant de prendre des décisions à la fois de dimensionnement des batches, de leur placement sur les ressources et de leur séquencement temporel (en cas de partage d'une même ressource). Un programme linéaire en variables mixtes (MILP) résout un problème de *lot-sizing* à capacité finie prenant en compte les setups. Ce MILP est résolu par *branch and bound*. Tous les trois nœuds de l'arbre d'exploration, une affectation des activités aux machines pour la période courante est réalisée et un problème de séquencement est alors posé. Pour résoudre ce problème de séquencement, un modèle de programmation par contraintes (PPC) est construit puis résolu. Trois issues sont alors possibles :

1. on trouve une séquence réalisable pour l'affectation proposée sur cette période, on en tire alors les informations de setup que l'on transmet au MILP dont on continue la résolution pour le traitement des périodes suivantes,
2. on ne trouve pas de solution d'ordonnancement réalisable mais le modèle PPC constate que l'affectation courante peut être rendue faisable en ajoutant localement (à ce nœud) une contrainte au MILP (le nœud courant est alors résolu à nouveau),
3. l'affectation courante n'est toujours pas faisable et ne peut l'être, elle est purement et simplement retirée de l'espace des solutions.

Ce modèle combinant MILP et PPC a été testé sur des problèmes provenant directement de l'industrie chimique. Les résultats concernant le temps de calcul et la qualité des solutions trouvées sont, selon l'auteur, difficiles à estimer, cependant, des solutions acceptables ont pu être trouvées sur des instances qui, auparavant, restaient non-résolues.

Modélisation continue du temps aux niveaux tactique et opérationnel. Toujours, dans le domaine de la chimie, un autre modèle coopératif faisant usage de la programmation linéaire et de la PPC est proposé par Maravelias et Grossmann dans [88, 85]. À nouveau, ils

utilisent une modélisation continue du temps. Il s'agit de résoudre dans un premier temps un problème d'affectation des batches aux ressources (MILP), puis d'ordonnancer ces batches sur les ressources (PPC). L'objectif ici peut être de trois types : la minimisation de la durée totale de l'ordonnancement (*makespan*) ou du coût total de production pour un ensemble fixé de demandes, ou encore la maximisation du profit dans un horizon de temps donné. On est donc dans une situation où le problème se place à plus court terme que dans l'exemple précédent, qui traitait plusieurs périodes. La coopération s'effectue par contre de façon similaire, le MILP, une fois résolu, propose au module de PPC une affectation des activités et teste ainsi la faisabilité de la solution. Si celle-ci n'est pas réalisable, elle est retirée de l'espace des solutions. En outre, par des propriétés de bornes fournies par le MILP d'une part et par le module de PPC d'autre part, on peut être en mesure de prouver l'optimalité des solutions fournies.

Domaine de l'industrie de process. Dans le domaine plus général de l'industrie de *process*, van Beek et Günther ont rassemblé dans [54] un ensemble d'articles relatifs à la problématique de planification et ordonnancement. On y trouve notamment l'article de Kallrath [68] qui décrit un benchmark inspiré d'un processus de production réel de l'industrie chimique allemande. Ce benchmark très complet présente une complexité qui ajoute à son intérêt. L'auteur présente une revue des problèmes de planification, de ceux d'ordonnancement, et des approches rencontrées pour résoudre le benchmark présenté. Blömer et Günther, dans [15], étudient cette même instance de Westenberger et Kallrath [68] en présentant un programme linéaire en variables mixtes qui tient compte de multiples contraintes. Ils proposent aussi deux heuristiques basées sur la programmation linéaire afin de résoudre des problèmes de taille réelle (ils étendent pour cela l'instance de [68] en en modifiant les valeurs numériques). Dans [16], les mêmes auteurs reviennent sur ces travaux en présentant une heuristique en deux phases à nouveau basée sur la programmation linéaire, qu'ils testent alors sur deux benchmarks — celui de Westenberger et Kallrath [68] et celui de Kondili *et al.* [72] — , pour lesquels ils définissent des ensembles d'instances en changeant à nouveau les valeurs numériques. Neumann, Schwindt et Trautmann se sont aussi intéressés à l'instance décrite dans [68]. Ils développent dans [94] un algorithme de résolution en deux phases, la première consistant à déterminer la taille des batches de façon à minimiser le temps total moyen d'occupation des ressources (qui dépend du nombre total de batches), la seconde consiste à résoudre le problème d'ordonnancement (affectation des ressources et des dates de début et de fin, pour chaque batch), dans lequel on optimise une fonction régulière. La modélisation du premier problème comme un programme linéaire en variables mixtes mène vers une résolution par un solveur classique tandis que la solution au second problème est déterminée par une procédure de recherche par faisceaux (*beam search*). Les auteurs comparent les performances de cette méthode décomposée à celles de la méthode monolithique proposée par Blömer et Günther dans [15]. La technique proposée s'avère alors systématiquement plus rapide et mène à des solutions meilleures ou équivalentes.

Conclusion. La recherche d'un optimum global, pour les problèmes de planification et d'ordonnancement, a poussé les chercheurs à définir et développer de nouveaux modèles : intégration de nouvelles contraintes et caractérisation des critères d'optimisation sont les principales questions posées lors de la modélisation. En effet, il n'est pas raisonnable, hormis sur des cas artificiellement simples ou de petite taille, de tenter de résoudre les deux problèmes simultanément. Il faut donc identifier les contraintes les plus pertinentes à ajouter aux modèles indépendants pré-existants de planification ou d'ordonnancement, afin d'intégrer certains aspects d'un des problèmes dans la modélisation de l'autre. En effet, la fusion de deux problèmes déjà complexes conduit à de nouveaux problèmes plus complexes encore et la question de la résolution se pose dans un second temps ; on assiste alors à l'émergence de modèles coopératifs, faisant appel à diverses méthodes de résolution (programmation linéaire, relaxation lagrangienne [131], programmation par contraintes, heuristique ou métaheuristique [131, 71]).

Ces travaux ont permis l'obtention de très bons résultats, tant sur le plan de la qualité des solutions que sur celui des temps de calcul. Cependant, les modèles proposés sont en général dédiés à la résolution d'une instance ou d'un benchmark particulier et la stabilité de leur comportement ne peut être assurée dans la perspective de résolution de problèmes plus généraux. De surcroît, la modélisation d'une contrainte additionnelle n'est pas toujours possible. Cela résulte en partie du fait qu'intégrer planification et ordonnancement, dans un cas particulier, nécessite l'étude approfondie du problème que l'on projette de résoudre, et qu'on peut alors extraire de sa structure des caractéristiques et propriétés spécifiques, qui, une fois exploitées, mènent à de très bonnes modélisations du problème. Concernant les cas les plus purs (contraintes classiques), les modèles de Dauzère-Pérès et Lasserre [36] et Wolosewicz *et al.* [131], par exemple, se concentrent sur des problèmes qui restent structurellement simples (dans le premier cas, il s'agit du *job shop*, dans le second, des précédences quelconques entre opérations peuvent être considérées) ; à nouveau, la nature de ces problèmes permet l'élaboration de procédures de résolution efficaces mais exploitables dans peu de cas pratiques.

Néanmoins, toutes les approches examinées ici semblent mettre en évidence le fait que dans une perspective de cohérence, de compatibilité et d'optimisation, l'intégration de la planifiaction et de l'ordonnancement passe par l'absorption de certaines contraintes d'un des problèmes dans le second. Les modèles coopératifs montrent une bonne efficacité sur les cas pour lesquels ils sont formulés ; cependant, il ne faut pas exclure ce type d'approche dans une perspective plus globale, les modules intervenant dans une éventuelle coopération n'étant pas nécessairement les problèmes de planification ou d'ordonnancement eux-mêmes, mais des problèmes plus composites. Un module coopératif peut par exemple constituer une entité d'un système hiérarchique et séquentiel. Ces architectures modulaires et hiérarchisées sont en effet celles que l'on trouve dans les solutions logicielles. La flexibilité qu'elles offrent permet de traiter de manière générique divers problèmes en optimisation de production, bien qu'une spécialisation à un domaine particulier se dégage en général. En outre, une telle décomposition séquentielle permet aux décideurs d'intervenir entre chaque niveau afin de modifier, si nécessaire, la solution proposée (plan de production, batches, affectation de ressources, ordonnancement...), avant de la considérer comme

satisfaisante.

1.2.4.3 Connexion entre moyen et court termes dans les APS

Il l'a déjà été mentionné, les APS du marché sont en général composés de deux modules principaux, chargés de résoudre respectivement un problème de planification et un problème d'ordonnancement. Les aspects tactiques constituant des décisions majeures dans de tels systèmes, les modèles de planification implantés travaillent sur de longs horizons et sont basés sur des formulations classiques macropériodiques permettant de traiter efficacement des problèmes complexes. Ce choix de modélisation induit un plus grand risque d'incohérence entre le module tactique et le module opérationnel que dans les modèles intégrés ou fortement coopératifs. Dans cette section, nous mettons en évidence les points sur lesquels notre attention s'est portée au cours de la recherche de cohérence entre planification et ordonnancement au sein de l'APS ILOG PPO et étudions la façon dont ils sont traités dans les APS du marché.

Modèles bi-modulaires et résolution séquentielle. Planification et ordonnancement sont deux problèmes conduisant à la prise de décisions relatives à la production, considérant des aspects qui peuvent être communs (temporels, gestion de la capacité des ressources, niveaux d'inventaire, date de livraison, setup) ou non (quantité totale de production, taille des batches physiques, satisfaction de demande, précédences). Les contraintes qui se trouvent à l'intersection de la planification et de l'ordonnancement ne sont pas prises en compte de la même façon dans les deux modèles, en particulier si l'on considère un modèle macropériodique pour le niveau tactique. En effet, les contraintes temporelles et de capacité sont en général approximées ou agrégées en planification, alors qu'elles sont considérées de façon exacte en ordonnancement. De cela résulte que le traitement de ces deux problèmes de manière indépendante est préjudiciable dans une perspective d'optimisation globale du fonctionnement d'une usine. De fait, un plan de production, optimal de son point de vue, et un ordonnancement optimal ne sont pas forcément compatibles, un plan, optimal ou non, ne mène pas nécessairement à un ordonnancement optimal, ni même à un problème d'ordonnancement réalisable. Le Tableau 1.1 synthétise les principales différences entre planification macropériodique et ordonnancement, en ce qui concerne la prise en compte des contraintes.

Forts de ces constats, les chercheurs se sont donc dirigés, comme nous l'avons vu, vers des approches intégrées des deux problèmes. Cependant celles-ci n'offrent pas, à ce jour, la flexibilité et la robustesse nécessaires aux solutions logicielles génériques. Dans la pratique, la planification est une étape préliminaire à laquelle fait suite l'ordonnancement. Il est alors crucial de connecter ces deux modules de la manière la plus astucieuse possible, en fonction de l'environnement global dans lequel s'inscrit le problème. L'approche générale est d'absorber certaines contraintes de bas niveau dès le problème de planification, afin de réduire les disparités observées. Pour des raisons de complexité, il n'est aujourd'hui pas raisonnable de tenter de résoudre les deux problèmes

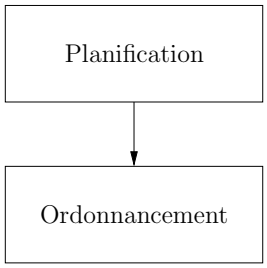

FIG. 1.4 – Résolution séquentielle bi-modulaire du problème général

	Planification	Ordonnancement
Temps	divisé en longues périodes	modélisé à chaque instant
Capacité	agrégée par période	considérée exactement
Niveaux d'inventaire	calculés aux bords des périodes	contrôlés à chaque instant
Dates de livraison	approximées par période	considérées exactement
Setups	approximés par période et ressource et/ou agrégés par produit	calculés exactement
Quantités de production	calculées globalement par période	données en entrée
Tailles des batches	généralement non prises en compte	considérées exactement
Satisfaction des demandes	planifiée par période	donnée en entrée
Précédences temporelles	non prises en compte	considérées exactement

TAB. 1.1 – Prise en compte des contraintes aux niveaux tactique et opérationnel

simultanément. Le plus souvent, ce sont les contraintes de capacité, de taille de batches et de setup qui sont alors considérées dès le niveau tactique.

Aspects retenus dans cette thèse. Nous nous sommes particulièrement intéressés au problème lié à la taille des batches. En effet, des contraintes économiques ou techniques imposent très souvent que celle-ci soit bornée inférieurement et/ou supérieurement. La durée des périodes définies dans la phase de planification est bien plus élevée que l'ordre de grandeur de la durée d'un batch respectant les bornes imposées sur sa taille. De cela résulte que les quantités de production planifiées par période ne tiennent pas compte de ces contraintes et la compatibilité entre le plan ainsi calculé et l'ordonnancement final n'est possible que si l'on procède à une étape d'identification des batches, soit au cours de la planification, soit en aval.

En outre, la modélisation de l'utilisation de la capacité des ressources est souvent à l'origine des incompatibilités entre plan et ordonnancement. Bien que focalisés sur la problématique de dimensionnement des batches, nos travaux s'attachent aussi à réduire ces incompatibilités. En effet, la prévision des périodes de temps où l'exécution est planifiée n'est pas toujours réalisable car les décisions à moyen terme sont prises au sein d'un modèle agrégé, alors que l'ordonnancement

considère une formulation exacte des contraintes.

Découpage en batches dans les APS. Tout APS possède sa propre façon de résoudre le problème de définition des batches physiques. Dans la suite, nous faisons référence à cette procédure par le terme de *batching*. Très souvent intégrées au calcul du plan de production, ces décisions n'y apparaissent pas comme cruciales et ne figurent pas parmi les critères optimisés. En général, lorsque la planification est résolue par un programme linéaire, une taille de batch est décidée à l'avance, après analyse du cas d'étude, et les quantités de production se contentent alors d'être des multiples de cette taille (FuturMaster, SAP APO dans SNP). Lorsque la méthode de résolution le permet, une certaine flexibilité peut être introduite, en exigeant seulement que la quantité totale soit telle qu'il existe une solution de *batching* respectant les bornes inférieure et supérieure sans imposer une taille fixe ; le découpage effectif se fait alors en sortie du module de planification de manière gloutonne (SAP APO dans PP de PP/DS) ou dans le module d'ordonnancement lui-même (OMPartners), et les décisions de découpage n'interviennent pas dans les critères d'optimisation.

Par ailleurs, le module en charge du *batching* peut dépendre du problème étudié, si l'APS le permet. Dans SAP APO, il est en effet possible de traiter le problème dès la planification (dans SNP) ou bien juste avant l'ordonnancement (dans PP de PP/DS). Dans cet APS, quel que soit le module concerné, différentes politiques communes peuvent être utilisées en fonction de l'usine considérée et des exigences de l'industriel : taille fixe (économique), lot-pour-lot (proche des algorithmes MRP I), taille minimale ou maximale [40]. De telles politiques de *batching* sont aussi disponibles dans les APS OMPartners et INFOR, et sont applicables par l'un ou l'autre module.

Cette étape nécessaire permettant la continuité entre planification à moyen terme et ordonnancement à court terme est en général dissimulée et ne fait jamais appel à des techniques d'optimisation à proprement parler. Des heuristiques très simples sont donc implantées pour résoudre le problème. Dans l'APS ILOG PPO, ce module (dit de *batching*) forme une étape à part entière s'intercalant entre la planification et l'ordonnancement et s'interfaçant avec ces deux modules. Le découpage des quantités de production planifiées en batches y est posé comme un véritable problème d'optimisation. La recherche de cohérence entre planification et ordonnancement, et l'objectif d'obtention de meilleures solutions pour l'ordonnancement final par cette nouvelle approche est le moteur de tous les travaux qui ont été menés au cours de cette thèse. Dans le chapitre suivant, nous nous intéressons à la définition précise que nous donnons à ce problème. À cette occasion, nous mettons en évidence les correspondances qui peuvent être faites entre notre sujet et les problèmes connus de *lot-streaming* et d'ordonnancement par batches. Cependant, avant cela, nous présentons le contexte industriel auquel nous nous intéressons dans notre thèse afin de bien définir notre cadre d'étude, puis nous formulons clairement les motivations et objectifs qui ont guidé nos travaux.

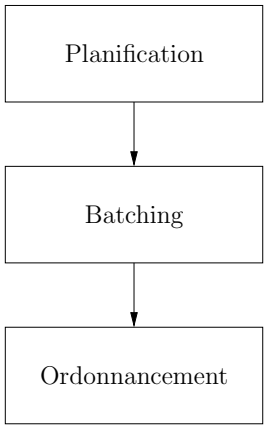

FIG. 1.5 – Résolution séquentielle tri-modulaire par l'APS ILOG PPO

1.3 Positionnement du travail de thèse

1.3.1 Environnement de production

Nous l'avons déjà évoqué, le domaine de l'industrie de production définit de multiples contextes. Dans cette thèse, nous nous concentrons sur l'industrie de production de type *process*, bien que l'intersection entre celle-ci et l'industrie de production discrète ne soit pas vide, et que, par suite, la plupart de nos résultats soient généralisables à n'importe quel environnement de production, pourvu que les contraintes qui le définissent soient prises en compte dans notre cadre d'étude. Nous manipulons donc des concepts généraux, communs à toute industrie de production. Ceux-ci sont définis dans une première section. Lorsque cela est nécessaire, des informations complémentaires relatives à l'industrie de *process* sont données. Nous présentons dans un second temps les spécificités liées à l'industrie de *process*, en particulier agroalimentaire, qui ont retenu notre attention au cours de nos travaux.

1.3.1.1 Concepts fondamentaux en industrie de production

Produits ou matériaux. Les industries de production transforment des matériaux. Ceux-ci peuvent être initiaux (matières premières entrant dans la chaîne de production), intermédiaires ou finaux (produits finis destinés aux clients). Un produit peut posséder des propriétés spécifiques (contraintes physiques) ou nécessiter des attentions particulières (contraintes sanitaires). Un temps de MATURATION peut être requis après la production afin d'obtenir effectivement le produit souhaité (pour à nouveau le transformer, si c'est un intermédiaire, ou le livrer, si c'est un produit fini). Il peut aussi être exigé qu'avant d'utiliser un produit, on le place en QUARANTAINE afin d'exécuter des tests bactériologiques et ainsi s'assurer de sa bonne qualité. Dans l'industrie de *process* et en particulier en agroalimentaire, on manipule des produits périssables, ils ont alors une DURÉE DE VIE maximale au sein de la chaîne de production (ce qui limite le temps autorisé

pour le stockage par exemple). Cela peut concerner tous les produits intervenant le long de la chaîne de production. Cependant, lorsqu'il s'agit de produits finis à livrer aux clients, cette durée de vie est définie de telle sorte que les dates de péremption soient suffisamment éloignées des dates de livraison ; le délai résulte alors non-seulement d'une contrainte physique mais aussi d'une négociation entre fournisseur et client.

Recettes. Il s'agit des procédés de transformation définis par une usine afin de réaliser sa production. Une recette utilise (consomme) des produits pour en fabriquer (produire) de nouveaux. Une recette peut être plus ou moins complexe selon le nombre de matériaux consommés ou produits (il peut n'y en avoir qu'un en entrée ou en sortie). Ainsi, une recette est définie par un multigraphe orienté où les nœuds sont des ACTIVITÉS génériques de production et les arcs, des relations de précédences entre ces activtités, modélisant en général des relations producteur/consommateur. Physiquement, ce sont des instances de ces activités qui réaliseront, dans l'usine, les opérations de transformation. En effet, il est important de noter que la définition d'un procédé de transformation est donnée pour une unité de recette exécutée, c'est donc une description normalisée d'un procédé de production. Pratiquement, la quantité réellement exécutée induira, en proportion, des quantités de consommation et de production de matériaux, ainsi que des temps et/ou capacité nécessaires à l'exécution de chacune des activités alors instanciées. Un BATCH d'une recette est l'exécution d'une certaine quantité de cette recette, il définit donc ces activités instanciées, dont temps et capacité requis sont proportionnels à la taille du batch (c'est-à-dire la quantité de recette exécutée). Par exemple, si une recette est définie par deux activités génériques dont les durées normalisées respectives sont 2 et 1, l'exécution d'un batch de 5 unités de la recette entraînera la création de deux activités instanciées, la première de durée 10, le seconde de durée 5. L'exécution d'un second batch de cette recette, mais cette fois-ci de taille 8, entraînera la création de deux nouvelles instances d'activités, la première de durée 16, la seconde de durée 8. Remarquons qu'en général, la durée d'une activité est une fonction affine de la taille du batch (et non-pas totalement proportionnelle comme dans cet exemple). Par ailleurs, la mise en production d'un batch entraîne, pour chacune des activités instanciées, un coût de production, qui est aussi une fonction affine de la taille du batch (cf. Chapitre 2, Section 2.2.1). En outre, en général, la taille de ces batches est bornée inférieurement et supérieurement.

Ressources. L'exécution des activités de production est réalisée par des ressources, en général, il s'agit de machines. Les activités de différentes recettes peuvent requérir les mêmes ressources. Inversement, une activité peut présenter des MODES alternatifs, ce qui laisse ouvert le choix de la ressource affectée. Ainsi, on pourra parfois définir un regroupement de ressources, en cas d'équivalence ; cependant, il se peut qu'une activité puisse être exécutée sur une ressource ou une autre mais avec différents coûts et en requérant différents temps et/ou capacités. Par ailleurs, les ressources ont une capacité donnée, cela doit être mis en relation avec le nombre d'activités pouvant être exécutées simultanément par une machine. Cette capacité peut varier au cours du temps, notamment, une ressource peut voir sa capacité réduite ou même être totalement

indisponible pendant une durée donnée. Des CALENDRIERS contiennent les informations relatives à ces variations.

Demandes. Une partie de la production réalisée au sein d'une usine est guidée par les demandes (ou commandes) des clients en produits finis. Dans notre étude, nous n'aborderons pas les aspects liés à l'incertitude pouvant être considérés dans la gestion des demandes, celles-ci sont des données déterministes de notre problème. Il s'agit donc de quantités fixées requises pour des dates de livraison négociées au préalable (dates dues). Ces dates n'induisent en général pas de contraintes dures (dans ce cas, la demande devrait être livrée impérativement avant la date butoir) mais par contre, des coûts d'avance et de retard peuvent être définis, ce qui place nos problèmes dans une problématique de livraisons "juste-à-temps". En outre, il peut être autorisé de satisfaire une demande en plusieurs livraisons, ainsi que de ne pas la satisfaire intégralement ; à nouveau, ces flexibilités sont négociées entre le fournisseur et le client.

1.3.1.2 Spécificités fonctionnelles considérées dans la thèse

Stockage. D'un point de vue physique, les produits sont stockés dans des tanks ou réservoirs. Il est interdit de mélanger des produits dans un même tank, mais il se peut qu'un tank puisse accueillir différents matériaux, pourvu que cela ne soit pas au même moment. De plus, la capacité de stockage d'un tank est finie. D'un point de vue global, on considère aussi le niveau d'inventaire des produits, qui est la quantité totale des matériaux actuellement en stock. Généralement, des préférences quant aux niveaux minimal (stock de sécurité) et maximal (fonctionnement type "flux tendu", gestion de la durée de vie d'un matériau) sont données. Ainsi, le maintien de niveaux de stock minimaux peut aussi intervenir dans les décisions de production d'une usine, ces stocks minimaux correspondant en réalité à l'anticipation d'une augmentation probable de la demande réelle.

Nettoyage. Des règles sanitaires peuvent parfois contraindre les ressources ou les tanks de stockage à être nettoyés. Cela peut être à fréquence régulière (après l'exécution d'un nombre donné d'activités, quelle que soit leur durée, ou lorsqu'une certaine durée s'est écoulée depuis le dernier nettoyage). La nécessité de nettoyage peut également dépendre de la succession des événements. Nous avons vu que, d'une part, les activités de production peuvent partager une même ressource, d'autre part, les produits peuvent être stockés dans les mêmes tanks. Cela nécessite que l'on nettoie ces ressources ou ces tanks entre deux utilisations de types différents. Les nettoyages dépendent alors de la séquence des activités exécutées sur la ressource ou des produits mis consécutivement en stock.

Setup. D'une manière général, comme une machine peut éventuellement exécuter des activités de types différents, il est parfois nécessaire d'opérer une reconfiguration de celle-ci. En effet, il peut se trouver que la machine possède différents états de fonctionnement ou de configuration et que, selon qu'elle doit exécuter une activité d'un type ou d'un autre, elle doit être paramétrée

dans un état ou un autre. Nous avons appelé ces changements de configuration "setups". Les temps et coûts induits par ces setups sont souvent cruciaux car longs et élevés, particulièrement dans les industries pharmaceutiques et chimiques. En agroalimentaire, la difficulté réside plus dans le fait qu'on est souvent obligé de changer de type d'activité pour ne pas violer les impératifs sanitaires (e.g. périssabilité). Dans tous les cas, les setups dépendent de la séquence des activités sur la ressource concernée.

Production continue et discrète. Dans l'industrie de *process*, les produits sont en général fluides et il n'est pas rare que la production ou la consommation de certains matériaux se fasse en continu, par opposition à un mode discret (cf. Figure 1.6). La production continue d'un matériau implique qu'il commence à être disponible (et peut-être mis en stock, s'il n'est pas immédiatement utilisé) dès le début de l'exécution de l'activité qui réalise cette production. Si une activité consomme un produit en continu, son exécution peut commencer avant que toute la quantité requise soit disponible (le produit peut être en cours de fabrication, de manière continue) ; si tout le produit était en stock au début de l'exécution, la baisse du niveau de stock se fait alors progressivement. Inversement, une activité produisant de façon discrète ne délivrera le matériau qu'à la fin de son exécution, et celui-ci ne sera pas disponible avant. L'exécution d'une activité consommant de manière discrète nécessite que la totalité de la quantité requise soit disponible dès le début de l'exécution.

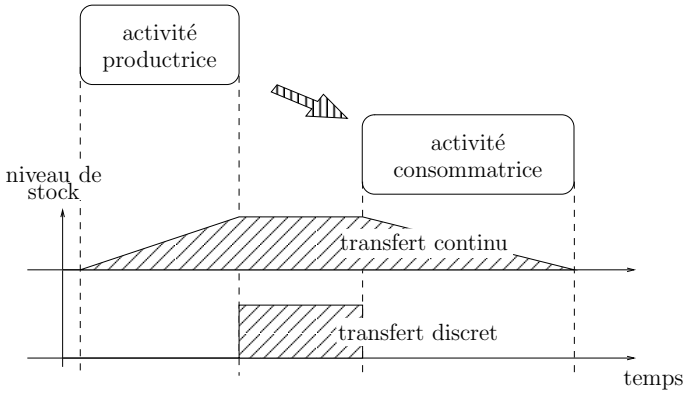

FIG. 1.6 – Évolution du stock en production continue ou disrète

1.3.2 Conclusion et mise en évidence des motivations et objectifs de la thèse

Dans ce chapitre nous avons étudié les problèmes de planification et d'ordonnancement de la production. C'est volontairement que nous en avons donné une vue d'ensemble afin de mettre en évidence les difficultés qu'ils sous-tendent et présenter les modèles et méthodes classiquement employés pour les formuler et résoudre. Dans ce contexte général de l'optimisation de la production à moyen et court termes, nous nous sommes intéressés à la coordination entre ces deux

niveaux de prises de décisions, tant du point de vue de problèmes spécifiques que de celui de cas plus généraux, au travers de l'élaboration de solutions logicielles génériques : les APS. C'est en effet dans cet environnement de travail que s'est déroulée notre thèse puisque nos travaux s'inscrivent dans le développement de l'APS ILOG PPO.

Dans notre étude, la coordination entre planification et ordonnancement passe par la définition des batches de production. L'observation d'APS du marché a permis d'établir que ce problème, dit de *batching*, y est certes une étape nécessaire mais traité de manière secondaire dans le processus d'optimisation globale de la production. Il est en général résolu de manière heuristique dans les solutions logicielles proposées et n'a pas été identifié comme un véritable problème d'optimisation combinatoire. En effet, ces décisions n'apparaissent pas comme cruciales dans une démarche de coordination entre planification et ordonnancement. La position d'ILOG a été, à l'inverse, de pressentir que ce problème, s'il était posé de manière plus fine et résolu par des méthodes d'optimisation efficaces, pouvait mener à de meilleures solutions finales. Le chapitre suivant décrit précisément le problème de *batching* que nous traitons dans cette thèse, et c'est à cette occasion que nous présentons des travaux réalisés sur des problématiques proches, souvent dans l'optique d'optimiser l'ordonnancement final. L'objectif de cette thèse est donc multiple, et néanmoins concentré autour du problème de *batching*. Il consiste à définir précisément ce nouveau problème, l'analyser d'un point de vue de sa complexité, et confirmer sa pertinence en tant que problème d'optimisation. Cette validation se fait à travers l'élaboration de méthodes de résolution adaptées et la réalisation d'études expérimentales.

Chapitre 2

Le problème couplé de *lot-streaming* et *pegging*, ou *"batching"*

Ce chapitre est consacré à la formulation du problème de *batching* qui a motivé nos travaux de thèse, c'est-à-dire le découpage de la production planifiée en batches (*lot-streaming*) et l'établissement des flots de produits (*pegging*). Dans une première partie, nous en faisons une description en langage naturel en montrant dans quelle mesure il peut être important de poser le sujet en termes de problème d'optimisation, nous mettons alors en évidence les intérêts qu'il présente d'un point de vue opérationnel. Nous poursuivons par la position formelle du *batching* en introduisant d'abord le problème global résolu par l'APS ILOG *Plant PowerOps* (PPO). Puis, la présentation de l'approche séquentielle en trois étapes adoptée nous permet d'extraire le rôle du *batching* comme module intermédiaire entre planification et ordonnancement. Nous en formalisons le problème *"cœur"*, qui nous occupera jusqu'au Chapitre 4. Cette définition précise nous permet, dans un troisième temps, d'établir les liens entre notre problématique et la thématique de l'ordonnancement par batch qui a déjà été bien étudiée dans le domaine de la Recherche Opérationnelle.

2.1 Description générale et intérêts opérationnels

Dans ce qui suit, nous donnons une description informelle du problème de *batching* afin que le lecteur appréhende la difficulté qu'il peut parfois contenir ainsi que les avantages que les industriels de la production peuvent tirer d'un "bon" *batching*. Nous illustrons nos propos avec un exemple volontairement simple.

2.1.1 Une quantité de production à partager

Position du problème. D'une manière très générale, le problème qui nous occupe consiste à découper une quantité totale de production en un certain nombre de batches physiques de production. Dans notre terminologie, cela signifie diviser en batches un nombre fixé d'unités

d'exécution d'une recette donnée. Par souci de simplification, nous manipulons dans ce chapitre des nombres entiers, cependant, en pratique, il peut s'agir de nombres réels. La quantité totale à découper a été évaluée de manière à satisfaire un certain nombre de demandes dont chacune requiert une quantité de produit fini fabriqué par la recette en question.

Exemple. Nous considérons une unique recette permettant de fabriquer un litre de yaourt par unité de recette exécutée, les ingrédients nécessaires ne sont pas modélisés. La ressource permettant d'exécuter l'unique activité de production de la recette est un mélangeur placé dans une cuve. La capacité de cette cuve est de 20 litres. En outre, l'hélice du mélangeur, placée au fond de la cuve, nécessite qu'une quantité minimale de 5 litres se trouve dans la cuve pour pouvoir fonctionner (l'hélice doit tremper intégralement pour pouvoir être mise en rotation). Ces contraintes impliquent que la taille des batches de notre recette doit être comprise entre 5 et 20 unités. Le yaourt n'est disponible qu'à la fin de l'exécution d'un batch. Il n'est pas nécessaire d'indiquer, pour l'exemple, la durée normalisée de l'activité générique qui est ici proportionnelle à la quantité. Considérons que la durée d'une activité instanciée est égale à la taille du batch qui l'instancie. Dans cet exemple, 5 demandes, $dem_1, \ldots dem_5$, indépendantes requérant respectivement $rq_1 = 2$, $rq_2 = 5$, $rq_3 = 3$, $rq_4 = 15$ et $rq_5 = 5$ litres de yaourt doivent être satisfaites et livrées pour des dates dues distinctes dt_d, pour d allant de 1 à 5, que l'on suppose triées selon d (cf. Figure 2.1).

L'exécution de 30 unités de la recette a été planifiée et le problème de *batching* consiste alors à définir des batches de production respectant les contraintes imposées sur les tailles (entre 5 et 20 unités), et ce de la meilleure manière possible. Le découpage est nécessaire afin de satisfaire des *contraintes*, il peut aussi s'avérer avantageux dans une optique d'*optimisation de critères* qui sont présentés dans la suite, par exemple si dem_2 doit être livrée avec un minimum de retard.

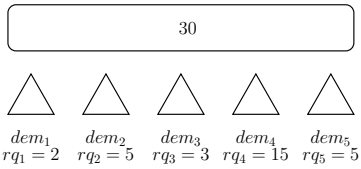

FIG. 2.1 – Exemple pour le problème de *batching*

Problème de *lot-streaming***.** L'action de subdiviser une grosse quantité en plusieurs batches est appelée dans la littérature *lot-streaming*. Ce problème est en général traité dans des contextes de flow-shop [60, 31, 6, 127, 53] ou de job-shop [37], afin de diminuer le temps total d'exécution d'un processus. Diverses techniques de résolution de cas simples (en général polynomiaux) ont été étudiées [125, 7], mais l'optimalité d'une solution est étroitement liée au contexte dans lequel le *lot-streaming* est réalisé. Nous allons voir en Section 2.2.4 que notre approche propose un

critère d'optimalité très éloigné de ce que l'on peut trouver dans la littérature, ce qui rend les études référencées ci-dessus peu exploitables.

Sur notre exemple, si l'on s'en tient au strict point de vue du respect des contraintes, plusieurs découpages des 30 unités de production en batches de taille valide sont possibles (en l'absence des contraintes d'intégrité des tailles des batches, il en existe même une infinité). On peut par exemple définir 6 batches, tous de taille 5 (taille minimale Figure 2.2.a), ou 2 batches, un de taille 20 et un de 10 (taille maximale Figure 2.2.b), ou encore 3 batches de taille 10 (taille fixe Figure 2.2.c). Les critères d'optimisation vont permettre de guider ces décisions.

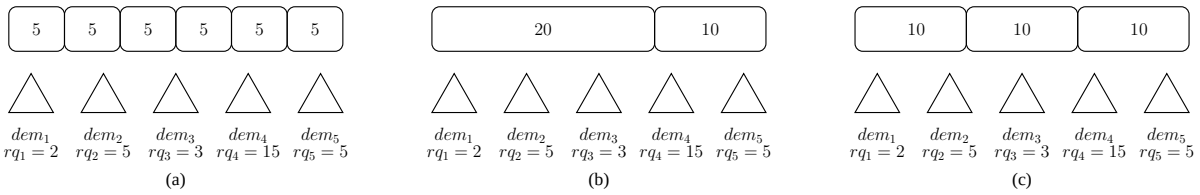

FIG. 2.2 – Exemple : problème de *lot-streaming*

Problème de *pegging***.** Au sein de notre problématique générale qui consiste à élaborer un APS efficace et performant, l'objectif global est de produire un bon ordonnancement final. Dans notre contexte, les critères principaux optimisés dans ce dernier module sont inspirés de ceux du "juste-à-temps", c'est-à-dire des pénalités d'avance et de retard des livraisons de chaque demande, relativement à la date due associée. Nous l'avons dit, la production d'un batch n'est délivrable qu'à la fin de l'exécution de celui-ci. Or, la durée d'un batch étant proportionnelle à sa taille, il est pertinent de décider, en amont de l'ordonnancement, des batches qui serviront chaque demande, afin d'estimer par anticipation les coûts encourus. Ainsi, un second problème est introduit lors de l'étape de *batching*. Couplé à celui de *lot-streaming*, le problème de *pegging* consiste alors à prendre des décisions sur les flux de produits circulant entre les batches et les demandes. Dans le cas général, un batch peut être lié à plusieurs demandes et réciproquement, une demande peut être satisfaite par différents batches. La terminologie *pegging* provient de [40, Chapitre 3, Section 6] où ce terme est employé pour décrire le transfert de matériaux entre les batches et les demandes. De le même façon, dans cette thèse, nous définissons l'existence d'un "arc de *pegging*" comme celle d'un transfert d'une quantité non-nulle de produit entre un batch et une demande. Cependant, cette notion se rapproche plus de celle d'ordonnancement par batch, problématique dans laquelle il s'agit de regrouper des jobs (qui correspondent à nos demandes) dans des batches afin de les exécuter sur des machines spéciales fonctionnant exclusivement par batch (et non par job). Nous reviendrons à la fin de ce chapitre en Section 2.3, sur les similitudes et disparités que l'on peut distinguer entre notre problème général de *batching* et ceux d'ordonnancement par batch.

Sur l'exemple, les décisions pour les flots circulant entre les batches et les demandes ne

sont pas triviales, même à découpage en batches fixé. La Figure 2.3 propose deux affectations possibles pour les flots de produits entre batches et demandes, à partir du découpage proposé au paragraphe précédent, Figure 2.2.a. À nouveau, les critères définissant une bonne solution vont orienter ces décisions.

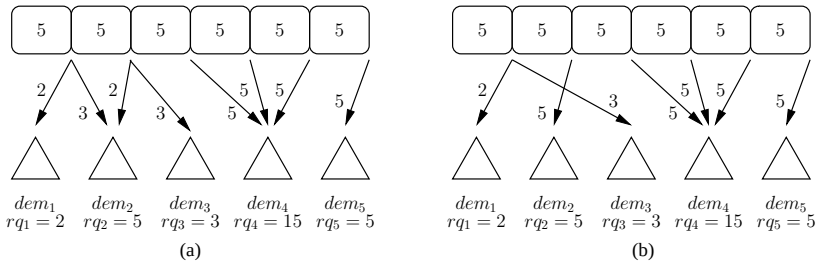

FIG. 2.3 – Exemple : problème de *pegging*

Problème couplé. Le problème complet consiste donc à définir d'une part des batches, dont les tailles satisfont les contraintes, afin de réaliser la quantité de production planifiée, d'autre part le flot de produit entre ces batches et les demandes. La structure du problème de *batching* est donc double, ce qui engendre naturellement sa combinatoire, Figure 2.4.

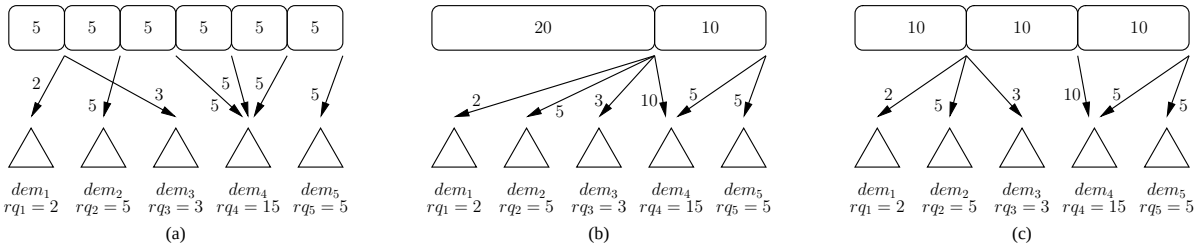

FIG. 2.4 – Exemple : problème couplé de *lot-streaming* et *pegging*

2.1.2 Définition informelle d'une bonne solution

2.1.2.1 Intention générale

Comme nous l'avons dit, une solution au problème de *batching* a pour rôle de pouvoir poser en aval le problème d'ordonnancement, en définissant des batches de production à ordonnancer, ceci dans une perspective d'obtention d'ordonnancements de qualité. De cela résulte que nous avons orienté le critère de qualité du *batching* relativement à la solution d'ordonnancement finalement obtenue. Ce souci d'obtention d'un ordonnancement de qualité est l'élément clé permettant de juger d'une solution de *batching*. Ainsi il n'est pas pertinent de définir de fonction

objectif pour le problème général de *batching*. Toutefois, nous en spécifions une pour le problème de *batching* traité dans cette thèse, ce qui nous permet de comparer différentes approches de résolution pour ce problème particulier. Cependant, même en l'absence de fonction objectif formellement définie pour le problème général nous pouvons confronter différentes méthodes de résolution en comparant la qualité de la solution obtenue par le module d'ordonnancement (cf. Section 1.2.4.3) qui agit comme un oracle, et non-pas celle du *batching* en tant que fin en soi. Bien que l'ordonnancement ne soit lui aussi qu'une approximation de la réalité, nous justifions ce choix par la précision intrinsèque au problème résolu dans ce module.

2.1.2.2 Orientation spécifique pour la thèse

Dans notre contexte, le problème d'ordonnancement minimise des pénalités liées à l'avance et au retard des livraisons. Il ne s'agit pas ici de minimiser le délai entre la date réelle de livraison et la date due, mais de minimiser le produit de cette durée par la quantité de produit livrée. En effet, cela est naturel puisqu'une demande peut être satisfaite par plusieurs batches et donc livrée en plusieurs fois. Notons que si un batch satisfait une certaine quantité d'une demande donnée, la livraison de cette quantité concorde avec la fin de l'exécution du batch en question.

En outre, comme il existe un coût de production correspondant à l'exécution d'un batch et que ce dernier se décompose en une partie fixe payée pour la création d'un batch et une partie variable proportionnelle à sa taille, nous introduisons un terme exprimant ces coûts encourus lors de l'étape de *batching*. Notons que seule la partie fixe de ces coûts intervient dans les décisions puisque la quantité totale à exécutée est connue à l'avance, et que, de ce fait, la partie variable est fixée. Dans la suite, les termes de coût de production et coût d'exécution sont employés indifféremment.

Cette optique est raisonnable car ces critères sont d'une part réalistes et effectivement optimisés dans la plupart des cas réels. D'autre part ils orientent les solutions de *batching* dans des directions opposées et sont donc difficilement conciliables ; nous détaillons ce point dans le paragraphe suivant et l'illustrons dans la section suivante. Ainsi, comme nous souhaitons définir une bonne solution de *batching* comme une solution menant vers un bon ordonnancement, nous allons dans cette thèse optimiser ces mêmes coûts dès le module de *batching*.

Contradiction des deux critères. Puisqu'il existe un coût fixe lié à la création d'un nouveau batch, la seule optimisation de ces coûts induirait des solutions dans lesquelles un mimimum de batches seraient créés, tous de taille maximale (dans la mesure du possible). Cependant, ce type de solution n'est pas du tout orientée vers une production "juste-à-temps", qui est le second critère que nous souhaitons optimiser, et même, ces deux critères s'opposent puisque la production "juste-à-temps" vise à faire des batches de taille adaptée aux besoins à un instant donné. En général, la taille résultant de cette politique ("lot-pour-lot"), ne prenant pas en compte les contraintes, n'est pas égale à la taille maximale d'un batch, au contraire, elle est même plus souvent inférieure à la taille minimale, qui est alors violée. Il s'agit donc de trouver un compromis entre la création d'un petit nombre de gros batches et celle d'un grand nombre de

petits batches, afin d'optimiser simultanément les coûts de production et ceux d'avance-retard, tout en respectant les contraintes liées à la taille des batches.

Coûts d'exécution. L'identification d'un terme modélisant les coûts de production dans une solution de *batching* est directe puisque les décisions sur le nombre et la taille des batches (qui induisent précisément ces coûts) sont prises lors de la résolution de ce problème.

Coûts d'avance-retard. Par opposition aux coûts d'excution, une estimation fine des pénalités d'avance-retard ne peut être effectuée car elles sont extrêmement liées aux choix qui seront faits dans l'ordonnancement. C'est pourquoi nous introduisons dans les modèles qui sont présentés dans cette thèse la notion de date d'exécution estimée pour chacun des batches, ce qui va nous permettre d'évaluer un délai s'écoulant entre la livraison d'une demande et sa date due, et ainsi de manipuler des informations temporelles. Si un arc de *pegging* est défini entre un batch et une demande, l'estimation d'une date d'exécution de ce batch nous permet ainsi de mesurer l'avance ou le retard de la livraison correspondante. On calcule alors le produit du délai entre la date de livraison et la date due par la quantité livrée pour obtenir un terme représentant une estimation du coût réel de l'ordonnancement ; en effet, dans le cas général, le batching ne résout pas le problème d'ordonnancement qui considère, lui, les contraintes liées aux ressources.

Synthèse : problème de *batching* **posé dans cette thèse.** Dans notre contexte, le problème de *batching* est donc posé comme celui du *lot-streaming* couplé à celui du *pegging*. Comme nous l'avons expliqué précédemment, le premier problème est responsable de la découpe en batches et le second de l'établissement des flux de produits entre batches et demandes. D'une part, le *lot-streaming* correspond au choix de la création d'un batch, induisant un coût fixe de production, et de son dimensionnement, induisant un coût variable proportionnel à la taille. D'autre part, le *pegging*, en établissant un flot de produit permet d'estimer un délai entre l'instant de production qui correspond à la livraison et la date due de la demande correspondante. De plus, le flot de produit circulant sur un tel arc de *pegging* permet de connaître la quantité transférée et ainsi, par multiplication avec le délai, d'obtenir un terme estimant celui de l'ordonnancement. En effet, ce sont ces décisions de *pegging* que l'on retrouve dans la fonction objectif de l'ordonnancement.

2.1.3 Un problème de décision difficile

En considérant l'exemple de la Figure 2.4, on observe que les décisions ne sont pas triviales dans la perspective d'optimisation de la fonction objectif définie ci-dessus. En supposant qu'il existe une solution au problème (cette question est traitée au Chapitre 3, Section 3.2.1), une solution optimale n'est pas nécessairement facile à déterminer. Les demandes étant triées relativement à leur date due, et ayant le même degré d'importance, idéalement, dem_{d-1} doit être livrée avant dem_d, pour d allant de 2 à 5. La quantité requise par chaque demande est relativement petite au vu des valeurs minimale et maximale pour la taille des batches de la recette ; différentes solutions pertinentes peuvent donc être envisagées selon que l'on privilégie l'optimisation des

coûts de production, des coûts d'avance-retard ou un équilibre des 2. La Figure 2.5 représente 6 solutions raisonnables possibles.

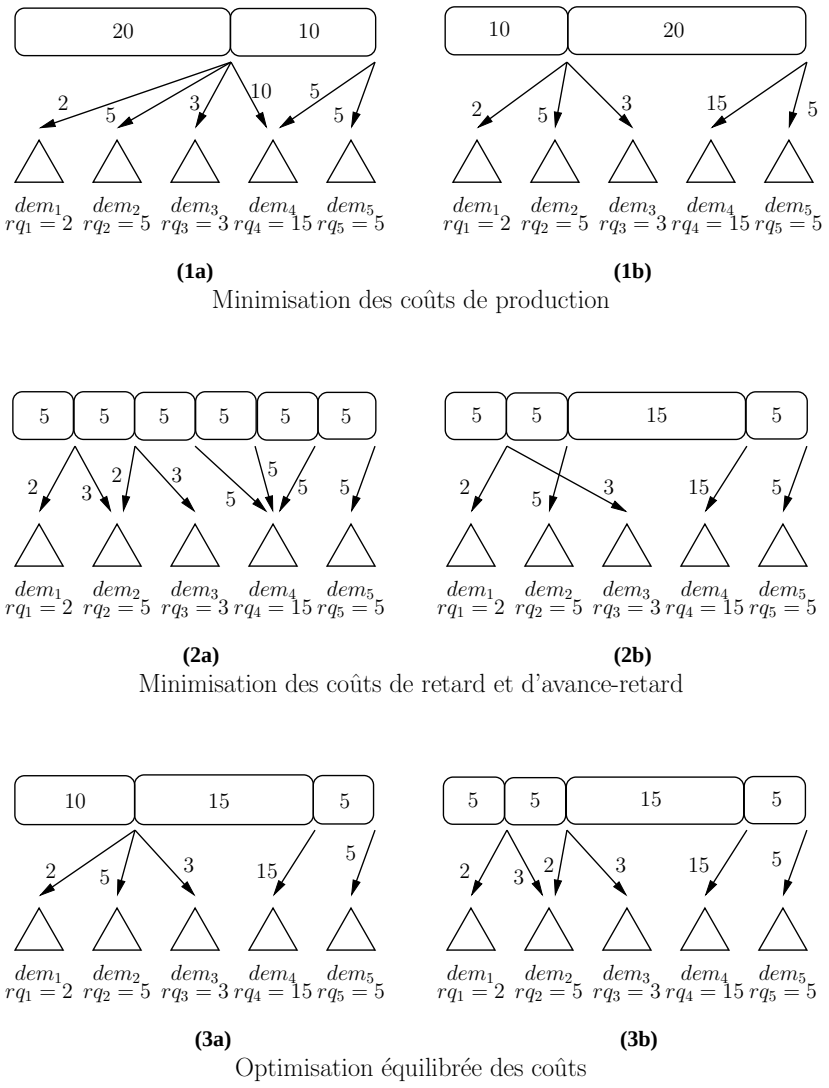

FIG. 2.5 – Optimisation des différents coûts pour le problème de *batching*

Analyse des solutions. Les solutions (1a) et (1b) de la Figure 2.5 correspondent à une politique de création d'un minimum de batches. Pour le critère des coûts de production, elles sont donc optimales. Cependant, dans le cas où le retard dans les livraisons est critique, on constate que ce type de solution présente une structure défavorable du point de vue de ce second critère. En effet, la durée d'exécution d'un batch étant proportionnelle à sa taille, plus un batch est gros, plus les matériaux qu'il produit sont disponibles tardivement, ce qui peut entraîner des

pénalités de retard. En revanche, la solution (2a) est dominante du point de vue du critère du retard. En effet, dans le cas où les coûts de production sont ignorés, la création de batches de taille minimale rend disponibles les produits le plus tôt possible et ainsi les livraisons peuvent se faire au plus tôt. *A contrario*, il s'agit de la solution la moins bonne pour ce qui concerne les coûts de production. Lorsque l'on considère en plus les coûts d'avance, cette solution n'est plus nécessairement dominante (cela dépend de la date due de chaque demande). Par exemple, supposons que la demande dem_4 ait une date due assez éloignée dans le futur, il est préférable de la livrer intégralement le plus tard possible, cette solution est représentée en (2b).

Finalement, on observe bien qu'optimiser simultanément des coûts de production, d'avance et de retard n'est pas trivial et il n'existe pas de propriété évidente de dominance dans le cas général. Les solutions (3a) et (3b) représentent des solutions de compromis, l'optimalité du problème dépendant de l'importance donnée de chaque critère.

Modélisation mono-critère d'un problème multi-critère. Le problème défini ci-dessus est clairement multi-critère, cependant, nous introduisons dans la suite un unique objectif, somme pondérée des coûts de production et d'avance-retard, que nous optimisons comme un critère unique. Malgré la difficulté qui serait engendrée par une approche de résolution multi-critère, il ne serait pas déraisonnable d'envisager une telle orientation pour résoudre ce problème.

2.2 Formulation du problème central ou problème *"cœur"*

Cette section est maintenant consacrée à la définition formelle du problème de *batching*. Dans un premier temps, nous introduisons l'ensemble des notations mathématiques employées dans la suite du manuscrit (cf. Annexe A, Tableau A.1). Toutefois, au cours des Chapitres 3 et 4, nous nous concentrons sur une version restreinte du problème, que nous nommons problème *"cœur"*, et des notations simplifiées correspondantes sont aussi introduites (cf. Annexe A, Tableau A.2). Dans un second temps, nous revenons sur le problème global résolu par l'APS ILOG PPO et formalisons les contraintes devant être satisfaites par une solution ainsi que sa fonction objectif. L'architecture originale de ILOG PPO décompose ce problème global en trois étapes, ce qui mène à la formalisation du rôle du *batching*, seconde étape située entre planification et ordonnancement. Ce problème est alors défini d'une manière d'abord générale (sans critère d'optimisation), puis dans la perspective spécifique envisagée dans la thèse où nous introduisons une fonction objectif. Un rappel des principales notations employées dans le manuscrit est proposé dans l'Annexe A.

2.2.1 Description des données, notations générales et simplifiées

Nous introduisons ci-après, pour chacune des notions prises en compte dans les données du problème global, les notations employées dans le cas général. Celles-ci seront utilisées lors de l'exposé de nos travaux concernant les extensions du problème *"cœur"* sur lequel nous nous

concentrons dans un premier temps. Dans cette version restreinte, nous adoptons des notations simplifiées.

Ressources. On considère un ensemble de ressources $\mathcal{R}es = \{res_1 \dots res_{|\mathcal{R}es|}\}$, chacune dotée de la caractéristique suivante :

- cap_r, pour $r = 1$ à $|\mathcal{R}es|$; capacité instantanée de la ressource res_r (cette notion est à mettre en relation avec le nombre d'activités pouvant être exécutées au même instant).

Dans le problème *"cœur"*, les simplifications sont les suivantes :

- $|\mathcal{R}es| = 1$ et on note $\mathcal{R}es = \{res\}$;
- $cap_1 = 1$.

Matériaux. On considère un ensemble de matériaux $\mathcal{M}at = \{mat_1 \dots mat_{|\mathcal{M}at|}\}$, chacun doté des caractéristiques suivantes :

- IS_m, pour $m = 1$ à $|\mathcal{M}at|$; stock initial du matériau indiquant sa quantité disponible à l'instant 0 ;
- stc_m, pour $m = 1$ à $|\mathcal{M}at|$; coût de stockage pour une unité de matériau mat_m gardée en stock pendant 1 unité de temps.

Dans le problème *"cœur"*, les simplifications sont les suivantes :

- $IS_m = 0$, pour $m = 1$ à $|\mathcal{M}at|$;
- $stc_m = 0$, pour $m = 1$ à $|\mathcal{M}at|$.

Recettes. On considère un ensemble de recettes, comme il a été défini au Chapitre 1, Section 1.3.1.1, $\mathcal{R}ec = \{rec_1 \dots rec_{|\mathcal{R}ec|}\}$, chacune dotée des caractéristiques suivantes :

- bs_i, pour $i = 1$ à $|\mathcal{R}ec|$; taille minimale pour un batch de production de la recette ;
- BS_i, pour $i = 1$ à $|\mathcal{R}ec|$; taille maximale pour un batch de production de la recette.

Activités. Pour chaque recette rec_i, on considère un ensemble d'activités génériques $\mathcal{A}ct_i = \{act_{i,1} \dots act_{i,|\mathcal{A}ct_i|}\}$. Dans le problème *"cœur"*, les simplifications sont les suivantes :

- $|\mathcal{A}ct_i| = 1$ pour $i = 1$ à $|\mathcal{R}ec|$ et on note act_i l'unique activité de la recette rec_i.

Modes. Pour chaque recette rec_i, pour chaque activité $act_{i,j}$ de la recette rec_i, on considère un ensemble de modes possibles pour l'exécution de l'activité $act_{i,j}$, $\mathcal{M}od_{i,j} = \{mod_{i,j,1} \dots mod_{i,j,|\mathcal{M}od_{i,j}|}\}$, chacun doté des caractéristiques suivantes :

- $mres_{i,j,k} \in \mathcal{R}es$, pour $i = 1$ à $|\mathcal{R}ec|$, $j = 1$ à $|\mathcal{A}ct_i|$ et $k = 1$ à $|\mathcal{M}od_{i,j}|$; ressource requise pour l'exécution de $act_{i,j}$ dans le mode $mod_{i,j,k}$;
- $mcap_{i,j,k}$; capacité instantanée requise de la ressource $mres_{i,j,k}$ pour l'exécution de $act_{i,j}$ dans le mode $mod_{i,j,k}$;
- la durée d'une instance de l'activité $act_{i,j}$ dans le mode $mod_{i,j,k}$ est déterminée par une fonction affine de la quantité du batch qui l'instancie ; l'ordonnée à l'origine de cette fonction

est $fp_{i,j,k}$ (partie fixe) et son coefficient directeur $vp_{i,j,k}$ (partie variable), cf. Section 2.2.2, équation (2.1) ;
- le coût d'exécution d'une instance de l'activité $act_{i,j}$ dans le mode $mod_{i,j,k}$ se calcule comme sa durée, la partie fixe du coût est notée $fc_{i,j,k}$, et sa partie variable $vc_{i,j,k}$.

Dans le problème *"cœur"*, les simplifications sont les suivantes :
- $mres_{i,j,k} = res$;
- $mcap_{i,j,k} = 1$;
- $|\mathcal{M}od_{i,j}| = 1$, pour $i = 1$ à $|\mathcal{R}ec|$ et $j = 1$ à $|\mathcal{A}ct_i|$, il n'y a qu'un mode possible pour l'exécution de l'unique activité act_i de la recette rec_i, que l'on note mod_i ;
- comme $|\mathcal{A}ct_i| = 1$ et $|\mathcal{M}od_{i,j}| = 1$, pour $i = 1$ à $|\mathcal{R}ec|$, les abus de notation suivants sont effectués pour les temps et coûts d'exécution de act_i : fp_i, vp_i, fc_i, et vc_i.

Consommation et production. Pour chaque recette rec_i, pour chaque activité $act_{i,j}$ de la recette rec_i, on considère d'une part un ensemble de consommations (utilisations) $\mathcal{C}m_{i,j} = \{cm_{i,j,1} \dots cm_{i,j,|\mathcal{C}m_{i,j}|}\}$ et d'autre part un ensemble productions (fabrications) $\mathcal{P}m_{i,j} = \{pm_{i,j,1} \dots pm_{i,j,|\mathcal{P}m_{i,j}|}\}$ de matériaux. Chaque $cm_{i,j,c}$, respectivement $pm_{i,j,p}$, est dotée des caractéristiques suivantes :
- $cmat_{i,j,c} \in \mathcal{M}at$, respectivement $pmat_{i,j,p}$, pour $i = 1$ à $|\mathcal{R}ec|$, $j = 1$ à $|\mathcal{A}ct_i|$ et $c = 1$ à $|\mathcal{C}m_{i,j}|$, respectivement $p = 1$ à $|\mathcal{P}m_{i,j}|$; matériau consommé, respectivement produit, lors de l'exécution de $act_{i,j}$;
- $cq_{i,j,c}$, respectivement $pq_{i,j,p}$, quantité normalisée de $cmat_{i,j,c}$ consommée, respectivement de $pmat_{i,j,p}$ produite, pour l'exécution d'une unité de la recette rec_i ; la consommation, respectivement la production, prend effet au début, respectivement à la fin, de l'exécution de $act_{i,j}$.

Dans le problème *"cœur"*, les simplifications sont les suivantes :
- $|\mathcal{C}m_{i,j}| = 0$, pour $i = 1$ à $|\mathcal{R}ec|$ et $j = 1$ à $|\mathcal{A}ct_i|$;
- $|\mathcal{P}m_{i,j}| = 1$, pour $i = 1$ à $|\mathcal{R}ec|$ et $j = 1$ à $|\mathcal{A}ct_i|$;
- $pq_{i,j,p} = 1$: lors de l'exécution de l'activité $act_{i,j}$ de la recette rec_i, il n'y a qu'un matériau $pmat_{i,j,p}$ produit et une unité d'exécution de rec_i produit une unité de $pmat_{i,j,p}$;
- comme $|\mathcal{A}ct_i| = 1$ et $|\mathcal{P}m_{i,j}| = 1$, pour $i = 1$ à $|\mathcal{R}ec|$, les abus de notation suivants sont effectués pour le matériau produit et sa production : $pmat_i$ et pq_i.

Demandes. On considère un ensemble de demandes $\mathcal{D}em = \{dem_1 \dots dem_{|\mathcal{D}em|}\}$, chacune dotée des caractéristiques suivantes :
- $rmat_d \in \mathcal{M}at$, pour $d = 1$ à $|\mathcal{D}em|$; matériau requis par la demande dem_d ;
- rq_d, quantité de $rmat_d$ requis ;
- dt_d, date due ;
- ec_d, pénalité unitaire de livraison en avance ;
- tc_d, pénalité unitaire de livraison en retard ;
- uns_d, pénalité unitaire de non-livraison de la demande.

Dans le problème *"cœur"*, les simplifications sont les suivantes :

- $\mathit{uns}_d = +\infty$ pour $d = 1$ à $|\mathcal{D}em|$, c'est-à-dire que toute demande doit être satisfaite intégralement.

Dans la suite, on notera souvent n le nombre de demandes $|\mathcal{D}em|$.

Dans un premier temps, nos travaux de thèse se concentrent sur l'étude du problème *"cœur"* et nous adoptons alors les notations simplifiées.

2.2.2 Solution générale du problème *"cœur"*

2.2.2.1 Informations décisionnelles finales

Sous les hypothèses simplificatrices relatives au problème *"cœur"*, une solution au problème global résolu par ILOG PPO consiste en la prise des décisions ci-dessous.

Batches (*lot-streaming*). Pour chaque recette rec_i, $i = 1$ à $|\mathcal{R}ec|$, un ensemble de batches $\mathcal{B}at_i = \{\mathit{bat}_{i,1} \dots \mathit{bat}_{i,|\mathcal{B}at_i|}\}$ doit être déterminé avec les attributs suivants :

- $q_{i,b}$, taille du batch $\mathit{bat}_{i,b}$, pour $b = 1$ à $|\mathcal{B}at_i|$,
- $\mathit{st}_{i,b}$, date de début de l'activité instanciée par $\mathit{bat}_{i,b}$, $\mathit{act}_{i,b}$, dans l'ordonnancement final ;
- $\mathit{et}_{i,b}$, date de fin de l'activité instanciée par $\mathit{bat}_{i,b}$, $\mathit{act}_{i,b}$, dans l'ordonnancement final.

Puisque $|\mathcal{A}ct_i| = 1$, tout batch $\mathit{bat}_{i,b}$ instancie l'unique activité générique act_i de la recette rec_i en une unique activité physique $\mathit{act}_{i,b}$. Il y a donc autant d'instances de l'activité act_i à ordonnancer que de batches $\mathit{bat}_{i,b}$, et cela pour chaque recette rec_i. La durée d'exécution de l'instance $\mathit{act}_{i,b}$ de act_i est alors calculée en sommant la partie fixe fp_i et le produit de la partie unitaire vp_i par la taille $q_{i,b}$ du batch instanciant l'activité, et on note $p_{i,b}$ cette durée (2.1) :

$$\forall i, \forall b, \qquad p_{i,b} = \mathit{fp}_i + \mathit{vp}_i q_{i,b} \tag{2.1}$$

Dans la suite, on notera souvent m_i le nombre de batches $|\mathcal{B}at_i|$ de la recette rec_i et m le nombre total de batches.

Flot de matériaux (*pegging*). Pour chaque batch $\mathit{bat}_{i,b}$ de chaque recette rec_i, $i = 1$ à $|\mathcal{R}ec|$ et $b = 1$ à $|\mathcal{B}at_i|$, pour chaque demande dem_d, la quantité $f_{i,b,d}$ de matériau produite par l'activité instanciée $\mathit{act}_{i,b}$ de la recette rec_i, livrée pour satisfaire la demande dem_d doit être identifiée. On considère que pour toute recette rec_i et pour toute demande dem_d telles que $\mathit{rmat}_d \neq \mathit{pmat}_i$ on a $f_{i,b,d} = 0$.

2.2.2.2 Contraintes satisfaites par une solution

Les prises de décisions décrites ci-dessus, et qui définissent sans ambiguïté une solution du problème global, doivent respecter les contraintes suivantes :

$$\forall i, \forall b, \quad bs_i \leq q_{i,b} \leq BS_i \tag{2.2}$$

$$\forall i, \forall b, \quad et_{i,b} = st_{i,b} + p_{i,b} \tag{2.3}$$

$$\forall i, \forall b, \quad \sum_d f_{i,b,d} = q_{i,b} \tag{2.4}$$

$$\forall d, \quad \sum_i \sum_b f_{i,b,d} = rq_d \tag{2.5}$$

$$\forall i, b, i', b', \quad (et_{i,b} \leq st_{i',b'}) \text{ ou } (et_{i',b'} \leq st_{i,b}) \tag{2.6}$$

Les inégalités (2.2) contraignent les batches définis dans une solution finale à respecter les bornes inférieure et supérieure sur la taille des batches. Les équations (2.3) relient dates de début, de fin et durée d'exécution des activités instanciées. Les contraintes (2.4) conservent les flots sortant des batches et les contraintes (2.5) conservent ceux entrant dans les demandes. Ces deux derniers ensembles d'équation garantissent que le *pegging* définit bien un flot pour chaque produit. Enfin, on exprime le fait que la ressource ne peut exécuter deux activités simultanément par la disjonction (2.6).

2.2.2.3 Fonction objectif du problème global.

Enfin, la solution optimale du problème général, restreint aux hypothèses simplificatrices du problème *"cœur"*, est celle qui minimise la fonction OBJ_{global} (2.7) ci-dessous : le premier terme calcule les coûts de production associés d'une part à la création des batches définis dans la solution, via une partie fixe, et d'autre part à leur taille, via une partie variable ; les second et troisième termes expriment les pénalités dues aux livraisons en avance et en retard, en vue de satisfaire les demandes. On note que ces termes considèrent le délai d'avance ou de retard ainsi que la quantité effectivement livrée ; dans la suite, on les nomme "termes quadratiques" d'avance ou de retard puisqu'il s'agit du produit de deux quantités variables.

$$\begin{aligned} OBJ_{global} &= \sum_i \sum_b (fc_i + vc_i q_{i,b}) \\ &+ \sum_i \sum_b \sum_d ec_d f_{i,b,d} (dt_d - et_{i,b})^+ \\ &+ \sum_i \sum_b \sum_d tc_d f_{i,b,d} (et_{i,b} - dt_d)^+ \end{aligned} \tag{2.7}$$

2.2.3 Architecture de résolution

Comme il a été annoncé dans le chapitre précédent, l'APS ILOG PPO présente une structure modulaire distinguant trois étapes, ce qui contribue à la nouveauté de l'approche de résolution

du problème général présenté ci-dessus. Chacune des étapes se voit attribuer un rôle décisionnel spécifique bien défini : planification de la production par période, *batching* de la production et *pegging* des batches aux demandes, puis ordonnancement des activités induites par les batches. Nous détaillons dans cette section l'approche de résolution particulière adoptée par ILOG PPO ainsi que la fonction occupée par chaque module ; des notations complémentaires sont alors introduites.

2.2.3.1 Planification

Dans la première étape, l'horizon est découpé en un ensemble de périodes $\mathcal{T} = \{T_1 \dots T_{|\mathcal{T}|}\}$, chacune dotée des caractéristiques suivantes :

– S_t et E_t, dates de début et fin de la période T_t, pour $t = 1$ à $|\mathcal{T}|$; on doit avoir $S_t < E_t$;

On supposera que les T_t sont triées chronologiquement selon t, c'est-à-dire que si $t < t'$ alors $E_t \leq S_{t'}$.

Le rôle de ce module est de décider, pour chaque période de la quantité de chaque recette à exécuter ainsi que de la part de chaque demande qui doit être livrée dans la période, il faut donc déterminer les valeurs suivantes :

– pour $i = 1$ à $|\mathcal{R}ec|$, pour $t = 1$ à $|\mathcal{T}|$, Q_i^t, quantité de production planifiée pour la recette rec_i pour la période T_t ;
– pour $d = 1$ à $|\mathcal{D}em|$, pour $t = 1$ à $|\mathcal{T}|$, Q_d^t, quantité de produit $rmat_d$ à livrer pendant la période T_t, afin de satisfaire la demande dem_d.

La procédure de résolution employée pour résoudre ce problème (très proche des problèmes de *lot-sizing*, [42] et Chapitre 1, Section 1.2.2) n'est pas le sujet qui nous occupe dans cette thèse, c'est pourquoi nous ne la détaillerons pas. La solution de planification (l'ensemble des Q_i^t et des Q_d^t) vient alors enrichir, comme nous allons le voir, les données d'entrée pour l'étape suivante dite de *batching*. Notons que, dans le cas général, la somme des productions planifiées Q_i^t, dans la période T_t, de toutes les recette rec_i permettant d'obtenir un même matériau $pmat_i$ n'est pas nécessairement égale à la quantité livrée Q_d^t, dans cette même période, à toutes les demandes dem_d telles que $rmat_d = pmat_i$, du fait des critères optimisés lors de la planification. Ainsi, il est possible que des flots de produit "traversent" des périodes et le problème ne peut donc pas être décomposé en $|\mathcal{T}|$ sous-problèmes indépendants.

2.2.3.2 Batching (lot-streaming + pegging)

La seconde étape a pour fonction de déterminer les batches et les arcs de *pegging*, problème sur lequel se concentre notre thèse. Comme le découpage du temps, selon les périodes définies au cours de la planification, est conservé, les ensembles $\mathcal{B}at_i$, définis pour chaque recette rec_i dans le problème général sont partitionnés en sous-ensembles $\mathcal{B}at_i^t = \{bat_{i,1}^t \dots bat_{i,|\mathcal{B}at_i^t|}^t\}$, pour $t = 1$ à $|\mathcal{T}|$. Les batches du sous-ensemble $\mathcal{B}at_i^t$ réalisent physiquement la quantité Q_i^t de production

de la recette rec_i planifiée d'être exécutée dans la période T_t. Réciproquement, on note $T_{i,b}$ la période d'exécution planifiée pour le batch $bat_{i,b}$, pour $i = 1$ à $|\mathcal{R}ec|$, pour $b = 1$ à $|\mathcal{B}at_i|$; en particulier si deux batches $bat_{i,b}$ et $bat_{i,b'}$ appartiennent au même sous-ensemble $\mathcal{B}at_i^t$, on aura $T_{i,b} = T_{i,b'}$.

Comme la planification des livraisons des demandes est aussi donnée par période, dans l'étape de *batching*, on décompose chaque demande dem_d définie initialement en autant de sous-demandes correspondant à la planification d'une partie de sa satisfaction dans une période. En d'autres termes, pour une demande dem_d donnée, on définit une sous-demande pour chaque période T_t telle que $Q_d^t > 0$. Chaque sous-demande est alors considérée comme une demande indépendante, se voyant attribuée les mêmes caractéristiques que sa demande "mère", à l'exception de la quantité rq_d remplacée par Q_d^t. On note l'ensemble de ces sous-demandes $\tilde{\mathcal{D}}em = \{\tilde{dem}_1, \ldots \tilde{dem}_{|\tilde{\mathcal{D}}em|}\}$. Ainsi, dans l'étape du batching, on ignore les demandes de l'ensemble initial $\mathcal{D}em$ et on ne manipule que les demandes de $\tilde{\mathcal{D}}em$. À chaque demande $\tilde{dem}_d$ de $\tilde{\mathcal{D}}em$, on peut alors faire correspondre l'unique période de planification T_d qui lui est associée.

Lors de cette étape, les flots de produit $f_{i,b,d}$ entre les batches et les demandes doivent être déterminés. Pour résoudre le problème de *pegging*, on considère les demandes de $\tilde{\mathcal{D}}em$. Du fait de la décomposition du temps en périodes, on note $f_{i,b,d}^t$ la quantité de flot correspondant, au cours de cette étape, à la satisfaction de la demande $\tilde{dem}_d$ par le batch $bat_{i,b}^t$.

Une solution au problème de *batching* doit respecter les contraintes (2.2,2.4,2.5) définies lors de la description du problème global. Enfin, avant de transmettre ces décisions (batches et arcs de *pegging*) au module final d'ordonnancement, les flots de produit sont réagrégés par demandes "mère", les sous-demandes n'étant plus considérées dans la suite. En outre, le partitionnement des batches selon les périodes de planification est abandonné, ce qui induit la disparition des variables t et le retour à l'indiçage à deux champs (i et b) des concepts relatifs aux batches.

2.2.3.3 Ordonnancement

Ce dernier module a pour fonction d'attribuer des dates de début et de fin à toutes les activités induites par les batches calculés précédemment. De ce fait, un ordonnancement doit respecter les contraintes (2.3,2.6) définies en Section 2.2.2. Dans le problème restreint, une seule ressource est considérée, il n'y a donc pas de problème d'affectation au sein du problème d'ordonnancement. En outre, la notion de période, issue de la planification et conservée dans le *batching*, disparaît ici. Par ailleurs la fonction objectif optimisée dans cette étape est celle qui a été définie lors de la description du problème général (2.7). Cependant, les variables $f_{i,b,d}$ apparaissant dans cette expression correspondent à des décisions prises en amont, l'objectif finalement minimisé est donné par l'expression suivante, dans laquelle le terme des coûts de production, désormais fixe, a été omis, de plus, les variables instanciées y ont été tildées (2.8).

$$\begin{aligned} OBJ_{ordo} &= \sum_i \sum_b \sum_d ec_d \tilde{f}_{i,b,d}(dt_d - et_{i,b})^+ \\ &+ \sum_i \sum_b \sum_d tc_d \tilde{f}_{i,b,d}(et_{i,b} - dt_d)^+ \end{aligned} \tag{2.8}$$

2.2.4 Position du problème de *batching* étudié dans cette thèse

Dans cette thèse, nous résolvons le problème de *batching* posé dans la section précédente en introduisant une fonction d'optimisation. En effet, nous rappelons que dans le cas général, aucune fonction objectif n'est définie pour ce problème, l'évaluation de la qualité d'une solution de *batching* se faisant par celle de l'ordonnancement finalement obtenu qui se comporte alors comme un oracle. L'originalité de nos travaux consistant à considérer ce problème comme un problème d'optimisation combinatoire, il était nécessaire de définir une fonction objectif. En outre, nous avons souhaité pouvoir évaluer la qualité d'une solution de *batching* en tant que telle, afin, d'une part, de pouvoir comparer plusieurs méthodes de résolution et d'autre part, d'étudier la corrélation entre la qualité d'une solution de *batching* et de celle de l'ordonnancement associé. Comme nous l'avons présenté de manière informelle dans la Section 2.1.2, une fonction proche de celle de l'objectif de l'ordonnancement est donc définie.

2.2.4.1 Variables intervenant dans la fonction objectif

Lot-streaming et coûts de production. Cette partie du problème consiste à découper, pour chaque période T_t, $t = 1$ à $|\mathcal{T}|$, pour chaque recette rec_i, $i = 1$ à $|\mathcal{R}ec|$, la quantité Q_i^t en batches respectant les contraintes de taille de batch imposées par le recette rec_i. Pour chaque batch $bat_{i,b}^t$ de $\mathcal{B}at_i^t$, il s'agit donc de déterminer la quantité $q_{i,b}^t$. La quantité totale à exécuter est connue, il s'agit de Q_i^t, pour chaque période et chaque recette. La partie variable des coûts de production est donc fixée ; elle apparaîtra cependant explicitement, pour plus de clarté. Par contre, le nombre de batches créés $|\mathcal{B}at_i^t|$ étant inconnu à l'avance, les coûts fixes de production interviennent logiquement dans la fonction objectif.

Pegging et coûts d'avance-retard. Dans cette seconde partie, il s'agit de déterminer les flots de produit $f_{i,b,d}^t$ entre les batches $bat_{i,b}^t$ et les demandes $\tilde{dem}_d$. Les flots eux-mêmes sont donc des variables et leur optimisation se fait en anticipant la répercussion de ces décisions sur les coûts d'avance et de retard dans l'ordonnancement. Comme il a été annoncé dans la Section 2.1.2, des variables temporelles sont introduites dès l'étape de *batching* afin d'estimer les dates d'exécution des batches et pouvoir évaluer des termes proches de ceux définis dans l'objectif de l'ordonnancement. On associe donc à chaque batch $bat_{i,b}^t$, deux variables exprimant les dates estimées de début et de fin d'exécution de ce batch : $est_{i,b}^t$ et $eet_{i,b}^t$.

2.2.4.2 Formulation exacte de la fonction objectif

Le terme des coûts de production s'exprime directement. Concernant les termes d'avance et de retard entre un batch $bat^t_{i,b}$ et une demande $\tilde{dem}_d$, les variables temporelles vont permettre de calculer un délai en mesurant la distance entre l'instant estimé de disponibilité du produit $eet^t_{i,b}$ et la date due dt_d. À l'image de la fonction objectif définie pour le problème d'ordonnancement, les délais d'avance et de retard sont alors multipliés par la quantité de matériau circulant sur l'arc de *pegging*, qui est ici une variable. Ainsi, on peut expliciter la fonction objectif $OBJ_{batching}$ que l'on cherche à optimiser pour le problème *"cœur"* de *batching* traité dans cette thèse (2.9) :

$$\begin{aligned} OBJ_{batching} &= \sum_t \sum_i \sum_b (fc_i + vc_i q^t_{i,b}) \\ &+ \sum_t \sum_i \sum_b \sum_d ec_d f^t_{i,b,d} (dt_d - eet^t_{i,b})^+ \\ &+ \sum_t \sum_i \sum_b \sum_d tc_d f^t_{i,b,d} (eet^t_{i,b} - dt_d)^+ \end{aligned} \tag{2.9}$$

2.3 Liens avec l'ordonnancement par batch

2.3.1 Points communs entre *batching* et ordonnancement par batch

Définition de l'ordonnancement par batch. Il existe un ensemble de problèmes d'ordonnancement sensiblement proches du problème de *batching* défini précédemment. La problématique de l'ordonnancement par batch consiste en effet à ordonnancer des tâches en les regroupant dans des batches qui sont alors exécutés par une machine ; la date de fin d'une tâche se confond alors avec la date de fin du batch auquel elle appartient. On distingue deux grandes classes de problèmes. Dans la première, les tâches d'un même batch sont exécutées en parallèle par la machine et la date de fin d'un batch correspond à la date de fin de la tâche de plus longue durée d'exécution, ce sont les problème de fournées, notés *p-batch* (*parallel batching*) [18, 17]. Dans la seconde classe, les tâches d'un même batch sont exécutées en série par la machine et la date de fin du batch correspond à la somme des durées d'exécution des tâches appartenant à ce batch ; de plus la mise en production d'un nouveau batch induit un temps fixe qui doit être ajouté à la durée d'exécution du batch. On peut trouver une présentation plus précise de ces problèmes dans [20]. En outre, Potts et Kovalyov en ont dressé un panorama dans [101]. Nous nous intéressons aux problèmes de la seconde famille, notés *s-batch* (*serial* ou *sum batching*).

Identification du problème de *batching* traité dans la thèse. Dans le problème d'ordonnancement par batch, lorsque les tâches à ordonnancer sont toutes de même type, cela signifie qu'une tâche peut être associée à n'importe quel batch. Chaque tâche est en outre dotée d'une durée d'exécution, d'une date due et de pénalités d'avance et de retard. Il s'agit alors de regrouper les tâches dans des batches de manière à exécuter ces derniers sur une unique machine.

Diverses contraintes peuvent s'ajouter, divers objectifs peuvent être optimisés, ce qui définit un ensemble de variantes pour ce problème général.

Dans notre contexte, considérons qu'une seule recette *rec* est modélisée, on peut donc fabriquer un unique produit *mat*. Il existe un ensemble de demandes dem_d telles que $rmat_d = mat$ pour tout d, chacune dotée d'une quantité rq_d, d'une date due dt_d et de pénalités d'avance ec_d et de retard tc_d. La mise en correspondance entre les demandes du *batching* et les tâches de l'ordonnancement par batch permet d'établir une équivalence entre différents problèmes issus de chacune de ces problématiques. En effet, le regroupement en batch des tâches coïncide avec le *pegging* des batches aux demandes. Le tableau 2.1 présente les équivalences entre les données de chaque problème.

batching		Ordonnancement par batch
demande dem_d	$\Leftrightarrow$	tâche
date due dt_d	$\Leftrightarrow$	date due
quantité rq_d	$\Leftrightarrow$	durée d'exécution
coûts d'avance/retard ec_d, tc_d	$\Leftrightarrow$	coût d'avance/retard
batch bat_b	$\Leftrightarrow$	batch
taille q_b du batch bat_b	$\Leftrightarrow$	somme de la durée des tâches dans ce batch
partie fixe fp de la durée du batch	$\Leftrightarrow$	temps fixe de mise en production

TAB. 2.1 – Équivalence des données entre *batching* et ordonnancement par batch

2.3.2 D'autres contraintes et de nouveaux objectifs

Malgré une grande similitude dans leur intention respective, les deux problèmes ne considèrent pas les mêmes contraintes et n'optimisent pas les mêmes fonctions objectifs. Le problème de *batching* introduit dans cette thèse pose ainsi de nouvelles variantes du problème d'ordonnancement par batch.

Contraintes. Une première caractéristique importante du problème général de *batching* est le fait qu'une demande peut être satisfaite par plusieurs batches, ce qui est équivalent à un problème d'ordonnancement par batch où la préemption est autorisée, cet aspect a été étudié dans le cadre de problème à machines parallèles [92, 93, 29]. Nous nommons multi-*pegging* la flexibilité liée à la possibilité pour une demande d'être satisfaite par plusieurs batches. Par opposition, la contrainte forçant une demande à être satisfaite par un unique batch est nommée mono-*pegging*, on sort alors du cas général préemptif. Cette contrainte a été rencontrée dans la pratique et sera étudiée dans le cadre de la complexité puis dans les extensions du problème. Par ailleurs, on peut imposer que chaque demande soit livrée entre deux instants restrictifs. Cette contrainte n'apparaît pas dans le cas du problème *"cœur"*, elle sera par contre mentionnée

comme extension au Chapitre 5. Elle est équivalente à la définition de dates de disponibilité et de dates butoir pour les tâches dans le cadre de l'ordonnancement par batch [8, 79, 23]. Une autre caractéristique importante est liée à la taille des batches, puisque dans cette thèse, on considère des bornes inférieure et supérieure sur la quantité de produit délivrable par un batch (donc sur la somme des durées d'exécution des tâches groupées dans un même batch). Les seules contraintes que nous avons rencontrées dans la littérature s'apparentant à celles-ci concernent le nombre de tâches à exécuter. Dans le cas *s-batch*, Cheng et Kovalyov [32] ont étudié le problème dans lequel le nombre de tâches regroupées dans un même batch ne doit pas dépasser un certain nombre k donné en entrée. Yuan *et al.* [133] ont récemment traité le cas dans lequel exactement k tâches doivent être exécutées dans un batch donné. Cette contrainte peut aussi être formulée dans le cas *p-batch* [21, 28, 64].

Critères. Dans le cas du problème de *batching* étudié dans cette thèse, le critère d'optimisation contient deux types de termes distincts. Dans un premier temps, un coût fixe étant associé à la création d'un nouveau batch, la fonction tend à minimiser le nombre total de batches. Ce critère n'est pas explicitement présent dans la littérature de l'ordonnancement par batch, bien que, comme un temps fixe est ajouté lors de la création d'un batch, les critères usuels tendent implicitement à trouver un compromis entre un trop grand nombre de batches et un petit nombre de batches trop grands. Dans un second temps, les termes d'avance-retard considèrent l'écart absolu de la date de fin d'un batch à la date due de chaque demande à laquelle il contribue, multiplié par la quantité livrée par ce batch. Dans le cas général, c'est-à-dire lorsqu'une demande peut être satisfaite par plusieurs batches (multi-*pegging*), cette quantité est une variable et le terme est donc quadratique. Dans le cas mono-*pegging*, on voit apparaître un critère original qui consiste à multiplier le délai (variable) par la quantité totale de la demande, ce qui, dans la terminologie de l'ordonnancement par batch s'exprime par le produit du délai par la durée d'une tâche (dans la notation classique, on cherche alors à minimiser $\sum p_i T_i$). Ce critère ne semble pas avoir été examiné, même dans le cadre des problèmes d'ordonnancement traditionnels, il attirera notre attention dans le chapitre suivant consacré à la complexité. En effet, les fonctions objectifs classiques en ordonnancement par batch reprennent les critères définis pour les problèmes d'ordonnancement traditionnels : somme pondérée ou non des dates de fin, du nombre de tâches en retard, des retards de toutes les tâches ou encore du retard algébrique maximal [20, 22].

2.3.3 Conclusion et extensions

Les problèmes de *batching* restreints à une machine et une recette peuvent être rapprochés de certains problèmes d'ordonnancement avec des machines à traitement par batch (série). Nous avons vu qu'il y avait certains aspects communs aux deux classes de problèmes. Cependant, les cas d'études particuliers qui nous intéressent dans cette thèse décrivent des contraintes et des critères d'optimisation qui n'ont pas encore été examinés dans leurs équivalents d'ordonnancement par batch. Cela permet de définir de nouveaux problèmes pour ce sous-domaine de l'ordonnancement.

Par ailleurs, la restriction à une machine et une recette permet de poser les problèmes de *batching* et d'ordonnancement simultanément, ce qui n'est plus possible dès qu'on ajoute une recette fabriquant un autre produit et requérant la même ressource. En effet, le problème de *batching* isolé ne considère pas les contraintes exactes de capacité de la ressource et ne formule pas les contraintes disjonctives sur l'exécution des batches. Cependant, l'extension à plusieurs recettes partageant une même ressource permet de retrouver les problèmes d'ordonnancement par batch dans lesquels on définit différentes familles de tâches et le regroupement par batch doit alors respecter l'appartenance à une même famille.

Chapitre 3

Étude de complexité

Nous effectuons dans ce chapitre une étude de la complexité de certaines variantes et cas particuliers du problème *"cœur"* de *batching* qui a été présenté dans le chapitre précédent. Le but est de déterminer des cas maximum polynomiaux d'une part, des cas minimum NP-difficiles, au sens fort et au sens faible, d'autre part. La question de la faisabilité est étudiée dans un premier temps, puis, nous nous intéressons aux problèmes d'optimisation en dissociant le critère des coûts de production et celui des pénalités de retard.

3.1 Présentation des paramètres et critères considérés

Pour l'étude de complexité qui suit, nous avons sélectionné les variantes de problèmes qui nous paraissent les plus pertinentes à examiner au vu des cas que nous résolvons en pratique. D'autres cas particuliers, plus exotiques, auraient pu être envisagés. En outre, nous allons le voir, les problèmes deviennent vite intraitables et l'ajout ou l'association de peu de contraintes suffit pour les faire basculer dans la classe des problèmes NP-difficiles. Dans ce qui suit, nous présentons les restrictions que nous adoptons pour l'analyse de complexité, puis nous introduisons les contraintes et critères d'optimisation que nous traitons.

3.1.1 Restrictions du problème pour l'étude de complexité

Intégrité des valeurs manipulées. Dans un souci de simplification, l'étude de complexité qui suit ne prend en considération que des cas dans lesquels toutes les durées d'exécution et toutes les tailles de batches sont entières. Cette restriction est raisonnable car, dans la pratique, il est possible d'approximer aussi précisément qu'il est utile une instance quelconque (manipulant des valeurs non-entières) en une instance où les valeurs numériques sont entières en rediscrétisant le temps et en choisissant une unité adaptée pour la taille des batches.

Recettes. Nous restreignons l'analyse à une chaîne de production ne définissant qu'une seule recette *rec*. Ainsi, un seul matériau *mat* peut être produit. Puisque nous étudions le problème

"cœur", la recette est composée d'une unique activité *act* ne pouvant s'exécuter que dans un seul mode. La partie variable *vp* du temps d'exécution de *act* est égale à 1, sa partie fixe *fp* est nulle. Ainsi, la durée de l'activité instanciée d'un batch bat_b de la recette ainsi définie est exactement égale à la taille du batch q_b. Comme nous l'avons fait remarquer au chapitre précédent, ce problème de *batching* restreint permet de résoudre simultanément le problème d'ordonnancement, ainsi, nous exprimons la date de fin du batch bat_b par la notation et_b. En outre, du fait de cette simplification, on note $f_{b,d}$ la quantité de produit livrée par le batch bat_b pour satisfaire la demande dem_d. Par ailleurs, la partie variable *vc* du coût d'exécution de *act* est nulle, en effet, la quantité d'exécution est fixée dans le *batching*, ce qui induit un coût constant ; par contre, sa partie fixe *fc* est égale à K. Ainsi, le terme de la fonction objectif relatif aux coûts de production est exactement égal à K fois le nombre de batches. Enfin, la taille d'un batch de la recette *rec* est bornée entre *bs* et BS. Lorsqu'il n'y a pas de contrainte sur la borne inférieure, la remarque sur l'intégrité des valeurs permet de poser $bs = 1$.

Demandes. Un ensemble $\mathcal{D}em$ de demandes dem_d pour d allant de 1 à $|\mathcal{D}em|$ $(= n)$ est défini de la façon suivante : chaque demande dem_d requiert une quantité rq_d de produit *mat* à livrer de préférence à la date due dt_d. Le poids ec_d associé à la pénalité d'avance des livraisons de la demande dem_d est nul, on ne considère que le critère de retard.

Périodes et planification. Dans ce chapitre, on considère implicitement une unique période de temps dans laquelle toutes les demandes doivent être livrées et la production planifiée permet de satisfaire cette exigence. C'est pourquoi, par souci d'allègement des notations, la notion de période n'apparaît pas dans ce chapitre.

Cette description fixe un certain nombre de paramètres pour tous les problèmes considérés lors de l'étude de complexité. Le Tableau 3.1 dresse un récapitulatif de ces restrictions, ainsi que des notations simplificatrices employées dans la suite du chapitre. En fonction des variantes, les paramètres restant à définir seront à leur tour fixés.

3.1.2 Paramètres et contraintes critiques

Taille des batches. Dans le cas général, elle est bornée inférieurement et supérieurement, cependant, nous étudions aussi des cas particuliers où l'une ou l'autre des bornes est non-restrictive. On peut donc avoir d'une part, $bs = 1$ ou *bs* quelconque (en ce cas on supposera que la quantité à exécuter $Q = \sum_d rq_d$ est supérieure à *bs*, sinon il n'y a pas de solution faisable), et d'autre part, $BS = +\infty$ ou $BS < +\infty$.

Mono ou multi-*pegging*. Ces termes ont été introduits au chapitre précédent. Ils désignent respectivement l'obligation qu'une demande soit satisfaite par un unique batch ou au contraire l'autorisation de recourir à différents batches pour livrer une demande. La contrainte associée

Recette *rec*	activité	*act*
	produit fabriqué	*mat*
	partie variable de la durée d'exécution	$vp = 1$
	partie fixe de la durée d'exécution	$fp = 0$
	partie variable du coût d'exécution	$vc = 0$
	partie fixe du coût d'exécution	$fc = K$
	date de fin d'exécution	et_b
	quantité livrée à la demande dem_d	$f_{b,d}$
	borne inférieure sur la taille des batches	$bs \geq 1$
	borne supérieure sur la taille des batches	BS
Demande $dem_d \in \mathcal{D}em$	produit requis	*mat*
	pénalité d'avance	$ec_d = 0$

TAB. 3.1 – Restrictions et notations pour l'étude de complexité

au mono-*pegging* ne fait pas partie du problème *"cœur"* mais elle est souvent requise en pratique et s'avère présenter un intérêt du point de vue de la complexité.

Poids des pénalités de retard. Nous avons vu que, dans la fonction objectif, le terme correspondant au retard d'une livraison pour la demande dem_d était le produit de la quantité livrée par la durée du retard par le poids (ou coût) tc_d. Dans l'analyse de complexité, nous distinguons les cas où $tc_d = 1$ pour tout d des cas où ce poids est quelconque.

3.1.3 Critères pour la fonction objectif

Coûts d'exécution. Nous avons spécifié que la partie variable vc du coût d'exécution d'une unité de la recette est nulle. En effet, comme la quantité totale Q à exécuter est fixée lors du *batching* (puisque c'est la planification qui prend cette décision), toute solution de *batching* possède dans son coût final un terme incompressible égal à $vc.Q$. Il est donc légitime de supprimer ce terme dans notre analyse. En revanche, le coût fixe fc est très important puisqu'il reflète le nombre de batches créés. Deux cas sont considérés dans notre étude, celui où l'on ne prend en compte que ces coûts (on posera alors $fc = 1$) et le cas inverse où ne subsistent dans l'objectif que les pénalités de retard ($fc = 0$).

Coûts de retard. De la même façon que pour les coûts de production, nous étudions des cas où les coûts de retard sont pris en compte ($tc_d \neq 0$ pour d allant de 1 à n) ou non (ne subsistent alors dans l'objectif que les coûts fixes de production). Comme nous l'avons dit, au sein de l'étude des pénalités de retard, nous distinguons deux cas, celui où $tc_d = 1$ pour tout d, et celui où les poids sont quelconques.

3.2 Étude de problèmes de faisabilité et d'optimisation

Nous abordons maintenant l'analyse de la complexité à proprement parler. Dans un premier temps nous nous intéressons aux problèmes de faisabilité (existence d'une solution, sans fonction d'optimisation), puis, dans un second temps, à celui de la minimisation des coûts de production, et enfin, à celui de la minimisation des coûts de retard. Dans chaque cas, nous synthétisons les problèmes abordés et les résultats de complexité sous forme d'arbres dont chaque feuille correspond à un problème spécifique (Figures 3.1, 3.2, 3.6).

3.2.1 Décider de l'existence d'une solution

La faisabilité est liée à la possibilité de créer ou non des batches de manière à réaliser la quantité de production Q tout en respectant les contraintes sur la taille des batches. Imposer la livraison des demandes en un seul batch est une contrainte supplémentaire qui est aussi examinée. Il y a donc trois aspects à considérer : l'existence d'une borne supérieure BS sur la taille de batch, celle d'une borne inférieure bs et la contrainte de *pegging*. L'arbre de la Figure 3.1 décrit les huit problèmes induits par la conjonction de ces contraintes.

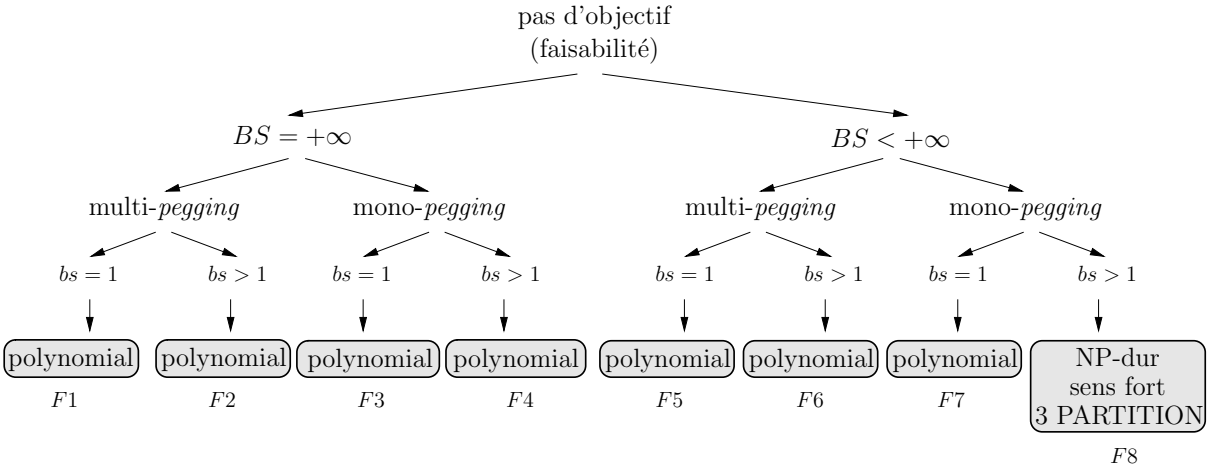

FIG. 3.1 – Existence d'une solution : problèmes abordés

$F1$, $F2$, $F3$ et $F4$ sont trivialement résolus puisque la solution où l'on crée un unique batch est réalisable. Pour $F2$ et $F4$ l'hypothèse $Q \geq bs$ nous est utile. $F5$ admet la solution où l'on crée $\lfloor Q/BS \rfloor$ batches de taille maximale et un dernier de taille $Q - BS \lfloor Q/BS \rfloor$, si nécessaire, c'est-à-dire dans le cas où Q n'est pas un multiple de BS. $F7$ admet une solution suivant les valeurs des données : si pour tout d, on a $rq_d \leq BS$, alors la solution où l'on crée un batch par demande est réalisable ; sinon, alors au moins une demande ne peut être satisfaite par un unique batch, ce qui viole la contrainte de mono-*pegging* et il n'y a donc pas de solution.

Condition de faisabilité pour $F6$. Nous pouvons aussi décider en temps polynomial si la variante $F6$ admet une solution ou non. Pour cela nous démontrons un critère nécessaire et suffisant non-trivial, calculable en temps polynomial, permettant de répondre à la question.

Propriété 1. Il existe une solution au problème $F6$ si et seulement si la relation suivante est vérifiée :

$$\left\lceil \frac{Q}{BS} \right\rceil \leq \left\lfloor \frac{Q}{bs} \right\rfloor$$

Preuve. Montrer que $\lceil Q/BS \rceil > \lfloor Q/bs \rfloor$ implique qu'il n'y a pas de solution est immédiat. En effet, il est clair que le nombre de batches d'une solution est minoré par $\lceil Q/BS \rceil$ et majoré par $\lfloor Q/bs \rfloor$. Donc si $\lceil Q/BS \rceil > \lfloor Q/bs \rfloor$, on ne peut pas trouver un nombre de batches permettant d'exhiber une solution satisfaisant les contraintes sur la taille des batches.

Réciproquement, montrons que si $\lceil Q/BS \rceil \leq \lfloor Q/bs \rfloor$ alors il existe nécessairement une solution. Sous cette hypothèse, construisons une solution réalisable. Formons tout d'abord un maximum de batches, tous de taille bs, il y en a $\lfloor Q/bs \rfloor$. Il reste donc une quantité $r = Q - bs \lfloor Q/bs \rfloor$ à répartir dans les batches ainsi formés, tout en ne dépassant pas la taille maximale BS. Dans ces batches, on peut encore placer une quantité égale à $R = (BS - bs) \lfloor Q/bs \rfloor$. Or, $Q/BS \leq \lceil Q/BS \rceil$ et par hypothèse $\lceil Q/BS \rceil \leq \lfloor Q/bs \rfloor$ donc $Q/BS \leq \lfloor Q/bs \rfloor$, soit $Q \leq BS \lfloor Q/bs \rfloor$, ce qui est équivaut à écrire que $r \leq R$. Il est donc possible de répartir la quantité restante r dans l'espace libre R, ce qui termine la construction de la solution. Notons que, de la même manière, on peut construire $\lceil Q/BS \rceil$ batches de taille BS puis retirer $BS \lceil Q/BS \rceil - Q$ unités à ces batches. En effet, comme $Q/bs \geq \lceil Q/BS \rceil$, on a $Q/\lceil Q/BS \rceil \geq bs$. Ainsi :

$$\frac{BS \left\lceil \frac{Q}{BS} \right\rceil - Q}{\left\lceil \frac{Q}{BS} \right\rceil} \leq BS - bs$$

Les $BS \left\lceil \frac{Q}{BS} \right\rceil - Q$ unités peuvent donc être retirées équitablement (à une unité près) tout en conservant des batches de taille supérieure ou égale à bs. ⊛

Remarques. Notons que ce critère est assez intuitif. Par hypothèse, $bs \leq BS$, donc $Q/BS \leq Q/bs$. Le critère de non-existence de solution est $\lceil Q/BS \rceil > \lfloor Q/bs \rfloor$. Dans le cas où il est vérifié, cela signifie que, même si $Q/BS \leq Q/bs$, ces valeurs sont néanmoins très proches puisqu'elles se trouvent entre deux nombres entiers consécutifs, d'où l'inversion de l'inégalité lorsque l'on passe à l'entier supérieur pour Q/BS et inférieur pour Q/bs. Une telle instance de problème paraît en effet, plus complexe à résoudre car elle offre moins de flexibilité, nous avons montré qu'il est même impossible d'en exhiber une solution réalisable. Notons que si $bs = BS$, il existe une solution si et seulement si Q est un multiple de BS, c'est-à-dire que $Q/BS = Q/bs$ est un entier et donc que $\lceil Q/BS \rceil = \lfloor Q/bs \rfloor$, ce qui est en accord avec le critère proposé. En outre, on peut facilement montrer que ce critère s'écrit de différentes façons :

$$Q \leq \left\lfloor \frac{Q}{bs} \right\rfloor BS \quad \Leftrightarrow \quad \left\lceil \frac{Q}{BS} \right\rceil \leq \left\lfloor \frac{Q}{bs} \right\rfloor \quad \Leftrightarrow \quad Q \geq \left\lceil \frac{Q}{BS} \right\rceil bs$$

Discussion sur l'appartenance à NP. Il est entendu que si décider de l'existence d'une solution peut être parfois trivial, cela n'implique pas qu'expliciter une telle solution en listant les batches et les arcs de *pegging* soit faisable en temps polynomial. En effet, une solution de *batching* consiste effectivement en la donnée, d'une part, d'un ensemble de batches et d'autre part, d'un ensemble d'arcs de *pegging*. Dans certains cas où, par exemple, la taille minimale bs est égale à 1, une solution optimale peut contenir Q batches de taille unitaire. Il y a alors $O(nQ)$ arcs de *pegging*. Une manière non compacte d'exprimer ces données peut conduire à un codage pseudopolynomial (polynomial en n et Q). Cependant il est clair qu'on peut exhiber un codage polynomial de la solution (il y a Q batches de taille 1 et chaque demande dem_d est livrée par rq_d batches consécutifs, il suffit donc de donner pour chaque demande l'indice du premier batch qui la livre). D'un manière plus générale, pour certains problèmes, il est possible de coder une solution sous forme compacte, en ayant recours à ce type de codage raisonnable. De tels codages sont employés dans l'étude des problèmes d'ordonnancements "à grande multiplicité" (*high multiplicity scheduling*). Dans la suite, nous précisons pour chaque variante ambiguë du problème si son appartenance à NP est démontrée.

Remarquons que toutes les variantes où la contrainte de mono-*pegging* est imposée sont clairement dans NP. En effet, une solution consiste en la donnée d'un ensemble $\mathcal{C}$ de couples (d, b) tels que le batch bat_b sert demande la dem_d. La contrainte de mono-*pegging* induit qu'il y a exactement n couples. La taille q_b de chaque batch bat_b est obtenue en sommant les quantités rq_d des demandes dem_d qu'il livre, ce qui se fait en temps polynomial. Pour obtenir le certificat, il suffit de vérifier que la valeur q_b est minorée par bs et majorée par BS, et de contrôler qu'il n'existe pas deux couples faisant intervenir deux batches différents et la même demande afin de s'assurer que la contrainte de *pegging* n'est pas violée.

Théorème 1. Le problème $F8$ est NP-complet au sens fort.

Preuve. Puisqu'il s'agit d'une variante mono-*pegging*, $F8$ est dans NP. Montrons alors que $F8$ est NP-difficile au sens fort en faisant une réduction de 3-PARTITION (problème défini dans [52] et en Section 3.3). À partir d'une instance I_{3P} de ce problème NP-complet au sens fort [52], on construit une instance I_{F8} de notre problème en définissant $3n$ demandes en correspondance avec les $3n$ éléments de 3-PARTITION. Chaque demande dem_d, pour d allant de 1 à $3n$, requiert une quantité rq_d égale à la taille de l'élément de I_{3P} auquel elle est associée. Il s'agit alors de former des batches afin de satisfaire ces demandes. D'une part, une demande ne peut être satisfaite que par une unique batch et d'autre part, la taille des batches doit être comprise entre une borne inférieure bs et une borne supérieure BS. Dans I_{F8}, on pose $bs = BS = l$, où l est la limite imposée dans 3-PARTITION. La construction de I_{F8} se fait clairement en temps polynomial.

Il est évident qu'une solution de 3-PARTITION fournit une solution de I_{F8}. Réciproquement, si l'on dispose d'une solution de I_{F8}, les contraintes imposées par I_{3P} sur les quantités rq_d vis-à-vis de la limite l induisent de manière nécessaire que tous les batches de la solution livrent exactement 3 demandes. Ces batches étant de taille $bs = BS = l$, cette solution de I_{F8} fournit aussi une solution de I_{3P}. Ce qui termine la réduction et conclut la preuve. ⊛

3.2.2 Minimiser les coûts de production

Dans cette partie, nous nous concentrons sur l'optimisation du critère des coûts de production, ce qui est équivalent à la minimisation des coûts fixes liés à l'exécution des batches, en d'autres termes, du fait des hypothèses simplificatrices présentées au début de ce chapitre, cela revient à chercher une solution avec un nombre minimal de batches, ou encore, à décider de l'existence d'une solution avec au plus M batches, avec $M \geq 1$ fini et donné en entrée.

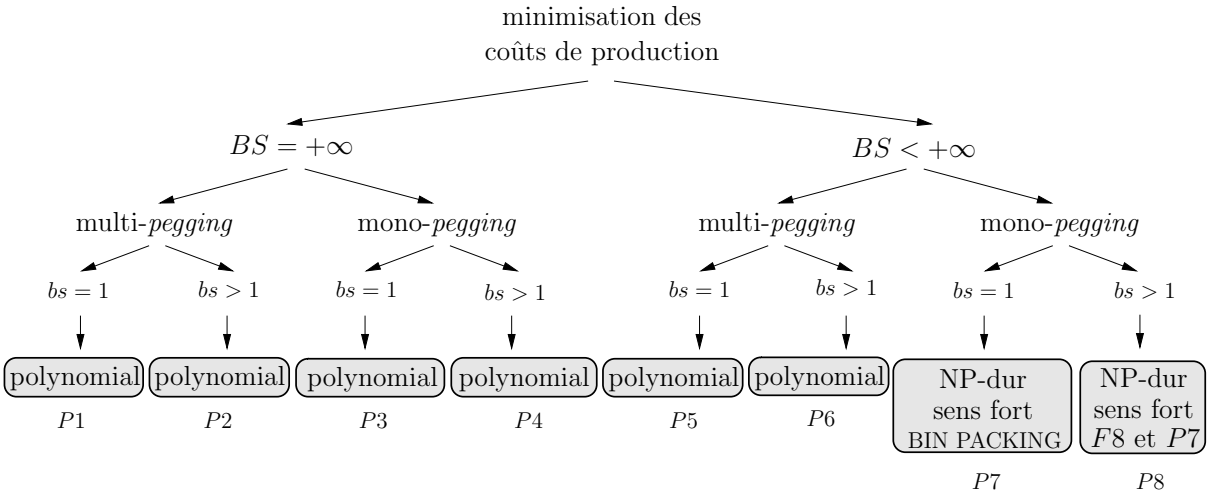

FIG. 3.2 – Minimisation des coûts de production : problèmes abordés

$P1$, $P2$, $P3$ et $P4$ sont trivialement résolus puique la solution où l'on crée un unique batch est réalisable et optimale. Le problème $P5$ correspond au problème de faisabilité $F5$. La procédure permettant d'exhiber une solution à ce problème qui a été décrite dans la section précédente crée exactement $\lceil Q/BS \rceil$ batches qui est une borne inférieure du nombre nécessaire de batches, c'est donc la solution optimale. Pour le problème $P6$ (qui correspond au problème de faisabilité $F6$), si le critère d'existence d'une solution est vérifié, alors, la solution où l'on crée $\lceil Q/BS \rceil$ batches est aussi réalisable et optimale.

Le problème $P7$ peut être reformulé comme le problème de BIN PACKING où les boîtes sont des batches et les articles sont des demandes. Les boîtes ont une capacité maximale qui correspond à la taille maximale des batches BS et placer toutes les demandes dans un minimum de batches (sans couper une demande, du fait de la contrainte de mono-*pegging*) équivaut à placer tous les articles dans un minimum de boîtes. Cela définit donc bien le problème de BIN PACKING. Comme BIN PACKING est NP-complet au sens fort [52], le problème $P7$ l'est aussi.

Enfin, le problème $P8$ correspond au problème de faisabilité $F8$ auquel on a ajouté une fonction d'optimisation, il est donc plus difficile que $F8$. Comme ce dernier a été prouvé NP-complet au sens fort, il en est de même pour $P8$. On peut aussi noter que $P8$ étant au moins

aussi difficile que $P7$, déjà NP-complet au sens fort, il en est naturellement de même pour $P8$.

3.2.3 Minimiser les coûts de retard

La définition d'un objectif faisant référence au retard sur les livraisons tend à nous rapprocher d'un problème d'ordonnancement et notamment du problème d'ordonnancement par batch présenté dans la Section 2.3. Nous étudions trois caractéristiques critiques : les poids pour les coûts de retard (tous égaux ou non), la contrainte de *pegging* (mono ou multi) et enfin la valeur de la borne inférieure bs sur la taille des batches ($bs = 1$ ou $bs > 1$). En fonction des variantes induites par les différentes conjonctions de ces caractéristiques, des propriétés de dominances peuvent être dégagées, nous en proposons quatre. Puis nous procéderons à l'analyse de complexité à proprement parler.

3.2.3.1 Dominances

Propriété 2 (Bloc). Dans tous les cas étudiés, les ordonnancements en un seul bloc (sans temps mort) et calés à gauche sont dominants.

Justification. Comme le début de l'exécution des batches n'est pas contraint, il est possible d'exécuter les batches en un seul bloc. En outre, comme il n'y a pas de pénalité liée à la livraison en avance des demandes, il est préférable de les exécuter au plus tôt, ce qui conduit bien à la dominance ci-dessus. □

Propriété 3 (Taille minimale). Lorsque la taille des batches n'est pas bornée inférieurement ($bs = 1$), les ordonnancements où l'on crée Q batches de taille 1, en multi-*pegging*, ou un batch par demande en mono-*pegging* sont dominants.

Justification. Le nombre de batches n'est pas critique puisqu'il n'intervient pas dans le coût, il est donc possible d'en faire autant que l'on souhaite. Pour les cas multi-*pegging*, supposons que l'on dispose d'une solution optimale dans laquelle il existe un batch de taille q_b supérieure à 1, il est toujours possible de découper ce dernier en q_b batches de taille 1 sans augmenter le coût de la solution puisque les livraisons ont lieu plus tôt. Pour le cas mono-*pegging*, supposons que l'on dispose d'une solution optimale dans laquelle un batch satisfait N demandes notées dem_d, d allant de 1 à N, avec $N > 1$. De la même façon, on peut découper ce batch en N batches de taille respective rq_d, pour d allant de 1 à N. À nouveau, chaque demande étant livrée plus tôt, cette solution ne peut augmenter le coût de la fonction objetif. □

Par contre, dès que la taille minimale est restrictive ($bs > 1$), les solutions avec un nombre de batches maximal ne sont pas dominantes.

Contre-exemple. Nous traitons ici le cas où $bs > 1$, avec poids unitaires sur les coûts de retards et autorisant le multi-*pegging*. Exhibons une instance de ce problème dans laquelle il faut créer strictement moins de batches que le nombre de batches maximal possible :

- la quantité à produire est $Q = 16$,
- la taille minimale des batches est $bs = 5$,
- il y a 2 demandes dem_1 et dem_2,
- les quantités requises sont $rq_1 = rq_2 = 8$,
- les dates dues sont $dt_1 = 8$ et $dt_2 = 15$.

Le nombre maximal de batches est $\lfloor Q/bs \rfloor = 3$. Dans toute solution à 3 batches, il y a 1 batch de taille 6 et 2 batches de taille 5. Distinguons le cas où le dernier batch participe à la satisfaction de dem_1 du cas où il ne produit que pour dem_2 :

1. Si le dernier batch produit en partie pour dem_1, il y a au moins une unité de dem_1 en retard de 8 unités et au moins une autre, livrée par le second batch (puisque la taille du premier batch ne peut excéder 6), donc en retard de 2 unités de temps ; cela implique un coût supérieur ou égal à 10.
2. Si le dernier batch ne produit que pour dem_2, il y a au moins 2 unités de dem_1 livrées par le deuxième batch, donc en retard de 2 unités de temps et au moins 5 unités de dem_2 en retard d'une unité de temps ; cela implique un coût supérieur ou égal à 9.

La solution où l'on crée deux batches de taille 8, le premier pour la demande dem_1, le second pour dem_2, présente un coût de 8. La Propriété 3 n'est donc pas vérifiée ici, ce qui montre la limite de sa portée.

Propriété 4 (EDD). Dans les cas multi-*pegging*, avec poids unitaires sur les coûts de retards, les ordonnancements dans lesquels les demandes sont satisfaites dans l'ordre des dates dues croissantes sont dominants.

Preuve. Supposons que l'on dispose d'une solution optimale S avec deux demandes dem_d et $dem_{d'}$ (telles que $dt_d > dt_{d'}$) et deux batches bat_b et $bat_{b'}$ (tels que $et_b < et_{b'}$), telle que $f_{b,d} > 0$ et $f_{b',d'} > 0$. Nous allons montrer qu'il est possible de construire une solution $\tilde{S}$ dans laquelle $\tilde{f}_{b,d} = 0$ ou $\tilde{f}_{b',d'} = 0$ et dont le coût total est inférieur ou égal au coût de S ($\tilde{f}_{b,d}$ désigne la partie de la demande dem_d satisfaite par le batch bat_b dans $\tilde{S}$).

Soit $\phi = \min(f_{b,d}\ ,\ f_{b',d'})$. Nous allons montrer que la solution $\tilde{S}$, construite à partir de S, dans laquelle $\tilde{f}_{b,d} = f_{b,d} - \phi$, $\tilde{f}_{b,d'} = f_{b,d'} + \phi$, $\tilde{f}_{b',d} = f_{b',d} + \phi$ et $\tilde{f}_{b',d'} = f_{b',d'} - \phi$ est de coût inférieur à celui de S (cf. Figure 3.3). Intuitivement, dans $\tilde{S}$, on échange la partie de la demande dem_d satisfaite par le batch bat_b avec celle de la demande $dem_{d'}$ satisfaite par le batch $bat_{b'}$, et rien d'autre n'est modifié dans S. Six cas (illustrés sur la Figure 3.3) dépendant des placements relatifs des dates dues dt_d et $dt_{d'}$, et des dates de fin de et_b et $et_{b'}$, doivent être examinés. Pour chacun des cas, on compare la contribution $c(S)$ de $(f_{b,d}, f_{b,d'}, f_{b',d}, f_{b',d'})$ dans S et la contribution $c(\tilde{S})$ de $(\tilde{f}_{b,d}, \tilde{f}_{b,d'}, \tilde{f}_{b',d}, \tilde{f}_{b',d'})$ dans $\tilde{S}$. La contribution de toutes les autres décisions de *pegging* est la même dans S et dans $\tilde{S}$.

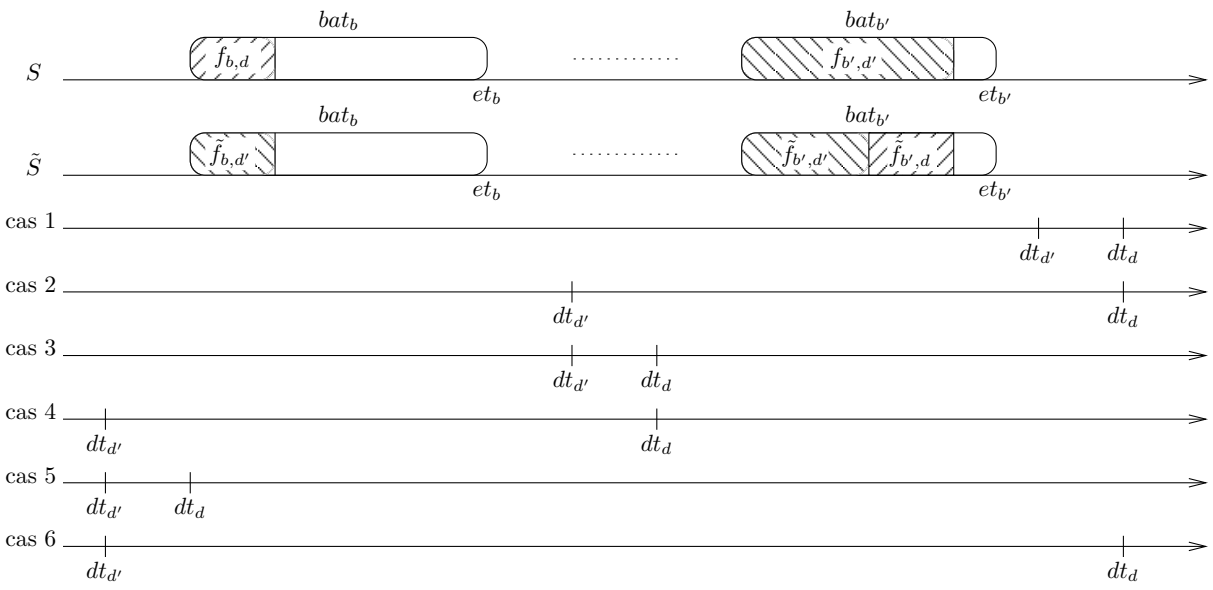

FIG. 3.3 – Illustration de la preuve de la Propriété 4

cas 1. $et_b < et_{b'} \leq dt_{d'} < dt_d$

$$c(S) = c(\tilde{S}) = 0$$

cas 2. $et_b \leq dt_{d'} \leq et_{b'}$ et $et_{b'} \leq dt_d$

$$\begin{aligned} c(S) - c(\tilde{S}) &= (f_{b',d'} - \tilde{f}_{b',d'})(et_{b'} - dt_{d'}) = \phi(et_{b'} - dt_{d'}) \\ &\text{et} \quad et_{b'} \geq dt_{d'}, \text{ donc, } c(S) - c(\tilde{S}) \geq 0 \end{aligned}$$

cas 3. $et_b \leq dt_{d'} \leq et_{b'}$ et $et_b \leq dt_d \leq et_{b'}$

$$\begin{aligned} c(S) - c(\tilde{S}) &= (f_{b',d} - \tilde{f}_{b',d})(et_{b'} - dt_d) + (f_{b',d'} - \tilde{f}_{b',d'})(et_{b'} - dt_{d'}) \\ &= -\phi(et_{b'} - dt_d) + \phi(et_{b'} - dt_{d'}) = \phi(dt_d - dt_{d'}) \\ &\text{et} \quad dt_d > dt_{d'}, \text{ donc, } c(S) - c(\tilde{S}) > 0 \end{aligned}$$

cas 4. $dt_{d'} \leq et_b$ et $et_b \leq dt_d \leq et_{b'}$

$$\begin{aligned} c(S) - c(\tilde{S}) &= (f_{b',d} - \tilde{f}_{b',d})(et_{b'} - dt_d) + (f_{b',d'} - \tilde{f}_{b',d'})(et_{b'} - dt_{d'}) + (f_{b,d'} - \tilde{f}_{b,d'})(et_b - dt_{d'}) \\ &= -\phi(et_{b'} - dt_d) + \phi(et_{b'} - dt_{d'}) - \phi(et_b - dt_{d'}) = \phi(dt_d - et_b) \\ &\text{et} \quad dt_d \geq et_b, \text{ donc, } c(S) - c(\tilde{S}) \geq 0 \end{aligned}$$

cas 5. $dt_{d'} \leq et_b$ et $dt_d \leq et_b$

$$\begin{aligned} c(S) &= (f_{b',d} - \tilde{f}_{b',d})(et_{b'} - dt_d) + (f_{b',d'} - \tilde{f}_{b',d'})(et_{b'} - dt_{d'}) \\ &+ (f_{b,d} - \tilde{f}_{b,d})(et_b - dt_d) + (f_{b,d'} - \tilde{f}_{b,d'})(et_b - dt_{d'}) \\ &= -\phi(et_{b'} - dt_d) + \phi(et_{b'} - dt_{d'}) + \phi(et_b - dt_d) - \phi(et_b - dt_{d'}) \\ &= 0 \end{aligned}$$

cas 6. $dt_{d'} \leq et_b$ et $et_{b'} \leq dt_d$

$$\begin{aligned} c(S) - c(\tilde{S}) &= (f_{b,d'} - \tilde{f}_{b,d'})(et_b - dt_{d'}) + (f_{b',d'} - \tilde{f}_{b',d'})(et_{b'} - dt_{d'}) \\ &= -\phi(et_b - dt_{d'}) + \phi(et_{b'} - dt_{d'}) = \phi(et_{b'} - et_b) \\ &\text{et} \quad et_{b'} > et_b, \text{ donc, } c(S) - c(\tilde{S}) > 0 \end{aligned}$$

Finalement, nous avons montré que, dans les 6 cas possibles, soit $c(S) = c(\tilde{S})$, soit $c(S) \geq c(\tilde{S})$, ou $c(S) > c(\tilde{S})$. La dernière inégalité implique que nous avons construit une solution de coût strictement meilleur que celui de S, S n'est donc pas optimale ce qui contredit l'hypothèse et prouve la Propriété. ⊛

Contre-exemple 1. Dans un premier temps, nous montrons que la Propriété 4 ne s'applique plus dès que la contrainte de mono-*pegging* est imposée. Nous exhibons donc une instance avec des poids unitaires sur les retards et sans taille de batch minimale mais où le mono-*pegging* est requis, pour laquelle il est préférable de livrer les demandes selon un autre ordre que EDD :

- la quantité à produire est $Q = 10$,
- il y a 2 demandes dem_1 et dem_2,
- les quantités requises sont $rq_1 = 2$ et $rq_2 = 8$,
- les dates dues sont $dt_1 = 3$ et $dt_2 = 4$.

D'après la Propriété 3, les ordonnancements où l'on crée un batch par demande sont dominants, il y a donc deux possibilités. La Figure 3.4 présentent ces deux solutions selon qu'on livre dem_1 avant ou après dem_2.

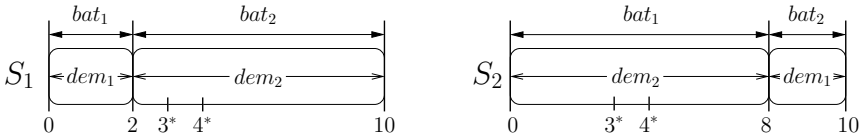

FIG. 3.4 – Limites de la Propriété 4 : premier contre-exemple.

Nous avons alors $OBJ(S_1) = 6 \times 8 = 48$ et $OBJ(S_2) = 8 \times 4 + 2 \times 7 = 46$. La solution S_2 est donc de meilleur coût alors que la demande dem_2 y est livrée en premier et cependant sa date due est plus élevée que celle de dem_1. Cela prouve que la Propriété 4 ne s'applique plus dès que l'on impose le mono-*pegging*.

Contre-exemple 2. Montrons maintenant que la Propriété 4 ne s'applique plus dès que l'on a des poids non-nécessairement égaux à 1 sur les coûts de retard. Nous exhibons donc une instance où de tels poids sont définis sur les retards, sans taille de batch minimale et où le multi-*pegging* est autorisé, pour laquelle il est préférable de livrer les demandes selon un autre ordre que EDD :

- la quantité à produire est $Q = 5$,
- il y a 2 demandes dem_1 et dem_2,
- les quantités requises sont $rq_1 = 2$ et $rq_2 = 3$,
- les dates dues sont $dt_1 = 2$ et $dt_2 = 1$,
- les poids sont $tc_1 = 10$ et $tc_2 = 2$.

D'après la Propriété 3, les ordonnancements où l'on crée $Q = 5$ batches de taille 1 sont dominants. La Figure 3.5 présentent deux solutions admissibles selon qu'on livre dem_1 avant ou après dem_2.

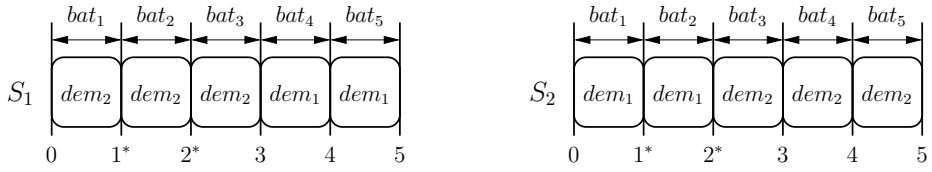

FIG. 3.5 – Limites de la Propriété 4 : second contre-exemple.

Nous avons alors $OBJ(S_1) = 2 \times (1+2) + 10 \times (2+3) = 56$ et $OBJ(S_2) = 2 \times (2+3+4) = 18$. La solution S_2 est donc de meilleure coût alors que la demande dem_1 y est livrée en premier et cependant sa date due est plus élevée que celle de dem_2. Cela prouve que la Propriété 4 ne s'applique plus dès que l'on a des poids quelconques sur les pénalités de retard.

Propriété 5 (BS implicite). Dans les cas multi-*pegging*, les ordonnancements où la taille des batches est comprise entre bs et $2bs - 1$ sont dominants (on a donc une borne supérieure sur la taille des lots).

Preuve. Soit une instance du problème de *batching* où des poids quelconques peuvent être définis sur les pénalités de retard, la taille de batch peut être bornée inférieurement et le multi-*pegging* est autorisé. Supposons que l'on dispose d'une solution optimale S pour cette instance dans laquelle il y a un batch bat_b de taille q_b au moins égale à $2bs$. Il est toujours possible de redécouper bat_b en un certain nombre de batches de taille comprise entre bs et $2bs - 1$, et ainsi, les demandes qui étaient livrées par bat_b dans S sont partitionnées en deux sous-ensembles : celles qui sont désormains livrées plus tôt, et celles qui sont livrées par le dernier batch du redécoupage. Les pénalités de retard encourues pour les demandes du premier sous-ensemble sont au pire les mêmes (il n'y en avait pas) et au mieux diminuées. Les pénalités associées aux demandes du second sous-ensemble sont les mêmes que dans S. Cette propriété ne s'applique pas dans le cas général du mono-*pegging* car les quantités rq_d des demandes peuvent tout à fait dépasser $2bs - 1$. ⊛

3.2.3.2 Synthèse sur les dominances, complexité

Outre les résultats de complexité que nous avons obtenus et qui sont analysés à la suite, nous donnons sur la Figure 3.6 les variantes de problèmes auxquelles s'appliquent les quatre dominances (Propriétés 2 à 5) présentées ci-dessus.

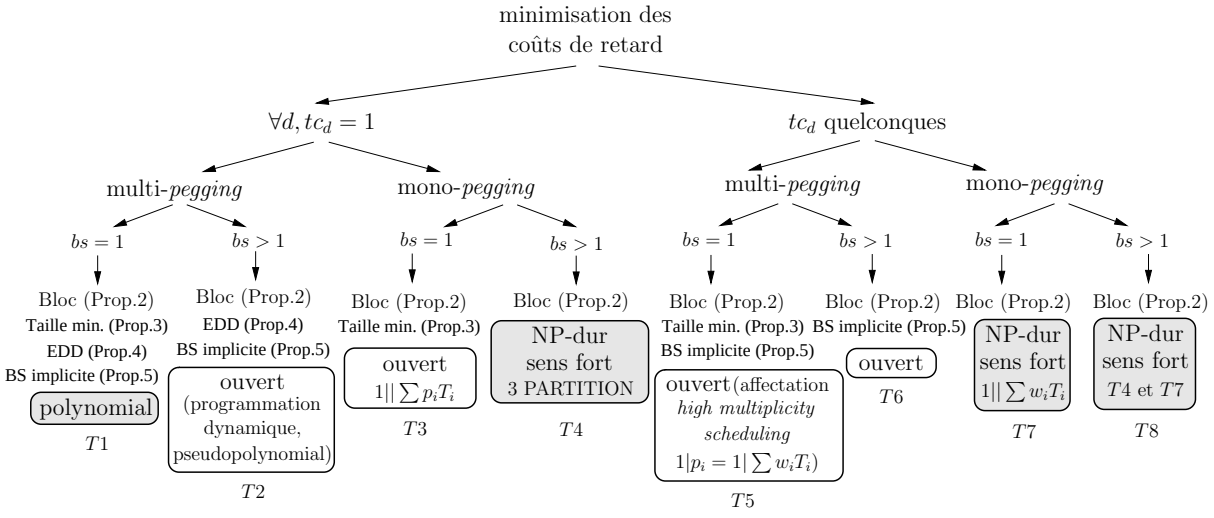

FIG. 3.6 – Minimisation des coûts de retard : problèmes abordés

3.2.3.3 Cas polynomial

D'après les Propriétés 2, 3 et 4, il existe une solution optimale pour le problème $T1$ qui peut être déterminée en temps polynomial ($O(n \log n)$, où n est le nombre de demandes) par application de la règle EDD. Il convient ici de se préoccuper de la formulation de la solution. En effet, pour exhiber une solution optimale, nous utilisons la Propriété 3 qui mène à la création de Q batches. Il s'agit donc de ne pas exprimer la solution batch par batch, ce qui serait pseudopolynomial, mais de les partitionner en un nombre polynomial de sous-ensembles. Un sous-ensemble regroupant tous les batches servant une demande donnée, il y a donc autant de sous-ensembles de batches que de demandes, ce qui est bien polynomial.

3.2.3.4 Problèmes ouverts.

Cas non NP-durs au sens fort. Bien qu'il ne soit pas prouvé que ce problème n'est pas polynomial, nous savons que $T2$ n'est pas NP-complet au sens fort car nous présentons au Chapitre 4, en Section 4.1, une méthode pseudopolynomiale de résolution par un algorithme de programmation dynamique. Cependant, la Propriété 3 ne s'appliquant pas dans ce cas (dominance des solutions avec un nombre maximal de batches), il nous semble peu probable que ce

problème soit polynomial. En outre, $T2$ appartient à NP, ce résultat n'étant pas trivial, nous présentons le codage raisonnable nécessaire à l'expression d'une solution en Section 4.1.

Par ailleurs, bien que nous ne puissions statuer de la complexité de $T5$, nous savons qu'il n'est pas NP-dur au sens fort. En effet, d'après les Propriétés 2 et 3, une solution optimale peut être obtenue en créant Q batches de taille 1, ordonnancés les uns à la suite des autres, sans trou, et calés à gauche. Il s'agit donc de décider pour chaque batch, à quelle demande livrer sa production. On suppose que les batches sont indicés selon la séquence de l'ordonnancement, la date de fin du batch bat_b est donc égale à b. L'affectation du batch bat_b à la demande dem_d induit une coût nul si le batch se termine avant la date due de la demande ($b \leq dt_d$), il est égal à $tc_d(b - dt_d)$ sinon. La solution optimale à ce problème est obtenue en résolvant un problème d'affectation classique pour lequel il existe des algorithmes polynomiaux (flot à coût minimal, [120, 25, 45]). Cependant, ici, la taille de l'entrée du problème d'affectation est de l'ordre de Q, ce qui est pseudopolynomial. Il s'agit du problème classique d'ordonnancement $1|p_i = 1|\sum w_i T_i$ en *high multiplicity scheduling* dont la complexité est ouverte à ce jour. Notons que la question de son appartenance à NP est elle-aussi ouverte, en effet, il n'est pas évident qu'un codage raisonnable puisse être trouvé pour en exprimer une solution.

Cas ouverts. L'introduction de poids quelconques sur les retards des livraisons empêche la généralisation de l'algorithme cité ci-dessus (pour $T2$) au problème $T6$, sa complexité est donc ouverte, $T5$ en étant un cas particulier, nous ne pouvons pas non-plus répondre à la question de son appartenance à NP. Le cas du problème $T3$ est resté non-résolu. D'après la Propriété 3 et la contrainte de mono-*pegging*, il existe une solution optimale dans laquelle un batch est affecté exclusivement à une demande. La taille d'un batch bat_b, qui est aussi sa durée, est alors égale à la quantité rq_d de la demande dem_d qu'il sert. L'objectif consiste à minimiser le produit de rq_d par le retard, c'est-à-dire le délai entre la date de fin du batch et la date due dt_d de dem_d. En termes d'ordonnancement, en utilisant la notation classique de Graham [55], on obtient alors le problème $1||\sum p_i T_i$. Le cas particulier où les tâches sont toutes de durée 1 nous ramène au problème $1|p_i = 1|\sum T_i$, qui est polynomial. Nous revenons sur ce problème dans la section suivante en donnant quelques observations.

3.2.3.5 Problèmes NP-complets au sens fort.

Nous apportons une preuve de la NP-complétude du problème $T4$ dans la suite du chapitre, par réduction de 3-PARTITION. Par ailleurs, en utilisant des arguments similaires que pour le problème $T3$, on constate que le problème $T7$ est équivalent au problème classique d'ordonnancement $1||\sum w_i T_i$, qui est NP-difficile au sens fort [73], ce qui permet de conclure sur la complexité de $T7$. Par conséquent, le problème $T8$ étant plus général que $T4$ d'une part et $T7$ d'autre part, il est aussi NP-difficile au sens fort.

3.2.3.6 Remarques au sujet du problème $1||\sum p_iT_i$

Cette variante du problème nous intéresse particulièrement, car non-seulement nous la résolvons souvent dans la pratique : chaque unité de produit requis nécessite la même durée 1 et est affectée du même poids $tc_d = 1$, ce qui mène à $w_i \Leftrightarrow p_i$. Elle présente de plus un intérêt du point de vue de la théorie de l'ordonnancement. Nous présentons donc dans une section dédiée les premières idées que nous avons dégagées sur ce sujet.

$1||\sum w_iT_i$ et algorithme de Lawler. Dans le cas général, le problème $1||\sum w_iT_i$ est NP-complet au sens fort. Cependant, Lawler propose, dans [73], une procédure pseudopolynomiale s'appliquant dans le cas particulier où $\forall i, j, p_i < p_j \Rightarrow w_i \geq w_j$. Or dans notre cas, $w_i = p_i$, donc la condition requise par Lawler n'est pas satisfaite. Cela ne signifie pas nécessairement que notre problème est NP-complet au sens fort, mais cela nous laisse penser que si une telle condition est nécessaire à l'applicabilité de l'algorithme de Lawler, il y a de fortes chances qu'il soit difficile (voire impossible) de trouver une procédure pseudopolynomiale fonctionnant dans des cas où elle n'est pas vérifiée.

$1|d_i = d|\sum p_iT_i$. Il s'agit du cas particulier où il n'y a qu'une date due, commune à toutes les demandes (ou tâches). Dans ce problème, seules les tâches ordonnancées après d contribuent à l'augmentation de la fonction objectif. On va donc s'intéresser à la résolution du problème $1||\sum w_iC_i$, sur les tâches en retard, avec $w_i = p_i$. Ce problème peut être résolu grâce à la règle de Smith, qui consiste à placer les tâches dans l'ordre des p_i/w_i croissants. Or dans notre cas, pout tout i, $p_i/w_i = 1$, donc la séquence n'a finalement pas d'importance. Il reste donc à déterminer, dans le problème initial, la tâche "pivot" $J_{\hat{i}}$ ordonnancée à cheval sur d ainsi que l'ensemble des tâches qui vont être ordonnancées après $J_{\hat{i}}$. Nous allons montrer qu'un ordonnancement des tâches selon la règle LPT (*Largest Processing Time*, c'est-à-dire selon les durées décroissantes) est dominant et permet donc de déterminer la tâche pivot et l'ensemble des tâches en retard.

Propriété 6. Pour le problème $1|d_i = d|\sum p_iT_i$, les ordonnancements dans lesquels les tâches sont exécutées dans l'ordre des durées décroissantes sont dominants.

Preuve. Supposons que l'on dispose d'une solution optimale S avec deux tâches consécutives J_i et J_{i+1} telles que $p_i < p_{i+1}$ (les tâches sont indicées dans l'ordre de la séquence de S et on note C_j la date de fin de toute tâche J_j). Nous allons montrer qu'il est possible de construire un ordonnancement $\tilde{S}$ de coût inférieur au coût de S dans lequel J_i et J_{i+1} sont inversées.

(1) Si $i+1 < \hat{i}$, les tâches J_i et J_{i+1} ne sont pas en retard, leur contribution dans S est donc nulle et leur inversion est donc sans effet sur le coût global.

(2) Si $i+1 = \hat{i}$, deux cas se présentent alors : soit J_{i+1} reste à cheval sur d dans $\tilde{S}$ (Figure 3.7, cas (a)), soit J_i devient la tâche pivot (Figure 3.7, cas (b)). Dans le premier cas, un calcul simple montre que la différence entre les contributions de J_i et J_{i+1} dans S et dans $\tilde{S}$ est égale à $p_i(p_{i+1} - C_{i+1} + d)$, et par construction $C_{i+1} - p_{i+1} < d$, la différence est donc

strictement positive. Dans le second cas, cette différence est égale à $(C_{i+1} - d)(p_{i+1} - p_i)$, elle est par hypothèse strictement positive. L'inversion des deux tâches J_i et J_{i+1} améliore donc le coût de l'ordonnancement.

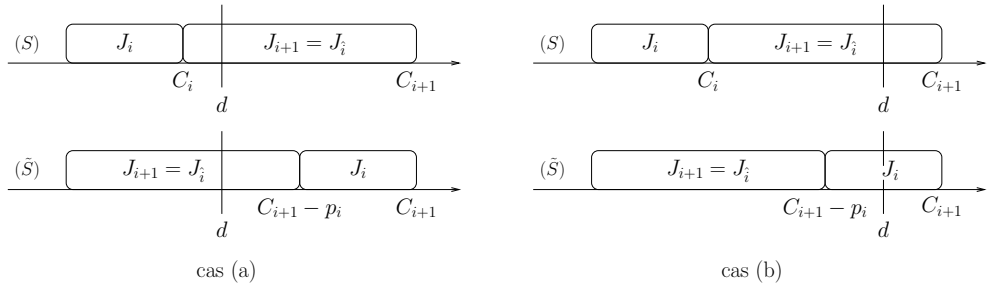

FIG. 3.7 – Illustration de la preuve de la Propriété 6 (2)

(3) Si $i \geq \hat{i}$ dans S, un calcul simple montre que la différence entre les contributions de J_i et J_{i+1} dans S et dans $\tilde{S}$ est égale à $p_i(C_i - C_{i+1} + p_{i+1})$, et comme $C_i = C_{i+1} - p_{i+1}$, ce terme est nul, l'inversion de J_i et J_{i+1} dans $\tilde{S}$ est donc sans effet sur le coût global.

Finalement, l'ordonnancement $\tilde{S}$ où les tâches J_i et J_{i+1} sont placées dans l'ordre des durées déroissantes est de coût inférieur ou égal à celui de la solution S, cette dernière n'est donc pas optimale, ce qui contredit l'hypothèse et prouve la dominance. ⊛

La Propriété 6 permet de résoudre problème $1|d_i = d|\sum p_i T_i$ en temps polynomial ($O(n \log n)$). Malgré l'intérêt contenu dans l'étude en elle-même de ce cas particulier, cela ne permet pas de conclure sur le cas général.

3.3 Preuve de complexité pour le problème $T4$

3.3.1 Définition du problème d'optimisation

Soit un ensemble $\mathcal{D}em$ de n demandes dem_d pour un même produit. Chaque demande possède les caractéristiques suivantes :

- rq_d est la quantité de produit requise par dem_d,
- dt_d est la date due de dem_d.

On dispose d'une machine permettant d'élaborer le produit demandé. Cette machine fonctionne en exécutant des batches bat_b de quantité (ou taille) variable q_b. La taille d'un batch correspond à la quantité de produit obtenue à la fin de son exécution. Une restriction contraint ces quantités q_b à être au moins égales à une taille minimale bs. En outre, la durée d'exécution d'un batch bat_b est égale à sa taille q_b et on note et_b sa date de fin.

Une demande dem_d ne peut être satisfaite que par un unique batch bat_b (mono-*pegging*). Si le batch bat_b satisfait dem_d et que bat_b se termine après la date due de dem_d ($et_b > dt_d$), une pénalité de retard est alors encourue. Cette pénalité est exprimée par le terme suivant : $rq_d(et_b - dt_d)^+$.

Le problème $T4$ consiste à former et ordonnancer des batches bat_b afin de satisfaire l'intégralité des demandes tout en respectant les contraintes sur la taille des batches et sur l'affectation unique d'un batch à une demande, de façon à minimiser la somme des pénalités de retard. Il s'agit donc d'exhiber un ensemble $\mathcal{C}$ de couples (d, b) tels que la demande dem_d est affectée à un unique batch bat_b et qui minimise l'expression suivante :

$$\sum_{(d,b)\,\in\,\mathcal{C}} rq_d(et_b - dt_d)^+$$

Notations. On note m le nombre de batches finalement créés, et Q la quantité totale de produit requise ($Q = \sum_d rq_d$).

3.3.2 Complexité

Problème de décision Π. Soit K un nombre réel. On appelle Π le problème de décision consistant à déterminer s'il existe une solution au problème $T4$ de valeur inférieure ou égale à K.

Théorème 2. Π est NP-complet au sens fort.

Preuve. Afin de prouver ce résultat, nous allons montrer les deux points suivants :

1. $\Pi \in$ NP ;
2. Π est NP-difficile au sens fort.

Remarques préliminaires

Codage d'une solution de Π. Pour obtenir le certificat en temps polynomial, il faut d'abord s'assurer qu'une solution peut se coder de manière polynomiale en la taille du problème. Une telle solution peut par exemple consister en la donnée des couples (d, b) tels que la demande dem_d est satisfaite par le batch bat_b. Comme une demande ne peut être satisfaite que par un unique batch, il y a donc exactement n couples à exhiber. Le codage d'une solution est bien polynomial en la taille du problème. Remarquons qu'il n'est pas utile d'expliciter la taille q_b de chaque batch puisqu'elle peut se déduire à partir de la donnée des couples (d, b).

Nombre de batches m. Il est possible de créer un nombre de batches entre 1 et $\lfloor Q/bs \rfloor$. Cependant, d'après la contrainte de mono-*pegging*, il y aura toujours au maximum n batches (cas limite où l'on crée un batch par demande). On a donc $m \leq min(n, \lfloor Q/bs \rfloor)$.

1. $\Pi \in$ NP

Soit une solution S_Π au problème Π, c'est-à-dire un ensemble $\mathcal{C}$ de couples (d, b). On peut déduire en temps polynomial les informations suivantes :

$$\forall b, \quad q_b = \sum_{d \setminus (d,b) \in \mathcal{C}} rq_d \qquad (\Rightarrow O(n))$$

$$\forall b, \quad et_b = \sum_{k \leq b} q_k \qquad (\Rightarrow O(m) \in O(n))$$

On a alors un certificat en temps polynomial en vérifiant que :

$$(i) \quad \forall b, \quad q_b \geq bs$$

$$(ii) \quad \forall d, \quad (\exists b, (d,b) \in \mathcal{C}) \text{ et } \left(\nexists b \neq b', (d,b) \in \mathcal{C} \text{ et } (d,b') \in \mathcal{C}\right)$$

$$(iii) \quad \sum_{(d,b) \in \mathcal{C}} rq_d (et_b - dt_d)^+ \leq K$$

2. Π est NP-difficile au sens fort. Pour montrer le second point, nous allons utiliser le problème 3-PARTITION, lui-même NP-complet au sens fort [52]. Nous montrons que ce problème peut être réduit à Π.

Le problème 3-PARTITION**.** On dispose d'un ensemble fini de $3n$ éléments, d'une taille a_i pour i allant de 1 à $3n$, associée à chaque élément, et d'une limite l. Les a_i vérifient $l/4 < a_i < l/2$ et $\sum_i a_i = nl$. Une solution au problème 3-PARTITION est une partition des a_i en n triplets $(a_{j_1}, a_{j_2}, a_{j_3})$ tels que $a_{j_1} + a_{j_2} + a_{j_3} = l$.

Construction en temps polynomial d'une instance I_Π de Π. On fait correspondre chaque élément de l'instance de 3-PARTITION à une demande dem_d. Pour chaque dem_d, d allant de 1 à $3n$, la quantité requise rq_d est égale à a_i et la date due dt_d est fixée à 0, début de l'horizon d'exécution des batches. Il s'agit alors de déterminer s'il existe un ensemble de batches bat_b, chacun de taille $q_b \geq l$ ($bs = l$), permettant de satisfaire toutes les demandes dem_d de façon à ce que chaque demande ne soit satisfaite que par un unique batch, et que le coût de la solution soit inférieur ou égal à $K = \frac{n(n+1)}{2} l^2$. Il est clair que cette instance I_Π de Π se construit bien en temps polynomial.

Une solution de 3-PARTITION **fournit une solution à I_Π.** Supposons que l'on dispose d'une solution S_{3P} au problème 3-PARTITION, montrons que cette solution peut être transformée en une solution S_{I_Π} de I_Π. Chaque triplet $(a_{j_1}, a_{j_2}, a_{j_3})$ de la solution S_{3P} est associé à un batch qui satisfait alors les demandes dem_{j_1}, dem_{j_2} et dem_{j_3} de l'instance I_Π. On construit donc n batches de taille l, puisque chacun des triplets $(a_{j_1}, a_{j_2}, a_{j_3})$ vérifie $a_{j_1} + a_{j_2} + a_{j_3} = l$. La solution S_{I_Π} vérifie bien qu'une demande n'est satisfaite que par un unique batch, et que

chaque batch bat_b est de taille supérieure ou égale à l. Calculons maintenant la valeur d'une telle solution (cf. Figure 3.8). En supposant que les batches sont indicés selon la séquence de l'ordonnancement, on peut exprimer les tailles q_b et les dates de fin et_b en fonction de b et l :

$$\forall b = 1...n, \qquad q_b = l$$
$$\forall b = 1...n, \qquad et_b = bl$$

Tous les batches sont ordonnancés en retard par rapport à la date échue commune 0 et la fonction objectif peut donc s'écrire de la façon suivante :

$$\begin{aligned} \sum_{(d,b)\in\mathcal{C}} rq_d et_b &= \sum_{b=1}^{n} q_b et_b \\ &= \sum_{b=1}^{n} l(bl) = \frac{n(n+1)}{2} l^2 \leq K \end{aligned}$$

Une solution S_{3P} au problème 3-PARTITION fournit donc aussi une solution à I_Π.

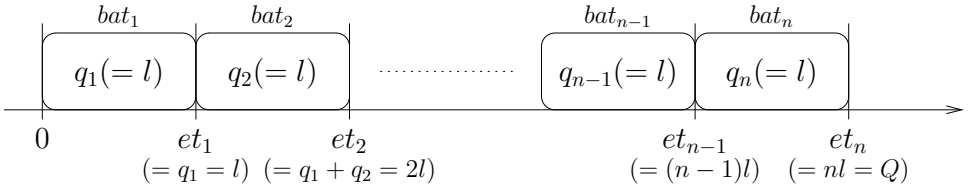

FIG. 3.8 – Ordonnancement des batches

Une solution de I_Π fournit une solution à 3-PARTITION**.** Réciproquement, montrons qu'une solution S_{I_Π} de I_Π correspond nécessairement à une solution du problème 3-PARTITION. La preuve s'effectue en spécifiant la structure d'une solution S_{I_Π}, ce qui permet de faire apparaître une solution de 3-PARTITION. Plus précisément, nous montrons que S_{I_Π} contient exactement n batches de taille l.

(1) Dans un premier temps, nous montrons qu'une solution S_{I_Π} quelconque à m batches est de coût supérieur ou égal à une solution particulière où l'on crée m batches de taille égale, c'est-à-dire que $q_b = Q/m$, pour tout b. Rappelons que :

$$m \leq \lfloor Q/l \rfloor = n$$

(2) Dans un second temps, nous montrons qu'une solution où l'on crée k batches est de coût strictement inférieur à une solution dans laquelle on crée $k-1$ batches, quel que soit $k \in \{2 \dots n\}$, ce qui implique qu'il est plus intéressant de créer un maximum de batches.
(3) Enfin, nous montrons que le coût de la solution minimale à n batches de taille $Q/n = l$ est exactement égal à K, ce qui interdit toute autre structure pour S_{I_Π}.

Considérons un ensemble de couples (d, b). Chacun de ces couples associe une demande dem_d, d allant de 1 à $3n$, à un batch bat_b, b allant de 1 à m. Comme il s'agit d'une solution au problème I_Π, nous avons :

$$Q = \sum_{b=1}^{m} q_b = \sum_{d=1}^{3n} rq_d = nl$$
$$\text{et } \forall b, \qquad q_b \geq l$$

En outre, cette solution vérifie l'inégalité :

$$\sum_{b=1}^{m} q_b et_b \leq \frac{n(n+1)}{2} l^2$$

Montrons le premier point (1). La fonction objectif F_m de la solution S_{I_Π} à m batches peut s'écrire de la façon suivante :

$$\begin{aligned} F_m(q_1, \ldots, q_m) &= q_1^2 + q_2(q_1 + q_2) + q_3(q_1 + q_2 + q_3) + \ldots + q_{m-1} \sum_{b=1}^{m-1} q_b + q_m \sum_{b=1}^{m} q_b \\ &= \sum_{b=1}^{m-1} (q_b \sum_{k=1}^{b} q_k) + (Q - \sum_{b=1}^{m-1} q_b) Q \end{aligned}$$

Les dérivées partielles de F_m s'écrivent :

$$\forall b = 1...m, \qquad \frac{\partial F_m(q_1, \ldots, q_m)}{\partial q_b} = \sum_{k=1}^{b-1} q_k + 2q_b + \sum_{k=q+1}^{m-1} q_k - Q$$

Le vecteur $(Q/m, \ldots, Q/m)^{\mathrm{T}}$ est l'unique solution du système linéaire carré de dimension m suivant, où $H(F_m)$ est la matrice hessienne de F_m (notons que comme $m \leq n$, on a bien $(Q/m) \geq l$) ;

$$H(F_m) \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{pmatrix} = \begin{pmatrix} 2 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 2 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{pmatrix} = \begin{pmatrix} Q \\ Q \\ \vdots \\ Q \end{pmatrix}$$

Montrons que la valeur du point critique de la fonction F_m prise en $(Q/m, \ldots, Q/m)^{\mathrm{T}}$ constitue l'unique minimum global de F_m. Pour cela, nous montrons que la matrice hessienne est définie positive. Par propriété, nous savons que $H(F_m)$ est symétrique. Le critère de Sylvester fournit une condition nécessaire et suffisante pour qu'une matrice symétrique soit définie positive. Il s'agit de montrer que les m déterminants mineurs principaux de $H(F_m)$ sont positifs. En notant h_{ij} le coefficient de la $i^{\text{ème}}$ ligne et de la $j^{\text{ème}}$ colonne de $H(F_m)$, on définit les déterminants mineurs principaux $\mathcal{H}_k(F_m)$, pour k allant de 1 à m, comme suit :

$$\forall k = 1...m, \qquad \mathcal{H}_k(F_m) = \begin{vmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & \cdots & h_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ h_{k1} & h_{k2} & \cdots & h_{kk} \end{vmatrix} = \begin{vmatrix} 2 & 1 & \cdots & 1 \\ 1 & 2 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 2 \end{vmatrix} = \det\left(H(F_k)\right)$$

Compte tenu de la structure de la matrice hessienne considérée, nous avons donc :

$$\forall k = 1...m, \mathcal{H}_k(F_m) \geq 0 \qquad \Leftrightarrow \qquad \forall k = 1...m, \det(H(F_k)) \geq 0$$

Lemme. $\forall k \geq 1, \quad \det(H(F_k)) = k + 1$

Preuve. Nous montrons la propriété par récurrence.

Base. $k = 1$

$$\det(H(F_1)) = 2 = k + 1$$

Induction. Supposons que $\det(H(F_{k-1})) = k$, montrons que $\det(H(F_k)) = k+1$. Le développement en colonnes de $\det(H(F_k))$ est le suivant :

$$\begin{aligned} \det\left(H(F_k)\right) \;=\; 2\det\left(H(F_{k-1})\right) &- \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 1 & \cdots & \vdots \\ 1 & 1 & 2 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & \cdots & \cdots & 2 \end{vmatrix} + \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ 2 & 1 & 1 & \cdots & \vdots \\ 1 & 1 & 2 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & \cdots & \cdots & 2 \end{vmatrix} \\ &- \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ 2 & 1 & 1 & \cdots & \vdots \\ 1 & 2 & 1 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & \cdots & \cdots & 2 \end{vmatrix} + \cdots + (-1)^{k-1} \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ 2 & 1 & 1 & \cdots & \vdots \\ 1 & 2 & 1 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & \cdots & \cdots & 2 & 1 \end{vmatrix} \end{aligned}$$

Les termes sont numérotés de $i = 1$ à k. Pour $i = 2$ à k, on effectue une permutation circulaire des $(i-1)^{\text{èmes}}$ premières colonnes afin d'obtenir à chaque fois la même sous-matrice que dans le terme $i = 2$. Quand i est pair, respectivement impair, il y a un nombre pair, respectivement impair, d'échanges et le signe du $i^{\text{ème}}$ terme est inchangé, respectivement inversé. En outre, quelle que soit la parité de k, le dernier terme est toujours considéré négativement. Finalement tous les termes de $i = 2$ à k sont négatifs et on obtient l'expression suivante, en faisant intervenir l'hypothèse d'induction :

$$\det\left(H(F_k)\right) = 2k - (k\ -\ 1)\begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 1 & \cdots & \vdots \\ 1 & 1 & 2 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & \cdots & \cdots & 2 \end{vmatrix}$$

En retranchant la première colonne aux suivantes, on triangule la matrice de façon à ne faire apparaître que des 1 sur sa diagonale, le déterminant de cette matrice est donc égal à 1, quelle que soit la valeur de k. On a alors :

$$\det(H(F_k)) = 2k - (k-1) = k+1$$

Ce qui conclut la démonstration. □

Il résulte immédiatement du lemme précédent que $\det(H(F_k)) \geq 0$ pour tout k puisque $k \geq 1$. Aussi, les $\mathcal{H}_k(F_m)$ pour k allant de 1 à m sont positifs et la matrice hessienne $H(F_m)$ est donc bien définie positive. La fonction F_m atteint son minimum global en $(Q/m, \ldots, Q/m)^{\mathrm{T}}$. Ainsi, si la solution S_{I_Π} possède m batches, la solution où l'on ne crée que des batches de taille $q_b = Q/m$ est l'unique optimum, ce qui démontre le premier point (1).

Montrons le second point (2). Notons $F_k(Q)$ le coût minimal pour produire la quantité Q en k batches. D'après ce qui précède, il s'agit d'une solution où l'on crée des batches de tailles toutes égales (pour b allant de 1 à k, $q_b = Q/k$). Montrons que pour k allant de 2 à n, $F_k(Q) < F_{k-1}(Q)$:

$$\begin{aligned} \forall k \geq 1,\ \ F_k(Q) &= \frac{Q^2(k+1)}{2k} \\ \frac{F_k(Q)}{F_{k-1}(Q)} &= \frac{(k+1)(k-1)}{k^2} = \frac{k^2-1}{k^2} < 1 \end{aligned}$$

Ce qui prouve la relation ci-dessus et donc le second point (2).

Enfin, montrons le troisième point (3). D'après (1), une solution au problème I_Π de coût minimal contient un nombre $m \leq n$ de batches de tailles toutes égales à Q/m. Or, le coût de la solution de I_Π dans laquelle on crée n batches de taille l a un coût exactement égal à $K = \frac{n(n+1)}{2} l^2$, et, d'après (2), une solution dans laquelle on créerait strictement moins de batches aurait un coût strictement supérieur à K et ne serait donc pas une solution de I_Π. Ainsi, la seule solution S_{I_Π} de l'instance I_Π est obtenue en réalisant n batches de taille l. Compte tenu des restrictions faites sur les quantités rq_d requises par chaque demande dem_d, chaque batch satisfait exactement 3 demandes. Une solution S_{I_Π} produit bien une solution au problème de 3-PARTITION.

Nous avons donc bien réduit 3-PARTITION à notre problème Π, et ce en temps polynomial, ce qui prouve que Π et NP-complet au sens fort. ⊛

3.4 Conclusion

L'étude de complexité que nous avons effectuée dans ce chapitre montre les intérêts du problème de *batching* que nous posons dans cette thèse. En effet, différentes variantes faisant intervenir ou non les contraintes principales du problème ont été présentées et leur complexité s'étend de la trivialité à la NP-difficulté au sens fort, qu'il s'agisse juste de décider de l'existence d'une solution ou d'optimiser une fonction objectif. La question de l'appartenance à NP n'a pas été fermée pour tous les problèmes. Concernant les problèmes d'optimisation, le critère quadratique (produit du retard par la quantité délivrée) est particulièrement intéressant. Il a permis de poser de nouveaux problèmes dont certains ont pu être fermés, quatre sont restés ouverts à ce jour, bien que plusieurs propriétés de dominances aient pu être dégagées. Pour l'un d'eux nous présentons dans le chapitre suivant un algorithme de programmation dynamique de complexité pseudopolynomiale ce qui permet de d'affirmer que, dans le pire des cas, ce problème est NP-complet au sens faible. Sa variante avec des poids quelconques sur les pénalités de retard reste par contre ouverte. Enfin, nous avons dégagé un nouveau problème d'ordonnancement traditionnel (sans regroupement par batch), $1||\sum p_i T_i$. Bien qu'une attention particulière ait été portée sur ce problème, la question de sa complexité est restée sans réponse.

Chapitre 4

Résolution du problème *"cœur"* de *batching*

Ce chapitre est dédié aux méthodes que nous avons envisagées pour résoudre le problème central du *batching* formulé précédemment. Comme nous l'avons vu ce problème se décline en plusieurs variantes. Dans un premier temps, nous présentons l'algorithme de programmation dynamique annoncé dans le cadre de l'étude de complexité. Cette procédure a pu être généralisée à d'autres variantes du problème qui n'ont pas été considérées du point de vue de la complexité. Dans un second temps, nous proposons une approche par modélisation mathématique d'une approximation linéaire du problème alliée à un algorithme de *branch and cut* exécuté par le solveur CPLEX. Des travaux préliminaires relatifs à cette méthode ont été publiés dans [103, 106, 105]. Les derniers travaux ont par ailleurs donné lieu à un article en cours de soumission et ont été présentés dans [104]. Enfin, nous proposons une étude expérimentale permettant de comparer les deux méthodes. Il convient de noter que d'autres analyses de performances de l'approche par programmation linéaire seront présentées au Chapitre 6, dans le cas de la résolution de toutes les variantes et généralisations du problème général de *batching*. L'étude expérimentale proposée dans ce chapitre se limite donc aux instances de problème pouvant être résolues par l'algorithme de programmation dynamique et par la programmation linéaire.

4.1 Programmation dynamique

Nous avons annoncé dans le chapitre précédent l'existence d'un algorithme de programmation dynamique permettant de résoudre le problème que nous avons nommé $T2$ (pas de coût de production, pondération du retard égale à 1, multi-*pegging* autorisé, taille minimale des batches restrictive). Cela nous a permis d'établir que ce problème n'est pas NP-complet au sens fort. Nous présentons dans cette section cette procédure, généralisée à un cas englobant le problème $T2$. En effet, nous traitons ici une conjonction de contraintes et objectif qui n'a pas été étudiée en tant que telle au chapitre précédent mais qui ne fait appel qu'à des caractéristiques qui

ont été examinées ou qui n'apportent pas de complexité supplémentaire, et dont $T2$ est un cas particulier. En ce sens, l'algorithme de programmation dynamique présenté ici résout une version étendue du problème $T2$, notée $\mathcal{P}_{dp}$. Par ailleurs, comme nous restons dans le cadre d'un problème où une unique recette est modélisée, nous conservons les notations simplifiées du chapitre précédent.

4.1.1 Problème traité, propriété et expression d'une solution

4.1.1.1 Définition du problème et dominance

La variante $\mathcal{P}_{dp}$ du problème *"cœur"* que nous nous proposons de résoudre est décrite ci-dessous :

- elle possède les caractéristiques et contraintes suivantes :
  - multi-*pegging* (il s'agit donc bien du problème central de *batching*),
  - durée variable quelconque pour l'exécution d'un batch ($vp \neq 0$),
  - borne inférieure bs sur la taille des batches strictement supérieure à 1,
  - borne supérieure BS sur la taille des batches restrictive ;
- la fonction objectif s'exprime comme la somme des termes suivants :
  - pénalités non-pondérées du produit du retard par les quantités ($tc_d = 1$ pour toute demande dem_d),
  - coûts fixes de production non-nuls ($fc \neq 0$), il existe donc un coût de création d'un batch,

Extension de la Propriété 4. La Propriété 4 (EDD), présentée au chapitre précédent, expose la dominance des ordonnancements dans lesquels les demandes sont livrées dans l'ordre croissant de leur date due. La preuve de cette propriété ne fait pas intervenir d'hypothèses contradictoires avec les caractéristiques présentées ici. Les ordonnancements où les demandes sont livrées selon la règle EDD sont donc aussi dominants pour le problème P_{dp}.

4.1.1.2 Codage polynomial d'une solution

Obtenir un certificat qu'une solution du problème P_{dp} est de coût inférieur à une certaine constante K en temps polynomial n'est pas trivial. Ainsi, si l'on a une seule demande dem_d de taille $rq_d = Q$, le nombre de batches est de l'ordre de $O(\lfloor Q/bs \rfloor)$, ce qui est pseudopolynomial. L'objet de cette section est de proposer un codage raisonnable pour une solution de P_{dp} qui soit suffisant pour avoir ce certificat en temps polynomial, afin de montrer l'appartenance de P_{dp} à NP.

Expression pseudopolynomiale d'une solution. D'après ce qui précède, il existe une solution optimale au problème dans laquelle les demandes sont livrées dans l'ordre croissant des dates dues. Nous verrons que l'algorithme de programmation dynamique utilise cette propriété de sorte que tout ordonnancement fourni en sortie aura cette caractéristique. Cela nous permet

d'exprimer une solution au problème $\mathcal{P}_{dp}$ par la donnée d'un ensemble de valeurs $f_{b,d}$ indiquant la quantité du batch bat_b qu'il est décidé de livrer à la demande dem_d. On note $\mathcal{C}$ l'ensemble des couples (d, b) tels que $f_{b,d} > 0$. Nous avons donc :

$$\begin{aligned} OBJ_{\mathcal{P}_{dp}} &= \sum_{b} (fc + vcq_b) \\ &+ \sum_{(b,d)\in\mathcal{C}} f_{b,d}(dt_d - et_b)^+ \end{aligned} \tag{4.1}$$

Si nous exprimons une solution sous cette forme, c'est-à-dire en listant les batches et le flot transféré sur chaque arc de *pegging*, le codage est pseudopolynomial. Pour prouver que $P_{dp} \in NP$, nous allons présenter dans un premier temps une représentation polynomiale et montrons dans un second temps que ce codage est suffisant pour exprimer une solution au problème P_{dp}.

Théorème 3. Le problème P_{dp} appartient à NP.

Preuve. Comme il a déjà été montré, il existe une solution optimale au problème dans laquelle les n demandes sont livrées les unes après les autres par des batches ordonnancés sans temps mort. Lorsqu'elle est définie par la liste des batches et des arcs de *pegging*, le certificat que cette solution est de coût inférieur ou égal à K s'obtient en temps pseudopolynomial. Nous allons proposer, à partir de cette solution, un codage raisonnable permettant d'attester de manière suffisante qu'il existe au moins une solution de coût inférieur ou égal à K.

La disposition des demandes sur l'horizon met en évidence au maximum $2n$ événements remarquables : les n dates de fin des demandes, notées C_d, d'une part et les n dates dues dt_d d'autre part, pour d allant de 1 à n. De part et d'autre de chacun de ces $2n$ événements, il y a au maximum deux changements de batch (un avant, à l'instant t_i et un après, à l'instant t_{i+1}). Sur la Figure 4.1, les instants t_2 et t_3 "entourent" l'événement remarquable dt_1. Si la date de fin d'un batch coïncide avec un événement remarquable, il n'y a qu'un seul instant t_i considéré. Cela nous permet de repérer au plus $4n$ événements particuliers (t_1 à t_N, $N \leq 4n$) (cf. Figure 4.1). Il est alors possible de donner le nombre N_i de batches situés entre les instants t_i et t_{i+1}. Sur la Figure 4.1, il y a 13 batches, 6 événements remarquables et nous considérons 7 instants t_i. Nous avons $N_1 = 4$ (il y a 4 batches entre t_1 et t_2), $N_2 = 1 \ldots$ Les nombres d'instants t_i et de N_i correspondants sont tous deux égaux à au plus $4n$, ce qui est bien polynomial en la taille des entrées du problème P_{dp}. En faisant cette transformation, une certaine partie de l'information décrivant la solution initiale est perdue, cependant, le lemme suivant montre que la donnée des (t_i, N_i) est suffisante pour obtenir le certificat qu'il existe une solution de coût inférieur à K.

Lemme 1. Supposons les N valeurs t_i et les $(N-1)$ valeurs N_i correspondants donnés. Il est possible de calculer en temps polynomial le coût de la meilleure solution telle que les instants t_i soient des changements de batches et qu'il y ait N_i batches entre t_i et t_{i+1}.

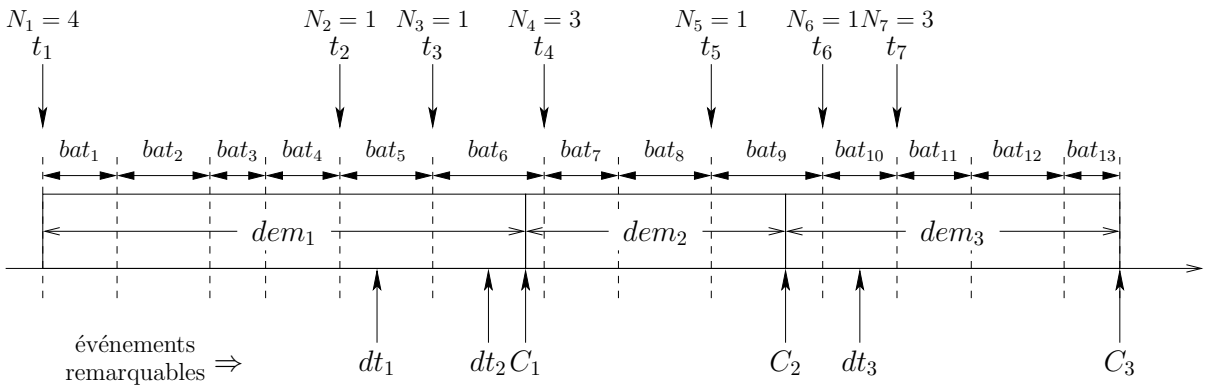

FIG. 4.1 – Codage raisonnable d'une solution au problème P_{dp}

Preuve. Nous allons prouver que, pour une solution optimale, entre t_i et t_{i+1}, la production est découpée en $N_i = n_i + n'_i$ batches $(n_i, n'_i \geq 0)$, tel que n_i batches ont une taille q et n'_i de taille $q+1$. Notons que si $N_i = 1$, on crée un unique batch de taille $(t_{i+1} - t_i)/vp$.

Supposons qu'il existe une solution S optimale dans laquelle, entre deux événements remarquables t_i et t_{i+1}, la production totale $Q = (t_{i+1} - t_i)/vp$ soit découpée en un ensemble de N_i batches, indicés de $b = 1$ à N_i, tel qu'il existe bat_{b_1} et bat_{b_2}, pour lesquels $q_{b_1} - q_{b_2} \geq 2$. Nous construisons une solution $\tilde{S}$, dans laquelle :

$$\tilde{q}_b = q_b \text{ pour } b \notin \{b_1, b_2\}, \tilde{q}_{b_1} = q_{b_1} - 1 \text{ et } \tilde{q}_{b_2} = q_{b_2} + 1$$

Par hypothèse, une seule demande dem_d est livrée dans l'intervalle $[t_i, t_{i+1}]$. Comme le nombre de batches est fixé à N_i et que la quantité globale produite dans l'intervalle n'est pas modifiée, le coût de production est le même dans les solutions S et $\tilde{S}$. En outre, si $dt_d \geq t_{i+1}$, l'équilibrage des tailles de bat_{b_1} et bat_{b_2} n'a aucun effet sur le coût de la solution. Étudions alors la différence entre le coût de retard de la solution S et celui de la solution $\tilde{S}$, dans le cas où $dt_d < t_i$. Le coût de retard donné par l'expression générale :

$$\sum_{b=1}^{N_i} q_b \left((t_i + vp \sum_{b'=1}^{b} q_{b'}) - dt_d \right)$$

Tous les batches étant en retard, le sous-problème étudié est équivalent à $1||\sum p_i C_i$ et la séquence n'a pas d'influence sur le coût (cf. Chapitre 3, Section 3.2.3.6). On peut donc réordonnancer les batches entre t_i et t_{i+1} dans un ordre quelconque sans modifier le coût total. Plaçons bat_{b_1} à l'instant t_i et bat_{b_2} immédiatement à sa suite, le coût du nouvel ordonnancement est égal à celui de S. La différence entre les coûts de S et de $\tilde{S}$ est donnée par :

$$\begin{aligned} c(S) - c(\tilde{S}) &= q_{b_1}^2 + q_{b_2}(q_{b_1} + q_{b_2}) - \left((q_{b_1} - 1)^2 + (q_{b_2} + 1)(q_{b_1} + q_{b_2})\right) \\ &= q_{b_1} - q_{b_2} - 1 \geq 1 \end{aligned}$$

Le coût de la solution $\tilde{S}$ est strictement inférieur à celui de S. S n'est donc pas optimale. En itérant la procédure d'équilibrage entre tous les batches ordonnancés entre t_i et t_{i+1}, on converge vers une solution optimale où les batches sont partitionnés en deux sous-ensembles tels que la différence entre la taille d'un batch du premier sous-ensemble et celle d'un batch du second est exactement égale à 1 (il est possible que l'un des sous-ensembles est vide et tous les batches entre t_i et t_{i+1} sont de même taille).

Il reste à exprimer les nombres n_i et n'_i, ainsi que la quantité q. Nous savons que N_i batches sont créés avec $N_i = n_i + n'_i$. En outre, la quantité totale Q produite entre t_i et t_{i+1} est égale à $n_i q + n'_i(q+1)$. Les contraintes d'intégrité sur les valeurs de n_i, n'_i et q imposent nécessairement que :

$$q = \left\lfloor \frac{Q}{N_i} \right\rfloor \text{ et donc, } n'_i = Q - N_i \left\lfloor \frac{Q}{N_i} \right\rfloor \text{ et } n_i = N_i \left(1 + \left\lfloor \frac{Q}{N_i} \right\rfloor \right) - Q$$

Les valeurs de q, n_i et n'_i peuvent donc être déterminées en temps polynomial, en fonction des valeurs de Q et N_i. Notons que lorsque Q est un multiple de N_i, les N_i batches sont identiques ($n'_i = 0$ et $n_i = N_i$). □

Par construction, le Lemme 1 conduit à une solution de meilleur coût au sens large que la solution initiale à partir de laquelle nous avons extrait les t_i et les N_i. Le certificat que la solution ainsi construite est de coût inférieur ou égal à K s'obtient en temps polynomial, ce qui démontre l'appartenance à NP du problème P_{dp}. ⊛

4.1.2 Schéma de l'algorithme de programmation dynamique

4.1.2.1 Principe, notations et formule générale de récurrence

D'après les Propriétés 2 et 4 étendue, il existe une solution optimale dans laquelle les demandes sont livrées sans temps mort et dans l'ordre croissant des dates dues. Notre algorithme donne en sortie une telle solution et nous supposons dans la suite un indiçage de 1 à n des $|\mathcal{D}em|$ demandes respectant cet ordre. Comme il n'y a pas de durée fixe associée à la création d'un batch, bien que le nombre de batches ne soit pas connu *a priori*, la date de fin de l'ordonnancement, notée H, est, elle, connue et égale au produit de la durée variable d'exécution d'un batch par la somme Q des quantités requises, c'est-à-dire $H = vp.Q = vp \sum_d rq_d$. Dans un diagramme de Gantt, les demandes sont donc "placées" sur l'horizon, à partir de l'instant $t = 0$ dans l'ordre croissant des dates dues. La durée de production de la quantité rq_d requise par la demande dem_d est égale à $vp.rq_d$. La date de fin minimale de livraison de demande C_d, de la demande dem_d, pour d allant de 1 à n, est donc donnée par l'expression suivante (cf. Figure 4.2) :

$$\forall d = 1 \ldots n, C_d = vp \sum_{k=1}^{d} rq_d$$

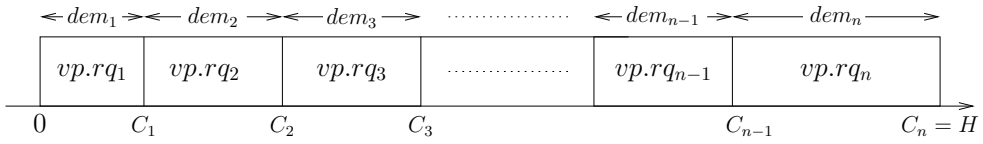

FIG. 4.2 – Placement sur l'horizon de la production affectée à chaque demande

L'enjeu est donc de décider comment livrer ces demandes, c'est-à-dire, comment découper la durée totale H en un ensemble de m batches de façon à minimiser l'objectif (4.1), tout en respectant les contraintes sur la taille des lots. La définition du problème impose que la taille q_b de chaque batch bat_b, pour b allant de 1 à m, soit supérieure ou égale à bs et inférieure ou égale à BS. Cependant, d'après la Propriété 5, il existe une borne supérieure implicite égale à $2bs-1$. Par conséquent, la taille des batches est inférieure à $\widetilde{BS} = \min(BS, 2bs-1)$. Dans la suite on emploie les notations M^{min} et M^{max} pour désigner respectivement les nombres $\left\lceil Q/\widetilde{BS} \right\rceil$ et $\lfloor Q/bs \rfloor$, bornes inférieure et supérieure du nombre total m de batches. De la même façon, on note D_{min} et D_{max} les durées minimale et maximale d'un batch, c'est-à-dire, $D_{min} = vp.bs$ et $D_{max} = vp.\widetilde{BS}$.

Ainsi, si l'on note $P_j(t)$ le coût du sous-problème prenant fin à la date $t \leq H$ dans lequel on crée au maximum j batches, $j \leq M^{max}$, la solution optimale du problème $\mathcal{P}_{dp}$ est obtenue en déterminant une solution qui minimise $P_{M^{max}}(H)$, coût d'une solution à au plus M^{max} batches, se terminant à la date H. Soit $c(t',t)$ la contribution d'un batch commençant à la date t' et se terminant à la date t, la formule de récurrence s'écrit alors :

$$\forall t = D_{min} \dots H, \forall j = \left\lceil \frac{t}{D_{max}} \right\rceil \dots \left\lfloor \frac{t}{D_{min}} \right\rfloor \quad , \quad P_j(t) = \min \left\{ \left(\min_{\substack{t' \geq (t-D_{max})^+ \\ t' \leq t-D_{min}}} \left(P_{j-1}(t') + c(t',t) \right) \right), P_{j-1}(t) \right\} \tag{4.2}$$

$$\forall t = D_{min} \dots D_{max}, \forall j = 1 \dots M^{min} \quad , \quad P_j(t) = c(0,t) \tag{4.3}$$

$$\forall t = 0 \dots D_{min} - 1, \forall j \quad , \quad P_j(t) = \infty \tag{4.4}$$

$$P_0(0) = 0 \tag{4.5}$$

$$P_{-1}(t) = \infty \tag{4.6}$$

L'équation (4.2) exprime le fait que le meilleur coût d'une solution se finissant à l'instant t avec au plus j batches est le minimum entre la somme du meilleur coût d'une solution se finissant à l'instant t' avec au plus $j-1$ batches et du coût d'un batch s'exécutant entre les dates t' et t, et le meilleur coût d'une solution se finissant à t avec $j-1$ batches. Les conditions limites sont décrites de la façon suivante : lorsque t est inférieur ou égal à D_{max}, il n'est plus possible de découper la production résiduelle sans violer les contraintes sur la taille des batches, la création

d'un unique batch entre les instants 0 et t est le seul choix réalisable (4.3) ; ainsi, lorsque t est strictement inférieur à D_{min} et qu'il faut créer au moins un batch, il n'y a pas de solution réalisable (le coût est infini) (4.4) ; le coût de la solution partielle réalisable se terminant à $t = 0$ à 0 batch a un coût nul (4.5) ; enfin, avant l'instant 0, il n'y a pas de solution réalisable, le coût est infini (4.6).

4.1.2.2 Formule générale pour le calcul des $c(t', t)$

Le coût d'exécution d'un batch bat_b entre les instants t' et t peut être calculé *a priori* puisque la séquence et le placement de la production relative à chaque demande sont connus. Le couple (t', t) est relié à la taille q_b du batch bat_b par la relation $t' - t = vp.q_b$. En notant $d(t^+)$ l'indice de la demande associée à la production ayant lieu entre t et $t + 1$, et $d(t^-)$ l'indice de la demande associée à la production ayant lieu entre $t - 1$ et t (cf. Figure 4.3), le coût $c(t', t)$ s'exprime comme suit :

$$\text{si } d(t'^+) = d(t^-), \quad c(t', t) = \quad fc + vc.q_b + q_b(t - dt_{d(t^-)})^+ \tag{4.7}$$

$$\text{sinon}, \quad c(t', t) = \quad fc + vc.q_b + \frac{C_{d(t'^+)} - t'}{vp}(t - dt_{d(t'^+)})^+ + \sum_{k=d(t'^+)+1}^{d(t^-)-1} rq_k(t - dt_k)^+ + \frac{t - C_{d(t^-)-1}}{vp}(t - dt_{d(t^-)})^+ \tag{4.8}$$

L'équation (4.7) donne la contribution du batch bat_b placé entre les instants t' et t lorsqu'il ne

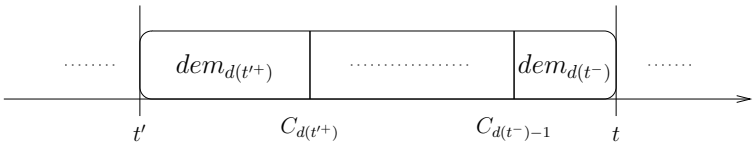

FIG. 4.3 – Contribution d'un batch entre les instants t' et t

livre qu'une seule demande $dem_{d(t^-)}$. On somme alors le coût d'exécution du batch $(fc + vc.q_b)$ avec le coût quadratique de retard (s'il est positif), ce dernier est donné par le produit de la quantité livrée à $dem_{d(t^-)}$, qui est ici l'intégralité de la production du batch (q_b), par la différence entre la date de fin du batch (t) et la date due de $dem_{d(t^-)}$ $(dt_{d(t^-)})$. L'équation (4.8) donne la contribution du batch bat_b placé entre les instants t' et t lorsqu'il livre plusieurs demandes. À nouveau, on somme le coût d'exécution avec la somme des coûts quadratiques des retards absolus. Pour calculer cette dernière, on distingue trois termes, le coût associé à la demande $dem_{d(t'^+)}$ en cours entre t' et $t' + 1$ (la quantité qui lui est livrée est égale à $\frac{C_{d(t'^+)} - t'}{vp}$), le coût associé aux demandes servies intégralement et le coût associé à la demande $dem_{d(t^-)}$ en cours entre $t - 1$ et t (la quantité qui lui est livrée est égale à $\frac{t - C_{d(t^-)-1}}{vp}$).

Comme le temps est discrétisé par pas d'une unité, la valeur d'un instant t est entière. Par ailleurs, les couples (t', t) réalisables doivent être tels que $0 \leq t' \leq H - D_{min}$, $D_{min} \leq t \leq H$. De plus, la contrainte sur la taille des batches impose $D_{min} \leq t - t' \leq D_{max}$, donc pour chacune des $H - D_{min}$ valeurs possibles pour t', il existe au maximum $D_{max} - D_{min}$ valeurs possibles pour t.

Formule de récurrence pour les $c(t', t)$. Le calcul de la fonction $c(t', t)$, pour les valeurs de t' et t définies ci-dessus peut être réalisé récursivement de la façon suivante :

$$\forall (t', t) \text{ réalisable}, \qquad c(t'+1, t) = c(t', t) - \frac{vc}{vp} - \frac{1}{vp}(t - dt_{d(t'^+)})^+ \tag{4.9}$$

L'équation (4.9) exprime la contribution d'un batch placé entre les instants $t'+1$ et t en fonction de celle d'un batch placé entre t' et t. Il suffit de *retrancher* à cette dernière la contribution de la portion de production du batch entre t' et $t'+1$, c'est-à-dire le coût fixe d'une unité de production ($\frac{vc}{vp}$) et le coût quadratique de retard, soit, le produit de la quantité livrée ($\frac{1}{vp}$) par le délai entre $dt_{d(t'^+)}$ (puisque $dem_{d(t'^+)}$ était la demande livrée entre t' et $t'+1$) et t (date de fin du batch). Comme le batch se termine au même instant dans les deux cas, la contribution du reste du batch relativement aux demandes suivantes est inchangée. La Figure 4.4 illustre cette équation. Nous

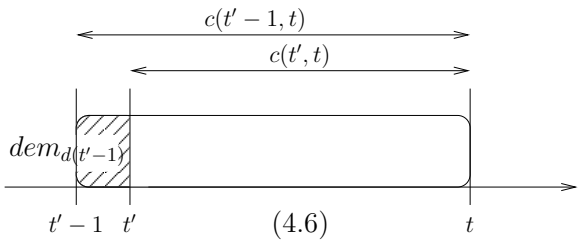

FIG. 4.4 – Récursivité du calcul des $c(t', t)$

présentons ci-dessous l'algorithme permettant de calculer la valeur de $c(t', t)$ pour chaque couple (t', t) valides, au sens du domaine de définition.

Pour $t = D_{max}$ **à** H
  calcul de $c(t - D_{max}, t)$ par l'équation (4.7) ou (4.8)
  Pour $t' = t - D_{max} + 1$ **à** $t - D_{min}$
    calcul de $c(t', t)$ par l'équation (4.9)

4.1.3 Complexité de l'algorithme

Calcul des $c(t', t)$. Les $c(t', t)$ sont calculés pour $H - D_{max} + 1$ valeurs de t. Pour chaque t, chacune des $D_{max} - D_{min} + 1$ valeurs de $c(t', t)$ est calculée en temps constant. À la constante vp

multiplicative près, on obtient, en fonction des données du problème, l'expression suivante pour la complexité du calcul des $c(t', t)$: $O\left((Q - BS)(BS - bs)\right) = O(Qbs)$.

Calcul des $P_j(t)$. Dans une solution optimale, il y a au maximum $\lfloor Q/bs \rfloor$ batches. Pour tout j, la date de fin t d'une solution partielle à j batches est un $O(H)$. Chaque $P_j(t)$ est en outre calculé en $O(D_{max} - D_{min})$ qui est en $O(bs)$. En outre comme $O(H)$ est en $O(Q)$, la complexité sptiale de l'algorithme est en $O(Q^2/bs)$, et sa complexité temporelle est en $O(Q^2)$.

4.2 Modélisation linéaire

Dans cette section, nous présentons une toute autre approche pour la résolution du *batching*. Comme nous le verrons dans les chapitres suivants, les extensions du problème *"cœur"* mènent à des variantes complexes, pour lesquelles nous souhaitions proposer une méthode de résolution générique. En effet, dans le but de s'intégrer à l'APS ILOG *Plant PowerOps* (PPO), il était indispensable d'envisager ce type d'approche. Nous nous sommes alors dirigés vers une formulation linéaire du problème dans laquelle la fonction objectif quadratique est remplacée par une approximation linéaire, choix que nous justifions dans la suite.

Nous introduisons donc le programme linéaire en variables mixtes formulant l'approximation linéaire du problème *"cœur"* de *batching* qui nous a occupé jusqu'à présent. Dans un premier temps nous précisons la variante exacte du problème que nous modélisons ici et introduisons les notations complémentaires, spécifiques au programme. Dans un second temps, nous procédons à la description de la fonction objectif et des contraintes qui constituent le programme.

4.2.1 Problème abordé et complément de notations

4.2.1.1 Position du problème traité

Dans les études précédentes (complexité au Chapitre 3 et programmation dynamique ci-dessus), nous avons restreint le problème *"cœur"* de *batching* au cas où une unique recette est définie. En outre, la notion de période de temps y a été omise, faisant l'hypothèse implicite qu'une seule période était considérée. Nous revenons maintenant au problème *"cœur"* tel qu'il a été défini au Chapitre 2 et qui est une généralisation des variantes étudiées jusqu'ici. Une seule ressource est toujours définie mais ici, plusieurs recettes permettant de fabriquer différents produits finis peuvent être modélisées et les instances des activités de production relatives aux batches de ces différentes recettes requièrent donc toute la ressource pour être exécutées. Par ailleurs, nous nous plaçons dans le cas général où plusieurs périodes de temps ont été définies et le problème de *batching* consiste à résoudre le *lot-streaming* et le *pegging* sur tout l'horizon de planification.

Nous rappelons que le problème *"cœur"* optimise une fonction objectif composée de trois termes : les coûts de production, les pénalités quadratiques de retard et les pénalités quadratiques d'avance. Ces dernières ont été laissées de côté lors des études précédentes et nous les

réintroduisons ici afin de nous placer dans le cas général.

4.2.1.2 Notations employées dans le programme linéaire

Nous manipulons dans la suite les notations présentées au Chapitre 2 pour le problème *"cœur"*. Rappelons que celles-ci ont été simplifiées afin de permettre d'alléger les formules. De surcroît, elles ont été encore épurées dans les études de complexité et pour l'algorithme de programmation dynamique. Il est ici nécessaire de revenir à des notations permettant de modéliser un nombre quelconque de recettes ainsi que les périodes de temps. Par ailleurs, la formulation du programme linéaire nécessite l'introduction de notations complémentaires, notamment celle de bornes (constantes) et de variables booléennes ou semi-continues intermédiaires.

Constantes. Le problème de *lot-streaming* s'applique à la production de chaque recette planifiée par période. Ainsi, nous notons $M^t_{i,min}$, respectivement $M^t_{i,max}$, le nombre minimum, respectivement maximum, de batches pouvant être créés pour la recette rec_i dans la période T_t, compte tenu des valeurs de bs_i, BS_i et Q^t_i. On a donc :

$$\forall t = 1 \dots |\mathcal{T}|, \forall i = 1 \dots |\mathcal{R}ec|, \qquad M^t_{i,min} = \left\lceil \frac{Q^t_i}{BS_i} \right\rceil \text{ et } M^t_{i,max} = \left\lfloor \frac{Q^t_i}{bs_i} \right\rfloor$$

Variables de production. Lors de la résolution du problème de *lot-streaming*, il s'agit de créer, dans chaque période T_t et pour chaque recette rec_i (telle que $Q^t_i > 0$), l'ensemble de batches $\mathcal{B}at^t_i$. La taille $q^t_{i,b}$ de chaque batch $bat^t_{i,b}$ a déjà été introduite, en outre, nous associons à la création d'un batch une variable booléenne $x^t_{i,b}$ prenant la valeur 1 si le batch $bat^t_{i,b}$ est effectivement créé, et 0 sinon. Ainsi les variables $q^t_{i,b}$, pour b allant de 1 à $M^t_{i,max}$, deviennent semi-continues, c'est-à-dire qu'elles peuvent prendre la valeur 0, en cas de non-création du batch $bat^t_{i,b}$, ou bien une valeur bornée par bs_i et BS_i dans le cas contraire.

Variables de flot. Lors de la résolution du problème de *pegging*, il s'agit de déterminer, pour chaque batch $bat^t_{i,b}$ et chaque demande dem_d la quantité de flot $f^t_{i,b,d}$. On associe à ce flot une variable booléenne $y^t_{i,b,d}$ prenant la valeur 1 s'il existe effectivement un flot non-nul entre le batch $bat^t_{i,b}$ et la demande dem_d. La variable $f^t_{i,b,d}$ n'en devient pour autant pas semi-continue au sens que l'on a défini ci-dessus, en effet, aucune contrainte n'empêche de transférer une quantité infiniment petite d'un batch à une demande. La nécessité de l'introduction des variables $y^t_{i,b,d}$ prendra tout son sens lorsque nous présenterons la fonction objectif.

Variables temporelles. Par variables temporelles nous entendons les variables relatives aux dates estimées d'exécution des activités relatives aux batches. Les notations $eet^t_{i,b}$ et $est^t_{i,b}$ ont déjà été mentionnées précédemment. Afin de pouvoir évaluer un délai entre la date $eet^t_{i,b}$ et la date due de la demande dem_d à laquelle pourrait être lié le batch $bat^t_{i,b}$ nous introduisons une

variable semi-continue $z^t_{i,b,d}$ prenant la même valeur que la variable $eet^t_{i,b}$ s'il existe un flot non-nul entre le batch $bat^t_{i,b}$ et la demande dem_d (c'est-à-dire si $y^t_{i,b,d} = 1$), et 0 sinon. À nouveau, l'introduction de ces variables $z^t_{i,b,d}$ sera justifiée lors de la présentation de la fonction objectif.

Dans la suite nous présentons la formulation mathématique du problème *"cœur"* de *batching*. Nous procédons dans un premier temps à la définition de la fonction objectif linéaire, à partir de la fonction quadratique (2.9) définie en Section 2.2.4 du Chapitre 2. Nous introduisons ensuite les contraintes du programme, celles-ci sont de deux types : les contraintes nécessaires assurent que les solutions réalisables du programme sont bien des solutions du problème de *batching*, les contraintes additionnelles viennent, de manière optionnelle, enrichir la formulation de base afin de la rendre plus réaliste ou plus performante.

4.2.2 Approximation linéaire de la fonction objectif

Nous rappelons ci-dessous la fonction objectif que nous avons définie pour le problème de *batching* traité dans cette thèse (2.9) :

$$\begin{aligned} OBJ_{batching} &= \sum_t \sum_i \sum_b (fc_i + vc_i q^t_{i,b}) \\ &+ \sum_t \sum_i \sum_b \sum_d ec_d f^t_{i,b,d} (dt_d - eet^t_{i,b})^+ \\ &+ \sum_t \sum_i \sum_b \sum_d tc_d f^t_{i,b,d} (eet^t_{i,b} - dt_d)^+ \end{aligned}$$

Les coûts de production sont déjà linéaires dans l'objectif initial, ce terme n'est donc pas modifié ici, on multiplie seulement le coût fixe de production fc_i d'un batch $bat^t_{i,b}$ par la variable booléenne de création $x^t_{i,b}$. Pour ce qui concerne les coûts quadratiques d'avance ou de retard, le procédé est le suivant. Nous conservons variable la partie temporelle, c'est-à-dire le délai entre la date de fin estimée $eet^t_{i,b}$ du batch $bat^t_{i,b}$ et la date due dt_d de dem_d, tandis que nous remplaçons le flot variable $f^t_{i,b,d}$ par une borne supérieure, plus exactement par la valeur $\min(BS_i, rq_d)$. Le choix d'orienter notre modèle vers une formulation temporelle est motivé par deux remarques. D'une part, la borne sur la valeur du flot est assez bonne en pratique et la minimisation du délai tend à saturer les quantités transférées lorsqu'un arc est ouvert (c'est-à-dire lorsqu'un flot non-nul circule entre un batch et une demande), la borne supérieure est donc souvent atteinte. D'autre part, comme la qualité de la solution du *batching* est relative à celle de l'ordonnancement final, ce dernier prenant des décisions temporelles, l'optimisation des délais nous paraît prépondérante.

On obtient alors l'expression suivante (4.10) :

$$\begin{aligned} OBJ'_{batching} &= \sum_t \sum_i \sum_b (fc_i x^t_{i,b} + vc_i q^t_{i,b}) \\ &+ \sum_t \sum_i \sum_b \sum_d ec_d \min(BS_i, rq_d)(dt_d - eet^t_{i,b})^+ \\ &+ \sum_t \sum_i \sum_b \sum_d tc_d \min(BS_i, rq_d)(eet^t_{i,b} - dt_d)^+ \end{aligned} \tag{4.10}$$

Cependant, la valeur du flot ne pouvant plus prendre la valeur 0 dans la fonction objectif, étant donné que rien n'empêche l'expression du délai $(dt_d - eet^t_{i,b})$ d'être non-nulle même si aucun flot ne circule entre le batch $bat^t_{i,b}$ et la demande dem_d, le terme de l'avance (ou du retard) n'est donc plus correctement calculé. C'est pourquoi les variables booléennes $y^t_{i,b,d}$ et semi-continues $z^t_{i,b,d}$ sont introduites afin de garantir qu'on ne considère dans l'objectif que les batches et demandes pour lesquels un flot circule effectivement. On obtient alors l'expression finale pour la fonction objectif de notre programme linéaire (4.11) :

$$\begin{aligned} OBJ^{lin}_{batching} &= \sum_t \sum_i \sum_b (fc_i x^t_{i,b} + vc_i q^t_{i,b}) \\ &+ \sum_t \sum_i \sum_b \sum_d ec_d \min(BS_i, rq_d)(dt_d y^t_{i,b,d} - z^t_{i,b,d})^+ \\ &+ \sum_t \sum_i \sum_b \sum_d tc_d \min(BS_i, rq_d)(z^t_{i,b,d} - dt_d y^t_{i,b,d})^+ \end{aligned} \tag{4.11}$$

Remarques. Notons que les expressions désignant les délais absolus de l'avance $(dt_d y^t_{i,b,d} - z^t_{i,b,d})^+$ ou du retard $(z^t_{i,b,d} - dt_d y^t_{i,b,d})^+$ ne sont pas linéaires. La linéarisation de ces expressions se fait en introduisant deux nouvelles variables, $earl^t_{i,b,d}$ et $tard^t_{i,b,d}$, exprimant l'avance et le retard. Nous posons alors simultanément $earl^t_{i,b,d} = dt_d y^t_{i,b,d} - z^t_{i,b,d}$ et $earl^t_{i,b,d} \geq 0$ d'une part, $tard^t_{i,b,d} = z^t_{i,b,d} - dt_d y^t_{i,b,d}$ et $tard^t_{i,b,d} \geq 0$ d'autre part.

Par ailleurs, une autre méthode aurait consisté à ne pas introduire les variables $z^t_{i,b,d}$ et à écrire, en plus des contraintes de positivité sur les variables $earl^t_{i,b,d}$ et $tard^t_{i,b,d}$, des contraintes de la forme $earl^t_{i,b,d} \geq dt_d - eet^t_{i,b} + M_{earl}(y^t_{i,b,d} - 1)$ et $tard^t_{i,b,d} \geq eet^t_{i,b} - dt_d + M_{tard}(y^t_{i,b,d} - 1)$. Cette formulation est proposée parmi les perspectives de nos travaux.

4.2.3 Contraintes du programme linéaire

Nous divisons en deux parties la présentation des contraintes du programme linéaire. Dans un premier temps nous exprimons les contraintes nécessaires à la formulation du problème de *batching* tel qu'il a été présenté au Chapitre 2 (Section 2.2.4). Dans un second temps, nous introduisons des contraintes additionnelles permettant de considérer des caractéristiques rendant le modèle plus réaliste, et en particulier de prendre en compte la capacité de la ressource. La dernière section de ce chapitre est consacrée à une étude expérimentale des deux méthodes

présentées (programmation dynamique et programmation linéaire). À cette occasion, nous mesurons l'impact sur la qualité des solutions de l'ajout, au cours de la résolution du *batching*, de ces contraintes additionnelles.

4.2.3.1 Contraintes nécessaires

Contraintes de *batching*

$$\forall t, \forall i, \forall b = 1 \dots M^t_{i,max} \qquad x^t_{i,b} bs_i \leq q^t_{i,b} \leq x^t_{i,b} BS_i \tag{4.12}$$

$$\forall t, \forall i \qquad \sum_{b=1}^{M^t_{i,max}} q^t_{i,b} = Q^t_i \tag{4.13}$$

Les doubles-inégalités (4.12) correspondent à celles présentées en Section 2.2.2 lors de la description du problème général (2.2), il s'agit d'exprimer les restrictions sur la taille des batches. Les équations (4.13) imposent que la totalité de la production planifiée dans chaque période et pour chaque recette soit effectivement réalisée par les batches créés.

Contraintes de *pegging*

$$\forall t, \forall i, \forall b = 1 \dots M^t_{i,max} \qquad q^t_{i,b} = \sum_d f^t_{i,b,d} \tag{4.14}$$

$$\forall d, \qquad rq_d = \sum_t \sum_i \sum_b f^t_{i,b,d} \tag{4.15}$$

Les deux ensembles d'équations ci-dessus sont les contraintes de conservation de flot. Le flot sortant d'un batch, exprimé par (4.14), correspond aux équations (2.4) et le flot entrant dans une demande, exprimé par (4.15), correspond aux équations (2.5).

$$\forall t, \forall i, \forall b = 1 \dots M^t_{i,max}, \forall d \qquad f^t_{i,b,d} \leq \min(BS_i, rq_d) y^t_{i,b,d} \tag{4.16}$$

Les inégalités (4.16) relient les variables booléennes $y^t_{i,b,d}$ aux variables de flot $f^t_{i,b,d}$, ce qui rend bien ces dernières semi-continues.

Contraintes temporelles. L'expression de la durée d'un batch a été présentée au Chapitre 2, Section 2.2.2 lors de la définition du problème *"cœur"* (2.1). Nous la rappelons ici, dans le contexte où le temps est partitionné en périodes, ce qui modifie la notation. En outre, la variable booléenne $x^t_{i,b}$ relative à la création du batch $bat^t_{i,b}$ est introduite afin de ne considérer que les batches effectivement créés (4.17). Les variables associées aux dates estimées de début et de fin d'un batch sont alors reliées par l'équation (4.18). Enfin, l'exécution d'un batch $bat^t_{i,b}$ doit avoir lieu dans la période T_t, cela s'exprime par une contrainte de borne sur chacune des variables

$est^t_{i,b}$ et $eet^t_{i,b}$ (4.19,4.20) :

$$\forall t, \forall i, \forall b = 1 \dots M^t_{i,max}, \qquad p^t_{i,b} = x^t_{i,b} fp_i + q^t_{i,b} vp_i \tag{4.17}$$

$$eet^t_{i,b} = est^t_{i,b} + p^t_{i,b} \tag{4.18}$$

$$est^t_{i,b} \geq S_t \tag{4.19}$$

$$eet^t_{i,b} \leq E_t \tag{4.20}$$

Afin d'imposer aux variables semi-continues $z^t_{i,b,d}$ la valeur $eet^t_{i,b}$ quand il existe effectivement un flot non-nul entre le batch $bat^t_{i,b}$ et la demande dem_d, l'ensemble de contraintes ci-dessous est nécessaire :

$$\left.\begin{array}{ll} \forall t, \forall i, \forall b = 1 \dots M^t_{i,max}, \forall d, & z^t_{i,b,d} \geq S_t y^t_{i,b,d} \\ & z^t_{i,b,d} \leq E_t y^t_{i,b,d} \\ & z^t_{i,b,d} \geq eet^t_{i,b} + E_t(y^t_{i,b,d} - 1) \\ & z^t_{i,b,d} \leq eet^t_{i,b} + S_t(y^t_{i,b,d} - 1) \end{array}\right\}$$

La conjonction des quatre inégalités ci-dessus permet d'exprimer de la disjonction suivante : si $y^t_{i,b,d} = 1$ alors $z^t_{i,b,d} = eet^t_{i,b}$, et si $y^t_{i,b,d} = 0$ alors $z^t_{i,b,d} = 0$. Cette disjonction relie les variables $z^t_{i,b,d}$ et $y^t_{i,b,d}$. En effet, les deux premières inégalités fixent les bornes inférieure et supérieure sur les variables $z^t_{i,b,d}$ tout en leur imposant la valeur 0 en cas d'absence de flot entre le batch $bat^t_{i,b}$ et la demande dem_d ($y^t_{i,b,d} = 0$). Les deux dernières inégalités forcent $z^t_{i,b,d}$ à prendre la valeur $eet^t_{i,b}$ si l'arc de *pegging* est créé ($y^t_{i,b,d} = 1$) tout en permettant à $z^t_{i,b,d}$ de prendre la valeur 0 dans le cas contraire.

Domaines de définition des variables.

$$\begin{array}{ll} \forall t, \forall i, \forall b = 1 \dots M^t_{i,max}, & x^t_{i,b} \in \{0,1\} \\ & q^t_{i,b} \in \mathbb{R} \\ & eet^t_{i,b} \in \mathbb{R} \\ & est^t_{i,b} \in \mathbb{R} \\ \forall t, \forall i, \forall b = 1 \dots M^t_{i,max}, \forall d & y^t_{i,b,d} \in \{0,1\} \\ & f^t_{i,b,d} \in \mathbb{R} \\ & z^t_{i,b,d} \in \mathbb{R} \end{array}$$

4.2.3.2 Contraintes additionnelles

Les contraintes formulées ci-dessus permettent de modéliser exactement le problème de *batching* défini au Chapitre 2. Cependant, une telle formulation omet de considérer la question de la disponibilité de la ressource. En effet, comme nous l'avons dit, dans le problème *"cœur"*, l'unique ressource est de capacité finie égale à 1 (elle ne peut exécuter qu'une activité à un instant donné) et toutes les activités induites par les batches devront se partager cette ressource. Or, ne pas considérer la disponibilité de la ressource conduit à une modélisation très optimiste, et

parfois très mauvaise, de la réalité. Nous proposons deux ensembles de contraintes additionnelles permettant de prendre en compte ces aspects. En outre, nous introduisons deux ensembles de contraintes permettant de rompre certaines symétries du problème. Le bénéfice tiré de l'ajout de ces contraintes facultatives sera évalué au Chapitre 6.

Contraintes de capacité. Nous définissons la notion d'*énergie* d'une ressource sur un intervalle de temps donné comme la quantité de travail que la ressource peut réaliser dans cet intervalle. Dans le problème *"cœur"*, la capacité de l'unique ressource *res* étant égale à 1, son énergie dans l'intervalle $[t_1, t_2]$ est exactement égale à la durée de cet intervalle, c'est-à-dire $t_2 - t_1$. En d'autres termes, il y a égalité entre les valeurs prises par le "temps disponible" et l'"énergie disponible" de la ressource ; cependant, leurs unités respectives ne sont pas les mêmes et dans le cas général, l'énergie est égale au produit de la capacité cap_r par la durée $t_2 - t_1$. Ainsi, en considérant la durée $p^t_{i,b}$ de chaque batch, nous pouvons ajouter une contrainte, dite d'*énergie périodique*, sur l'utilisation de la ressource au cours d'une période. En effet, la durée de l'ensemble des batches devant être exécutés ne doit pas dépasser l'énergie (ou ici, la durée) totale disponible (cf. Figure 4.5), on obtient les inégalités (4.21) :

$$\forall t, \qquad \sum_i \sum_b p^t_{i,b} \leq E_t - S_t \tag{4.21}$$

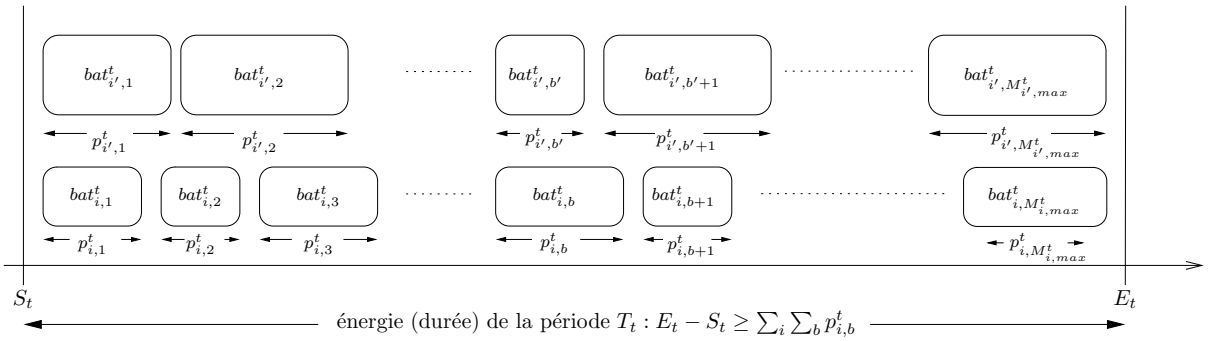

FIG. 4.5 – Contrainte périodique relative à l'énergie disponible

Dans l'étape de planification de la production précédant le *batching*, le même type de contrainte est déjà formulé, c'est en effet une contrainte classique en *lot-sizing*. Bien que l'énergie requise par la production y soit approximée — en effet, le nombre de batches n'est pas encore connu et la somme des durées fixes liées à l'exécution des batches est donc inexacte — les contraintes (4.21) définies dans le *batching* peuvent être redondantes et se révéler inefficaces en pratique. De ce fait, nous nous sommes dirigés vers l'expression de contraintes sémantiquement similaires en rediscrétisant chaque période de planification en un ensemble d'intervalles plus courts. Pour chaque période T_t, t allant de 1 à $|\mathcal{T}|$, on définit alors H_t sous-périodes $\theta_{t,h}$, pour h allant de 1 à H_t, chacune de même longueur. Puis, pour chaque sous-période $\theta_{t,h}$, pour chaque

recette rec_i, on affecte un ensemble $B_i^{t,h} = \{b_{i,1}^{t,h} \dots b_{i,|B_i^{t,h}|}^{t,h}\}$ d'indices désignant les batches $bat_{i,b}^t$ devant être exécutés dans la sous-période $\theta_{t,h}$. Les $B_i^{t,h}$, pour t allant de 1 à $|\mathcal{T}|$, i allant de 1 à $|\mathcal{R}ec|$, et h allant de 1 à H_t, forment donc une partition de l'ensemble $\{1 \dots M_{i,max}^t\}$ des indices b des batches de $\mathcal{B}at_i^t$. On formule alors les contraintes d'*énergie sous-périodique* suivantes :

$$\forall t, \forall h, \qquad \sum_i \sum_{b \backslash b \in B_i^{t,h}} p_{i,b}^t \leq \frac{E_t - S_t}{H_t} \tag{4.22}$$

Sans exprimer les contraintes disjonctives (2.6) du problème d'ordonnancement (c'est-à-dire l'exigence que toutes les activités définissent une séquence dans la solution), les contraintes d'énergie par sous-période tendent à rapprocher le modèle d'une formulation dans laquelle la granularité du temps est diminuée, ce qui mène à une plus grande précision. Ainsi, les contraintes (4.22) répartissent *a priori* l'exécution des batches le long de l'horizon, une solution ne définit donc pas exactement une séquence mais un partitionnement des batches relativement aux tranches de temps que définissent les sous-périodes (cf. Figure 4.6). Cela donne une estimation de ce que pourrait être la séquence finale de l'ordonnancement.

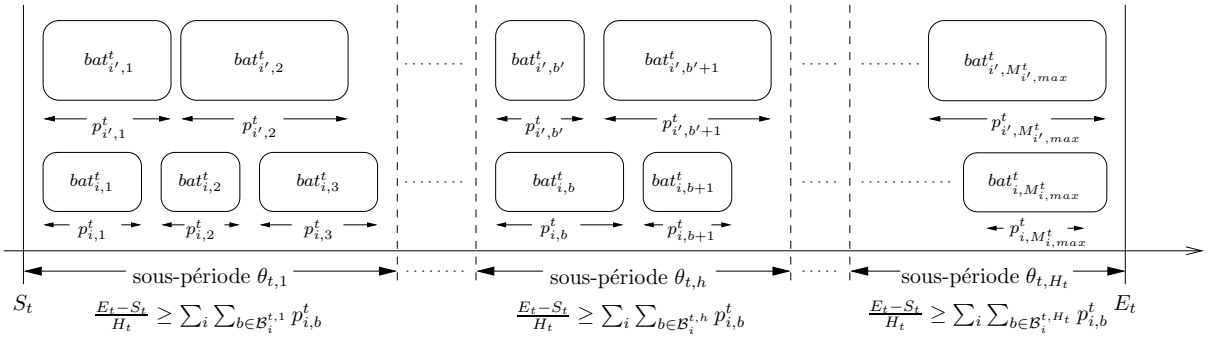

FIG. 4.6 – Contrainte sous-périodique relative à l'énergie disponible

On note que les contraintes (4.22) contiennent les contraintes (4.21), en effet, si $H_t = 1$ pour tout t, une seule sous-période est définie, ce qui est équivalent à considérer la période T_t elle-même. De cela résulte que nous n'étudions pas de programme mathématique comportant les deux ensembles de contraintes. Nous notons $\mathcal{P}_0$ le modèle dans lequel aucune des contraintes d'énergie n'est formulée, il s'agit du modèle de base ne contenant que les contraintes dites nécessaires. $\mathcal{P}_1$, respectivement $\mathcal{P}_2$, désigne $\mathcal{P}_0$ enrichi des contraintes (4.21), respectivement (4.22).

Ruptures de symétries. L'indiçage des batches pour une même recette rec_i dans une période T_t, ou une sous-période $\theta_{t,h}$, selon la modélisation considérée, mène à des symétries dans la formulation. On peut rompre aisément certaines d'entre elles en imposant un ordre sur la création des batches, c'est-à-dire sur les variables $x_{i,b}^t$, soit au sein de la période dans le modèle $\mathcal{P}_1$ (4.23,4.24)

— on peut alors fixer les $M^t_{i,min}$ premières variables booléennes à la valeur 1 — soit au sein de chaque sous-période (4.25) — dans ce cas, il est impossible de fixer les $M^t_{i,min}$ variables qui seront à 1. En pratique, ces contraintes sont toujours intégrées aux programmes $\mathcal{P}_0$, $\mathcal{P}_1$ et $\mathcal{P}_2$ (nous ne changeons pas les notations pour désigner ces modèles).

$$\text{cas } \mathcal{P}_0 \text{ ou } \mathcal{P}_1 : \quad \forall t, \forall i, \forall b = 1 \dots M^t_{i,min} \qquad x^t_{i,b} = 1 \tag{4.23}$$

$$\forall t, \forall i, \forall b = M^t_{i,min} \dots M^t_{i,max} - 1 \qquad x^t_{i,b+1} \leq x^t_{i,b} \tag{4.24}$$

$$\text{cas } \mathcal{P}_2 : \quad \forall t, \forall h, \forall i, \forall b = b^{t,h}_{i,1} \dots b^{t,h}_{i,|B^{t,h}_i|-1} \qquad x^t_{i,b+1} \leq x^t_{i,b} \tag{4.25}$$

En outre, on peut rompre la symétrie sur la séquence estimée selon laquelle sont exécutés les batches d'une même recette dans une période (ou sous-période), c'est-à-dire imposer un ordre sur les variables $est^t_{i,b}$. Dans le cas du problème *"cœur"*, la ressource étant de capacité 1, elle ne peut exécuter qu'une seule activité à la fois (ce ne sera plus vrai dans le cas général), donc, on peut ici contraindre à ce que la date de début d'un batch soit supérieure ou égale à la date de fin du batch précédent (dans le cas général, on ne pourra écrire l'inégalité que sur les dates de début). Rappelons que pour tout batch $bat^t_{i,b}$, nous avons $eet^t_{i,b} = est^t_{i,b} + p^t_{i,b}$ puisque la préemption n'est pas autorisé et que la ressource exécute une unité de travail par unité de temps. À nouveau, pour écrire les contraintes de rupture de symétrie sur les dates estimées d'exécution, on distingue les modèle $\mathcal{P}_0$ et $\mathcal{P}_1$ (4.26) du modèle $\mathcal{P}_2$ (4.27). L'ajout de ces contraintes est noté d'un + en exposant du problème enrichi ($\mathcal{P}^+_0$, $\mathcal{P}^+_1$ ou $\mathcal{P}^+_2$) :

$$\text{cas } \mathcal{P}^+_0 \text{ ou } \mathcal{P}^+_1 : \quad \forall t, \forall i, \forall b = 1 \dots M^t_{i,max} - 1 \qquad eet^t_{i,b} \leq est^t_{i,b+1} \tag{4.26}$$

$$\text{cas } \mathcal{P}^+_2 : \quad \forall t, \forall h, \forall i, \forall b = b^{t,h}_{i,1} \dots b^{t,h}_{i,|B^{t,h}_i|-1} \qquad eet^t_{i,b} \leq est^t_{i,b+1} \tag{4.27}$$

4.2.4 Synthèse sur les différents modèles présentés

Nous avons exprimé le problème de *batching* tel qu'il a été présenté en Section 2.2.2 du Chapitre 2 en formulant des contraintes linéaires ; l'approximation de la fonction objectif originale du problème par une fonction linéaire a permis d'exprimer le problème sous forme d'un programme linéaire en variables mixtes (MILP). Le modèle ne contenant que les contraintes nécessaires à la formulation du *batching* est noté $\mathcal{P}_0$. L'introduction de contraintes d'énergie périodique mène au modèle $\mathcal{P}_1$, lorsque ces contraintes agissent sur des sous-périodes, on note le modèle $\mathcal{P}_2$. En outre, des ruptures de symétries sur la création des batches sont intégrées à ces modèles. Comme nous souhaitons étudier l'impact de l'ajout de ruptures de symétries sur l'exécution des batches, nous distinguons les modèles qui les intègrent. Cela mène respectivement aux problèmes $\mathcal{P}^+_0$, $\mathcal{P}^+_1$ ou $\mathcal{P}^+_2$. Le Tableau 4.1 dresse un récapitulatif de chaque modèle en fonction des contraintes additionnelles qu'il prend en compte.

Modèles / Contraintes	$\mathcal{P}_0$	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_0^+$	$\mathcal{P}_1^+$	$\mathcal{P}_2^+$
Énergie périodique (4.21)		X			X	
Énergie sous-périodique (4.22)			X			X
Symétrie de séquence (4.26,4.27)				X	X	X

TAB. 4.1 – Contraintes optionnelles dans les modèles linéaires pour le *batching*

4.3 Étude expérimentale

Dans cette dernière section, nous réalisons une étude comparative des deux approches présentées dans ce chapitre. En effet, comme la programmation dynamique est une méthode exacte (mais difficile à étendre à des classes de problèmes plus générales), contrairement à l'approche trimodulaire (qui en revanche est plus facilement généralisable), il est intéressant de caractériser la qualité des ordonnancements que cette dernière permet d'obtenir, en comparaison des solutions optimales atteignables par la première méthode. Nous traitons donc le sous-cas du problème *"cœur"* qui est résolu par le programme dynamique et décrit en Section 4.1.1. La variante du programme linéaire en nombres entiers testée ici est $\mathcal{P}_2$, c'est en effet, la variante employée par défaut dans PPO, nous le justifions au Chapitre 6, au cours duquel nous présentons une autre étude expérimentale. L'algorithme $\mathcal{P}_{dp}$ étant une méthode de résolution exacte, alors que la résolution de $\mathcal{P}_2$, suivie de l'ordonnancement, est une méthode approchée, il est intéressant de comparer les solutions obtenues par cette dernière avec les solutions optimales.

Jeu de données. Les expérimentations ont été conduites sur un ensemble de 40 instances générées aléatoirement de la façon suivante. Pour toutes les instances, le nombre de demandes est fixé à 10 et les quantités rq_d sont réparties uniformément entre 5 et 15 de sorte que la quantité totale produite Q est de l'ordre de 100. La durée unitaire vp d'un batch est fixée à 1, ainsi, la durée d'un batch est donc égale à sa taille et l'horizon H est égal à Q. En outre, le coût fixe de création d'un batch est égal à 5 et le coût unitaire à 10.

Sur ces bases communes, nous formons ensuite deux classes, chacune composée de 20 instances en fonction de la répartition des dates dues. Dans la première (`1`), les dates dues sont réparties uniformément entre 0 et $H/2$ (au moins la moitié des demandes seront livrées en retard), dans la seconde (`2`), elles sont réparties entre 0 et H, ce qui est moins contraignant pour les pénalités de retard. Enfin, pour ces deux classes, nous distinguons encore deux sous-classes `a` et `b`. Dans les instances de type `a`, la recette considérée s'exécute par batches de grande taille, relativement aux quantités rq_d, nous choisissons ici $bs = 13$ et $BS = 19$. *A contrario*, dans les intances de type `b`, la taille des batches de la recette doit être comprise entre $bs = 8$ et $BS = 12$.

Critère d'évaluation. Comme nous souhaitons estimer l'erreur commise par l'approche décomposée (planification, *batching*, ordonnancement) par rapport à la méthode exacte de programmation dynamique, nous comparons les valeurs des solutions obtenues par la première approche à celles des solutions optimales, respectivement notées $S_{\mathcal{P}_2}$ et S^*. Nous présentons donc la moyenne des erreurs commises, sur l'intégralité des instances, puis en analysant plus précisément chaque catégorie. Pour chaque instance, l'erreur est donnée par la formule $(S_{\mathcal{P}_2} - S^*)/S^*$. Le Tableau 4.2 présente les résultats ainsi obtenus.

Temps de calcul. Le programme dynamique résout chaque instance en un temps compris entre quelques secondes et 5 minutes (sur un processeur Pentium 1.8 GHz, et avec 2 Go de mémoire). Pour la méthode tri-modulaire, la planification est résolue instantanément, 30 secondes sont alors attibuées à chacun des deux autres modules (*batching* et ordonnancement), ce qui implique que le *branch and cut* résolvant le programme $\mathcal{P}_2$ peut être tronqué (et donc non-résolu à l'optimalité). C'est en général le cas et l'écart entre la borne inférieure courante et la solution entière se situe entre 5 à 30 %.

Total (40 instances)	moyenne = 2.64 %	écart min. = 0.00 %	écart max. = 13.48 %
Classe `1` (20 instances)	moyenne = 2.12 %	type `a` (10 instances)	moyenne = 3.49 %
		type `b` (10 instances)	moyenne = 0.76 %
Classe `2` (20 instances)	moyenne = 3.16 %	type `a` (10 instances)	moyenne = 5.91 %
		type `b` (10 instances)	moyenne = 0.41 %
Instances de type `a` (20)	moyenne = 4.70 %		
Instances de type `b` (20)	moyenne = 0.58 %		

TAB. 4.2 – Écart relatif moyen aux solutions optimales

Analyse. Ces résultats montrent un bon comportement général de l'approche tri-modulaire de PPO, lorsque le *batching* est résolu par le programme $\mathcal{P}_2$. En effet, la moyenne des écarts aux solutions optimales est faible (2.64 %). Par ailleurs, une différence nulle n'est constatée que sur une seule instance sur les 40, la solution optimale n'est donc quasiment jamais atteinte. *A contrario*, l'écart maximal est de 13.48 % et seules 5 instances présentent un écart à la solution optimale de plus de 5 %.

Une analyse plus détaillée montre que les instances dans lesquelles les dates dues se répartissent sur tout l'horizon (classe `1`) sont légèrement moins bien résolues. Cela s'explique par le fait que les décisions de *batching* sont difficiles à prendre tout au long de l'horizon alors que dans les instances de classe `2`, la décision de terminer à un batch à une date donnée (c'est-à-dire de "couper" la production totale Q à cet instant) est surtout cruciale en début d'horizon ; sur sa seconde moitié, en effet, tous les batches sont nécessairement en retard. Cependant, la différence de performances entre ces deux classes n'est pas aussi claire que celle que l'on observe pour les catégories où l'on fait varier les intervalles pour la taille des batches (types `a` ou `b`). Sur

cet aspect, les résultats montrent en effet une difficulté de résolution significative des instances où les batches peuvent être de grande taille (type `a`) alors que les solutions obtenues sur les instances de type `b` sont de très bonne qualité en moyenne. Ceci est dû aux erreurs commises par l'algorithme de *batching* qui n'est pas suffisamment avisé des pénalités de retard encourues lors de la création des batches. Ainsi le nombre de batches créés est souvent inférieur à ce qu'il conviendrait de faire afin de réduire les coûts de retard.

Conclusion. Cette étude expérimentale révèle un bon comportement en moyenne de l'approche tri-modulaire qui, nous le constaterons par la suite, se veut très générique et de ce fait, nécessairement moins compétitive qu'un algorithme dédié au problème résolu ici. Cela est donc très satisfaisant vis-à-vis de l'objectif que nous avons de traiter différents types de problèmes, y compris des variantes plus élémentaires que les cas rencontrés dans les applications pratiques.

Chapitre 5

Extensions du problème *"cœur"*, classification des problèmes de *batching*

Jusqu'à présent, nous nous sommes concentrés sur l'étude du problème central de *batching*, que nous avons nommé problème *"cœur"*. Cependant, nous avons introduit au Chapitre 2 un ensemble de notations plus générales annonçant l'élargissement du problème à des situations plus complexes. Ces extensions sont l'objet du présent chapitre. Chacune d'elles fait référence à un concept classique de l'industrie de production ; ainsi, pour chaque nouvelle notion, nous donnons sa définition, sa modélisation dans l'APS ILOG *Plant PowerOps* (PPO), ainsi que sa formulation mathématique au sein du MILP que nous avons élaboré pour résoudre le problème de *batching* (Chapitre 4 en Section 4.2). Enfin, nous dressons un récapitulatif des différentes variantes du problème en en présentant une classification.

Dans la taxinomie qui est présentée au cours de ce chapitre, le problème *"cœur"* appartient à la classe `NCOS` (*No Calendar, One Step*). Cette classe englobe la variante du problème où l'on définit une seule ressource de disponibilité constante au cours du temps. Les recettes de cette classe ne se définissent que par une seule activité de production exécutable selon un seul mode. Elles produisent chacune un produit fini requis par un ensemble de demandes qu'il s'agit de satisfaire intégralement.

5.1 Arrêt complet et productivité variable des ressources

5.1.1 Description de l'extension

La disponibilité d'une ressource ainsi que la vitesse à laquelle elle exécute les activités peuvent varier au cours du temps. Par exemple, il arrive que pendant les week-ends ou une période de maintenance, le fonctionnement des ressources soit totalement interrompu ; il est possible en outre que, durant la nuit, les ressources fonctionnent à moindre régime. Pour cela, nous

définissons la notion de calendrier associé à une ressource. Un calendrier est défini par un certain nombre d'intervalles (périodiques ou non) et l'indisponibilité ou la productivité de la ressource (un pourcentage) est renseigné pour chacun d'entre eux. L'indisponibilité (*Break*) indique que la ressource ne peut exécuter aucune activité dans cet intervalle [110]. La productivité d'une ressource est à mettre en relation avec sa vitesse d'exécution et donc la durée effective de l'activité qu'elle exécute. La référence est de 1.0. Si sur un intervalle, la productivité devient 0.5 (par exemple pendant la nuit), alors les activités vont être exécutées en deux fois plus de temps. Au contraire, lorsque la productivité est égale à 2.0, deux unités de la durée de l'activité seront exécutées en une unité de temps.

Deux classes d'instances sont dérivées de ces extensions : `BROS` (*BReak, One Step*), dans laquelle surviennent des intervalles d'indisponibité, et `BPOS` (*Break and Productivity, One Step*), dans laquelle indisponibilités et variations de productivité peuvent intervenir.

5.1.2 Modélisation dans ILOG *Plant PowerOps*

Dans les modules de planification et de *batching*, les contraintes de calendrier sont approximées tandis que l'ordonnancement les considère de façon exacte (aucune activité n'est exécutée sur les périodes d'indisponibilité et la durée de chaque activité dépend de l'intervalle de temps dans lequel elle est exécutée). Dans les deux premiers modules, l'approximation est réalisée en modifiant l'énergie totale de la ressource (définition au Chapitre 4, Section 4.2.3.2) le long d'un intervalle de temps donné (périodes pour la planification et le modèle $\mathcal{P}_1$ de *batching*, sous-périodes pour le modèle $\mathcal{P}_2$ de *batching*, cf. Chapitre 4, Section 4.2.3.2). Une période d'indisponibilité est considérée comme un intervalle le long duquel la capacité de la ressource est nulle. Les variations de productivité sont considérées comme des changements temporaires de la capacité (augmentation ou diminution). Ainsi, il est possible de recalculer l'énergie disponible d'une ressource le long d'un intervalle donné, c'est-à-dire l'aire du rectangle de hauteur égale à la capacité modifiée, et de longueur égale à la durée de l'intervalle. On note Γ_r^t, respectivement, $\Gamma_r^{t,h}$, l'énergie de la ressource res_r pendant la période T_t, respectivement, pendant la $h^{\text{ème}}$ sous-période de la période T_t. Par exemple, considérons une ressource res_r avec le profil de productivité suivant (cf. Figure 5.1) : chaque jour, entre 0h et 8h, la ressource est en maintenance (sa productivité est égale à 0.0), entre 8h et 16h, elle fonctionne à plein régime (sa productivité est égale à 1.0) et de 16h à 0h, elle fonctionne à une vitesse réduite de moitié (sa productivité est égale à 0.5). Sur une période T_t d'une journée, son énergie Γ_r^t est alors égale à $8 + 8 \times 0.5 = 12$.

5.1.3 Intégration au programme linéaire en variables mixtes

Les contraintes d'énergie périodique (4.21), respectivement sous-périodique (4.22), formulées le long d'une période, respectivement d'une sous-période, définies au Chapitre 4, Section 4.2.3.2, sont remplacées par les contraintes (5.1), respectivement (5.2). En outre, on généralise ces

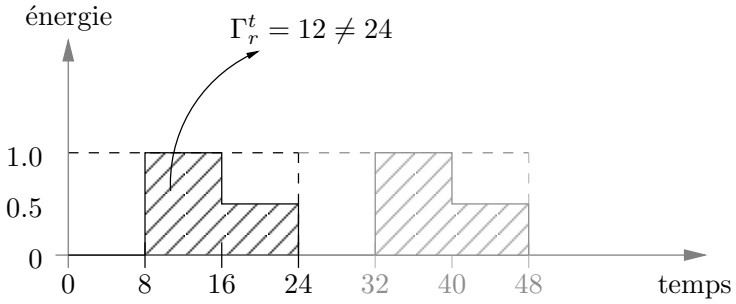

FIG. 5.1 – Énergie en cas de profil de productivité variable

contraintes aux cas à plusieurs ressources.

$$\forall r, \forall t, \qquad \sum_{i \backslash mres_i = res_r} \sum_{b} p_{i,b}^t \leq \Gamma_r^t \tag{5.1}$$

$$\forall r, \forall t, \forall h, \qquad \sum_{i \backslash mres_i = res_r} \sum_{b \backslash b \in B_i^{t,h}} p_{i,b}^t \leq \Gamma_r^{t,h} \tag{5.2}$$

5.2 Modes alternatifs et capacité quelconque pour les ressources

5.2.1 Description de l'extension

La notion de modes est associée à celle d'activité et permet de modéliser l'existence de ressources alternatives pour l'exécution d'une activité. Si plusieurs modes sont rattachés à une activité générique, alors plusieurs ressources sont capables d'exécuter une instance de celle-ci ; chaque instance pourra alors être exécutée par une de ces ressources. Les données relatives à la durée et au coût d'exécution d'une activité dans un mode donné, ainsi que la capacité qu'elle requiert, peuvent être différentes d'un mode à l'autre ou non ; dans ce dernier cas, les ressources sont strictement équivalentes. Lorsque le mode $mod_{i,k}$ est choisi pour exécuter une instance $act_{i,b}$ de l'activité de la recette rec_i, alors la ressource $mres_{i,k}$ exécute (entièrement) l'activité pendant une durée $p_{i,b} = fp_{i,k} + q_{i,b} vp_{i,k}$ et la capacité $mcap_{i,k}$ est requise par l'activité pendant cette durée. Les classes d'instances définissant des modes alternatifs sont notées `MM_XXXX` (*Multi-Mode*).

On généralise en outre dans cette section la capacité cap_r de chaque ressource res_r à des valeurs quelconques, ce qui implique, lorsque que $cap_r > 1$, que res_r peut exécuter plusieurs activités au même instant. Symétriquement, la capacité $mcap_{i,k}$ requise par une instance de l'activité act_i exécutée dans le mode $mod_{i,k}$ est aussi généralisée à des valeurs quelconques.

5.2.2 Modélisation dans ILOG PPO

Le module de planification affecte par anticipation un mode pour l'activité générique de chaque recette exécutée. Plus précisément, un mode n'est pas affecté pour chaque produc-

tion de recette globalement planifiée sur une période, en réalité, les recettes multi-mode sont décomposées en autant de recettes mono-mode qu'il existe de modes d'exécution possibles et la résolution se déroule alors comme s'il s'agissait de recettes indépendantes. Ainsi, la production planifiée d'une recette initiale notée Q_i^t est divisée en plusieurs $Q_{i,k}^t$. Le module de *batching* ne remet pas en cause ces décisions et les utilise comme informations supplémentaires pour poser le problème. Au contraire, l'ordonnancement les remet en question et affecte le mode final pour chaque activité.

La prise en compte de capacités quelconques pour les ressources et pour les activités est modélisée de la même façon dans les modules de planification et de *batching* : l'énergie disponible de la ressources est calculée comme la durée d'une période (ou sous-période le cas échéant) multipliée par la capacité de la ressource, de la même façon, l'énergie requise par une activité est alors donnée par le produit de sa durée par la capacité qu'elle requiert.

5.2.3 Intégration au programme linéaire en nombres entiers

Les modifications du programme sont mineures, il s'agit principalement d'ajouter l'indice k correspondant au mode $mod_{i,k}$ lorsque cela est nécessaire. La fonction objectif (4.11) et l'expression (4.17) de la durée de l'activité $act_{i,b}^t$ instanciée par le batch $bat_{i,b}^t$ se réécrivent alors respectivement en (5.3) et (5.4)

$$\begin{aligned} OBJ_{batching}^{lin} &= \sum_t \sum_i \sum_b (fc_{i,k} x_{i,b}^t + vc_{i,k} q_{i,b}^t) \\ &+ \sum_t \sum_i \sum_b \sum_d ec_d \min(BS_i, rq_d)(dt_d y_{i,b,d}^t - z_{i,b,d}^t)^+ \\ &+ \sum_t \sum_i \sum_b \sum_d tc_d \min(BS_i, rq_d)(z_{i,b,d}^t - dt_d y_{i,b,d}^t)^+ \end{aligned} \tag{5.3}$$

$$\forall t, \forall i, \forall b, \qquad p_{i,b}^t = x_{i,b}^t fp_{i,k} + q_{i,b}^t vp_{i,k} \tag{5.4}$$

Par ailleurs, on généralise au cas où la capacité cap_r de chaque ressource res_r est quelconque (et non-plus nécessairement égale à 1), on exprime alors les contraintes d'énergie périodique et sous-périodique pour chaque ressource par les inégalités suivantes (5.5,5.6) :

$$\forall r, \forall t \qquad \sum_{i \backslash mres_{i,k} = res_r} \sum_b mcap_{i,k} p_{i,b}^t \leq cap_r (E_t - S_t) \tag{5.5}$$

$$\forall r, \forall t, \forall h, \qquad \sum_{i \backslash mres_{i,k} = res_r} \sum_{b \backslash b \in B_i^{t,h}} mcap_{i,k} p_{i,b}^t \leq cap_r \frac{E_t - S_t}{H_t} \tag{5.6}$$

5.3 Recette multi-activités et précédences internes

5.3.1 Description de l'extension

N'ont été considérées jusqu'à présent que des recettes définissant une unique activité de production. Cette nouvelle extension permet de modéliser des recettes possédant plusieurs activités génériques ($|\mathcal{A}ct_i| \geq 1$). Ainsi, lorsqu'un batch $bat_{i,b}$ exécute une quantité $q_{i,b}$ de la recette rec_i, une instance $act_{i,j,b}$ pour chacune des activités de l'ensemble $\mathcal{A}ct_i$, pour j allant de 1 à $|\mathcal{A}ct_i|$, est alors créée. En pratique, ce type de recette est utile pour modéliser des procédés de transformation définissant plusieurs étapes étroitement liées. Considérons par exemple une certaine quantité de produit stockée dans une cuve, si l'exécution du procédé consiste à faire passer toute cette quantité dans un pasteurisateur, puis un fermenteur et enfin dans un stérilisateur, on le modélisera par une recette à trois activités. En effet, comme les ressources mises en jeu sont différentes et qu'elles peuvent être requises par d'autres activités (e.g. d'autres recettes), il faut distinguer ces trois activités (pasteurisation, fermentation, stérilisation). Cependant, elles sont toutes reliées par la quantité de produit transférée. Ainsi, les durées respectives de ces activités sont couplées puisqu'elles sont toutes calculées à partir de la taille $q_{i,b}$ du batch qui les instancie, ce qui définit une certaine proportionalité entre elles. Pour éviter toute ambiguïté, on impose qu'au sein d'une recette donnée, une seule activité est capable de produire un matériau donné. Par ailleurs, des contraintes de précédences peuvent être imposées entre les activités d'une recette. En pratique, La classe de problèmes utilisant de telles précédences est notée `NCGS` (*No Calendar, General Shop*).

5.3.2 Modélisation dans ILOG PPO

Après analyse du graphe de précédences représentant une recette, le module de planification agrège les données relatives à chacune des activités composant une recette afin de donner une approximation de l'énergie que celle-ci requiert, ainsi que de son coût. Au contraire, le module de *batching* considère toutes les activités ainsi que les précédences qui les lient. Cependant, bien que cela permette d'imposer un délai entre deux activités, et ainsi de mieux approcher la réalité, comme les contraintes de ressources ne sont pas formulées de manière exacte, le modèle en reste une approximation. Enfin, il est clair que l'ordonnancement doit satisfaire les contraintes de précédences entre activités.

5.3.3 Intégration au programme linéaire en nombres entiers

Tout d'abord, les variables relatives à la durée $p^t_{i,b}$ et aux dates de début $est^t_{i,b}$ et de fin d'un batch $eet^t_{i,b}$ sont démultipliées pour chaque activité $act^t_{i,j,b}$ instanciée par le batch $bat^t_{i,b}$ (on obtient alors pour toute période T_t, toute recette rec_i, tout batch $bat^t_{i,b}$, toute activité $act^t_{i,j,b}$, les

notations $p^t_{i,j,b}$, $est^t_{i,j,b}$ et $eet^t_{i,j,b}$). Les contraintes (4.17,4.18,4.19,4.20) deviennent alors :

$$\forall t, \forall i, \forall b, \forall j, \qquad p^t_{i,j,b} = x^t_{i,b} fp_{i,j} + q^t_{i,b} vp_{i,j} \tag{5.7}$$

$$eet^t_{i,j,b} = est^t_{i,j,b} + p^t_{i,j,b} \tag{5.8}$$

$$est^t_{i,j,b} \geq S_t \tag{5.9}$$

$$eet^t_{i,j,b} \leq E_t \tag{5.10}$$

Au sujet des variables de *pegging* (existence d'un flot $y^t_{i,b,d}$, quantité de flot $f^t_{i,b,d}$), comme, pour une recette donnée, une seule activité peut produire un matériau donné, il n'y a pas d'ambiguïté sur l'activité instanciée issue de la recette rec_i qui est réellement liée à la demande dem_d (une seule activité générique $act_{i,j}$ produit la matériau $rmat_d$). Ces notations ne sont donc pas modifiées, il en est par conséquent de même pour la fonction objectif.

Par ailleurs, on exprime les contraintes de précédences entre les activités d'une même recette par les inégalités suivantes (5.11) :

$$\forall t, \forall i, \forall b, \forall j, j' \text{ tels que } act_{i,j} \text{ précède } act_{i,j'}, \qquad eet^t_{i,j,b} \leq est^t_{i,j',b} \tag{5.11}$$

Si de plus un délai minimal $del^{i,j,j'}_{min}$ et/ou un délai maximal $del^{i,j,j'}_{max}$ doivent être respectés entre l'exécution de deux activités $act_{i,j}$ et $act_{i,j'}$ liées par une telle précédence, la contrainte (5.11) est remplacée par (5.12) et la contrainte (5.13) est ajoutée au programme linéaire.

$$\forall t, \forall i, \forall b, \forall j, j' \text{ tels que } act_{i,j} \text{ précède } act_{i,j'}, \qquad eet^t_{i,j,b} + del^{i,j,j'}_{min} \leq est^t_{i,j',b} \tag{5.12}$$

$$eet^t_{i,j,b} + del^{i,j,j'}_{max} \geq est^t_{i,j',b} \tag{5.13}$$

Notons qu'en général, la définition de multiples activités par recette induit naturellement la généralisation à un modèle à plusieurs ressources. En effet, les activités d'une même recette sont souvent de types différents et ne sont pas exécutées par les mêmes ressources. Cette généralisation est faite au fur et à mesure de l'introduction d'autres extensions et il n'est pas utile de rappeler ici les équations impliquées.

5.4 Précédences externes

5.4.1 Description de l'extension

Nous introduisons dans cette section la notion de niveaux de production mentionnée en Section 1.2.2.1 du Chapitre 1. Il s'agit maintenant de modéliser des recettes ne produisant pas qu'exclusivement des produits finis mais aussi des produits intermédiaires, pouvant être utilisés par d'autres recettes ; symétriquement, les recettes peuvent donc consommer des matières premières ou des produits intermédiaires. Ainsi, des précédences dites externes vont être exprimées entre recettes. Une telle précédence représente le transfert d'un matériau mat_m produit par une recette rec_i ($mat_m = pmat_i$) et consommé par une recette $rec_{i'}$ ($mat_m = cmat_{i'}$). Un nouveau type d'arcs de *pegging* apparaît puisqu'il va s'agir de décider des flots circulant entre deux batches $bat^t_{i,b}$ et

$bat^{t'}_{i',b'}$. Nous restreignons notre étude au cas où une recette est réduite à une unique activité, cependant la généralisation au cas multi-activités est immédiate, il suffit d'identifier l'activité de rec_i qui produit le matériau transféré et l'activité de $rec_{i'}$ qui le consomme (par hypothèse, ces activités sont uniques). Lorsqu'un arc de *pegging* est créé entre deux batches, une contrainte de précédence devra être respectée dans l'ordonnancement final entre les deux activités intervenant dans le transfert de produit. Les problèmes faisant intervenir ces précédences sont placés dans la même classe `NCGS` (*No Calendar, General Shop*) que ceux définissant des précédences internes.

5.4.2 Modélisation dans ILOG PPO

Ces précédences externes sont totalement ignorées dans le module de planification, dans lequel on considère négligeable le délai entre deux activités de production liées par une telle contrainte. En revanche, les quantités de matériaux globalement produits et consommés doivent être équilibrées. Le module de *batching* décide des arcs de *pegging* à créer entre batches. Pour cela, un nouveau type de précédence est formulé et de nouvelles contraintes de flot doivent être respectées. Un arc de *pegging* établi entre deux batches induit une contrainte de précédence entre les activités associées à ces batches, celle-ci doit être satisfaite dans l'ordonnancement final.

5.4.3 Intégration au programme linéaire en nombres entiers

À présent, un batch $bat^t_{i,b}$ peut consommer à partir d'un autre batch $bat^{t'}_{i',b'}$, et produire vers un autre $bat^{t''}_{i'',b''}$ (ou vers une demande dem_d, comme auparavant). Nous introduisons alors de nouvelles notations pour représenter le flot entre deux batches $bat^t_{i,b}$ et $bat^{t'}_{i',b'}$: l'existence d'un arc de *pegging* est notée $y^{t,t'}_{i,b,i',b'}$ et la quantité du flot est notée $f^{t,t'}_{i,b,i',b'}$. Ainsi, une contrainte de flot entrant dans un batch $bat^t_{i,b}$ doit être ajoutée (5.14), et la contrainte sur le flot sortant doit être modifiée (5.15). Nous nous restreignons toujours au cas où une recette rec_i n'est définie que par une seule activité générique act_i, cependant, nous élargissons au cas où cette activité peut non seulement produire un matériau $pmat_i$ ($|\mathcal{P}m_i| = 1$) mais aussi en consommer un, $cmat_i$, ($|\mathcal{C}m_i| = 1$). On considère que pour tout couple de recettes $(rec_i, rec_{i'})$, tel que $pmat_i \neq cmat_{i'}$, ou pour tout couple de périodes $(T_t, T_{t'})$ tel que $t' < t$, on a $f^{t,t'}_{i,b,i',b'} = 0$. En outre, nous généralisons au cas où la quantité produite ou consommée est proportionnelle à la taille du batch (en ajoutant les facteurs multiplicatifs quelconques cq_i ou pq_i) :

$$\forall t, \forall i, \forall b \qquad cq_i q^t_{i,b} = \sum_{t'} \sum_{i'} \sum_{b'} f^{t',t}_{i',b',i,b} \tag{5.14}$$

$$pq_i q^t_{i,b} = \sum_{d} f^{t}_{i,b,d} + \sum_{t'} \sum_{i'} \sum_{b'} f^{t,t'}_{i,b,i',b'} \tag{5.15}$$

Les variables $f^{t,t'}_{i,b,i',b'}$ et $y^{t,t'}_{i,b,i',b'}$ sont reliées par la relation (5.16) :

$$\forall t, t', \forall i, i', \forall b, b', \qquad f^{t,t'}_{i,b,i',b'} \leq \min(pq_i BS_i, cq_{i'} BS_{i'}) y^{t,t'}_{i,b,i',b'} \tag{5.16}$$

Comme nous souhaitons exprimer une précédence temporelle entre deux batches reliés par un arc de *pegging*, de nouvelles variables et contraintes doivent être introduites. Si les batches $bat^t_{i,b}$ et $bat^{t'}_{i',b'}$ peuvent être reliés, nous souhaitons manipuler le délai entre la date de fin de $bat^t_{i,b}$ ($eet^t_{i,b}$) et la date de début de $bat^{t'}_{i',b'}$ ($est^{t'}_{i',b'}$). Comme il en a été pour les variables de *pegging* entre un batch et une demande, cela nécessite l'introduction de deux ensembles de variables semi-continues prenant respectivement les valeurs $eet^t_{i,b}$ et $est^{t'}_{i',b'}$ quand un flot circule effectivement ($f^{t,t'}_{i,b,i',b'} > 0$), et 0 sinon. Nous définissons pour cela les variables $zp^{t,t'}_{i,b,i',b'}$ et $zc^{t,t'}_{i,b,i',b'}$. Les contraintes permettant de les relier respectivement à $eet^t_{i,b}$ et $est^{t'}_{i',b'}$ sont données ci-après :

$$\left.\begin{array}{ll} \forall t,t',\forall i,i',\forall b,b', & zp^{t,t'}_{i,b,i',b'} \geq S_t y^{t,t'}_{i,b,i',b'} \\ & zp^{t,t'}_{i,b,i',b'} \leq E_t y^{t,t'}_{i,b,i',b'} \\ & zp^{t,t'}_{i,b,i',b'} \geq eet^t_{i,b} + E_t(y^{t,t'}_{i,b,i',b'} - 1) \\ & zp^{t,t'}_{i,b,i',b'} \leq eet^t_{i,b} + S_t(y^{t,t'}_{i,b,i',b'} - 1) \end{array}\right\}$$

$$\left.\begin{array}{ll} \forall t,t',\forall i,i',\forall b,b', & zc^{t,t'}_{i,b,i',b'} \geq S_{t'} y^{t,t'}_{i,b,i',b'} \\ & zc^{t,t'}_{i,b,i',b'} \leq E_{t'} y^{t,t'}_{i,b,i',b'} \\ & zc^{t,t'}_{i,b,i',b'} \geq est^{t'}_{i',b'} + E_{t'}(y^{t,t'}_{i,b,i',b'} - 1) \\ & zc^{t,t'}_{i,b,i',b'} \leq est^{t'}_{i',b'} + S_{t'}(y^{t,t'}_{i,b,i',b'} - 1) \end{array}\right\}$$

On peut alors exprimer la contrainte de précédence entre deux batches reliés par un arc de *pegging*. L'inégalité (5.17) établit que s'il existe un flot non-nul entre $bat^t_{i,b}$ et $bat^{t'}_{i',b'}$, alors $zp^{t,t'}_{i,b,i',b'}$ et $zc^{t,t'}_{i,b,i',b'}$ sont respectivement égales à $eet^t_{i,b}$ et $est^{t'}_{i',b'}$, la précédence temporelle est donc bien posée. Dans l'autre cas, toutes les variables sont nulles et la contrainte résultante $0 \geq 0$ est toujours vraie.

$$\forall t,t',\forall i,i',\forall b,b', \qquad zc^{t,t'}_{i,b,i',b'} - zp^{t,t'}_{i,b,i',b'} \geq 0 \tag{5.17}$$

5.5 Ruptures sur les demandes et hypothèse de surcapacité

5.5.1 Description de l'extension

La classe des problèmes décrite dans cette section permet de modéliser la relaxation de deux contraintes : d'une part, la satisfaction intégrale de toutes les demandes, d'autre part, l'obligation d'allouer une ressource pour l'exécution d'une activité. Ainsi, dans ce contexte, il est possible de produire moins qu'il est nécessaire pour livrer toutes les demandes ; en outre, on peut considérer qu'une activité est exécutée par d'autres ressources que celles modélisées dans le problème. Les quantités réellement satisfaites sont des décisions qui concernent les niveaux de planification et de *batching*, qui prennent des décisions relatives aux quantités de production. L'ordonnancement, quant à lui, ne remet pas en cause ces décisions, cependant, si l'ensemble des batches créés par le module précédent ne peut être réellement exécuté, par manque de

capacité (les approximations faites dans les modules de planification et de *batching* ont été trop optimistes), l'ordonnancement peut avoir recours à la non-allocation de ressource pour certaines activités ; cependant une activité "non-exécutée" est supposée l'être par une autre ressource non-modélisée et la production est considérée comme réalisée. Ne pas satisfaire une partie des demandes ou ne pas allouer de ressource pour une activité sont des décisions coûteuses. De telles instances appartiennent aux classes `UNP_XXXX` (*UNPerformed*).

5.5.2 Modélisation dans ILOG PPO

Dans ce contexte, le module de planification peut fournir un plan de production ne permettant pas de satisfaire intégralement les demandes. Cette décision peut être due au manque de capacité que le module a su anticiper, ou à l'optimisation des coûts qui a mené à une telle solution. À son tour, le module de *batching* peut remettre en cause ces décisions. En effet, le degré de détail étant plus fin, une nouvelle optimisation des quantités de production est raisonnable. Une fois que le *batching* a été résolu, ces décisions ne sont plus remises en cause. Ainsi, si l'ensemble des batches proposés n'est pas réalisable, la seule alternative qu'a le module d'ordonnancement pour générer une solution faisable est de ne pas allouer de ressource pour toutes les activités.

5.5.3 Intégration au programme linéaire en nombres entiers

Pour chaque demande dem_d considérée au niveau du *batching*, nous notons $\bar{q}_d$ la quantité associée à la rupture sur la demande dem_d (quantité non-livrée), compte tenu des décisions de production et de *pegging*. Un terme associé à cette quantité non-satisfaite est ajouté à la fonction objectif qui est alors donnée par l'expression (5.18) :

$$\begin{aligned} OBJ^{lin}_{batching} &= \sum_t \sum_i \sum_b (fc_i x^t_{i,b} + vc_i q^t_{i,b}) \\ &+ \sum_t \sum_i \sum_b \sum_d ec_d \min(BS_i, rq_d)(dt_d y^t_{i,b,d} - z^t_{i,b,d})^+ \\ &+ \sum_t \sum_i \sum_b \sum_d tc_d \min(BS_i, rq_d)(z^t_{i,b,d} - dt_d y^t_{i,b,d})^+ \\ &+ \sum_d uns_d \bar{q}_d \end{aligned} \tag{5.18}$$

En outre, comme la production relative aux décisions de *batching* peut ne plus correspondre aux décisions de planification, les contraintes (4.13) sont donc retirées du programme. Par ailleurs, les contraintes de flots (4.15) sont remplacées par l'expression (5.19) :

$$\forall d, \qquad rq_d = \sum_t \sum_i \sum_b f^t_{i,b,d} + \bar{q}_d \tag{5.19}$$

5.6 Pénalisation du niveau de stock

5.6.1 Description de l'extension

Nous introduisons dans cette section la notion de pénalité liée au stockage des produits. En effet, il n'est pas réaliste de considérer que l'on peut conserver des matériaux en stock pendant une durée indéterminée sans que cela n'induise un coût au niveau de la gestion de l'usine. Dans ce contexte, lorsqu'une certaine quantité de matériau est produite, soit elle est mise en stock pour être utilisée plus tard, soit elle est immédiatement consommée. Ces deux possibilités ne sont pas équivalentes du point de vue de l'optimisation de la chaîne de production. Le calcul exact des pénalités de stockage tient compte du niveau de stock à tout instant du temps, le coût total est donc une valeur qui dépend de la quantité stockée, de la durée du stockage et du type de produit stocké.

5.6.2 Modélisation dans ILOG PPO

Dans les modules de planification et de *batching*, les dates d'exécution des activités de production n'étant pas connues, les pénalités de stockage sont donc approximées. En ce qui concerne la planification, il s'agit de faire le produit, pour chaque matériau, de sa quantité en stock à la fin de chaque période, par la durée de la période et par le coût spécifique au matériau. À nouveau, le *batching* bénéficiant d'un degré de détail plus élevé considère ces pénalités de manière plus fine, bien qu'il ne s'agisse encore que d'une approximation. On se place ici dans le cas de production multi-niveaux (avec des précédences entre recettes comme définies en Section 5.4) et différentes situations conduisent à considérer la mise en stock potentielle de produits :

- lorsque deux batches sont reliés par un arc de *pegging*, cela ne signifie pas que la consommation du produit a lieu immédiatement après sa production et il est tout à fait possible qu'un délai s'écoule durant lequel le produit doit être réservé ;
- jusqu'à présent, nous faisions l'hypothèse que l'instant de production d'un produit fini coïncidait avec sa date de livraison ; cette hypothèse est peu réaliste, en général, les quantités produites sont regroupées avant d'être livrées, souvent en une seule fois, au client ; cela induit que certaines portions produites en avance sont potentiellement stockées avant d'être livrées.

De plus, les quantités en stock au début et à la fin de l'horizon induisent de nouveaux types d'arcs de *pegging* (décrits au paragraphe suivant). Enfin, dans le module d'ordonnancement, lorsqu'une solution est calculée, la courbe de l'évolution du stock de chaque produit est connue à chaque instant et le coût exact peut alors être optimisé.

5.6.3 Intégration au programme linéaire en nombres entiers

Le stockage des produits intermédiaires (entre deux batches) est pris en compte en ajoutant un nouveau terme dans la fonction objectif. Comme c'est le cas pour le calcul des pénalités d'avance et de retard, le délai et la quantité transférée sont deux variables et leur produit

mène à un programme quadratique ; la quantité est donc approximée par une borne supérieure. Par contre, la modélisation du stockage des produits finis nécessite l'introduction de nouvelles variables associées à la date de livraison, qui n'est désormais plus confondue avec la date de production. Si un arc de *pegging* est créé entre le batch $bat^t_{i,b}$ et la demande dem_d, la variable notée $zd^t_{i,b,d}$ désigne alors la date de livraison de la quantité $f^t_{i,b,d}$ relative à cet arc de *pegging*. La valeur des variables $zd^t_{i,b,d}$ est conditionnée par l'existence d'un arc, ce qui implique qu'elles sont semi-continues. Le même raisonnement que pour les variables $z^t_{i,b,d}$ s'applique. Un ensemble de quatre contraintes (que nous ne mentionnons pas) est nécessaire pour obtenir la disjonction suivante : $zd^t_{i,b,d}$ est égale à la date de livraison si l'arc existe et 0 sinon. Nous sommes désormais en mesure de calculer la pénalité de stockage d'un produit fini entre sa date de production et sa date de livraison. Par ailleurs, la variable $zd^t_{i,b,d}$ devient la référence pour le calcul des pénalités d'avance et de retard des livraisons. On obtient alors la nouvelle fonction objectif suivante :

$$\begin{aligned}
OBJ^{lin}_{batching} &= \sum_t \sum_i \sum_b (fc_i x^t_{i,b} + vc_i q^t_{i,b}) \\
&+ \sum_t \sum_i \sum_b \sum_d ec_d \min(BS_i, rq_d)(dt_d y^t_{i,b,d} - zd^t_{i,b,d})^+ \\
&+ \sum_t \sum_i \sum_b \sum_d tc_d \min(BS_i, rq_d)(zd^t_{i,b,d} - dt_d y^t_{i,b,d})^+ \\
&+ \sum_m \sum_t \sum_{i \backslash pmat_i = mat_m} \sum_b \sum_d stc_m \min(BS_i, rq_d)(zd^t_{i,b,d} - z^t_{i,b,d}) \\
&+ \sum_m \sum_{t,t'} \sum_{i,i' \backslash pmat_i = mat_m} \sum_{b,b'} stc_m \min(BS_i, BS_{i'})(zc^{t,t'}_{i,i',b,b'} - zp^{t,t'}_{i,i',b,b'})
\end{aligned} \tag{5.20}$$

En outre, on introduit trois nouveaux types d'arc de *pegging* afin de prendre en compte les inventaires. En effet, les flots de matériaux peuvent désormais circuler du stock vers un batch (pour les produits initiaux ou intermédiaires), du stock vers la demande (pour les produits finis) ou encore d'un batch vers le stock (pour des produits intermédiaires ou finis). On définit alors trois paires de variables (booléennes et semi-continues) pour exprimer la création d'un arc et la quantité de flot :

- entre le stock de $cmat_i$ et un batch de rec_i : $y^t_{in,i,b}$ et $f^t_{in,i,b}$,
- entre un batch de rec_i et le stock de $pmat_i$: $y^t_{out,i,b}$ et $f^t_{out,i,b}$,
- entre le stock de $rmat_d$ et la demande dem_d : y_d et f_d.

Les trois contraintes de flot (flot entrant dans un batch, flot sortant d'un batch ou flot entrant

dans une demande) sont donc modifiées (5.21,5.22,5.23) :

$$\forall t, \forall i, \forall b \qquad q^t_{i,b} = \sum_{t'} \sum_{i'} \sum_{b'} f^{t',t}_{i',b',i,b} + f^t_{in,i,b} \tag{5.21}$$

$$q^t_{i,b} = \sum_{d} f^t_{i,b,d} + \sum_{t'} \sum_{i'} \sum_{b'} f^{t,t'}_{i,b,i',b'} + f^t_{out,i,b} \tag{5.22}$$

$$\forall d, \qquad rq_d = \sum_{t} \sum_{i} \sum_{b} f^t_{i,b,d} + f_d \tag{5.23}$$

Le niveau de stock du matériau mat_m final, FS_m (à la fin de l'horizon) peut être obtenu à partir de son niveau initial IS_m et des quantités issues du plan de production. Ainsi, on peut écrire les équations reliant chaque variable booléenne à sa correspondante semi-continue.

$$\forall m, \forall t, \forall i \backslash cmat_i = mat_m, \forall b, \qquad f^t_{in,i,b} \leq \min(BS_i, IS_m) y^t_{in,i,b} \tag{5.24}$$

$$\forall m, \forall t, \forall i \backslash pmat_i = mat_m, \forall b, \qquad f^t_{out,i,b} \leq \min(BS_i, FS_m) y^t_{out,i,b} \tag{5.25}$$

$$\forall m, \forall d \backslash rmat_d = mat_m \qquad f_d \leq \min(rq_d, IS_m) y_d \tag{5.26}$$

Enfin, pour interdire aux flots de produits d'entrer en stock puis d'en sortir au lieu d'être transférés directement d'un batch à un autre (bien qu'en pratique, si la consommation n'a pas lieu immédiatement après la production, le produit sera réservé), on impose les contraintes (5.27) et (5.28), afin de ne pas prélever du stock ou mettre en stock plus qu'il n'est possible :

$$\forall m, \qquad \sum_{t} \sum_{i \backslash cmat_i = mat_m} \sum_{b} f^t_{in,i,b} + \sum_{d \backslash rmat_d = mat_m} f_d \leq (IS_m - FS_m)^+ \tag{5.27}$$

$$\sum_{t} \sum_{i \backslash pmat_i = mat_m} \sum_{b} f^t_{out,i,b} \leq (FS_m - IS_m)^+ \tag{5.28}$$

Remarques. Les inégalités (5.27) et (5.28) sont très contraignantes. On pourrait en effet souhaiter autoriser plus de transfert de produit entre les stocks et la production, cela supposerait alors d'ajouter de nouveaux termes dans la fonction objectif afin de considérer l'ensemble des transferts et d'évaluer les durées de stockage ainsi que les quantités stockées. Notons que dans de nombreux cas pratiques, le stock initial est déjà affecté de manière confirmée à des batches de production existants ou à des livraisons, donc, à l'exception des matières premières, tout ce qui est ensuite planifié d'être consommé doit correspondre à une production supplémentaire.

Par ailleurs, l'introduction d'une variable exprimant à la date de livraison relative à une demande permet d'exprimer des contraintes de fenêtres de temps pour ces livraisons. En termes d'ordonnancement par batches (cf. Chapitre 2, Section 2.3), cela revient alors à formuler des contraintes de dates de disponibilité et de dates butoir sur les activités à ordonnancer.

5.7 Temps et coûts de setup

5.7.1 Description de l'extension

Cette extension permet de modéliser la notion de setup, telle qu'elle a été définie dans la Section 1.3.1.2 du Chapitre 1. Une activité exécutée par une ressource exige que celle-ci soit configurée dans un certain état. Ainsi, si la ressource n'est pas correctement configurée avant l'exécution de cette activité, une activité spécifique de durée fixe est exécutée et éventuellement, un coût est ajouté à la fonction objectif. C'est donc la séquence des activités qui impose la nécessité d'exécution des setups. On note `STC_XXXX` (*Setup Time and Cost*) les classes de problèmes intégrant cette notion.

5.7.2 Modélisation dans ILOG PPO

Comme il a été mentionné en Section 1.2.2.2 du Chapitre 1, le modèle de planification approxime la notion de setup par ressource et par recette. Bien que la séquence des activités ne soit pas connue à ce niveau, un état est "élu", pour chaque ressource, parmi ceux qui ont été actifs durant la période ; cet état est reporté sur la période suivante afin d'optimiser au mieux le coût de setup. L'approximation permet de ne compter qu'un seul setup si la production d'une même recette est planifiée pour deux périodes consécutives. Le module de *batching* ne prend pas en compte la notion de setup, cependant, comme les décisions de planification font partie de ses entrées (et donc en quelque sorte, de ses contraintes), l'approximation faite au niveau supérieur lui est implicitement transférée. Quant au module d'ordonnancement, il prend en compte ces contraintes de manière exacte.

5.8 Synthèse sur les variantes de problèmes

Les précédentes sections de ce chapitre présentent les extensions possibles au problème de *batching* que nous avons traitées dans notre thèse, dans le cadre d'études comparatives entre différentes approches de résolution. Ces expérimentations sont présentées au Chapitre 6. D'autres extensions spécifiques à des applications réelles seront présentées au Chapitre 7 mais elles ne sont pas prises en compte dans la classification proposée ici.

Caractéristiques. Nous répartissons les instances de problèmes de *batching* dans différentes classes en fonction de la chaîne de production et des contraintes mises en jeu. Pour chaque extension, nous avons donné la taxinomie employée pour y faire référence. En général, les extensions sont indépendantes les unes des autres et il est possible de composer une classe d'instances en sélectionnant un problème de base et un certain nombre d'extensions (sauf lorsque cela mène à des situations absurdes). Cette classe sera désignée par la concaténation des notations employées pour désigner chaque caractéristique, par exemple :

- `STC_BROS` désigne la classe d'instances à un niveau de production (`OS`), où des intervalles d'indisponibilité de la ressource sont définis (`BR`) et des temps et coûts de setup peuvent survenir au cours de la production (`STC`) ;
- `UNP_MM_NCGS` désigne la classe d'instances à plusieurs niveaux (internes ou externes) de production (`GS`), où les ressources sont toujours disponibles et ont une productivité constante au cours du temps (`NC`), les activités peuvent être exécutées dans différents modes (`MM`), les demandes peuvent ne pas être intégralement satisfaites et les activités peuvent être exécutées par des ressources non-modélisées (`UNP`).

Critères d'optimisation. Par ailleurs, toute instance d'une classe ainsi décrite peut se décliner encore en une variante selon les critères considérés dans la fonction objectif. Ainsi, une lettre (de `a` à `e`) est ajoutée à la fin du nom de classe afin de caractériser la fonction objectif :

- `a` : présence uniquement de coûts de production et de coûts de retard avec des poids sur les demandes menant à des batches d'importance similaire ;
- `b` : présence uniquement de coûts de production et de coûts de retard avec des poids quelconques ;
- `c` : présence de coûts de production et de coûts d'avance et de retard avec des poids sur les demandes menant à des batches d'importance similaire ;
- `d` : présence de coûts de production et de coûts d'avance et de retard avec des poids quelconques ;
- `e` : présence de coûts de production et de coûts d'avance, de retard et de stockage (ces instances sont concernées par l'extension de la Section 5.6).

Le chapitre suivant, consacré aux études expérimentales, présente un ensemble de jeux de données qui a été élaboré au cours de cette thèse afin d'évaluer les performances de différentes approches de résolution du problème général de planification et ordonnancement de la production. Ces jeux de données sont basés sur la classification présentée ci-dessus.

Chapitre 6

Expérimentations et analyse de performances

Dans ce chapitre, nous présentons les résultats d'études expérimentales qui ont été menées sur la résolution du problème de *batching* central (*"cœur"*) mais aussi sur ses variantes étendues, présentée au Chapitre 5 (contraintes classiques dans le contexte de la production, optimisation des coûts de production, d'avance-retard et de stockage). Dans un premier temps, nous décrivons le jeu de données sur lequel ont été réalisés nos tests, ainsi que les critères d'évaluation considérés. Un premier ensemble d'expérimentations est réalisé afin d'analyser les six variations du modèle linéaire présenté en Section 4.2 du Chapitre 4 ($\mathcal{P}_0$, $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_0^+$, $\mathcal{P}_1^+$ et $\mathcal{P}_2^+$), enrichi le cas échéant des extensions présentées dans le Chapitre 5. Dans une troisième partie, nous réalisons une étude comparative entre notre modèle et trois autres algorithmes de résolution du problème de *batching* qui ont été élaborés en parallèle de nos propres travaux.

6.1 Jeu de données et indicateurs de qualités

6.1.1 Benchmark pour l'évaluation d'algorithmes

Différents jeux de données pour l'intégration des problèmes de planification et d'ordonnancement existent dans la littérature [72, 68, 88, 16]. Ils ont été mentionnés en Section 1.2.4 du Chapitre 1. Nous avons dans un premier temps tenté de les adapter à notre problématique générale, afin de tester nos propres approches. Cependant, ces benchmarks sont élaborés pour l'expérimentation d'approches micropériodiques, ou en deux phases. De plus, les fonctions d'optimisation proposées ne correspondent pas aux nôtres ; en général, la minimisation de la durée de l'ordonnancement (makespan), ou la maximisation de la productivité sont les critères considérés. Par ailleurs, il existe un jeu d'instances permettant d'évaluer des algorithmes de planification résolvant des problèmes de *lot-sizing* ; cependant, la librairie LOTSIZELIB [9] n'est pas adaptée aux problèmes d'ordonnancement et ses critères d'évaluation sont ceux de la planification (coûts de production, de setup et d'inventaire calculés par période). C'est pourquoi un

jeu de données spécifique a été élaboré afin de tester des algorithmes d'optimisation de la production, relativement à nos critères. Il n'est bien entendu pas nécessaire que ces algorithmes adoptent une approche par décomposition tri-modulaire comme dans l'APS ILOG *Plant PowerOps* (PPO) (cf. Section 2.2.3 du Chapitre 2). Il est donc tout à fait possible de mesurer les performances d'algorithmes bi-modulaires, séquentiels ou coopératifs, ou encore d'approches totalement intégrées. L'originalité du jeu de données est que l'entrée d'une instance est typique de l'entrée d'un problème de planification, et que l'évaluation se fait sur l'ordonnancement final. Les tests présentés ici sont effectués sur un ensemble de 45 problèmes, chacun décliné en 4 ou 5 instances, en fonction des critères à optimiser (certaines instances ne s'étendent pas au cas où les coûts de stockage sont pris en compte). Cet ensemble de jeux de données a été présenté dans [75]. Les 193 instances qui le composent sont réparties en 11 classes. Les caractéristiques spécifiques à chaque classe correspondent aux extensions décrites de manière générale dans le Chapitre 5. La composition de l'ensemble de jeux de données sur lequel nous avons réalisé nos expérimentations est répertoriée dans le Tableau 6.1. Les 45 problèmes composant ce benchmark ne sont pas tous de taille équivalente. Au sein d'une classe donnée, on distingue chaque problème par un numéro à 2 chiffres choisi comme suit : le chiffre des dizaines caractérise la taille du problème, celui des unités permet d'identifier de manière unique le problème. Par ailleurs, les instances de cet ensemble de jeux de données proviennent de deux origines. Certaines sont issues de problèmes d'ordonnancement et ont été modifiées afin de poser un problème plus général, ayant trait à la fois à la planification et à l'ordonnancement. En particulier, cela permet de tester notre approche de résolution en séquence des 3 problèmes de planification, *batching* et ordonnancement. D'autres instances sont des adaptations de problèmes réels rencontrés dans le cadre d'applications concrètes des problèmes de planification et ordonnancement.

Classe	Description (nb. d'instances)
NCOS	*No Calendar, One Step* (24)
BROS	*BReaks, One Step* (16)
BPOS	*Breaks-Productivity, One Step* (30)
NCGS	*No Calendar, General Shop* (32)
STC_NCOS	*Setup Times-Costs, No Calendar, One Step* (15)
STC_BPOS	*Setup Times-Costs, Breaks-Productivity, One Step* (10)
UNP_BROS	*UNPerformed, BReaks, One Step* (16)
MM_STC_NCOS	*Multi-Mode, Setup Times-Costs, No Calendar, One Step* (24)
MM_STC_BPOS	*Multi-Mode, Setup Times-Costs, Breaks-Productivity, One Step* (8)
UNP_STC_BPOS	*UNPerformed, Setup Times-Costs, Breaks-Productivity, One Step* (10)
UNP_MM_STC_BPOS	*UNPerformed, Multi-Mode, Setup Times-Costs, Breaks-Productivity, One Step* (8)

TAB. 6.1 – Répartition en classes des instances du jeu de données

6.1.2 Critères de performance

Comme il a déjà été dit, le critère principal de référence pour évaluer la qualité d'une solution de *batching* est la valeur de l'ordonnancement finalement obtenu. Ceci se justifie clairement ici : qu'il s'agisse des variations du programme linéaire élaboré dans cette thèse ou d'un autre type d'approche (cf. Section 6.3), les problèmes de *batching* résolus ne sont pas les mêmes. Dans le cas où l'on compare les variantes du programme linéaire élaboré au cours de cette thèse (cf. Chapitre 4, Section 4.2), celles-ci conduisant à des modèles différents, il est naturel que la solution optimale obtenue pour un problème donné de *batching* ne soit pas la même pour toutes les variations ; ce qui nous importe alors est de déterminer quel modèle conduit, en moyenne, aux meilleures solutions finales pour le problème d'ordonnancement. Cela est encore plus clair lorsqu'il s'agit de comparer d'autres approches de résolution, en particulier lorsqu'elles ne manipulent pas d'informations temporelles. D'une part, il est alors impossible d'estimer les coûts de stockage, d'avance et de retard, d'autre part, ces approches ne définissent pas nécessairement de fonction objectif. Dans tous les cas, étant donné une solution de *batching* (ensembles de batches et d'arcs de *pegging*), un décideur ne peut évaluer cette solution sans avoir une idée de l'ordonnancement auquel elle peut conduire. C'est pourquoi, dans notre approche générale, le module d'ordonnancement agit comme un oracle afin de comparer les différentes méthodes. Nous étudions donc la moyenne des **erreurs relatives par rapport aux meilleures solutions connues** pour l'ordonnancement, commises par chacune des méthodes, ainsi que le **nombre de meilleures solutions atteintes** dans chaque cas.

Calcul de l'erreur moyenne. Le critère de performance central dans notre étude est relatif à la valeur de la meilleure solution connue pour l'ensemble des instances testées. Pour chacune des instances, la meilleure solution peut avoir été déterminée de différentes manières :

- obtention après que les trois problèmes définis dans l'architecture d'ILOG PPO (planification, *batching* et ordonnancement) aient été résolus en séquence, en ayant recours à différentes méthodes de résolution pour le *batching* ;
- calcul par un algorithme spécifique, dédié au problème.

On note O^* la valeur de l'objectif de la meilleure solution connue et $O_{\mathcal{A}}$ celle obtenue par la méthode $\mathcal{A}$ ($\mathcal{A}$ peut par exemple désigner la prodécure de *batching* employée). L'erreur relative (ou écart relatif) commise entre $O_{\mathcal{A}}$ et O^* est alors donnée par la formule $(O_{\mathcal{A}} - O^*)/O^*$ (notons que $O_{\mathcal{A}} - O^* \geq 0$ puisque l'ordonnancement est un problème de minimisation).

Nombre de meilleures solutions atteintes. Comme il a été annoncé, dans notre étude expérimentale, plusieurs méthodes de résolution de *batching* sont comparées. Chaque méthode conduit à l'ordonnancement de meilleur coût connu pour un certain nombre d'instances. Ce nombre de meilleures solutions atteintes par chaque méthose est le second critère pris en compte dans notre analyse.

6.2 Étude des variations du programme linéaire

Dans cette section nous procédons à l'étude des six différentes versions du programme linéaire en nombres entiers que nous avons élaboré pour résoudre le problème de *batching*, afin de mettre en évidence l'impact sur la qualité des solutions finalement obtenues dans l'ordonnancement, des contraintes optionnelles présentées en Section 4.2.3.2 du Chapitre 4.

6.2.1 Résultats numériques

Temps d'exécution. Un temps de calcul total limité est défini en fonction de la taille de chaque instance. En effet, pour chaque problème, on peut estimer le nombre d'activités qu'il sera nécessaire d'ordonnancer à la sortie du module de *batching*, ce qui permet de définir un temps limite de calcul raisonnable, relativement à la taille d'un problème donné. Le Tableau 6.2 donne le temps de calcul attribué à la résolution du problème global, pour chaque type de problème, pour un ordinateur cadencé à 1 GHz.

Taille de problème	$0x$	$1x$	$2x$	$3x$	$4x$	$5x$	$6x$	$7x$	$8x$
Estimation du nombre d'activités	10	25	50	75	100	200	500	1000	2500
Temps de calcul	60s	150s	300s	450s	600s	1200s	1800s	2400s	3600s

TAB. 6.2 – Temps limite attribué pour la résolution de chaque type de problème

Pour chaque programme linéaire de *batching* testé, le temps total attribué pour la résolution du problème global est réparti équitablement en trois, de sorte que chaque module (planification, *batching* et ordonnancement) a le même temps pour calculer sa solution. Ainsi, il peut arriver que le programme linéaire ne soit pas résolu à l'optimalité. Notons que cela n'est pas rédhibitoire puisque toute solution réalisable du programme linéaire est une solution de *batching* et non pas uniquement une solution optimale. L'optimisation d'une telle solution, bien qu'en général profitable, n'en est pas moins heuristique du point de vue de l'optimisation globale de la production. Le modèle présenté dans cette thèse approxime la réalité sur différents aspects et l'obtention d'une solution optimale n'est pas une nécessité du point de vue de la réalisabilité du problème d'ordonnancement posé à sa suite.

Le Tableau 6.3 donne, pour chacune des six variantes du programme linéaire élaboré dans cette thèse et pour chaque classe de problème, l'erreur relative moyenne par rapport aux meilleures solutions connues. Nous produisons aussi le nombre de meilleures solutions atteintes dans chaque cas.

6.2.2 Analyse

Comme il était attendu, les performances des modèles sans contraintes additionnelles d'énergie ($\mathcal{P}_0$ et $\mathcal{P}_0^+$) sont les moins bonnes. Ces résultats montrent immédiatement l'intérêt que nous

Classes (instances)	$\mathcal{P}_0$	$\mathcal{P}_1$	$\mathcal{P}_2$	$\mathcal{P}_0^+$	$\mathcal{P}_1^+$	$\mathcal{P}_2^+$
NCOS (24)	9.30 %	8.86 %	**4.75 %**	9.30 %	8.82 %	**4.75 %**
BROS (16)	12.80 %	12.53 %	**7.15 %**	12.13 %	11.52 %	8.46 %
BPOS (30)	15.84 %	15.89 %	**7.06 %**	15.59 %	15.89 %	7.71 %
NCGS (32)	23.26 %	21.94 %	17.98 %	23.21 %	22.07 %	**17.83 %**
STC_NCOS (15)	8.97 %	8.97 %	**6.04 %**	8.97 %	8.97 %	**6.04 %**
STC_BPOS (10)	17.78 %	18.06 %	**17.04 %**	18.18 %	17.78 %	17.11 %
UNP_BROS (16)	20.10 %	20.13 %	**9.47 %**	20.13 %	20.13 %	10.17 %
MM_STC_NCOS (24)	12.55 %	12.08 %	9.15 %	12.64 %	12.17 %	**9.05 %**
MM_STC_BPOS (8)	39.33 %	43.64 %	**12.25 %**	49.54 %	44.11 %	12.55 %
UNP_STC_BPOS (10)	10.03 %	9.71 %	**9.59 %**	9.75 %	10.86 %	**9.59 %**
UNP_MM_STC_BPOS (8)	160.32 %	160.36 %	**14.55 %**	160.32 %	160.36 %	**14.55 %**
Moyenne (193)	22.18 %	22.01 %	**10.15 %**	22.52 %	22.02 %	10.39 %
Nb. de meilleures solutions atteintes (/193)	39	39	**43**	39	39	**43**

TAB. 6.3 – Écart relatif moyen aux meilleures solutions connues : variantes du MILP

avons à modéliser l'occupation des ressources au cours du temps dans le module de *batching*. En effet, bien que prise en compte dans les modules de planification et d'ordonnancement, une modélisation intermédiaire s'avère nécessaire. De fait, en planification, les contraintes de ressources sont en général agrégées et approximées, tandis que lors de l'ordonnancement, elles sont prises en compte de manière exacte. Une transition directe entre ces deux optiques, en résolvant le *batching* aveuglément, mène à de mauvaises solutions pour l'ordonnancement. L'ajout des contraintes d'énergie périodique (c'est-à-dire sur un intervalle égal à une période de planification) ne conduit pas clairement à un meilleur modèle. Cela s'explique par le fait que ces contraintes sont déjà formulées dans le module de planification, le plan de production qu'il convient de suivre au cours du *batching* tient déjà compte de ces restrictions. Par conséquent, les ajouter à nouveau dans le programme linéaire présente une certaine forme de redondance et apporte peu, en terme de performances. On note que les classes pour lesquelles l'ajout de ces contraintes est même néfaste sont celles qui font intervenir des calendriers sur les ressources (définissant alors des périodes d'indisponiblité ou de variation de productivité). Cela est à mettre en relation avec la modélisation des contraintes de ressources qui sont encore plus difficiles à prendre en compte dans ces cas-là, et le degré d'approximation de la formulation conduit à des modèles plus éloignés encore de la réalité que lorsqu'aucune contrainte n'est ajoutée.

En revanche, les contraintes d'énergie sous-périodique (posées sur un intervalle plus court qu'une période de planification) sont réellement bénéfiques pour atteindre de meilleures solutions d'ordonnancement. Elles améliorent nettement la formulation et conduisent à des solutions tout à fait satisfaisantes : l'erreur relative moyenne calculée sur toutes les instances du benchmark reste faible. En effet, ces contraintes ont une sémantique proche d'une formulation micro-périodique

(définie au Chapitre 1), ce qui permet de rendre notre formulation macro-périodique plus précise, donc plus réaliste, et ainsi d'obtenir de meilleures solutions. Cependant, tout en améliorant la modélisation, il n'est pas impossible que la solution optimale du problème de *batching* dans lequel ces contraintes sont posées mène parfois à de moins bons ordonnancements que si elles avaient été omises. En effet, la discrétisation des périodes en sous-périodes est réalisée de manière heuristique : plus il y a de sous-périodes, plus le modèle est précis, mais aussi, plus l'optimum global risque de nous échapper, et même, le programme linéaire peut devenir infaisable. Il s'agit de trouver un compromis entre un nombre de sous-périodes qui permettent une formulation précise tout en posant un problème réalisable. En outre, lorsque le nombre de sous-périodes augmente, la taille des modèles augmente avec lui, ce qui est un facteur à prendre en compte lors de l'élaboration du modèle et du calcul des sous-périodes.

Contrairement à ce qui aurait pu être attendu, les contraintes de rupture de symétrie ne sont en moyenne pas intéressantes pour la formulation, et mènent globalement à de moins bonnes solutions finales. Il convient de faire remarquer que les classes de problèmes pour lesquelles l'ajout de ces contraintes est bénéfique sont celles qui ne définissent pas de calendrier sur les ressources, le séquencement *a priori* des activités n'est donc pas une approximation de la réalité ici, tandis que lorsqu'il s'agit d'instances où l'énergie de la ressource varie au cours du temps, cela n'étant pas modélisé de manière exacte, le séquencement *a priori* conduit à une formulation trop erronée du problème. En outre, l'ajout de ces contraintes renforce sensiblement le modèle, ce qui augmente la difficulté de sa résolution. Ainsi, comme nous ne résolvons pas toujours les problèmes de *batching* à l'optimalité, il est possible que la solution obtenue à la fin du temps imparti soit de moins bonne qualité que si les contraintes de rupture de symétrie n'avaient pas été posées, ce qui peut à son tour conduire à de moins bonnes solutions pour l'ordonnancement.

Conclusion. Au regard des expérimentations réalisées sur les six variantes du programme linéaire, il apparaît clairement que le modèle $\mathcal{P}_2$ (contraintes d'énergie sous-périodique et pas de rupture de symétrie) est le plus performant, c'est d'ailleurs celui qui permet d'atteindre le plus de meilleures solutions connues pour le problème d'ordonnancement associé. Dans la suite, c'est cette formulation qui est utilisée pour comparer notre modèle à d'autres approches de résolution du problème de *batching*.

6.3 Comparaisons de différentes méthodes de résolution

Dans cette seconde phase d'expérimentations, nous comparons les performances du programme $\mathcal{P}_2$ à trois autres méthodes implantées dans l'APS ILOG PPO pour résoudre le problème de *batching* : un second programme linéaire en nombres entiers résolu par ILOG CPLEX, et deux heuristiques. Ces algorithmes ont été élaborés en parallèle de nos travaux de thèse, de manière indépendante pour les heuristiques, en s'inspirant en partie de nos travaux pour le second programme linéaire en nombres entiers.

6.3.1 Description des trois autres approches de résolution

6.3.1.1 Le programme linéaire $\mathcal{C}$

Élaboré à la suite de la formulation de notre programme linéaire (e.g. $\mathcal{P}_2$), ce modèle s'en inspire en en modifiant certains aspects. La nécessité d'une nouvelle formulation, plus simple, s'est en effet fait ressentir, notre modèle présentant l'inconvénient de mener, dans certains cas, à des programmes de grande taille, difficiles à résoudre. Cet inconvénient est la contrepartie d'un modèle réaliste, flexible et extensible.

Contraintes communes. Le modèle $\mathcal{C}$ permet de résoudre un problème de *batching*, par conséquent, une solution obtenue à partir de sa résolution satisfait les contraintes de bornes sur la taille des batches et de flot de produit. Ce sont en effet, les contraintes nécessaires à l'obtention d'une solution de *batching* telle qu'elle a été définie au Chapitre 2 (Section 2.2.3.2).

Contraintes modifiées. La différence entre les modèles $\mathcal{C}$ et $\mathcal{P}_2$ se situe d'une part dans la modélisation des contraintes de capacité des ressources. En effet, le programme $\mathcal{C}$ définit des intervalles d'exécution pour les batches plus précis que la période où ils sont planifiés sans introduire de contraintes d'énergie (contrairement aux contraintes (4.21) et (4.22) de $\mathcal{P}_2$). D'autre part il est possible de créer des batches regroupant des quantités initialement planifiées dans des périodes différentes.

Fonction objectif. Le modèle $\mathcal{C}$ peut estimer par des constantes les dates d'exécution des batches (compatibles avec les intervalles mentionnés ci-dessus), ce qui permet de linéariser l'objectif original, soit en approximant les délais entre deux batches ou entre batches et demandes, soit en approximant la valeur des flots de matériaux sur les arcs de *pegging* (comme cela est fait dans $\mathcal{P}_2$) ; on peut alors multiplier le terme approximé par la valeur exacte (variable) du délai ou du flot, selon le cas. La formulation précise servant de référence dans ce chapitre approxime les délais et considère les flots exacts pour les coûts de stockage, et approxime les flots en considérant les délais variables pour les coûts d'avance et de retard. La multiplication des pénalités de stockage par un terme logarithmique constant (fonction du délai approximé) permet de rompre la symétrie du problème.

6.3.1.2 L'heuristique $\mathcal{H}_1$

Nous le verrons au Chapitre 7, le programme $\mathcal{P}_2$ est capable de résoudre des problèmes réels de l'industrie de production. Cependant, les temps de calcul étant prohibitifs (les modèles générés sont de très grande taille), il était nécessaire que l'APS ILOG PPO dispose de méthodes plus rapides de résolution. Ainsi, une première heuristique a été élaborée. Cette approche résout simultanément les sous-problèmes de *lot-streaming* — découpage en batches — et de *pegging* — établissement des flots de produits — (cf, Chapitre 2, Section 2.1.1). Au cœur de la procédure, une politique dérivée du "lot-pour-lot" (décrite au Chapitre 1, Section 1.2.4.3), respectant les

contraintes sur la taille des batches, est employée. Pour cela, l'algorithme regroupe les batches dont il est attendu qu'ils commencent dans un même intervalle de temps et établit ainsi les liens de *pegging*.

6.3.1.3 L'heuristique $\mathcal{H}_2$

Cette dernière méthode a aussi été implémentée en parallèle du modèle $\mathcal{P}_2$. Il s'agit d'une procédure gloutonne en deux phases :

1. le *lot-streaming* (découpage) est réalisé, pour chaque période, de façon à créer un minimum de batches, ils sont donc de taille maximale et un processus de réparation (retour arrière) intervient en cas d'infaisabilité de manière à ce que, si une solution existe, elle soit nécessairement trouvée ;
2. le *pegging* est ensuite calculé, en remontant des demandes vers les batches, et à chaque niveau, en progressant chronologiquement.

Temps d'exécution. Lorsque le *batching* est résolu de manière heuristique (par $\mathcal{H}_1$ ou $\mathcal{H}_2$), le temps non-utilisé par ce module peut alors être affecté à l'optimisation de l'ordonnancement.

6.3.2 Écart relatif moyen aux meilleures solutions connues

Le Tableau 6.4 donne, pour chacune des quatre approches de résolution que nous comparons ici l'erreur moyenne relative par rapport aux meilleures solutions connues. Nous produisons à nouveau le nombre de meilleures solutions atteintes dans chaque cas.

Classes (instances)	$\mathcal{P}_2$	$\mathcal{C}$	$\mathcal{H}_1$	$\mathcal{H}_2$
NCOS (24)	**4.75 %**	8.49 %	18.26 %	14.20 %
BROS (16)	**7.15 %**	12.28 %	15.36 %	19.72 %
BPOS (30)	**7.06 %**	14.35 %	17.73 %	18.42 %
NCGS (32)	17.98 %	**11.33 %**	23.56 %	15.22 %
STC_NCOS (15)	**6.04 %**	9.78 %	18.45 %	22.85 %
STC_BPOS (10)	17.04 %	**15.25 %**	22.66 %	19.13 %
UNP_BROS (16)	**9.47 %**	15.69 %	20.09 %	22.54 %
MM_STC_NCOS (24)	9.15 %	**3.36 %**	16.26 %	28.09 %
MM_STC_BPOS (8)	12.25 %	**3.63 %**	25.36 %	57.87 %
UNP_STC_BPOS (10)	9.59 %	5.73 %	**5.47 %**	7.32 %
UNP_MM_STC_BPOS (8)	**14.55 %**	30.79 %	31.57%	67.64 %
Moyenne (193)	**10.15 %**	11.18 %	19.14 %	22.50 %
Nb. de meilleures solutions atteintes (/193)	43	**46**	30	36

TAB. 6.4 – Écart relatif moyen aux meilleures solutions connues : comparaison de 4 méthodes

Analyse. Dans cette étude, il ne s'agit pas tant de comparer les quatre méthodes entre elles que chacune par rapport à une référence commune. En moyenne, l'erreur commise est minimale lorsque l'on utilise le programme $\mathcal{P}_2$. Cependant, le programme linéaire $\mathcal{C}$ est aussi très compétitif et se révèle même meilleur sur certaines classes de problèmes, c'est en outre l'approche qui permet d'atteindre le plus de meilleures solutions connues pour l'ordonnancement. D'une manière générale, les heuristiques sont, pour leur part, moins performantes. Une classe est néanmoins mieux résolue en moyenne par $\mathcal{H}_1$ que par les deux programmes linéaires, pourtant plus élaborés. Cela montre la difficulté inhérente à l'élaboration de méthodes de résolution génériques pour des problèmes difficiles comme ceux posés dans ce jeu de données. Le défi consiste en effet à rester globalement efficace en moyenne, et dans la plupart des situations.

Les performances obtenues par la résolution des deux programmes linéaires, qui sont donc des méthodes d'optimisation, démontrent l'importance que peuvent avoir les décisions de *batching* au sein d'un processus global d'optimisation d'une chaîne de production. En effet, les heuristiques $\mathcal{H}_1$ et $\mathcal{H}_2$ permettent aussi de résoudre le problème, mais de manière naïve, elles conduisent cependant à des solutions de qualité médiocre pour l'ordonnancement. À nouveau, nous tenons à faire remarquer que le choix de juger des performances d'une approche au regard de la solution finale de l'ordonnancement peut être discuté. D'une part, l'ordonnancement n'est qu'une approximation de ce qui sera finalement mis en œuvre au sein de l'usine. D'autre part, un ensemble de décisions proposé pour la production d'une usine peut être soumis à différents critères d'évalutation qui ne sont pas nécessairement représentés par un ordonnancement (gestion globale des stocks, fréquence des commandes en matières premières...). Cependant, il n'est pas déraisonnable d'utiliser un critère d'évaluation opérant sur un ensemble de décisions dont le degré de détail est élevé. Par ailleurs, les autres critères ayant trait à des niveaux décisionnels plus élevés (e.g. tactique), on peut supposer qu'ils sont considérés dans les étapes de prises de décisions qui se situent en amont.

Remarque sur les temps de calcul. Les heuristiques $\mathcal{H}_1$ et $\mathcal{H}_2$ s'exécutant très rapidement, l'obtention de la solution est instantanée, c'est pourquoi nous ne considérerons pas ces méthodes dans nos remarques. Le processus d'expérimentation mis en place dans cette étude fixe une durée maximale pour la résolution d'un programme linéaire (la même pour $\mathcal{P}_2$ ou $\mathcal{C}$), il se peut donc, dans certains cas, que la solution optimale ne soit pas atteinte, puisque la recherche est tronquée au bout d'un certain temps limite. Certaines instances dites "faciles" (correspondant en général à des problèmes élémentaires et de petite taille — `NCOS, BROS, BPOS`) sont résolues à l'optimalité instantanément, d'autres peuvent prendre beaucoup de temps avant que l'algorithme de *branch and cut* ne converge. Le gap (écart relatif entre la borne inférieure courante et la meilleure solution entière trouvée) peut rester très élevé, même à la fin du temps imparti. Notons cependant que cette valeur peut ne pas être pertinente pour évaluer la qualité de la solution entière finalement obtenue ; en effet, si le gap est élevé, cela ne signifie pas nécessairement que la solution trouvée se situe loin de l'optimum, la valeur courante de la borne inférieure pouvant être de mauvaise qualité.

6.3.3 Qualité de l'ordonnancement

Dans cette section, nous nous intéressons à nouveau à la valeur de la solution d'ordonnancement finalement obtenu après l'utilisation des différentes méthodes de résolution de *batching*. Nous ne les comparons cette fois-ci plus à la meilleure solution obtenue dans l'absolu, mais les unes par rapport aux autres. On note $O_{\mathcal{P}_2}$, respectivement $O_{\mathcal{C}}$, $O_{\mathcal{H}_1}$ et $O_{\mathcal{H}_2}$, la valeur de la solution d'ordonnancement obtenu après la résolution du *batching* par $\mathcal{P}_2$, respectivement $\mathcal{C}$, $\mathcal{H}_1$ et $\mathcal{H}_2$. Le Tableau 6.5 répertorie les résultats suivants :

- la seconde colonne donne le nombre d'instances pour lesquelles l'algorithme $\mathcal{A}$ et plus compétitif ($O_{\mathcal{A}} < O_{\mathcal{A}'}$), équivalent ($O_{\mathcal{A}} = O_{\mathcal{A}'}$) ou moins compétitif ($O_{\mathcal{A}} > O_{\mathcal{A}'}$), que l'algorithme $\mathcal{A}'$; ces mesures sont données dans un premier temps entre l'approche élaborée dans notre thèse ($\mathcal{P}_2$) et les trois autres méthodes, puis nous comparons ces trois dernières entre elles, principalement à titre informatif ;
- la quatrième colonne fournit des informations sur la déviation moyenne entre les solutions trouvées par deux méthodes différentes ; le pourcentage moyen d'amélioration, respectivement de dégradation, est calculé par rapport au nombre de solutions de strictement meilleure qualité, respectivement, de strictement moins bonne qualité.

▶ $\mathcal{P}_2$ vs $\mathcal{C}/\mathcal{H}_1/\mathcal{H}_2$	Nb. d'instances	Déviations moyennes	
$O_{\mathcal{P}_2} < O_{\mathcal{C}}$	77/193 (39.9 %)	% amélioration moyenne	11.17 %
$O_{\mathcal{P}_2} = O_{\mathcal{C}}$	41/193 (21.2 %)	% dégradation moyenne	8.16 %
$O_{\mathcal{P}_2} > O_{\mathcal{C}}$	75/193 (38.9 %)		
$O_{\mathcal{P}_2} < O_{\mathcal{H}_1}$	118/193 (61.1 %)	% amélioration moyenne	19.50 %
$O_{\mathcal{P}_2} = O_{\mathcal{H}_1}$	26/193 (13.5 %)	% dégradation moyenne	11.54 %
$O_{\mathcal{P}_2} > O_{\mathcal{H}_1}$	49/193 (25.4 %)		
$O_{\mathcal{P}_2} < O_{\mathcal{H}_2}$	109/193 (56.5 %)	% amélioration moyenne	24.12 %
$O_{\mathcal{P}_2} = O_{\mathcal{H}_2}$	41/193 (21.2 %)	% dégradation moyenne	8.60 %
$O_{\mathcal{P}_2} > O_{\mathcal{H}_2}$	43/193 (22.3 %)		
▶ $\mathcal{C}$ vs $\mathcal{H}_1$ vs $\mathcal{H}_2$	Nb. d'instances	Déviations moyennes	
$O_{\mathcal{C}} < O_{\mathcal{H}_1}$	115/193 (59.6 %)	% amélioration moyenne	15.23 %
$O_{\mathcal{C}} = O_{\mathcal{H}_1}$	35/193 (18.1 %)	% dégradation moyenne	8.16 %
$O_{\mathcal{C}} > O_{\mathcal{H}_1}$	43/193 (22.3 %)		
$O_{\mathcal{C}} < O_{\mathcal{H}_2}$	122/193 (63.2 %)	% amélioration moyenne	18.56 %
$O_{\mathcal{C}} = O_{\mathcal{H}_2}$	39/193 (20.2 %)	% dégradation moyenne	12.47 %
$O_{\mathcal{C}} > O_{\mathcal{H}_2}$	32/193 (16.6 %)		
$O_{\mathcal{H}_1} < O_{\mathcal{H}_2}$	92/193 (47.7 %)	% amélioration moyenne	11.23 %
$O_{\mathcal{H}_1} = O_{\mathcal{H}_2}$	41/193 (21.2 %)	% dégradation moyenne	12.06 %
$O_{\mathcal{H}_1} > O_{\mathcal{H}_2}$	60/193 (31.1 %)		

TAB. 6.5 – Résultats généraux sur les valeurs de l'ordonnancement

Analyse. La moitié supérieure du tableau montre l'efficacité de l'approche développée dans cette thèse. En effet, celle-ci permet d'obtenir des solutions meilleures ou équivalentes à celles obtenues par les heuristiques dans plus de 75 % des cas. Les performances des programmes $\mathcal{P}_2$ et $\mathcal{C}$ sont par ailleurs globalement comparables. En outre, les solutions sont significativement meilleures, surtout par rapport aux heuristiques (respectivement 19.50 % et 24.12 %), mais aussi cette fois par rapport à $\mathcal{C}$ (11.17 %). En revanche, les dégradations, lorsqu'elles ont lieu, restent modestes (entre 8.16 % et 11.54 %). À nouveau, le programme $\mathcal{C}$ se présente aussi comme une très bonne méthode de résolution, ce qui confirme l'impact d'une résolution affinée du problème de *batching* sur la qualité des solutions finales. En effet, cette approche est même plus robuste que $\mathcal{P}_2$ lorsqu'on la compare à l'heuristique $\mathcal{H}_1$, ce comportement ne se retrouve pas dans la comparaison avec $\mathcal{H}_2$, où $\mathcal{P}_2$ s'avère alors être une méthode plus performante.

Il est intéressant de constater, à travers ces résultats, qu'il est difficile d'établir un ordre total entre les méthodes de résolution. Bien qu'il apparaisse clairement une supériorité, en moyenne, des programmes linéaires sur les heuristiques, cela ne se vérifie pas systématiquement. En outre, la comparaison des deux programmes linéaires montre une meilleure performance du programme $\mathcal{P}_2$, cependant, cette comparaison s'effectue en moyenne et cette dominance n'est pas très prononcée.

6.3.4 Étude des différents critères d'optimisation

Nous présentons dans cette section une analyse comparative des quatre méthodes, relativement aux critères composant la fonction objectif. En effet, nous avons annoncé en Section 5.8 du Chapitre 5 cinq types d'instances vis-à-vis des critères. Toutes les instances prennent en compte les coûts de production de la même façon, ce qui n'est pas le cas pour les coûts d'avance, de retard et de stockage. Les instances de type `a` et `b` ne considèrent que les coûts de retard (avec des pondérations différentes), les instances de type `c` et `d` considèrent les coûts d'avance et de retard (avec des pondérations différentes) et enfin les instances de type `e`, considèrent, en plus des coûts d'avance et de retard, les pénalités de stockage. Le Tableau 6.6 donne, pour chaque méthode de résolution et pour chaque type de fonction objectif, la moyenne des erreurs relatives commises :

Classes (instances)	$\mathcal{P}_2$	$\mathcal{C}$	$\mathcal{H}_1$	$\mathcal{H}_2$
type `a` (45)	5.98 %	**4.82 %**	7.24 %	8.00 %
type `b` (45)	**13.66 %**	13.98 %	24.28 %	23.68 %
type `c` (45)	6.71 %	**6.52 %**	12.17 %	15.70 %
type `d` (45)	**11.38 %**	12.33 %	21.68 %	26.03 %
type `e` (13)	**8.20 %**	14.87 %	21.70 %	21.42 %

TAB. 6.6 – Écart relatif moyen : étude des différentes fonctions objectifs

Analyse. Les instances pour lesquelles les poids sont équivalents (`a` et `c`) sont inconstestablement plus faciles à résoudre. En effet, les quatre méthodes de résolution obtiennent une erreur relative moyenne faible, notamment sur les instances de type `a`, qui ne considèrent que le critère de retard. À nouveau, les performances des programmes linéaires en nombres entiers sont en moyenne équivalentes sur les instances de types `a`, `b`, `c` et `d`, tandis que les heuristiques perdent vite en compétitivité lorsque l'on ajoute le critère d'avance ou que l'on généralise les poids à des valeurs quelconques, ce qui s'explique bien par le fait que les heuristiques ne considèrent pas ces critères dans leur procédure. Les instances de type `e` sont quant à elles significativement mieux résolues par notre programme linéaire. En effet, la prise en compte des ce critère est plus précise dans notre modélisation que dans celle définie par $\mathcal{C}$, ce qui rend par conséquent son comportement meilleur en moyenne. À nouveau les heuristiques présentent de très faibles performances sur ces instances.

6.4 Conclusion

Afin d'évaluer la procédure de résolution du problème général de planification et ordonnancement de l'APS ILOG PPO, présentée en Section 2.2 du Chapitre 2, un ensemble de jeux de données a été élaboré. Ce benchmark a permis de tester les performances d'une part de différentes variantes du programme que nous avons formulé dans cette thèse afin de converger vers une version globalement plus performante. D'autre part, il a été possible de confronter cette approche à d'autres méthodes de résolution du problème de *batching*.

Nous avons pu établir que les modèles posant les contraintes d'énergie sous-périodique (4.22) (Section 4.2.3.2, Chapitre 4) pour approximer de manière fine l'occupation des ressources au cours du temps sont les plus performants. En particulier, le programme $\mathcal{P}_2$ présente le meilleur comportement en moyenne, pour la résolution des instances étudiées. Dans l'absolu, la qualité des solutions obtenues est tout à fait satisfaisante puisque l'écart relatif moyen aux meilleures solutions connues reste faible (10.15 %). En outre, la comparaison de ce programme à trois autres approches a montré sa compétitivité et même sa supériorité en moyenne, ce malgré la présence de plusieurs facteurs pouvant mener à de mauvaises solutions (objectif linéaire approximant le problème quadratique à l'origine, estimation parfois erronée des contraintes de capacité, résolution tronquée par le temps de calcul).

Par ailleurs, l'étude expérimentale réalisée dans ce chapitre a aussi permis de montrer la difficulté inhérente à la réalisation d'une méthode de résolution générique pour des problèmes complexes et variés comme ceux posés par le jeu de données. Cela explique que les comportements observés, pour chacune des quatre approches, ne sont pas toujours corrélés entre les classes de problèmes. Nous n'avons pas pu établir de classement net de ces méthodes, en terme de compétitivité. Cependant, les approches visant à optimiser de manière fine les solutions de batching, c'est-à-dire les deux programmes linéaires, sont clairement plus performantes que les heuristiques. Cela valide la problématique posée dans cette thèse : se poser la question des

décisions relatives au découpage des batches en terme de problèmes d'optimisation conduit effectivement à de meilleures solutions pour l'ordonnancement final.

Chapitre 7

Intégration logicielle et étude de problèmes industriels

Dans ce chapitre, nous nous intéressons à la mise en application des travaux que nous avons réalisés au cours de cette thèse, en particulier du programme mathématique présenté dans les Chapitres 4 et 5. Nous décrivons dans un premier temps l'APS ILOG *Plant PowerOps* (PPO), application logicielle définissant notre cadre technique de travail et dans laquelle nous avons intégré nos travaux. Dans un second temps, nous analysons un cas industriel dont le problème de *batching* intrinsèque a pu être résolu par une version étendue et adaptée de notre modèle (dénoté dans ce chapitre par $\mathcal{P}$). En effet, de nouvelles contraintes spécifiques ont été formulées afin de prendre en compte les particularités du cas pratique étudié.

7.1 L'APS ILOG *Plant PowerOps*

7.1.1 Présentation de l'application : vocation et objectifs

Le logiciel ILOG PPO est une application permettant de résoudre des problèmes de planification à moyen terme et d'ordonnancement détaillé. Les logiques métier qui orientent les modèles et les méthodes de résolution implantés appartiennent principalement à l'industrie de *process*, en particulier l'agroalimentaire. Cependant, certains aspects, classiques dans le milieu de la production, sont communs à différents types d'industrie, ce qui permet à PPO de traiter des problèmes issus d'autres environnements, en particulier, de l'industrie discrète, bien que cela ne soit pas sa vocation principale. Néanmoins, la spécialisation de PPO à l'industrie de *process* permet, d'une part de proposer des outils de modélisation permettant d'obtenir une formulation réaliste des problèmes posés quand ils appartiennent à ce cadre, d'autre part, de développer des méthodes de résolution adaptées aux contraintes particulières qui se retrouvent dans la majorité des problèmes issus de l'industrie de *process*.

Ce logiciel s'adresse aux entreprises de production désireuses de gérer et optimiser automatiquement leur fonctionnement, au niveau planification (cf. Figure 7.1) comme au niveau ordonnancement (cf. Figure 7.2). Il a pour but d'être implanté au niveau des usines dont il modélise précisément la chaîne de production afin de proposer des plans de production globale et des ordonnancements détaillés de l'activité quotidienne, réalisables en pratique et de grande qualité (relativement aux critères requis par l'entreprise). Il n'y a donc pas d'intermédiaire entre les concepteurs du logiciel et les utilisateurs finaux, qui sont des planificateurs sans connaissance de la Recherche Opérationnelle. Il est donc nécessaire de comprendre précisément les problèmes à résoudre et d'identifier, au sein d'un ensemble de contraintes métier, celles qui sont critiques de celles qui n'expriment que des préférences. Les expériences de chacun des intervenants, lors du démarrage d'un projet, sont si différentes qu'il peut s'écouler un certain temps avant que tout le monde parle enfin le même langage et que chacun puisse être compris sans ambiguïté par son interlocuteur. En ce qui concerne le concepteur du logiciel, c'est dans cette phase que vont se décider les orientations pour la modélisation du problème : contraintes et critères d'optimisation. L'une des difficultés dans le développement d'un tel logiciel réside dans la bonne connaissance des environnements et des contraintes métier, afin d'aboutir à l'intégration et au déploiement d'une application réaliste, viable, performante et robuste.

7.1.2 Architecture générale et intégration des travaux de thèse

7.1.2.1 Structuration du logiciel

Nous l'avons dit, l'APS ILOG PPO se divise en trois phases : planification, *batching* et ordonnancement. Chaque phase est prise en charge par un module logiciel que l'on peut qualifier d'indépendant vis-à-vis des autres, c'est-à-dire que les entrées et les sorties de chaque module sont clairement identifiées, et les modules adjacents doivent respecter ces informations afin que le système soit cohérent. Une connaissance de la méthode de résolution du problème relatif à un module n'est pas nécessaire aux interactions entre modules, chacun agit comme une *boîte noire* pour les autres.

L'architecture est hiérarchique et séquentielle. Au départ, le système contient les données principales (e.g. description de la chaîne de production, ressources, recettes, demandes). En premier lieu, un problème de planification est résolu et sa solution (sortie), ajoutée aux données principales, constitue l'entrée du module de *batching*. À son tour, il résout le problème associé et fournit en sortie les données nécessaires à la pose d'un problème d'ordonnancement, qui est alors résolu. Une seule méthode est implantée pour la résolution du problème de planification : un programme linéaire en variables mixtes basé sur les modèles de *lot-sizing* et intègrant un grand nombre de contraintes additionnelles et spécifiques à l'industrie de *process* est résolu par ILOG CPLEX. Au contraire, le problème de *batching* peut être résolu par différentes méthodes. Nous en avons mentionné quatre au Chapitre 6 : deux programmes linéaires ($\mathcal{P}$ et $\mathcal{C}$) résolus par ILOG CPLEX, et deux heuristiques ($\mathcal{H}_1$ et $\mathcal{H}_2$). Il existe en outre une cinquième approche

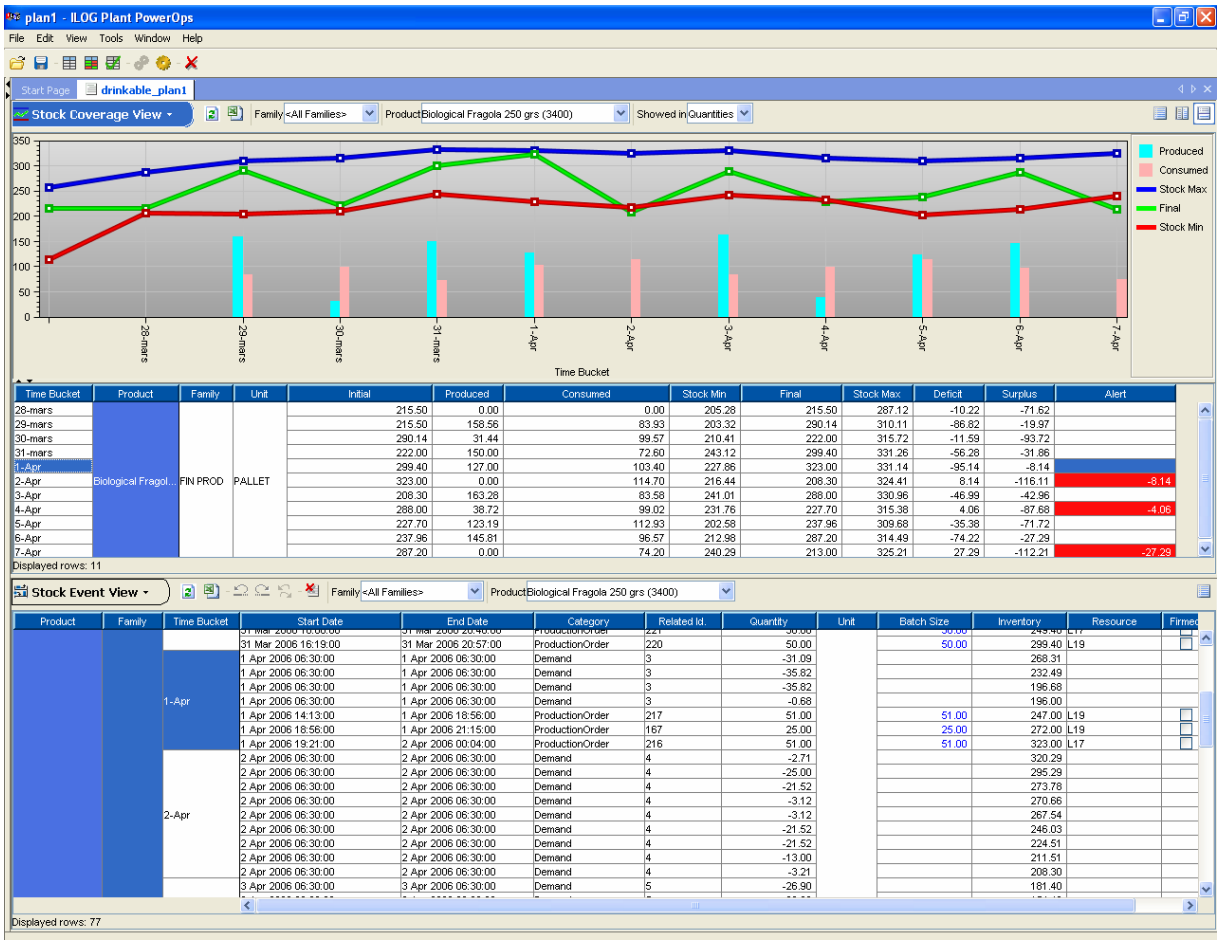

FIG. 7.1 – Visualisation de l'évolution des niveaux de stocks à partir du plan de production

basée sur la programmation par contraintes (PPC). Récemment développée, celle-ci n'est, pour le moment, pas suffisamment élaborée pour pouvoir être testée comme il a été fait pour les quatre autres méthodes. Dans cette dernière approche, le modèle de contraintes est résolu par ILOG Solver. Bien qu'interchangeables en théorie, ces cinq méthodes ne sont pas équivalentes en pratique, nous l'avons vu au Chapitre 6. Certaines contraintes, mais surtout les critères d'optimisation, sont mal pris en compte par les algorithmes les plus simples, voire totalement ignorés ; en revanche, ces derniers sont en général plus rapides à s'exécuter. Nous l'avons montré, aucune méthode n'est apparue comme ostensiblement plus performante et le choix de l'algorithme dépend du problème à traiter. Enfin, l'ordonnancement est résolu par une seule méthode : une modélisation du problème est réalisée au moyen de ILOG Scheduler et le modèle de contraintes est résolu par ILOG Solver. Des heuristiques spécifiques de recherche, basées sur les méthodes de voisinages, adaptées à la problématique générale, y sont ajoutées afin d'accélérer la convergence de la résolution et d'atteindre des solutions de très bonne qualité. De façon très simplifiée, la

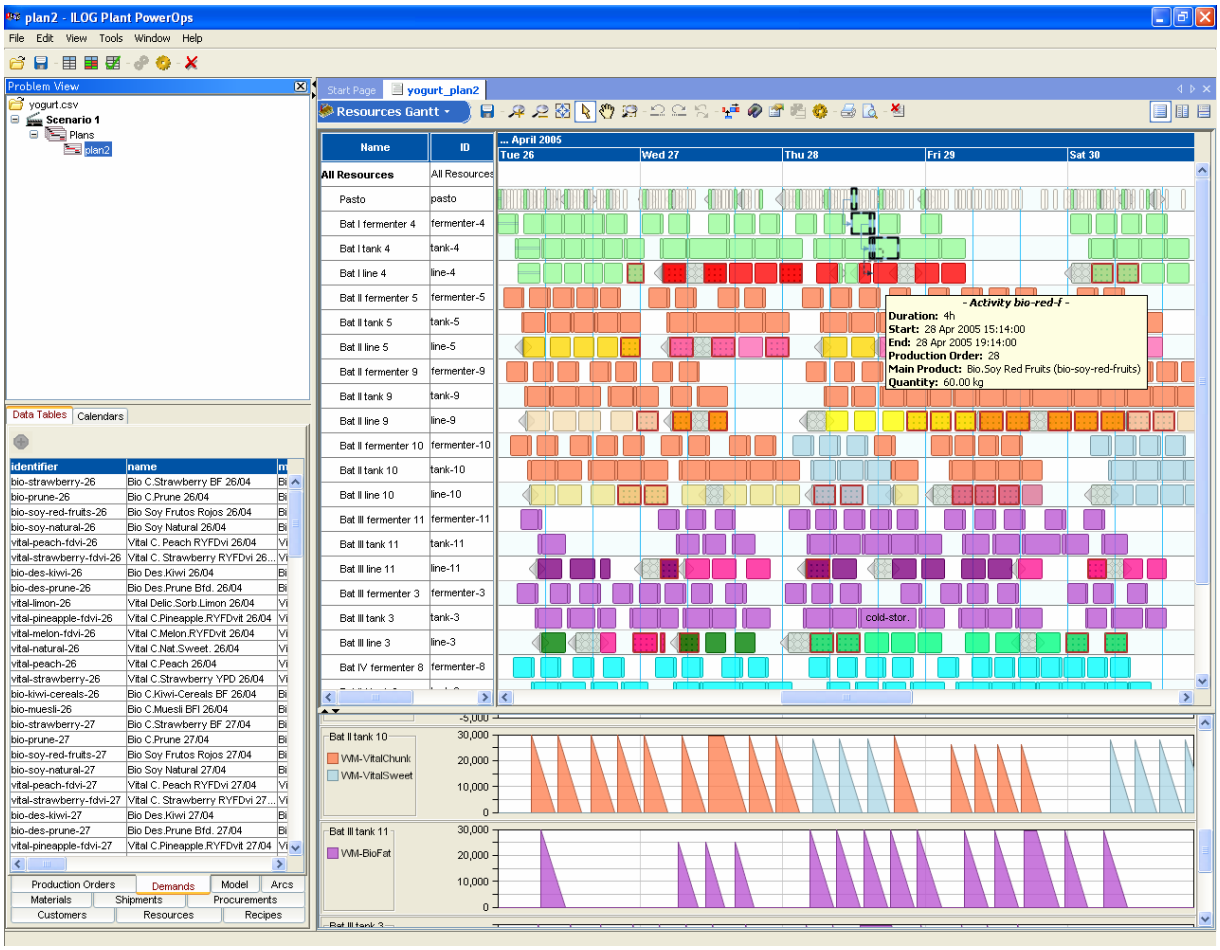

FIG. 7.2 – Diagramme de Gantt montrant l'ordonnancement final des activités de production

Figure 7.3 représente l'enchaînement global de résolution au sein de ILOG PPO.

7.1.2.2 Intégration du programme linéaire

Nous avons implanté le modèle linéaire $\mathcal{P}$ élaboré dans cette thèse au sein du logiciel ILOG PPO afin de l'intégrer totalement au système. Il constitue un algorithme possible pour le *batching* au même titre que les quatre autres, s'intercalant entre planification et ordonnancement de manière tout aussi transparente. Les exigences liées à la mise en application dans des situations réelles de cet algorithme ont été ajoutées au fur et à mesure de son élaboration, de façon à ce qu'il maintienne une cohérence totale avec le reste de l'application. Ces exigences sont en général liées à la robustesse de l'algorithme. En effet, faire fonctionner un tel programme sur des problèmes réels, avec des instances directement issues d'usines appartenant à l'industrie de production présente des difficultés que l'on ne retrouve pas lorsque l'on résout des "cas d'école".

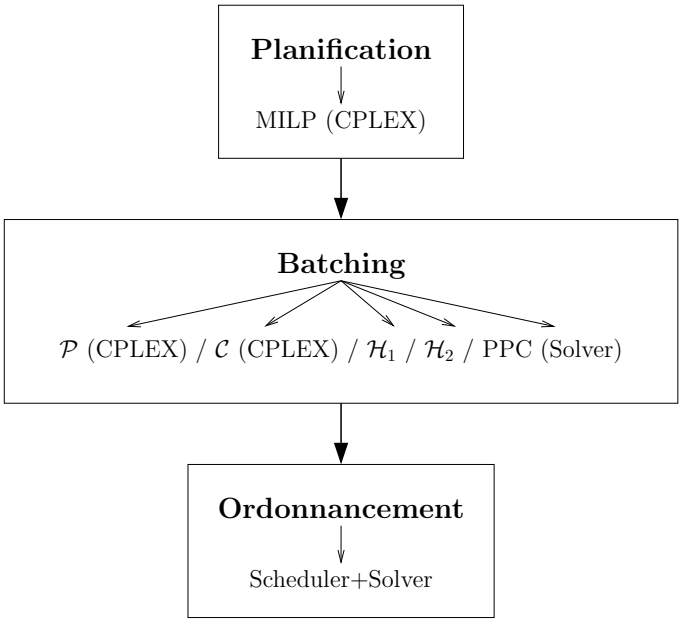

FIG. 7.3 – Structure trimodulaire de ILOG *Plant PowerOps*

Dans les cas pratiques, les données sont souvent incohérentes, certaines décisions peuvent être déjà fixées, et même si celles-ci violent des contraintes, il faut les prendre en compte (par exemple en relaxant ces contraintes violées). La robustesse que nous avons ajoutée à notre programme a permis de tester ses performances sur plusieurs problèmes industriels. Dans la section qui suit, nous présentons un cas sur lequel nous avons travaillé en profondeur et qui a permis de mettre en évidence la flexibilité du programme en intégrant de nouvelles extensions spécifiques à l'industrie de *process*.

7.2 Cas d'étude en industrie laitière

Dans cette section nous montrons comment le programme linéaire élaboré dans nos travaux de thèse a pu être adapté afin de résoudre des problèmes réels. Nous décrivons dans un premier temps le problème industriel qui nous a intéressé en identifiant les caractéristiques qui lui sont spécifiques. Nous présentons dans un second temps les contraintes mathématiques que nous avons formulées et intégrées au MILP d'origine dans le but de modéliser ces caractéristiques.

7.2.1 Analyse des procédés de transformation

Le cas étudié se place dans l'environnement particulier de l'agroalimentaire, et plus précisément, celui de l'élaboration de produits frais. Dans le contexte que nous étudions ici, il s'agit d'optimiser la production de produits laitiers dans une usine de grande taille : fabrication et approvision-

nement de 300 000 tonnes de produits laitiers par an. Ces produits se répartissent en trois catégories : yaourts, fromages et desserts. À chaque type de produits correspond un procédé (ou *process*) de fabrication faisant intervenir différents aspects techniques fondamentaux non-modélisés jusqu'alors dans notre formulation mathématique.

7.2.1.1 Principe général

Nous avons mentionné dans la Section 1.3.1 du Chapitre 1 certains aspects propres à l'industrie de *process*. Nous les retrouvons ici dans le cadre d'une application concrète :

- présence de "tanks" dans lesquels sont emmagasinés les produits intermédiaires où s'effectuent des modification d'état : préparation, fermentation, stockage ou cuisson ;
- présence de lignes de remplissage ; ressources exécutant des opérations de conditionnement des produits finis ;
- présence de "machines continues" sur lesquelles transitent les produits pour modifier à nouveau leur état : refroidissement, stérilisation, pasteurisation.
- opérations de nettoyage des ressources (dépendant de la séquence d'activités ou non) ;
- opérations de setup sur les ressources (dépendant de la séquence d'activités) ;
- spécificités dues à la nature de certains produits : durée de maturation nécessaire avant de pouvoir être utilisés, délai de périssabilité très contraignant par rapport à l'horizon ; les contraintes associées à ces caractéristiques doivent impérativement être satisfaites dans l'ordonnancement et sont difficiles à respecter car elles diminuent fortement le degré de flexibilité (et l'espace des solutions réalisables).

7.2.1.2 Particularités de l'usine considérée

Production et consommation dans les tanks. Nous avons mentionné dans le Chapitre 1 qu'il est possible de rencontrer des machines produisant ou consommant les matériaux de manière discrète ou continue, spécialement dans le contexte de l'industrie de *process*. C'est typiquement le cas des tanks utilisés dans l'usine que nous étudions : ils sont remplis en continu et, en fonction du processus en cours, peuvent être vidés en une fois ou de manière continue.

Lignes de remplissage et de conditionnement. Ces ressources, appelées "lignes", sont utilisées pour le remplissage des pots de yaourts ou des bouteilles de lait. Elles sont en général remplies en continu (les produits proviennent des tanks) et fabriquent leur produit fini de manière discrète (par exemple par palettes de yaourts).

Nombre de producteurs et de consommateurs. Nous avons introduit au Chapitre 3, lors de l'étude de complexité des problèmes, la possibilité de contraindre qu'une demande ne soit satisfaite que par un unique batch. Nous avons désigné cette contrainte par le terme de mono-*pegging* (par opposition au multi-*pegging* qui ne limite pas le nombre de batches). Cette extension est ici généralisée à tous les niveaux de la chaîne de production. Lorsqu'un batch d'un

produit intermédiaire donné est déversé dans un tank, il se peut qu'une contrainte physique ou sanitaire interdise de déverser un nouveau batch tant que le premier n'est pas intégralement consommé. Symétriquement, lorsqu'une certaine quantité de produit stockée dans un tank doit être consommée par un batch en aval, il peut être imposé que l'utilisation du produit se fasse en un unique pompage. Ces contraintes dites "de cardinalité de *pegging*" peuvent être reformulées de la façon suivante : un batch donné ne peut produire pour plus d'un batch consommateur, et/ou vice-versa. Notons que c'est l'identification de cette spécificité technique qui nous a conduit, au Chapitre 3, à poser la question de la difficulté théorique induite par cette contrainte.

Hétérogénéité des ressources. Dans l'usine étudiée, plusieurs machines peuvent exécuter la même opération, notamment les lignes de remplissage. Ce principe a été étudié dans les extensions présentées au Chapitre 5 (Section 5.2) et correspond à la notion de modes alternatifs pour l'exécution d'une activité. Dans le cas où ces ressources sont équivalentes, l'agrégation des machines en une seule machine à grande capacité est envisageable ; par ailleurs, les décisions d'affectation de ressource prises par la planification ne sont pas critiques. En revanche, dans le cas où une ressource est beaucoup plus lente que les autres (par exemple), les ressources ne sont pas équivalentes et une idée précise de l'affectation semble nécessaire pour pouvoir évaluer correctement les temps d'exécution, et ainsi respecter réellement les contraintes cruciales liées à la péremption des produits.

7.2.2 Modélisation des contraintes spécifiques

7.2.2.1 Production et consommation continue ou discrète

Jusqu'à présent, nous faisions l'hypothèse que la production et la consommation avaient lieu de manière discrète. Dans ce contexte, il est clair que si deux batches $bat^t_{i,b}$ et $bat^{t'}_{i',b'}$ sont liés par un arc de *pegging*, alors $bat^{t'}_{i',b'}$ ne peut commencer avant que $bat^t_{i,b}$ n'ait été complètement exécuté. Ainsi, les contraintes (5.17) présentées au Chapitre 5 en Section 5.4 (rappelées ci dessous), reliant la date estimée de fin $eet^t_{i,b}$ du batch producteur à la date estimée de début $est^{t'}_{i',b'}$ du batch consommateur (via les variables semi-continues $zp^{t,t'}_{i,b,i',b'}$ et $zc^{t,t'}_{i,b,i',b'}$) sont bien nécessaires et suffisantes à l'expression des précédences entre batches producteurs et consommateur (cf. Figure 7.4).

$$\forall t, t', \forall i, i', \forall b, b', \qquad zc^{t,t'}_{i,b,i',b'} - zp^{t,t'}_{i,b,i',b'} \geq 0$$

Lorsque $bat^t_{i,b}$ et $bat^{t'}_{i',b'}$ s'exécutent de manière continue, il n'est plus aussi simple d'exprimer la précédence temporelle que leurs dates d'exécution doivent satisfaire. Dans un premier temps, nous présentons les contraintes que nous avons formulées dans notre programme linéaire afin de modéliser des précédences plus souples que les contraintes (5.17). Cependant, ces nouvelles contraintes sont encore trop fortes et n'expriment pas la relation de précédence exacte qui doit

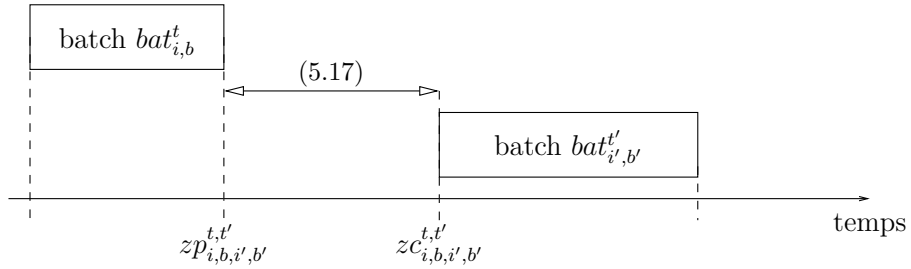

FIG. 7.4 – Précédence en cas de production et consommation disrètes

être établie entre $bat^t_{i,b}$ et $bat^{t'}_{i',b'}$. Nous proposons dans un second temps une formulation exacte mais complexe (et non mise en œuvre) de la contrainte de précedence réelle.

Modélisation "surcontrainte". Dans ce cas, deux contraintes linéaires sont posées (cf Figure 7.5). D'une part, on impose que le batch consommateur $bat^{t'}_{i',b'}$ commence après que le batch producteur $bat^t_{i,b}$ a démarré sa production (cette contrainte relie les deux dates de début $est^t_{i,b}$ et $est^{t'}_{i',b'}$). La contrainte (5.17) fait intervenir les variables semi-continues $zp^{t,t'}_{i,b,i',b'}$ (qui correspondent à la date de fin estimée du batch $bat^t_{i,b}$) et $zc^{t,t'}_{i,b,i',b'}$ (qui correspondent à la date de début estimée du batch $bat^{t'}_{i',b'}$). L'extension au cas totalement continu impose de retrancher la durée $p^t_{i,b}$ du batch $bat^t_{i,b}$, producteur, à la variable $zp^{t,t'}_{i,b,i',b'}$ afin de considérer la date de début. La contrainte (7.1) modélise alors la précédence entre les dates de début. Cette inégalité établit que s'il existe un flot non-nul entre $bat^t_{i,b}$ et $bat^{t'}_{i',b'}$, alors la précédence temporelle est bien posée. Dans le cas contraire, la contrainte résultante $p^t_{i,b} \geq 0$ est toujours vraie. D'autre part, on interdit que le batch consommateur s'exécute plus rapidement que le producteur et un seconde précédence est cette fois-ci imposée sur les dates de fin. Dans la nouvelle contrainte (7.2), on augmente donc la date de début $zc^{t,t'}_{i,b,i',b'}$ du batch $bat^{t'}_{i',b'}$ de sa durée $p^{t'}_{i',b'}$:

$$\forall t, t', \forall i, i', \forall b, b', \qquad zc^{t,t'}_{i,b,i',b'} - (zp^{t,t'}_{i,b,i',b'} - p^t_{i,b}) \geq 0 \tag{7.1}$$

$$(zc^{t,t'}_{i,b,i',b'} + p^{t'}_{i',b'}) - zp^{t,t'}_{i,b,i',b'} \geq 0 \tag{7.2}$$

Modélisation exacte. Les contraintes ci-dessus n'expriment pas avec exactitude la réalité. En effet, d'une part, si $bat^{t'}_{i',b'}$ est lié à plusieurs batches producteurs, son exécution peut commencer avant le début de $bat^t_{i,b}$ et exprimer une précédence entre leurs dates de début est de ce fait trop contraignant. D'autre part, $bat^{t'}_{i',b'}$ peut aussi se terminer avant la fin de $bat^t_{i,b}$ — ce dernier est alors relié à d'autres batches consommateurs s'exécutant plus tard —, en ce cas, exprimer une contrainte entre les dates de fin de $bat^t_{i,b}$ et de $bat^{t'}_{i',b'}$ est aussi trop fort. La contrainte réelle qui exprime les précédences entre $bat^t_{i,b}$ et $bat^{t'}_{i',b'}$ dans le contexte totalement continu fait intervenir

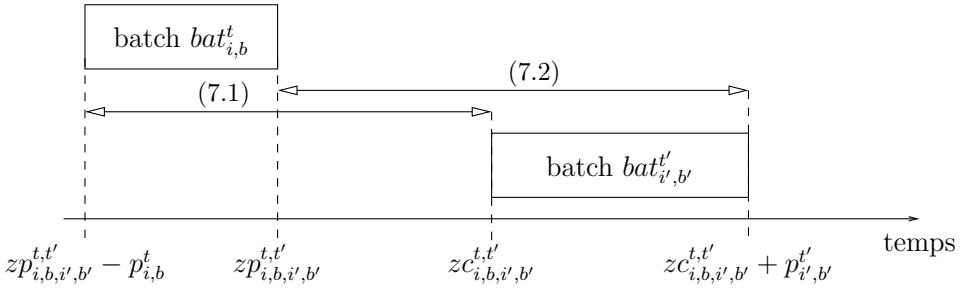

FIG. 7.5 – Précédence en cas de production et consommation contnues

la vitesse de production pr_i de $pmat_i$ par un batch $bat^t_{i,b}$ de rec_i et la vitesse de consommation $cr_{i'}$ de $cmat_{i'}$ par un batch $bat^{t'}_{i',b'}$ de $rec_{i'}$. On définit alors les fonctions $f^t_{i,b}(\tau)$ et $g^{t'}_{i',b'}(\tau)$ désignant respectivement la production du batch $bat^t_{i,b}$ et la consommation du batch $bat^{t'}_{i',b'}$ au cours du temps. Ces fonctions se définissent ainsi :

$$f^t_{i,b}(\tau) = \begin{cases} 0 & \text{si } \tau \leq est^t_{i,b} \\ pr_i(\tau - est^t_{i,b}) & \text{si } est^t_{i,b} \leq \tau \leq eet^t_{i,b} \\ pr_i(eet^t_{i,b} - est^t_{i,b}) = q^t_{i,b} & \text{si } \tau \geq eet^t_{i,b} \end{cases}$$

$$g^{t'}_{i',b'}(\tau) = \begin{cases} 0 & \text{si } \tau \leq est^{t'}_{i',b'} \\ cr_{i'}(\tau - est^{t'}_{i',b'}) & \text{si } est^{t'}_{i',b'} \leq \tau \leq eet^{t'}_{i',b'} \\ cr_{i'}(eet^{t'}_{i',b'} - est^{t'}_{i',b'}) = q^{t'}_{i',b'} & \text{si } \tau \geq eet^{t'}_{i',b'} \end{cases}$$

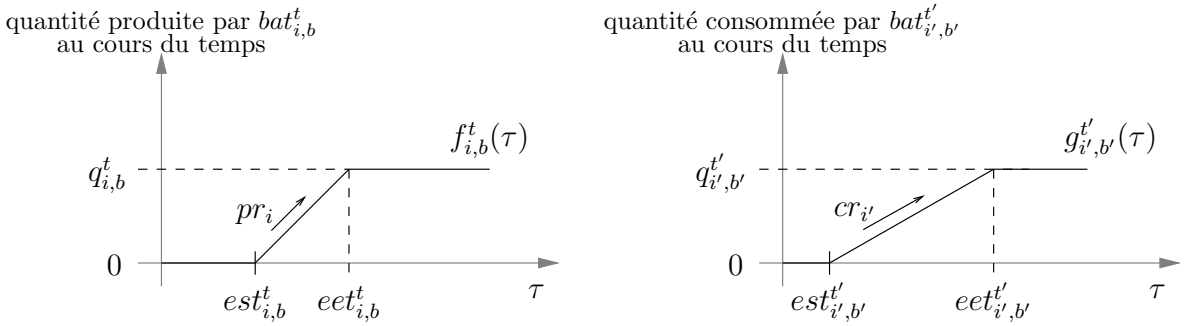

FIG. 7.6 – Fonctions de production et de consommation des batches $bat^t_{i,b}$ et $bat^{t'}_{i',b'}$

La contrainte exacte consiste alors en la détermination d'une fonction $h^{t,t'}_{i,b,i'b'}(\tau)$, pour tout i, i', τ, croissante au sens large, exprimant le transfert de produit entre les batches $bat^t_{i,b}$ et $bat^{t'}_{i',b'}$.

$h^{t,t'}_{i,b,i'b'}(\tau)$ est telle que :

$$\forall \tau, \qquad \sum_{t',i',b'} h^{t,t'}_{i,b,i'b'}(\tau) \leq f^{t}_{i,b}(\tau) \tag{7.3}$$

$$\forall \tau, \qquad \sum_{t,i,b} h^{t,t'}_{i,b,i'b'}(\tau) \geq g^{t'}_{i',b'}(\tau) \tag{7.4}$$

L'inégalité (7.3) impose que la quantité transférée ne dépasse pas la quantité produite. Symétriquement, l'inégalité (7.4) impose que la quantité transférée soit au moins aussi grande que la quantité consommée. En outre, la quantité transférée est nulle au moins tant que le batch producteur $bat^{t}_{i,b}$ n'a pas commencé son exécution (7.5) ; elle est égale au flot de produit $f^{t,t'}_{i,b,i'b'}$, au plus tard lorsque le batch consommateur $bat^{t'}_{i',b'}$ a terminé la sienne (7.6) :

$$\forall \tau \leq est^{t}_{i,b}, \qquad h^{t,t'}_{i,b,i'b'}(\tau) = 0 \tag{7.5}$$

$$\forall \tau \geq eet^{t'}_{i',b'}, \qquad h^{t,t'}_{i,b,i'b'}(\tau) = f^{t,t'}_{i,b,i'b'} \tag{7.6}$$

Cela permet de relier la date de début $est^{t}_{i,b}$ du batch producteur $bat^{t}_{i,b}$ à la date de fin $eet^{t'}_{i',b'}$ du batch consommateur $bat^{t'}_{i',b'}$: les inégalités (7.7) et (7.8) permettent de calculer une date minimale pour la fin du consommateur, compte tenu des vitesses de production pr_i de $bat^{t}_{i,b}$ et de consommation $cr_{i'}$ de $bat^{t'}_{i',b'}$, et de la quantité $f^{t,t'}_{i,b,i'b'}$ à transférer :

$$est^{t}_{i,b} + \frac{f^{t,t'}_{i,b,i'b'}}{pr_i} \quad \leq \quad eet^{t'}_{i',b'} \tag{7.7}$$

$$est^{t}_{i,b} + \frac{f^{t,t'}_{i,b,i'b'}}{cr_{i'}} \quad \leq \quad eet^{t'}_{i',b'} \tag{7.8}$$

Dans le cas général, la fonction $h^{t,t'}_{i,b,i'b'}(\tau)$ n'est pas nécessairement régulière, en pratique, on peut imaginer que la vitesse de transfert entre $bat^{t}_{i,b}$ et $bat^{t'}_{i',b'}$ varie au cours du temps, on ne peut alors rien dire de plus au sujet de $h^{t,t'}_{i,b,i'b'}(\tau)$. On peut s'intéresser au cas où la fonction de transfert $h^{t,t'}_{i,b,i'b'}(\tau)$ est linéaire et que la vitesse de transfert entre $bat^{t}_{i,b}$ et $bat^{t'}_{i',b'}$ est constante et égale à $tr_{i,i'}$. En introduisant les nouvelles variables $est^{t,t'}_{i,b,i'b'}$ et $eet^{t,t'}_{i,b,i'b'}$ désignant respctivement l'instant où commence le transfert et celui où il se termine, la fonction $h^{t,t'}_{i,b,i'b'}(\tau)$ s'écrit de la façon suivante :

$$h^{t,t'}_{i,b,i'b'}(\tau) = \begin{cases} 0 & \text{si } \tau \leq est^{t,t'}_{i,b,i'b'} \\ tr_{i,i'}(\tau - est^{t,t'}_{i,b,i'b'}) & \text{si } est^{t,t'}_{i,b,i'b'} \leq \tau \leq eet^{t,t'}_{i,b,i'b'} \\ f^{t,t'}_{i,b,i'b'} & \text{si } \tau \leq eet^{t,t'}_{i,b,i'b'} \end{cases}$$

$$\text{avec } tr_{i,i'} = \frac{f^{t,t'}_{i,b,i'b'}}{eet^{t,t'}_{i,b,i'b'} - est^{t,t'}_{i,b,i'b'}}$$

Les contraintes (7.3) et (7.4) peuvent alors être réécrites en faisant intervenir les variables $est^{t,t'}_{i,b,i'b'}$ et $eet^{t,t'}_{i,b,i'b'}$. En effet, il suffit d'écrire qu'en tout point du type $est^{t}_{i,b}$, $eet^{t}_{i,b}$, $est^{t'}_{i',b'}$, $eet^{t'}_{i',b'}$, $est^{t,t'}_{i,b,i'b'}$ et $eet^{t,t'}_{i,b,i'b'}$, les contraintes (7.3) et (7.4) sont vérifiées. (7.7) et (7.8) deviennent alors respectivement :

$$est^{t,t'}_{i,b,i'b'} \geq est^{t}_{i,b} \tag{7.9}$$

$$eet^{t,t'}_{i,b,i'b'} \leq eet^{t'}_{i',b'} \tag{7.10}$$

Dans le cas où le batch $bat^{t}_{i,b}$ produit de manière continue et le batch $bat^{t'}_{i',b'}$ consomme de manière discrète, on a :

$$\forall \tau \geq est^{t'}_{i',b'}, \qquad h^{t,t'}_{i,b,i'b'}(\tau) = f^{t,t'}_{i,b,i'b'} \tag{7.11}$$

Inversement, dans le cas où le batch $bat^{t}_{i,b}$ produit de manière discrète et le batch $bat^{t'}_{i',b'}$ consomme de manière continue, on a :

$$\forall \tau < eet^{t}_{i,b}, \qquad h^{t,t'}_{i,b,i'b'}(\tau) = 0 \tag{7.12}$$

7.2.2.2 Délai sur la consommation : maturation et périssabilité

Dans l'industrie agroalimentaire et en particulier lorsque les matériaux manipulés sont des produits frais, un délai de maturation peut être nécessaire entre la production et la consommation. Si le matériau mat_m possède un délai de maturation δ_m cela implique d'introduire un terme supplémentaire dans la définition de la contrainte de la précédence (5.17) définie en Section 5.4 du Chapitre 5. On formule alors cette contrainte par l'expression (7.13). Lorsque le matériau mat_m présente une durée Δ_m maximale durant laquelle il peut être stocké, après laquelle les durées de péremption ne sont plus respectées, on doit ajouter une contrainte bornant supérieurement le délai entre production et consommation (7.14) (cf. Figure 7.9). Notons que cette dernière restriction rend le problème intrinsèquement plus difficile. En effet, d'une manière générale en ordonnancement, lorsque les activités ne peuvent être retardées indéfiniment le long de l'horizon, l'existence d'une solution réalisable est un problème difficile ; notre modèle de *batching* s'apparentant à ce type de problème, l'ajout de ces contraintes augmente de la même façon sa difficulté de résolution.

$$\forall t, t', \forall i, i', \forall b, b', \qquad zc^{t,t'}_{i,b,i',b'} - zp^{t,t'}_{i,b,i',b'} \geq \delta_m y^{t,t'}_{i,b,i',b'} \tag{7.13}$$

$$zc^{t,t'}_{i,b,i',b'} - zp^{t,t'}_{i,b,i',b'} \leq \Delta_m \tag{7.14}$$

7.2.2.3 Cardinalité de *pegging*

Concernant la restriction du nombre de batches producteurs à N_d pour la demande dem_d (la contrainte de mono-*pegging* s'exprime en posant $N_d = 1$), elle s'exprime en bornant le nombre d'arcs de *pegging* arrivant sur la demande dem_d (7.15) :

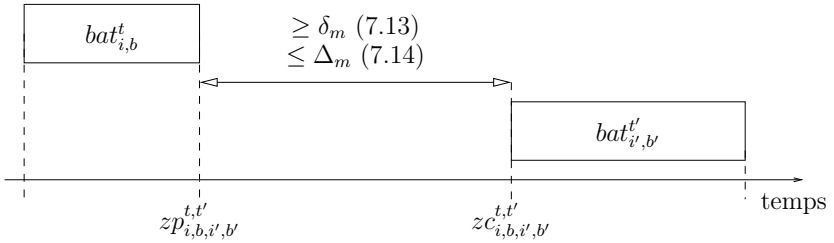

FIG. 7.7 – Délai de maturation et périssabilité du matériau transféré entre $bat^{t}_{i,b}$ et $bat^{t'}_{i',b'}$

$$\forall d, \qquad \sum_{t}\sum_{i}\sum_{b} y^{t}_{i,b,d} \leq N_d \tag{7.15}$$

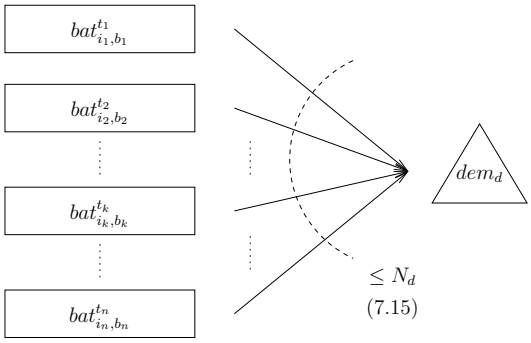

FIG. 7.8 – Limitation du nombre de batches servant la demande dem_d

Les contraintes limitant le nombre de batches consommateurs et producteurs, au niveau d'un batch $bat^{t}_{i,b}$ de la recette rec_i à respectivement N^{out}_i et N^{in}_i, s'expriment de façon similaire par (7.16) et (7.17) :

$$\forall t, \forall i, \forall b, \qquad \sum_{t'}\sum_{i'}\sum_{b'} y^{t,t'}_{i,b,i',b'} \leq N^{out}_i \tag{7.16}$$

$$\sum_{t'}\sum_{i'}\sum_{b'} y^{t',t}_{i',b',i,b} \leq N^{in}_i \tag{7.17}$$

Il convient de rappeler que ce type de contraintes rend le problème beaucoup plus difficile du point de vue de la complexité, ce qui se vérifie lors sa résolution. En effet, la structure intrinsèque de flot qui était jusqu'à présent conservée dans le modèle ne peut se retrouver ici. Lorsque cette contrainte est définie dans le modèle de *batching*, on optimise en général une

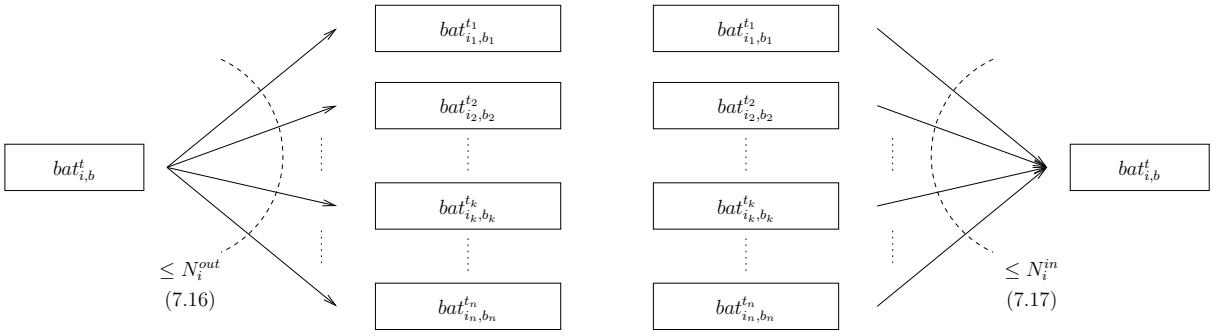

FIG. 7.9 – Limitation du nombre de consommateurs ou de producteurs de $bat^t_{i,b}$

fonction objectif restreinte aux coûts de production et l'obtention d'une solution réalisable est déjà un bon résultat. Cette simplification est tout à fait raisonnable dans la mesure où les solutions où l'on crée des batches de grande taille (donc un minimum de batches) sont préférées dans la pratique par les clients.

7.2.2.4 Ressources alternatives hétérogènes

Au niveau du *batching*, l'affectation des ressources provient des décisions, souvent optimistes, du plan de production, et ne sont pas remises en cause. Ainsi, la durée d'exécution d'un batch peut être sous-estimée. Le module de planification, en approximant les contraintes de capacité, peut avoir affecté à une ressource, une quantité à exécuter qui ne soit pas réalisable en cas de présence de délai borné sur la périssabilité des produits. Dans le meilleur des cas, l'ordonnancement devra alors modifier l'affectation de ressource estimée en amont. Dans le pire des cas, il n'y aura pas de solution réalisable au problème. Ainsi, si l'on ne prend en compte que les durées d'exécution, au niveau du *batching*, on peut considérer, à tort, qu'une certaine quantité de production pourra être consommée dans un délai non-réalisable en pratique.

Considérons un batch $bat^t_{i,b}$ produisant le matériau $pmat_i = mat_m$ dont le délai de périssabilité Δ_m est limité. Plus la taille $q^t_{i,b}$ de ce batch est élevée, plus le temps que prendra sa consommation sera long. Or, il se trouve qu'un tel batch peut être dimensionné à sa taille maximale et consommé intégralement sans violer la contrainte de péremption dans le cas où le batch consommateur est exécuté par une ressource "rapide". En revanche, si celui-ci est exécuté par une ressource "lente", la consommation d'un batch de taille maximale prendra un temps dépassant la durée Δ_m, ce qui doit être interdit. On ne peut donc pas contraindre explicitement la taille $q^t_{i,b}$ du batch $bat^t_{i,b}$ à être inférieure à une valeur strictement plus petite que BS_i. Nous allons alors, sur chaque ressource res_r pouvant exécuter un batch $bat^{t'}_{i',b'}$ consommateur de $cmat_{i'} = mat_m$, borner la quantité totale produite par $bat^t_{i,b}$, consommée sur cette ressource. Il s'agit donc, pour chaque ressource res_r, de considérer le flot de produit transféré entre $bat^t_{i,b}$ et tout batch $bat^{t'}_{i',b'}$ exécuté sur res_r, et de borner la durée de consommation par Δ_m (cf. Figure 7.10). On obtient alors la

contrainte (7.18) suivante pour la modélisation des ressources hétérogènes :

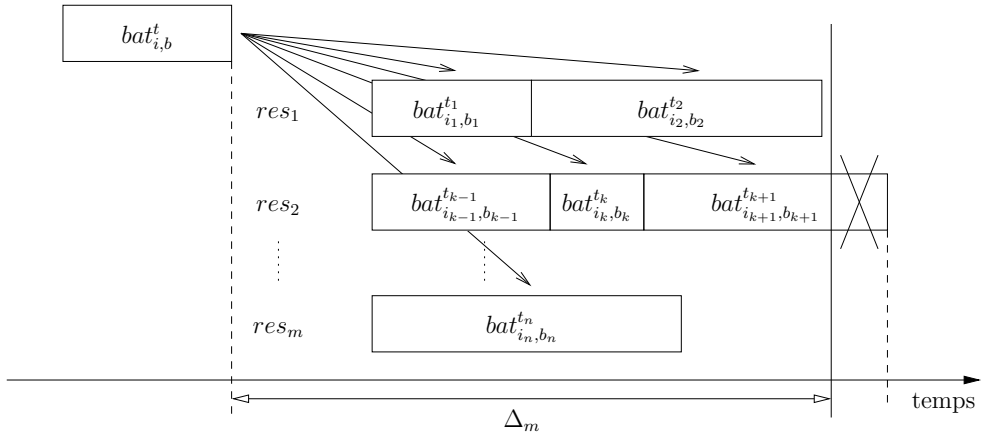

FIG. 7.10 – Limitation de la quantité consommée sur les ressources lentes

$$\forall t, \forall i, \forall b, \forall r, \qquad \sum_{t'} \sum_{i' \backslash mres_{i'} = res_r} \sum_{b'} vp_{i'} f^{t,t'}_{i,b,i',b'} cq_{i'} \leq \Delta_m \tag{7.18}$$

7.2.3 Synthèse

L'étude d'un cas industriel qui s'ancre dans le cadre d'un projet en cours à ILOG a permis de soumettre notre approche à la résolution de problèmes réels. L'analyse des spécificités fonctionnelles qui ont pu être identifiées a conduit à la formulation de nouvelles contraintes. Cela a démontré les capacités d'extensibilité et de flexibilité du programme linéaire élaboré au cours de nos travaux de thèse.

Concernant les résultats numériques, des données ont été transmises par les industriels de l'usine en question et toutes les instances des trois types de procédés ont pu être traitées par PPO (planification, *batching* utilisant notre formulation étendue, ordonnancement) menant à des solutions admissibles vis-à-vis des contraintes formulées par les clients. Le jeu de données, constitué d'un grand nombre d'instances (15 par *process*, soit un total de 60 instances) était sans cesse renouvelé par de nouvelles données, puisque les expérimentations se faisaient en parallèle de l'intégration de PPO dans le système existant dans l'usine. Les données variaient donc quotidiennement. Cela a permis d'attester de la robustesse du modèle de *batching* dans le cadre d'une utilisation réelle et quotidienne.

Néanmoins, les solutions obtenues pour l'ordonnancement, bien qu'admissibles du point de vue des contraintes, n'étaient pas satisfaisantes du point de vue de la fréquence des nettoyages et des setups. Ces défauts, plus du ressort du module d'ordonnancement (qui optimise explicitement ces contraintes), provenaient en réalité d'une résolution trop optimiste du *batching* en amont.

Ainsi, à l'heure actuelle, une décomposition spécifique a été mise place : une bonne séquence pour les setups et les activités de nettoyage est d'abord calculée, puis le processus de résolution séquentiel de planification, *batching* et ordonnancement est alors exécuté. Dans cette procédure, l'algorithme de *batching* utilisé est une heuristique dédiée.

Chapitre 8

Conclusions et perspectives

Synthèse générale

La problématique générale dans laquelle s'inscrit cette thèse est celle de la gestion de la chaîne d'approvisionnement (*Supply Chain Management*), dans le contexte particulier de l'industrie de *process*. Nous avons effectué nos travaux dans le cadre du développement de l'APS *Plant PowerOps* (PPO), au sein de la société ILOG S.A. Notre travail a consisté à définir, analyser et résoudre un nouveau problème d'optimisation s'intercalant entre les niveaux de décisions tactiques (planification globale de la production) et opérationnelles (ordonnancement des activités de production sur les ressources). Il s'agit de décider du nombre et de la taille des batches de production réellement exécutés dans une usine de production. Au travers de cette thèse, nous avons défini un cadre précis et apporté les premières bases théoriques et pratiques, pour l'étude de ce nouveau problème, qui présente de nombreuses variantes et applications.

Afin de mettre en évidence la problématique de la connexion entre les décisions de planification et celles d'ordonnancement, nous avons tout d'abord dressé un panorama de ces deux questions générales. Par le biais d'études théoriques d'une part, et de solutions logicielles d'autre part, nous avons identifié un niveau intermédiaire de décisions critiques intervenant dans le processus global d'optimisation d'une usine : il s'agit du problème de dimensionnement des batches de production, dit de *batching*. Résolue de manière heuristique dans les APS du marché, notre thèse, au contraire, considère cette question comme un problème d'optimisation à part entière.

Nous avons d'abord explicité ce nouveau problème de manière informelle, afin d'appréhender les difficultés sous-tendues : à partir de la quantité de production planifiée, les batches doivent être définis de façon à inférer des activités devant alors être ordonnancées. Au problème de "découpage" de la production (*lot-streaming*) s'ajoute celui de l'établissement de flots des produits (*pegging*) entre les batches et les demandes, pour finalement définir notre problème — couplé — de *batching*. Dans un second temps, nous avons donné une formulation mathématique du problème global résolu par l'APS PPO. La présentation de l'approche novatrice de résolution en trois étapes (planification, *batching*, ordonnancement), employée dans ce logiciel, a conduit à

la définition d'un cadre formel pour l'étude du problème de *batching* considéré dans cette thèse.

Nous avons alors abordé la question de la complexité de ses principales variantes. Pour cela, nous avons sélectionné certaines caractéristiques majeures du problème (contraintes et critères d'optimisation) qui nous ont permis d'identifier des sous-cas polynomiaux d'une part, et des sous-cas NP-complets au sens fort d'autre part. La conjonction de plusieurs contraintes, faciles à traiter quand elles sont considérées indépendamment les unes des autres, ou l'optimisation de divers critères, peut en effet mener à des problèmes difficiles. Nous n'avons pas fermé la question de la complexité pour toutes les variantes de *batching* considérées dans cette étude. En particulier, deux problèmes d'ordonnancement ouverts ont pu être identifiés : $1||\sum p_i T_i$ et $1|p_i = 1|\sum w_i T_i$ dans le contexte de l'ordonnancement à haute multiplicité (*high multiplicity scheduling*).

Nous nous sommes ensuite intéressés à la résolution de la variante élémentaire du problème de *batching*, que nous avons nommée problème *"cœur"*. Dans un premier temps, nous avons élaboré un algorithme de programmation dynamique pseudopolynomial résolvant un cas particulier du problème. Dans un second temps, un programme linéaire en nombres entiers a été formulé. Le modèle proposé autorise une formulation linéaire exacte du point de vue des contraintes de *batching*, ce qui est en soi un résultat intéressant ; cependant sa fonction objectif est une approximation linéaire de celle que nous avons initialement définie, cette dernière étant quadratique. Nous avons donc réalisé une étude expérimentale permettant de comparer la méthode exacte et la méthode approchée dans laquelle le programme linéaire est résolu par ILOG CPLEX. Ces expérimentations ont révélé un comportement très satisfaisant de la seconde approche puisque la moyenne des erreurs relatives commises est de 2.64 %.

Le programme linéaire proposé pour la résolution du problème *"cœur"* a ensuite été étendu afin de s'adapter à des situations plus complexes et cependant fréquentes en industrie de *process* : variation de la productivité des ressources au cours du temps, ressources alternatives, précédences temporelles, ruptures sur les demandes, prise en compte des coûts de stockage et setup. Nous avons modélisé chacun des ces aspects par de nouvelles contraintes linéaires, venant s'ajouter au programme initial. Cela a, en parallèle, conduit à une classification des différentes variantes des problèmes.

Nous avons ensuite mené une étude expérimentale visant à mesurer les performances du programme linéaire. L'ensemble de jeux de données sur lequel ont été réalisés nos tests a été élaboré au cours de notre thèse ; il est basé sur la classification mentionnée ci-dessus. Dans un premier temps, nous avons comparé six variantes de notre programme linéaire, suivant différentes modélisations des contraintes de capacité des ressource, afin d'identifier la meilleure formulation. Dans un second temps, la méthode de résolution basée sur ce modèle est comparée à trois autres approches : un second programme linéaire en nombres entiers et deux heuristiques. Notre programme s'est alors montré comme la méthode la plus compétitive avec une moyenne d'erreur relative par rapport aux meilleures solutions connues de 10.15 %. Par ailleurs, les deux méthodes issues de notre modèle de *batching* ont présenté des performances significativement meilleures que celles des heuristiques, ce qui confirme l'intérêt de se poser la question du dimensionnement

des batches en terme de problème d'optimisation.

Enfin, l'implantation de notre programme linéaire au sein de l'APS PPO, comme un algorithme de résolution du problème de *batching* à part entière, a rendu possible l'évaluation de son comportement sur des problèmes réels, issus de l'industrie de *process*. Nous avons mené une étude à partir d'un cas appartenant à l'industrie agroalimentaire et plus particulièrement, à la production laitière. Les contraintes de *batching* inhérentes à cet environnement, bien que difficiles à exprimer, ont pu être formulées linéairement, comme des extensions de notre modèle. Cette expérience a permis de montrer que notre approche est flexible et robuste, et il semble tout à fait raisonnable d'envisager d'avoir recours à ce type de modèles pour la résolution de problèmes réels.

Perspectives de travaux

Nous l'avons vu, la complexité de certains problèmes où l'on optimise les coûts quadratiques de retard est restée en suspens. Lorsque le retard relatif à chaque demande a la même importance, le cas multi-*pegging* (plusieurs batches peuvent livrer une demande) avec une taille de batch minimale restrictive d'une part, et le cas mono-*pegging* général ($1||\sum p_i T_i$) d'autre part, sont ouverts. Lorsqu'il existe des poids quelconques sur la livraison des demandes, le cas général multi-*pegging* (avec ou sans taille de batch minimale restrictive) est ouvert. Les études de complexité de ces problèmes font évidemment l'objet de perspectives de recherche intéressantes, sur le plan théorique. Par ailleurs, le rapprochement entre le *batching* tel que défini dans cette thèse, et les problèmes d'ordonnancement par batch est prometteur. En effet, parmi les problèmes couramment étudiés, on trouve par exemple des contraintes de précédences entre les tâches (graphe quelconque ou chaînes) [95, 4, 74, 77], ce qui, dans le contexte du *batching*, revient à exprimer le fait que certaines demandes doivent être livrées impérativement avant d'autres. Jusqu'ici, la nécessité de formuler ce type de contraintes n'a pas été mise en évidence par les cas étudiés dans notre thèse. Par ailleurs, différentes restrictions sur la durée des tâches ont été étudiées : durées toutes égales [4, 8] et durées unitaires [4, 78, 43]. Les équivalents de ces cas particuliers pour le problème de *batching* n'ont pas été examinés dans cette thèse. Il pourrait donc être intéressant d'analyser les correspondances découlant de ces remarques.

Concernant le programme linéaire élaboré dans cette thèse, on peut envisager la reformulation de certaines contraintes afin de comparer différentes modélisations. En outre, à moyen terme, il serait intéressant de fusionner les deux programmes ($\mathcal{P}$ et $\mathcal{C}$) en un unique qui profiterait des qualités de chacune des deux approches. Cela apporterait encore plus de flexibilité et de robustesse à la méthode de résolution.

Par ailleurs, nous envisageons d'étudier la possibilité d'une généralisation de l'algorithme de programmation dynamique à d'autres sous-cas du problème de *batching*, en particulier à la prise en compte des durées fixes associées à la création d'un batch, ainsi que celle des pénalités d'avance. On peut aussi imaginer que cette méthode de résolution intervienne dans une approche basée sur la recherche locale par grands voisinages, qui traiterait un problème général complexe

et dont les sous-problèmes pourraient être résolus par notre méthode exacte.

Enfin, dans une perspective à long terme, on pourrait réviser l'approche séquentielle de la résolution tri-modulaire proposée dans l'APS PPO et envisager une coopération plus étroite entre les modules de *batching* et d'ordonnancement, voire une intégration totale des décisions de *batching* à celles d'ordonnancement (Figure 8.1). On pourrait par exemple envisager de résoudre à tour de rôle le *batching* puis l'ordonnancement, en transmettant des informations complémentaires permettant de produire des solutions de *batching* menant à chaque itération à de meilleurs ordonnancements. Une approche totalement intégrée consisterait à passer directement du module de planification à celui d'ordonnancement, en considérant les tailles de batches et les flots de produits comme des variables au niveau de l'ordonnancement, ce qui, en contrepartie, conduirait à résoudre des problèmes d'ordonnancement significativement plus difficiles.

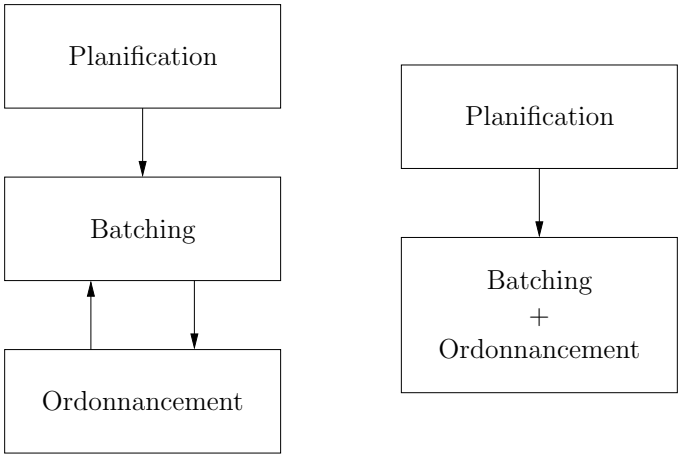

FIG. 8.1 – Résolution coopérative ou intégrée des problèmes de *batching* et d'ordonnancement

Annexe A

Notations mathématiques

Notion	Indice - Ensemble	Attribut	Notation
Ressource :	$\forall res_r \in \mathcal{R}es$	Capacité	cap_r
Matériau :	$\forall mat_m \in \mathcal{M}at$	Stock initial	IS_m
		Stock final	FS_m
		Coût unitaire	stc_m
Recette :	$\forall rec_i \in \mathcal{R}ec$	Taille minimale de batch	bs_i
		Taille maximale de batch	BS_i
		Ensemble d'activités	$\mathcal{A}ct_i$
Activité :	$\forall act_{i,j} \in \mathcal{A}ct_i$	Consommation	$\mathcal{C}m_{i,j}$
		Production	$\mathcal{P}m_{i,j}$
		Ensemble de modes	$\mathcal{M}od_{i,j}$
Mode :	$\forall mod_{i,j,k} \in \mathcal{M}od_{i,j}$	Ressource requise	$mres_{i,j,k}$
		Capacité requise	$mcap_{i,j,k}$
		Partie fixe de durée	$fp_{i,j,k}$
		Partie variable de durée	$vp_{i,j,k}$
		Partie fixe de coût	$fc_{i,j,k}$
		Partie variable de coût	$vc_{i,j,k}$
Consommation :	$\forall cm_{i,j,c} \in \mathcal{C}m_{i,j}$	Matériau consommé	$cmat_{i,j,c}$
		Quantité normalisée consommée	$cq_{i,j,c}$
Production :	$\forall pm_{i,j,p} \in \mathcal{P}m_{i,j}$	Matériau produit	$pmat_{i,j,p}$
		Quantité normalisée produite	$pq_{i,j,p}$
Demande :	$\forall dem_d \in \mathcal{D}em$	Matériau requis	$rmat_d$
		Quantité requise	rq_d
		Date due	dt_d
		Pénalité unitaire d'avance	ec_d
		Pénalité unitaire de retard	tc_d
		Pénalité unitaire de non-livraison	uns_d

TAB. A.1 – Concepts et notations intervenant dans le problème général

Notion	Indice - Ensemble	Attribut	Notation
Ressource :	$\mathcal{R}es = \{res\}$	Capacité	$cap_1 = 1$
Matériau :	$\forall mat_m \in \mathcal{M}at$	Stock initial	$IS_m = 0$
Recette :	$\forall rec_i \in \mathcal{R}ec$	Taille minimale de batch	bs_i
		Taille maximale de batch	BS_i
		Unique activité	$\mathcal{A}ct_i = \{act_i\}$
		Unique mode	$\mathcal{M}od_i = \{mod_i\}$
		Capacité requise	$mcap_i = 1$
		Partie fixe de durée	fp_i
		Partie variable de durée	vp_i
		Partie fixe de coût	fc_i
		Partie variable de coût	vc_i
		Consommation	$\mathcal{C}m_i = \emptyset$
		Production	$\mathcal{P}m_i = \{pm_i\}$
		Matériau produit	$pmat_i$
		Quantité normalisée produite	$pq_i = 1$
Demande :	$\forall dem_d \in \mathcal{D}em$	Pénalité unitaire de non-livraison	$uns_d = \infty$

TAB. A.2 – Notations simplifiées employées pour la définition du problème *"cœur"*

Notion	Indice - Ensemble	Attribut	Notation
Batch :	$bat_{i,b} \in \mathcal{B}at_i, \forall rec_i \in \mathcal{R}ec$	Taille	$q_{i,b}$
		Date de début	$st_{i,b}$
		Date de fin	$et_{i,b}$
		Durée	$p_{i,b}$
Flot de produits :	$\forall rec_i \in \mathcal{R}ec, \forall bat_{i,b} \in \mathcal{B}at_i \forall dem_d \in \mathcal{D}em,$	Quantité transférée	$f_{i,b,d}$

TAB. A.3 – Informations contenues dans une solution générale au problème *"cœur"*

Notion	Indice - Ensemble	Attribut	Notation
Période :	$T_t \in \mathcal{T}$	Date de début	S_t
		Date de fin	E_t

TAB. A.4 – Données complémentaires pour la résolution par ILOG PPO du problème *"cœur"*

Module	Notion	Notation
Planification :	Production totale pour chaque rec_i dans T_t	Q_i^t
	Quantité livrée à chaque dem_d dans T_t	Q_d^t
Batching :	Batches pour chaque rec_i par période T_t :	$bat_{i,b}^t \in \mathcal{B}at_i^t$
	· Taille de batch	$q_{i,b}^t$
	· Date estimée de début	$est_{i,b}^t$
	· Date estimée de fin	$eet_{i,b}^t$
	Flot de produit entre $bat_{i,b}^t$ et dem_d	$f_{i,b,d}^t$
Ordonnancement :	Date de fin pour tout batch $bat_{i,b}$	$et_{i,b}$

TAB. A.5 – Décomposition en 3 étapes, résolution du *batching* par le MILP $\mathcal{P}$

Notion	Variable	Domaine de Définition
lot-streaming		
Création du batch $bat^t_{i,b}$	$x^t_{i,b}$	$\{0,1\}$
Taille du batch $bat^t_{i,b}$	$q^t_{i,b}$	$\mathbb{R}$
pegging		
Flot transféré entre $bat^t_{i,b}$ et dem_d	$f^t_{i,b,d}$	$\mathbb{R}$
Existence d'un arc entre $bat^t_{i,b}$ et dem_d	$y^t_{i,b,d}$	$\{0,1\}$
temps		
Date estimée de début de $bat^t_{i,b}$	$est^t_{i,b}$	$\mathbb{R}$
Date estimée de fin de $bat^t_{i,b}$	$eet^t_{i,b}$	$\mathbb{R}$
Liaison entre la date estimée de fin de $bat^t_{i,b}$ et $y^t_{i,b,d}$	$z^t_{i,b,d}$	$\mathbb{R}$

TAB. A.6 – Variables du MILP $\mathcal{P}$ pour la résolution du problème *"cœur"* de *batching*

Bibliographie

[1] N. Absi. *Modélisation et Résolution de Problèmes de Lot Sizing à Capacité Finie*. PhD thesis, Université Pierre et Marie Curie (Paris 6), 2005.

[2] P. Afentakis and B. Gavish. Optimal lot-sizing for complex product structures. *Operations Research*, 34 :237–249, 1986.

[3] A. Aggarwal and J.K. Park. Improved algorithms for economic lot size problems. *Operations Research*, 41 :549–571, 1993.

[4] S. Albers and P. Brucker. The complexity of one-machine batching problems. *Discrete Applied Mathematics*, 47(2) :87–107, 1993.

[5] H.C. Bahl, LP. Ritzman, and J.N.D. Gupta. Determining lot sizes and resource requirements : a review. *Operations Research*, 35(3) :329–345, 1987.

[6] K.R. Baker. Lot streaming in the two-machine flow-shop with setup times. *Annals of Operations Research*, 57 :1–11, 1995.

[7] K.R. Baker and D. Jia. A comparative study of lot streaming procedures. *OMEGA International Journal of Management Science*, 21(5) :561–566, 1993.

[8] P. Baptiste. Batching identical jobs. *Mathematical Methods of Operation Research*, 52(3) :355–367, 2000.

[9] G. Belvaux and L.A. Wolsey. A library of models and matrices for lot-sizing problems, 1999. URL : ://www.core.ucl.ac.be :16080/wolsey/lotsizel.htm.

[10] G. Belvaux and L.A. Wolsey. bc-prod : A specialized branch-and-cut system for lot-sizing problems. *Management Science*, 46 :724–738, 2000.

[11] G. Belvaux and L.A. Wolsey. Modelling practical lot-sizing problems as mixed integer programs. *Management Science*, 47 :993–1007, 2001.

[12] M. Benisch, A. Greenwald, I. Grypari, R. Lederman, V. Naroditskiy, and M. Tschantz. A supply chain management agent. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 3, pages 1174–1181, 2005.

[13] P.J. Billington, J.O. McClain, and L.J. Thomas. Heuristics for multilevel lot-sizing with a bottleneck. *Management Science*, 32(8) :989–1006, 1986.

[14] G. Bitran and H.H. Yanasse. Computational complexity of capacitated lot-sizing problem. *Management Science*, 28(10) :1174–1186, 1982.

[15] F. Blömer and H.-O. Günther. Scheduling of a multi-product batch process in the chemical industry. *Computers in Industry*, 36 :245–259, 1998.

[16] F. Blömer and H.-O. Günther. LP-based heuristics for scheduling chemical batch processes. *International Journal of Production Research*, 38(5) :1029–1051, 2000.

[17] M. Boudhar and G. Finke. Scheduling on a batch machine with job compatibility. *JORBEL*, 40(1-2) :69–80, 2000.

[18] M. Boudhar and G. Finke. Problème d'ordonnancement de tâches sur une machine à traitement par batch. *Revue RMM*, 2001.

[19] N. Brahimi, S. Dauzère-Pérès, N.M. Najid, and A. Nordli. Single item lot sizing problems. *European Journal of Operational Research*, 168(1) :1–16, 2006.

[20] P. Brucker. *Scheduling Algorithms*. Springer, 2nd edition, 1997.

[21] P. Brucker, A. Gladky, J.A. Hoogeveen, M.Y. Kovalyov, C.N. Potts, T. Tautenhahn, and S.L. van de Velde. Scheduling a batching machine. *Journal of Scheduling*, 1 :31–54, 1998.

[22] P. Brucker and S. Knust. Complexity results of scheduling problems. URL :www.mathematik.uni-osnabrueck.de/research/OR/class/dateien/classes/bat_s/bat_s.

[23] P. Brucker, M.Y. Kovalyov, Y.M. Shafransky, and F. Werner. Batch scheduling with deadlines on parallel machines. *Annals of Operations Research*, 83(0) :23–40, 1998.

[24] W. Bruggemann and Jahnke H. The discrete lot-sizing and scheduling problem : complexity and modification for batch availability. *European Journal of Operational Research*, 124(3) :511–528, 2000.

[25] R.G. Busacker and P.J. Gowen. A procedure for determining a family of minimal cost network flow patterns. O.R.O. Technical report 15, Johns Hopkins University, 1961.

[26] D. Cattrysse, J. Maes, and L.N. Van Wassenhove. Set partitioning and column generation heuristics for capacitated dynamic lot-sizing. *European Journal of Operational Research*, 46 :38–48, 1990.

[27] B. Chaib-draa and J. Müller. *MultiAgent based Supply Chain Management*. Springer, 2006.

[28] V. Chandru, C.-Y. Lee, and R. Uzsoy. Minimizing total completion time on a batch processing machine with job families. *Operations Research Letters*, 13 :61–65, 1993.

[29] B. Chen. A better heuristic for preemptive parallel machine scheduling with batch setup times. *SIAM Journal of Computing*, 22(6) :1303–1318, 1993.

[30] H. Chen, D. Hearn, and C. Lee. A new dynamic programming method for the single item capacitated dynamic lot-sizing model. *Journal of Global Optimization*, 4 :285–300, 1994.

[31] J. Chen and G. Steiner. Approximation methods for discrete lot streaming in flow shop. *Operation Research Letters*, 21(3) :139–145, 1997.

[32] T.C.E. Cheng and M.Y. Kovalyov. Single machine batch scheduling with sequential job processing. *IIE Transactions*, 33(5) :413–420, 2001.

[33] M. Christopher. *Logistics and Supply Chain Management. Strategies for Reducing Cost and Improving Service*. London, 2nd edition, 1998.

[34] S. Croom, Romano P., and M. Giannakis. Supply chain management : an analytical framework for critical literature review. *European Journal of Purchasing and Supply Management*, 6 :67–83, 2000.

[35] S. Dauzère-Pérès and J.B. Lasserre. *An Integrated Approach in Production Planning and Scheduling*. Springer, 1994.

[36] S. Dauzère-Pérès and J.B. Lasserre. Integration of lot sizing and scheduling decisions in a job-shop. *European Journal of Operational Research*, 75(2) :413–426, 1994.

[37] S. Dauzère-Pérès and J.B. Lasserre. Lot streaming in job-shop scheduling. *Operations Research*, 45(4) :584–595, 1997.

[38] S. Dauzère-Pérès and J.B. Lasserre. On the importance of sequencing decisions in production planning and scheduling. *International Transactions in Operational Research*, 9(6) :779–793, 1999.

[39] N. Dellaert, J. Jeunet, and A. Jonard. A genetic algorithm to solve the general multi-level lot-sizing problem with timevarying costs. *International Journal of Production Economics*, 68 :241–257, 2000.

[40] J.T. Dickersbach. *Supply Chain Management with APO*. Springer, 2004.

[41] A. Drexl and K. Haase. Proportional lot-sizing and scheduling. *International Journal of Production Economics*, 40 :73–87, 1995.

[42] A. Drexl and A. Kimms. Lot sizing and scheduling - survey and extensions. *European Journal of Operational Research*, 99(2) :221–235, 1997.

[43] J. Du and J.Y.-T. Leung. Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, 15(3) :483–495, 1990.

[44] G. Dudek and H. Stadtler. Negotiation-based collaborative planning between supply chain partners. *European Journal of Operational Research*, 163(3) :668–687, 2005.

[45] J. Edmonds and R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the A.C.M.*, 19(2) :248–264, 1972.

[46] S.E. Elmaghraby. The economic lot scheduling problem (elsp) : review and extensions. *Management Science*, 24 :587–598, 1978.

[47] J. Esteves and J. Pastor. Enterprise resource planning systems research : an annotated bibliography. *Communications of AIS*, 7(8) :1–52, 2001.

[48] S.E. Fawcett and M.B. Myers. Product and employee development in advanced manufacturing : Implementation and impact. *International Journal of Production Research*, 39 :65–79, 2001.

[49] A. Federgruen and M. Tzur. A simple forward algorithm to solve genral dynamic lot-sizing models with n periods in $O(nlogn)$ or $O(n)$ time. *Management Science*, 37 :909–925, 1991.

[50] B. Fleishmann. The discrete lot-sizing and scheduling problem. *European Journal of Operational Research*, 44 :337–348, 1990.

[51] M.S. Fox, M. Barbuceanu, and Teigen R. Agent oriented supply chain management. *International Journal of Flexible Manufacturing Systems*, 12 :165–188, 2000.

[52] M.R. Garey and D.S. Johnson. *Computers and Intractability ; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[53] C.A. Glass, J.N.D. Gupta, and C.N. Potts. Lot streaming in three-stage production processes. *European Journal of Operational Research*, 75(2) :378–394, 1994.

[54] H.-O. Günther and P. van Beek. *Advanced Planning and Scheduling solutions in Process Industry*. Springer, 2003.

[55] R.E. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of Discrete Mathematics*, 4 :287–326, 1979.

[56] M. Grunow, H.-O. Günther, and M. Lehmann. Campaign planning for multi-stage batch processes in the chemical industry. *OR Spectrum*, 24(3) :281–314, 2002.

[57] D. Gupta and T. Magnusson. The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times. *Computers and Operations Research*, 32 :727–747, 2005.

[58] K. Haase. *Lot sizing and scheduling for production planning*. Springer, Berlin, Germany, 1994.

[59] K. Haase. Capacitated lot-sizing with sequence dependent setup costs. *OR Spectrum*, 18 :51–59, 1996.

[60] N.G. Hall, G. Laporte, E. Selavarajah, and C.. Sriskandarajah. Scheduling and lot streaming in flowshops with no-wait in process. *Journal of Scheduling*, 6 :339–354, 2003.

[61] J.R. Hardin, G.L. Nemhauser, and M.W.P. Savelsbergh. Analysis of bounds for capacitated single-item lot sizing problem. *Computers and Operations Research*, 2005.

[62] A.C. Hax and H.C. Meal. *Hierarchical Integration of Production Planning and Scheduling.* Amsterdam, 1975.

[63] M. He, A. Rogers, E. David, and N.R. Jennings. Designing and evaluating an adaptative trading agent for supply chain management applications. In *IJCAI 2005, Workshop on Trading Agent Design and Analysis*, 2005.

[64] D.S. Hochbaum and D. Landy. Scheduling semiconductor burn-in operations to minimize total flowtime. *Operations Research*, 45(6) :874–885, 1997.

[65] W. Hopp and M. Spearman. *Factory Physics.* McGraw-Hill/Irwin, 2000.

[66] H. Jodlbauer. An approach for integrated scheduling and lot-sizing. *European Journal of Operational Research*, 172(2) :386–400, 2006.

[67] C. Jordan and A. Drexl. Discrete lot-sizing and scheduling by batch sequencing. *Management Science*, 44(5) :698–713, 1998.

[68] J. Kallrath. Planning and scheduling in the process industry. *OR Spectrum*, 24 :219–250, 2002.

[69] B. Karimi, S.M.T. Fatemi Ghomi, and J.M. Wilson. The capacitated lot sizing problem : a review of models and algorithms. *OMEGA International Journal of Management Science*, 31(5) :365–378, 2003.

[70] U. Karmarkar and L. Schrage. The deterministic dynamic product cycling problem. *Operations Research*, 33 :326–345, 1985.

[71] A. Kimms. A genetic algorithm for multi-level, multi-machine lot sizing and scheduling. *Computers and Operations Research*, 26 :829–848, 1999.

[72] E. Kondili, C.C. Pantelides, and R.W.H. Sargent. A general algorithm for short-term scheduling of batch operations - i. milp formulation. In *Proceedings 3rd International Symposium on Process Systems Engeneering*, pages 62–75, 1993.

[73] E.L. Lawler. A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, 1 :331–342, 1977.

[74] E.L. Lawler. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics*, 2 :75–90, 1978.

[75] C. Le Pape and A. Robert. Jeux de données pour l'évaluation d'algorithmes de planification et ordonnancement. In *Actes du 8ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2007)*, pages 283–284, 2007.

[76] C.-Y. Lee, S. Çetinkaya, and A.P.M. Wagelmans. A dynamic lot sizing model with demand time windows. *Management Science*, 47(10) :1384–1395, 2001.

[77] J.K. Lenstra and A.H.G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1) :22–35, 1978.

[78] J.K. Lenstra and A.H.G. Rinnooy Kan. Complexity results for scheduling chains on a single machine. *European Journal of Operational Research*, 4(4) :270–275, 1980.

[79] J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1 :343–362, 1977.

[80] J. Leung, L. Kelly, and J.H. Anderson. *Handbook of Scheduling : Algorithms, Models, and Performance Analysis*. CRC Press, Inc., Boca Raton, FL, USA, 2004.

[81] X. Li and Q. Wang. Coordination mechanisms of supply chain systems. *European Journal of Operational Research*, 179(1) :1–16, 2007.

[82] M. Lütke Entrup. *Advanced Planning in Fresh Food Industries*. Springer, 2005.

[83] T. Lunn and S.A. Neff. *MRP, Integrating Material Requirements Planning and Modern Business*. Irwin, 1992.

[84] J. Maes, J.O. McClain, and L.N. Van Wassenhove. Multilevel capacitated lot sizing complexity and LP-based heuristics. *European Journal of Operational Research*, 53(2) :131–148, 1991.

[85] C.T. Maravelias and I.E. Grossmann. An hybrid MILP/CP decomposition approach for the scheduling of batch plants. In *CPAIOR'03*, 2003.

[86] C.T. Maravelias and I.E. Grossmann. Minimization of the makespan with a discrete-time state-task network formulation. *Industrial & Engineering Chemistry Research*, 42(24) :6252–6257, 2003.

[87] C.T. Maravelias and I.E. Grossmann. New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. *Industrial & Engineering Chemistry Research*, 42(13) :3056–3074, 2003.

[88] C.T. Maravelias and I.E. Grossmann. An hybrid MILP/CP decomposition approach for the short term scheduling of multipurpose batch plants. *Computers and Chemical Engineering*, 28(10) :1921–1949, 2004.

[89] C.T. Maravelias and I.E. Grossmann. On the relation of continuous and discrete time models for the state-task network formulation. *AIChE Journal*, 52(2) :843–849, 2006.

[90] H. Meyr, M. Wagner, and J. Rohde. *Structure of Advanced Planning Systems*, pages 99–104. Springer, 2nd edition, 2002.

[91] E. Monk. *Concepts in Enterprise Resource Planning*. Course Technology, 2nd edition, 2005.

[92] C.L. Monma and C.N. Potts. On the complexity of scheduling with batch setup times. *Operations Research*, 37(5) :798–804, 1989.

[93] C.L. Monma and C.N. Potts. Analysis of heuristics for preemptive parallel machine scheduling with batch setup times. *Operations Research*, 41(5) :981–993, 1993.

[94] K. Neumann, C. Schwindt, and N. Trautmann. Advanced production scheduling for batch plants in process industries. *OR Spectrum*, 24(3) :251–279, 2002.

[95] C.T. Ng, T.C.E. Cheng, and J.J. Yuan. A note on the single machine serial batching scheduling problem to minimize maximum lateness with precedence constraints. *Operations Research Letters*, 30 :66–68, 2002.

[96] R.K. Oliver and M.D. Webber. *Supply Chain Management : Logistics catches up with strategy*. London, 1992. Reprint from Outlook 1982.

[97] J. Orlicky. *The Successful Computer System : its Planning development and Management in a Business Enterprise*. McGraw-Hill, 1968.

[98] M. Pinedo. *Scheduling : Theory, Algorithms, and Systems*. Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 2002.

[99] G.W. Plossl. *Orlicky's Material Requirement Planning*. McGraw-Hill, 2nd edition, 1994.

[100] Y. Pochet and L.A. Wolsey. Solving multi-item lot sizing problems using strong cutting planes. *Management Science*, 37(1) :53–67, 1991.

[101] C.N. Potts and M.Y. Kovalyov. Scheduling with batching : a review. *European Journal of Operational Research*, 120(2) :228–249, 2000.

[102] K. Rajaram and U.S. Karmarkar. Campaign planning and scheduling for multiproduct batch operations with applications to the food-processing industry. *Manufacturing and Service Operations Management*, 6(3) :253–269, 2004.

[103] A. Robert and C. Le Pape. Lot-streaming and pegging to build good and plan-consistent schedules. In *12th IFAC Symposium on Information Control Problems in Manufacturing*, pages 703–708, May 2006.

[104] A. Robert and C. Le Pape. Optimizing size of batches between production planning and scheduling to improve APS performance. In *8th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP 2007)*, Juillet 2007.

[105] A. Robert, C. Le Pape, F. Paulin, and F. Sourd. Une nouvelle approche de l'intégration de la planification de production et de l'ordonnancement. In *7ème Rencontre des Jeunes Chercheurs en Intelligence Artificielle, RJCIA 2005*, pages 15–28, 2005.

[106] A. Robert, C. Le Pape, and F. Sourd. Lot-streaming et pegging pour l'intégration de la planification et de l'ordonnancement de production. In *Actes du 7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2006)*, pages 273–290. Presses Universitaires de Valenciennes, 2006.

[107] F. Sahin and E.P. Robinson. Flow coordination and information sharing in supply chain : Review, implications, and directions for future research. *Decision Science*, 33(4) :505–535, 2002.

[108] E.A. Silver, D.F. Pyke, and R. Peterson. *Inventory Management and Production Planning and Scheduling*. John Wiley and Sons, 1998.

[109] D. Simchi-Levi, Kaminsky P., and E. Simchi-Levi. *Designing and Managing the Supply Chain. Concepts, strategies and case studies*. McGraw-Hill Education, 2003.

[110] F. Sourd. Scheduling with periodic availability constraints and irregular cost functions. *Recherche Opérationnelle - RAIRO*, 41(2) :141–154, 2007.

[111] C.R. Sox and Y. Gao. The capacitated lot-sizing problem with setup carry-over. *IIE Transactions*, 31(2) :173–181, 1999.

[112] C. Sürie. Campaign planning in time-indexed model formulations. *International Journal of Production Research*, 43(1) :49–66, 2005.

[113] C. Sürie and H. Stadtler. The capacitated lot sizing problem with linked lot sizes. *Management Science*, 49(8) :1039–1054, 2003.

[114] H. Stadtler. Mixed integer model formulations for dynamic multi-item multi-level capacitated lot sizing. *European Journal of Operational Research*, 94(3) :561–581, 1996.

[115] H. Stadtler. Reformulations of the shortest route model for dynamic multi-item multi-level capacitated lot sizing. *OR Spectrum*, 19 :87–96, 1997.

[116] H. Stadtler. Supply chain management and advanced planning - basics, overview and challenges. *European Journal of Operational Research*, 163(3) :575–588, 2005.

[117] H. Stadtler and C. Kilger. *Supply Chain Management and Advanced Planning — Concepts, Models, Software and Case Studies*. Springer, 3rd edition, 2005.

[118] A.T. Staggemeier and A.R. Clark. A survey of lot-sizing and scheduling models. In *23rd Symposium of the Brazilian Operational Research Society (SOBRAPO)*, 2001.

[119] K.C. Tan. A framework of supply chain management literature. *European Journal of Purchasing and Supply Management*, 7 :39–48, 2001.

[120] E. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica 5*, pages 247–255, 1985.

[121] H. Tempelmeier and M. Derstroff. A Lagrangian-based heuristic for dynamic multilevel multi-item constrained lot sizing with setup times. *Management Science*, 42 :738–757, 1996.

[122] H. Tempelmeier and S. Helber. A heuristic for dynamic multi-item multi-level capacitated lot sizing for general product structures. *European Journal of Operational Research*, 75(2) :296–311, 1994.

[123] D.J. Thomas and P.M. Griffin. Coordinated supply chain management. *European Journal of Operational Research*, 94(1) :1–15, 1996.

[124] C. Timpe. Solving planning and scheduling problems with combined integer and constraint programming. *OR Spectrum*, 24(4) :431–448, 2002.

[125] D. Trietsch and K.R. Baker. Basic techniques for lot streaming. *Operation Research*, 41(6) :1065–1076, 1993.

[126] C.P.M. van Hoesel and 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.

[127] R.G. Vickson. Optimal lot streaming for multiple products in a two-machine flow shop. *European Journal of Operational Research*, 85(3) :556–575, 1995.

[128] B.E. Vollman, Berry W.L., and D.C. Whybark. *Manufacturing Planning and Control Systems*. McGraw, 1997.

[129] S. Voß and D.L. Woodruff. *Introduction to Computational Optimization Models for Production Planning in a Supply Chain*. Springer, 2003.

[130] H.M. Wagner and T.M. Whitin. A dynamic version of the economic lot size model. *Management Science*, 5(1) :89–96, 1958.

[131] C. Wolosewicz, S. Dauzère-Pérès, and R. Aggoune. A new approach for solving integrated planning and scheduling problem. In A. Dolgui, G. Morel, and C.E. Pereira, editors, *12th IFAC Symposium on Information Control Problems in Manufacturing - INCOM'2006*, 2006.

[132] L.A. Wolsey. Progess with single-item lot-sizing. *European Journal of Operational Research*, 86(3) :395–401, 1995.

[133] J.J. Yuan, Y.X. Lin, T.C.E. Cheng, and C.T. Ng. Single machine serial-batching scheduling problem with a common batch size to minimize total weighted completion time. *International Journal of Production Economics*, 105(2) :402–406, 2007.

Resumé

La thèse s'est déroulée dans le cadre du développement de l'APS (*Advanced Planning and Scheduling*) ILOG *Plant PowerOps*, logiciel d'optimisation de la production à moyen et court termes (planification et ordonnancement). Dans ce contexte, nous nous sommes posé des questions relatives à la connexion entre ces deux niveaux de prises de décisions, notamment au dimensionnement des batches de production physiquement mis en oeuvre dans une usine, ce qui constitue la problématique centrale de notre thèse. Après avoir décrit la problématique dans laquelle s'inscrit notre sujet, nous avons formulé précisément le problème de *batching* qui est au coeur de nos travaux. Dans un premier temps, nous avons effectué une étude de la complexité de différents sous-cas. Puis, un algorithme de programmation dynamique a été proposé pour résoudre un cas particulier, et un programme linéaire en nombres entiers a été formulé afin de traiter des cas plus généraux, via sa résolution par le solveur ILOG CPLEX. Ce modèle a été étendu, par l'intégration de nombreuses contraintes additionnelles, permettant de tester ses performances sur un jeu de données composé de 320 instances, élaboré au cours de notre thèse. Enfin, sa flexibilité a été démontrée par l'étude d'un cas réel en industrie agroalimentaire dont nous avons pu résoudre les problèmes de *batching* sous-tendus, en intégrant de nouvelles contraintes, spécifiques à cette application industrielle. Ces expérimentations ont permis de valider les approches proposées et d'attester de leur robustesse dans la perspective d'une utilisation pratique.

Abstract

This thesis has been done during the development of the APS (*Advanced Planning and Scheduling*) ILOG *Plant PowerOps*, which is dedicated to the mid-term and short-term optimization of the production (planning and scheduling). In this context, we considered some questions related to the connection between these two levels of decisions, namely as far as the dimensioning of the production batches, physically executed in the plant, is concerned. This is the main question stated in the thesis. Afer a description of the problematic to which our subject belongs, we precisely formulated the *batching* problem. First, we studied the complexity of various subcases. Then, a dynamic program has been proposed in order to solve a particular case, and a mixed integer linear program has been elaborated to take into account more general cases, this program is solved by ILOG CPLEX. This model has been extended, by integrating various additional constraints, allowing to test its performance on a benchmark composed by 320 instances, which has been elaborated during our thesis. At last, its flexibility has been shown by studying a practical case coming from the food-processing industry. Indeed, we have been able to solve the related *batching* problems, by integrating specific constraints coming from this industrial application. These experiments allowed to validate the proposed approaches and to attest their robustness in the perspective of a practical use.